Runtime Systems Research

KC Sivaramakrishnan (kcsrk.info)

FSTTCS Mentoring workshop
16th December 2025

Who am I — KC Sivaramakrishnan

- CS Prof at IIT Madras
  - Programming languages, formal verification and systems
- A core maintainer of the ***OCaml*** programming language
- CTO at Tarides
  - Building functional systems using ***OCaml***
  - Maintainers of the OCaml compiler and platform tools

Programming language research

- What is PL research?
- A bridge between ***humans*** and ***machines***

PL research is about helping people build *reliable software* by *designing abstractions and systems* that make *intent precise*, *mistakes harder to make and behaviour efficient but predictable*.

- Tools
  - Language design (Expression), Formal semantics (Meaning), Type systems & Verification (Guarantees), Compilers & Runtime systems (Execution)

Runtime Systems Research

*How do we make the **promises** of a language hold in **reality**?*

- Themes
  - Memory management, JIT, Concurrency, Performance, etc.
  - Overlaps with Networks, Storage, OS research
- Runtime systems sit at the boundary between ***beautiful ideas*** and ***messy reality***
  - …. *and so does a researcher's career.*
- Goal of the talk
  - Insight into runtime systems research through my own journey

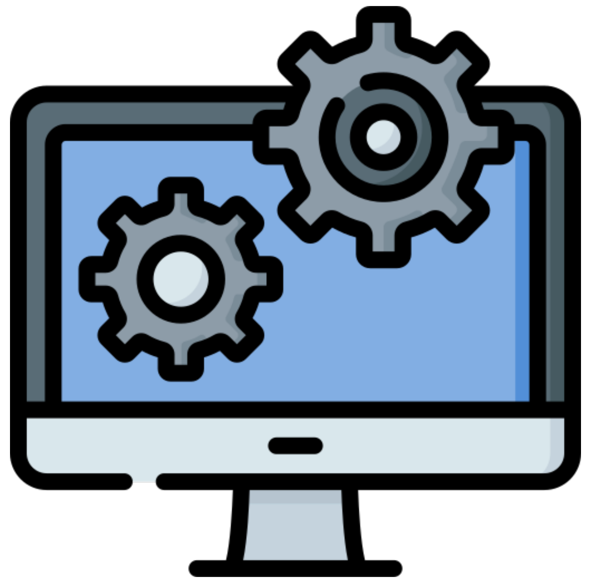
Language Design

Type Systems

Verification

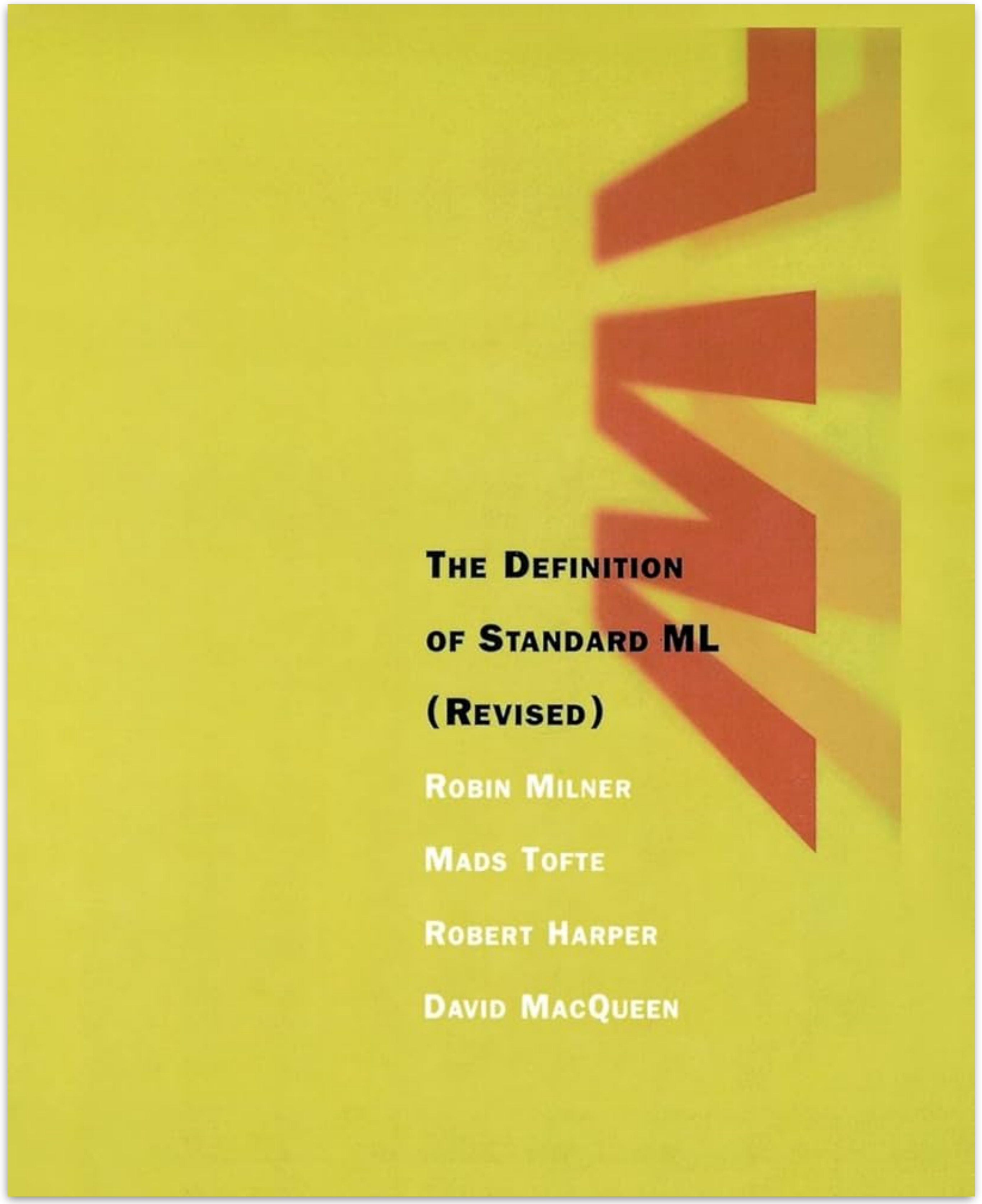
Compilation

Runtime Systems



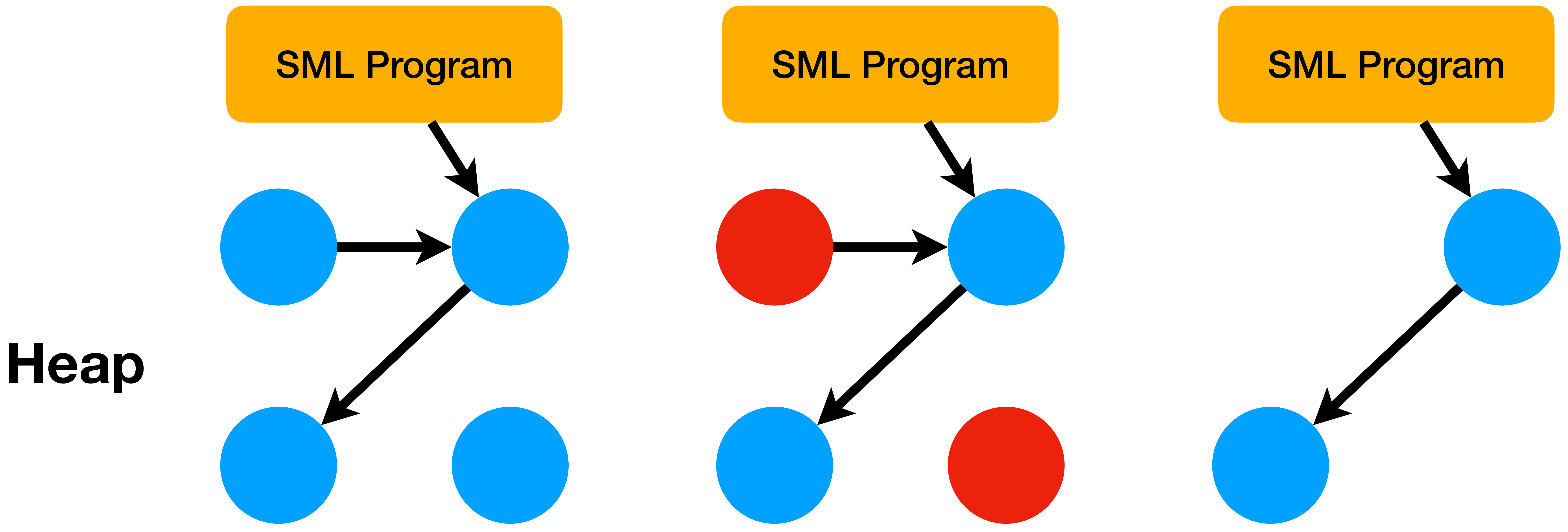
MultiMLton

- PhD starting project @ Purdue University
- MultiMLton — a multicore-aware extension of MLton Standard ML compiler
  - **Standard ML:** a rigorously specified FP language in ML family
  - **MLton:** a whole-program optimising compiler for the Standard ML language
- Extend this to take advantage of multi-core
  - Language design ← Lukasz Ziarek
  - Parallelism support in the runtime ← me

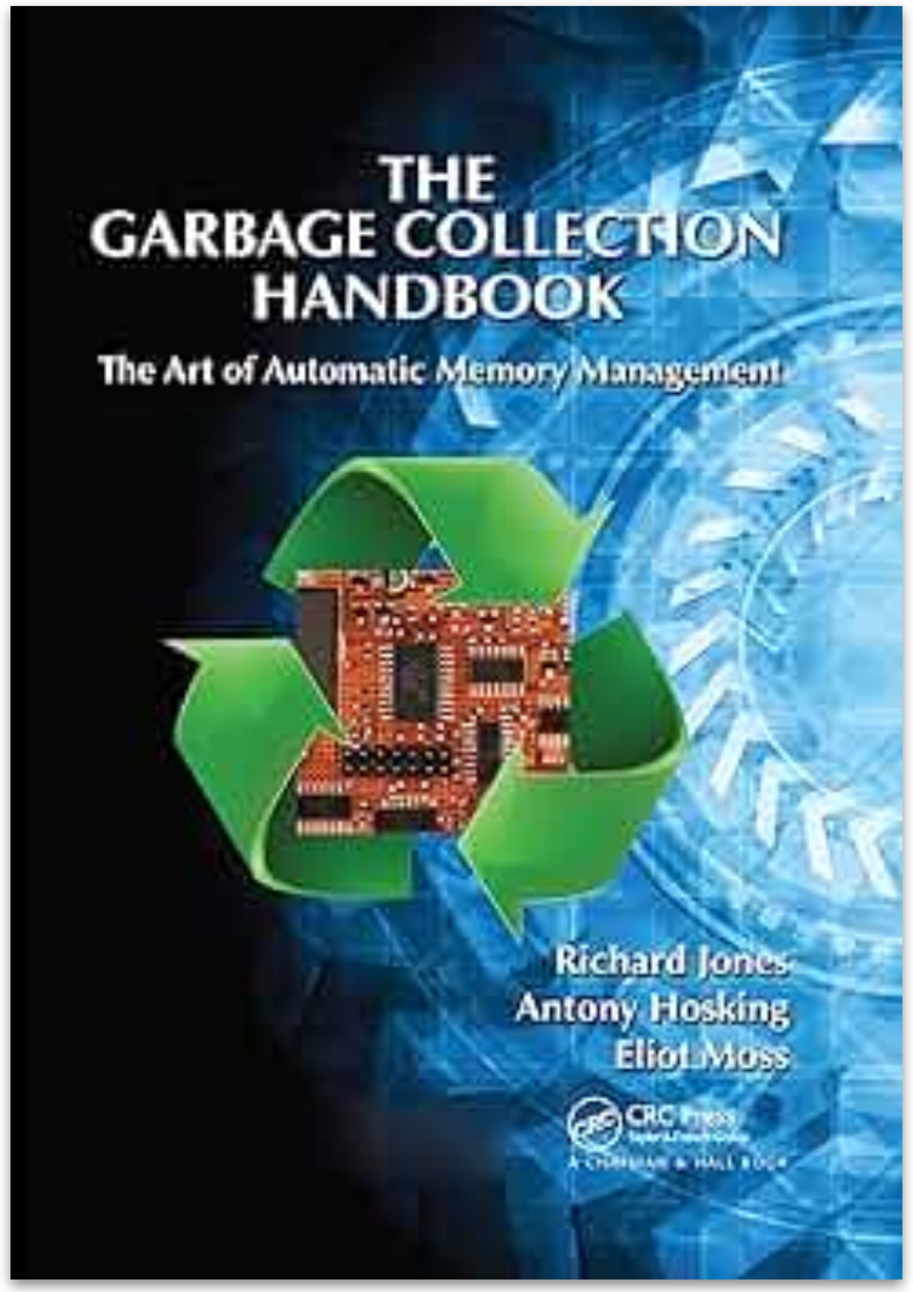


Parallelism support in the runtime

- Automatic memory management with a garbage collector (GC)

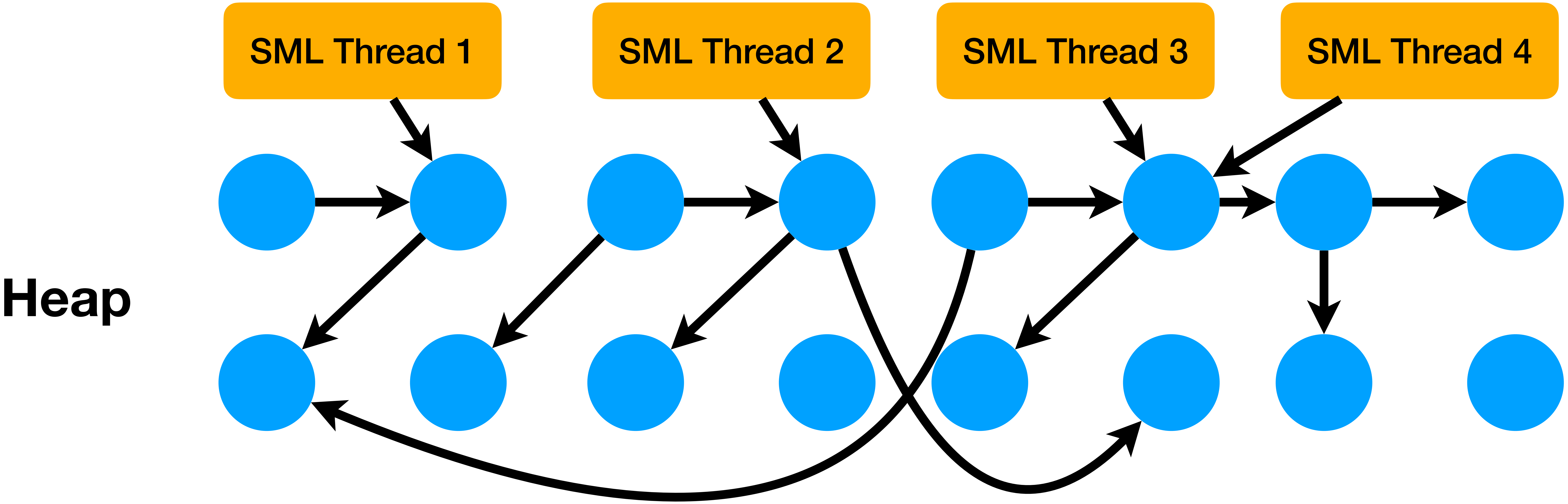


… also Java, .NET, Python, Go, JavaScript, OCaml, Haskell, JavaScript, etc.


THE GARBAGE COLLECTION HANDBOOK
The Art of Automatic Memory Management
Richard Jones
Antony Hosking
Eliot Moss
CRC Press

- No "perfect" GC
  - Trade-offs — throughput, latency, memory usage, complexity,…

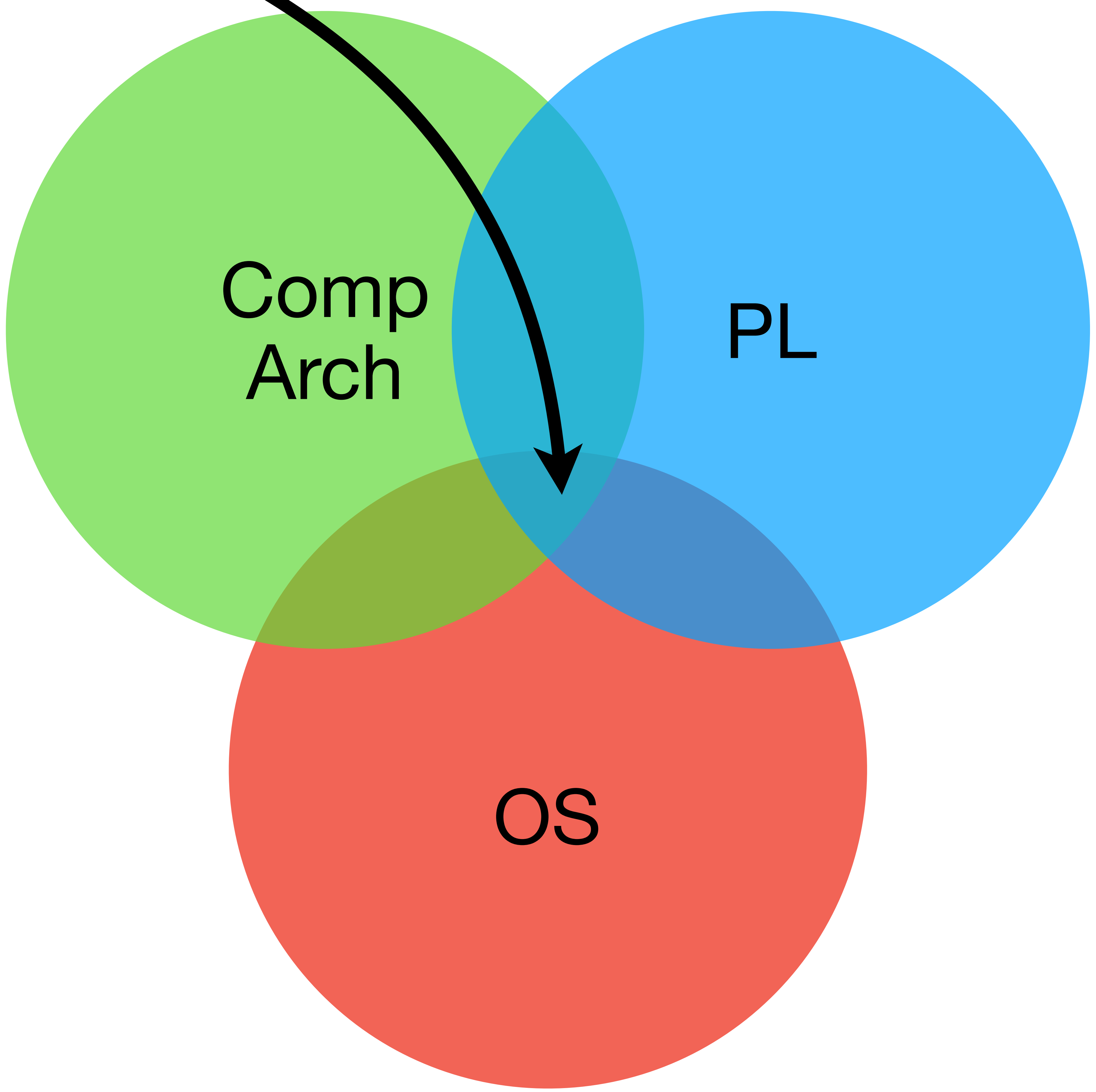
Parallel Garbage Collector



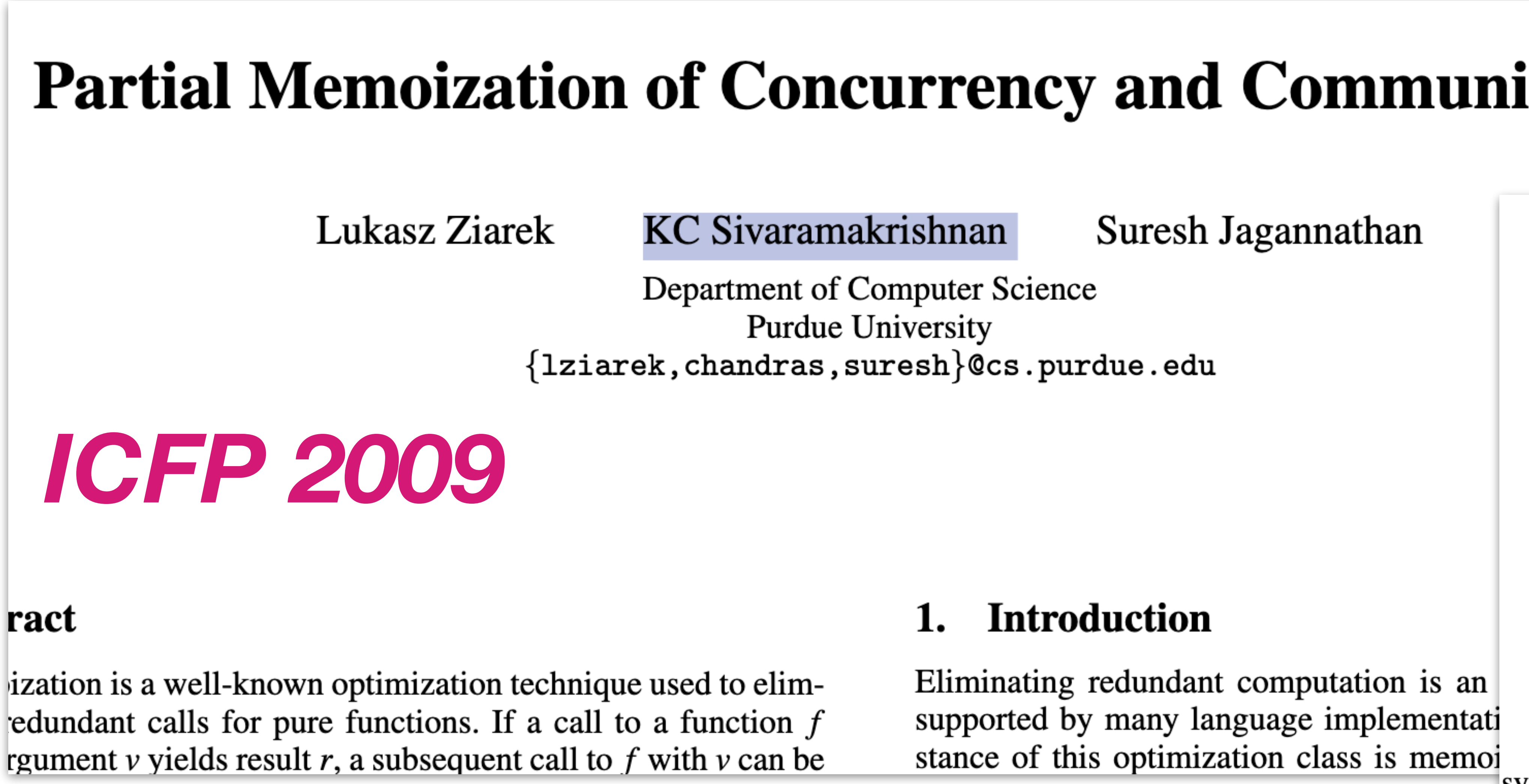
- Should we ***stop all*** the SML threads?
- Should we collect all the garbage at once or ***incrementally***?
- Should the GC run ***concurrently*** with the SML threads?
- How do the GC ***interact*** with the OS and (micro-)architectural quirks?

GC research area

- Quite challenging/rewarding
- Many ***open problems***, especially as the compute stack gets heterogeneous and distributed
- ***Impactful*** — any improvement can benefit all users of the language
- ***But…***
  - Requires massive, long-term engineering effort
    - Bugs are rare, timing-dependent, and hard to reproduce
    - Research ideas are tightly entangled with infrastructure
  - Many industrial-strength GCs; “Solved” problem?
    - Easy to get stuck building; hard to know when to publish



PhD Research



Partial Memoization of Concurrency and Communication

Lukasz Ziarek KC Sivaramakrishnan Suresh Jagannathan
Department of Computer Science
Purdue University
{lziarek,chandras,suresh}@cs.purdue.edu

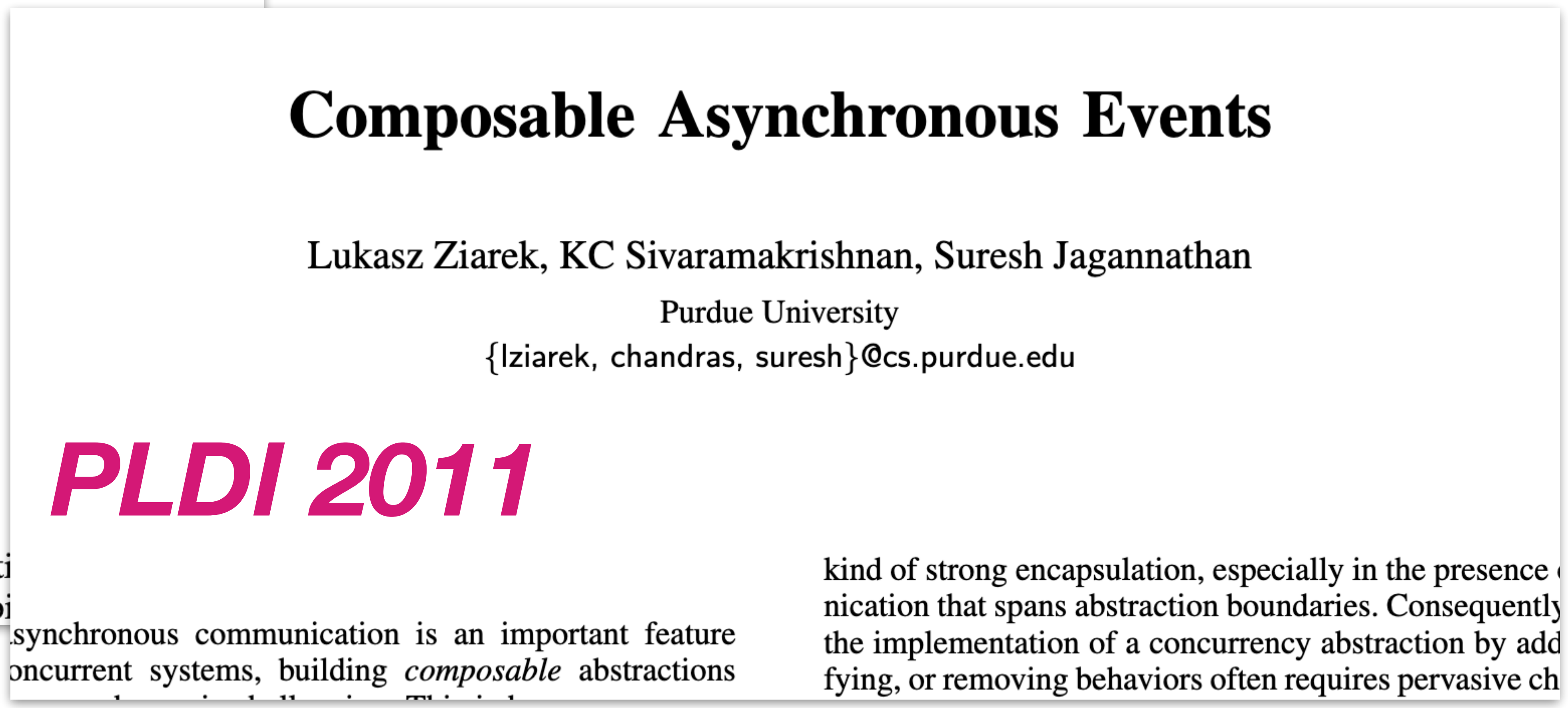
ICFP 2009

ract

ization is a well-known optimization technique used to elim-
edundant calls for pure functions. If a call to a function f
rgument v yields result r, a subsequent call to f with v can be

1. Introduction

Eliminating redundant computation is an
supported by many language implementati
stance of this optimization class is memoi



Composable Asynchronous Events

Lukasz Ziarek, KC Sivaramakrishnan, Suresh Jagannathan
Purdue University
{lziarek, chandras, suresh}@cs.purdue.edu

PLDI 2011

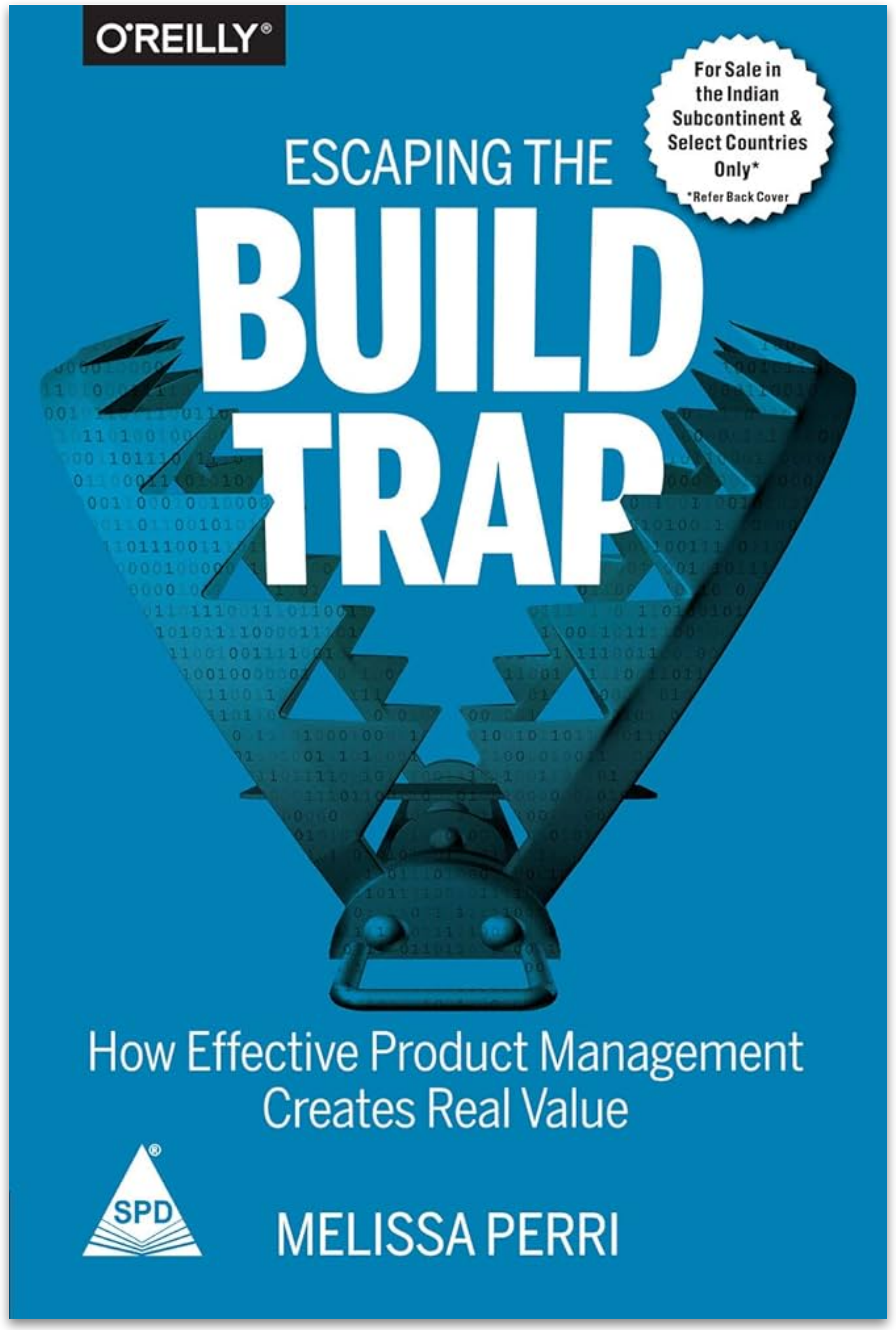
synchronous communication is an important feature
oncurrent systems, building *composable* abstractions

kind of strong encapsulation, especially in the presence
nication that spans abstraction boundaries. Consequently
the implementation of a concurrency abstraction by add
fying, or removing behaviors often requires pervasive ch

- Started my PhD in 2008
- I was supporting language research projects in my group
- ***What was my research goal?***
  - Build Trap!

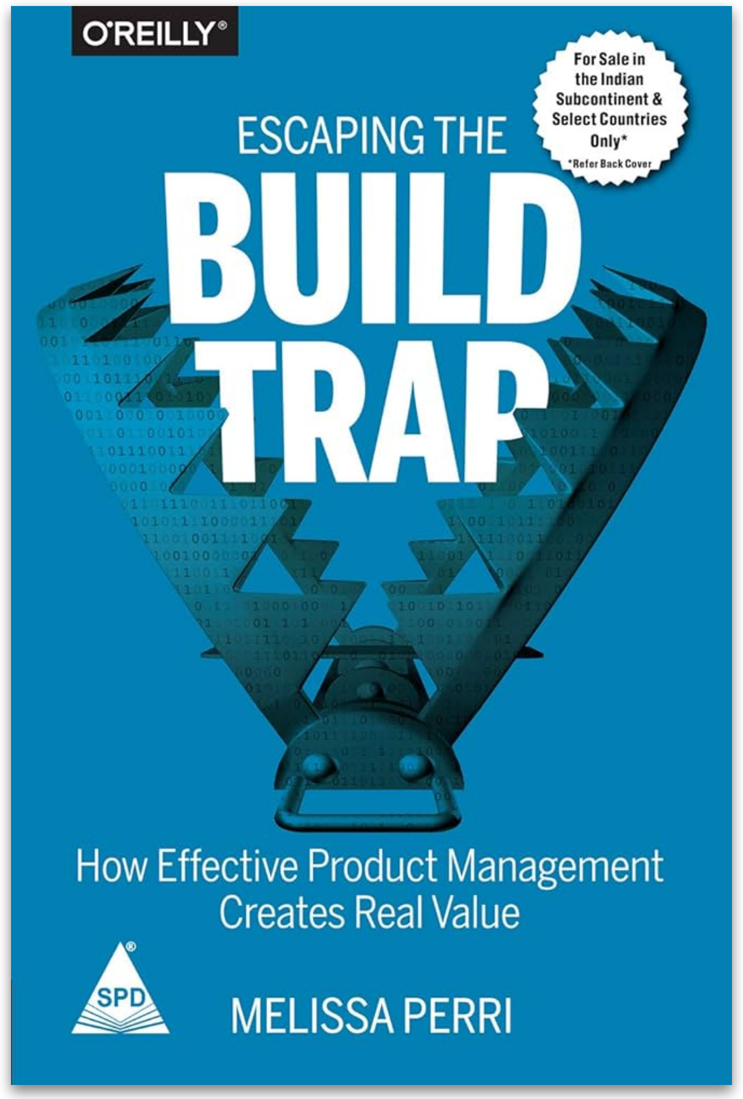
Build Trap!

- ***Keep building, value will emerge on its own!***
- Companies falling into measuring success
  - by ***outputs*** — features shipped, code written, systems built
  - Instead of ***outcomes*** — customer value, impact, learning
- For runtime systems research
  - ***Outputs*** — code written, systems built, bugs fixed
  - ***Outcomes*** — improving the state of the art ⇒ ***publishing papers***
- Risks in systems research
  - High-upfront cost, complexity of supporting infra, performance rabbit holes, insufficient checkpoints


O'REILLY
For Sale in the Indian Subcontinent & Select Countries Only*
*Refer Back Cover
ESCAPING THE
BUILD TRAP
How Effective Product Management Creates Real Value
SPD
MELISSA PERRI

Escaping the build trap

- **Start with the research question/hypothesis that you want to test**
  - Starting to build without one is a definite path to the build trap
- **Treat the system as an instrument, not the product**
  - “Avoid Success at All Costs”, SPJ about GHC
  - Research PL — Koka, Effekt, Links, Flix, Hazel…
- **Force early articulation of the paper**
  - If you can’t write the introduction, you don’t yet know why you’re building
  - SPJ, “Writing a research paper is the way to do research”
- **Stop when the marginal build effort doesn’t sharpen the claim**
  - Engineering progress feels tangible, but insight is fuzzier
  - Reviewers reward the latter!



O'REILLY

For Sale in the Indian Subcontinent & Select Countries Only* *Refer Back Cover

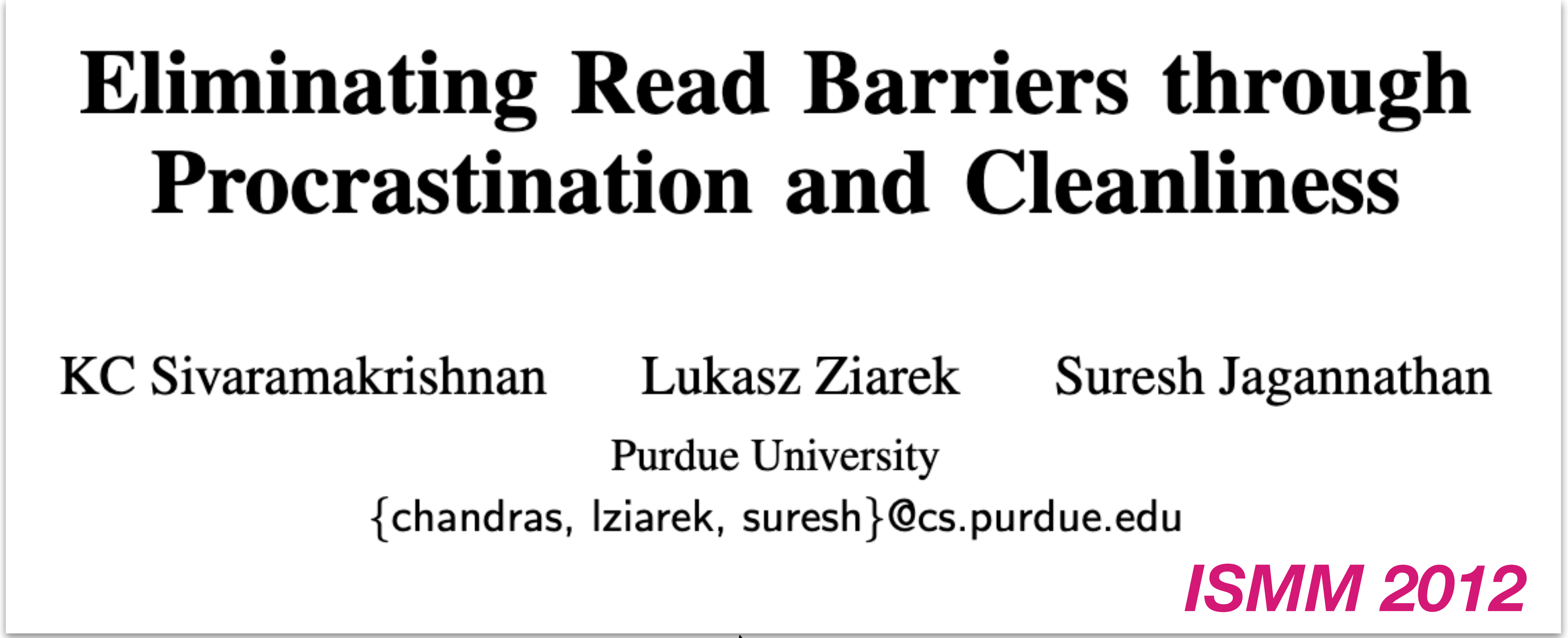
ESCAPING THE BUILD TRAP

How Effective Product Management Creates Real Value

SPD

MELISSA PERRI

Did escape the build trap eventually 😅

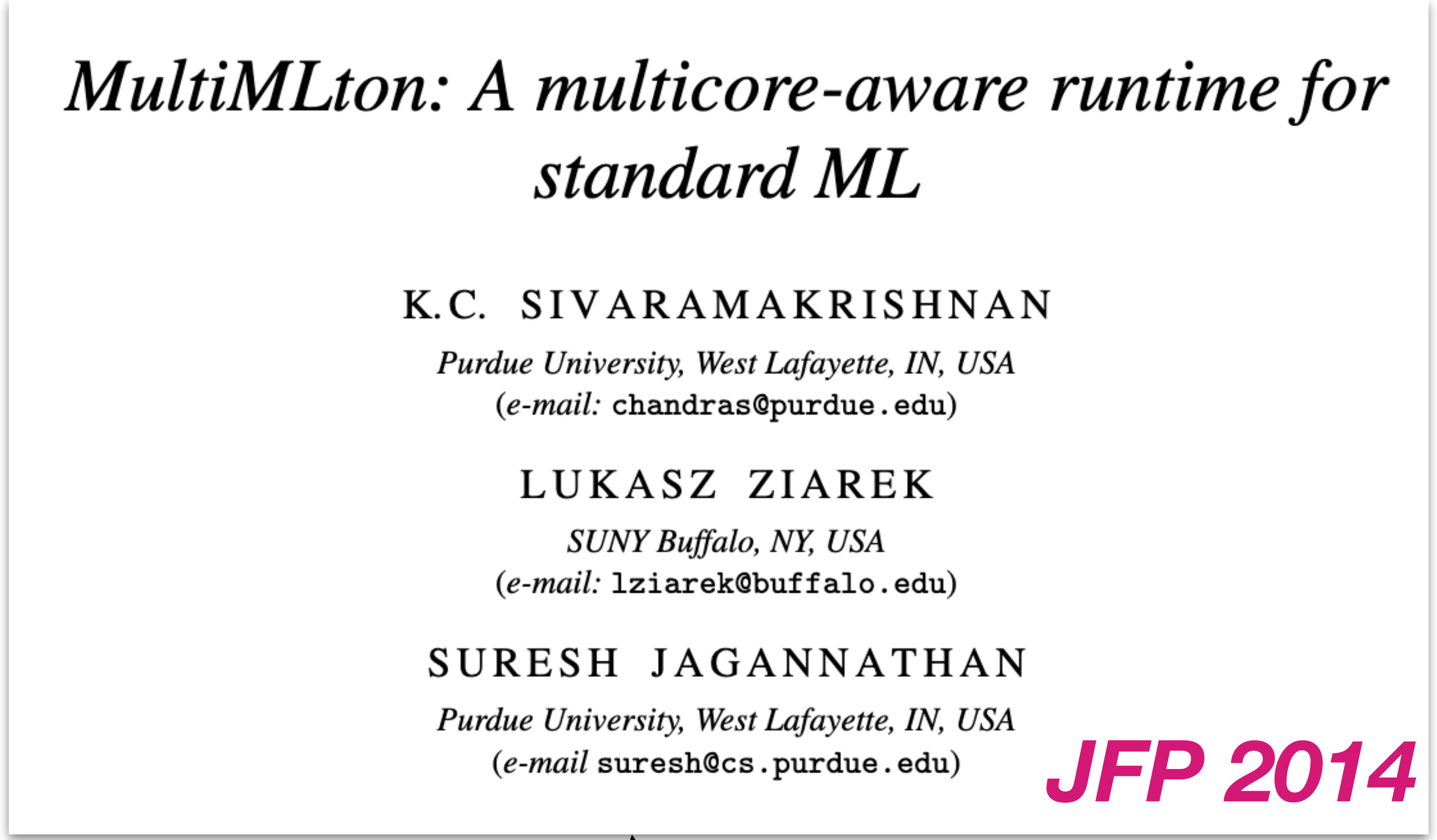


Eliminating Read Barriers through Procrastination and Cleanliness

KC Sivaramakrishnan Lukasz Ziarek Suresh Jagannathan

Purdue University
{chandras, lziarek, suresh}@cs.purdue.edu

ISMM 2012



MultiMLton: A multicore-aware runtime for standard ML

K.C. SIVARAMAKRISHNAN
Purdue University, West Lafayette, IN, USA
(*e-mail:* chandras@purdue.edu)

LUKASZ ZIAREK
SUNY Buffalo, NY, USA
(*e-mail:* lziarek@buffalo.edu)

SURESH JAGANNATHAN
Purdue University, West Lafayette, IN, USA
(*e-mail* suresh@cs.purdue.edu)

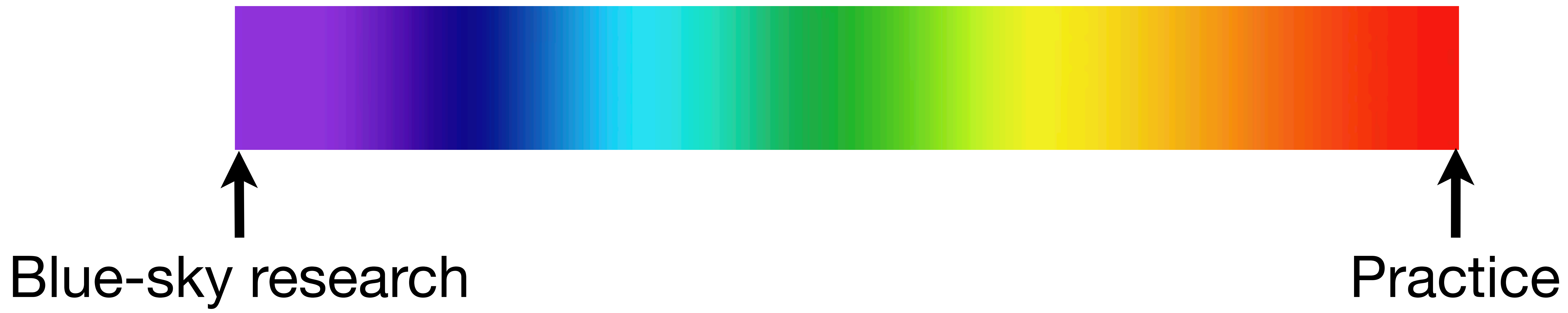
JFP 2014

Using pervasive concurrency in the language to trade off GC overheads

Complete language and runtime system; evaluation on 768-core behemoth

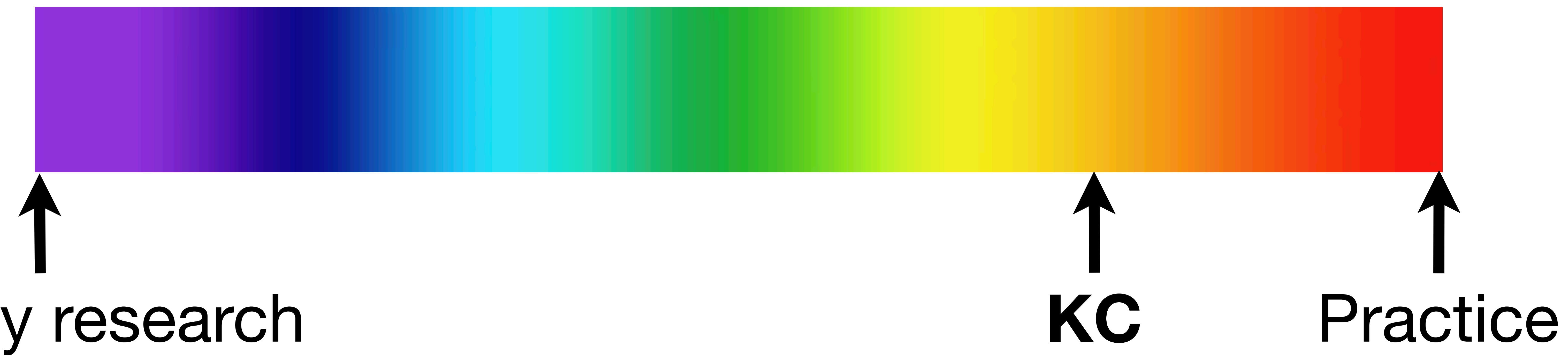
- Not Core A* (🙄) PL systems conferences — POPL, PLDI, ICFP, OOPSLA, ASPLOS…
    - *… and that was ok, I had fun doing this research*

Finding my research focus



- PL research takes a long time to mature
  - GC — 60s (Lisp), 00s (Java)
  - Strong static typing — 70s (ML), 00s (Java), 10s (TypeScript)
  - Concurrency — Research 80s, 10s (Go, Rust)
  - Static memory safety — Research 90s, 10s (Rust)

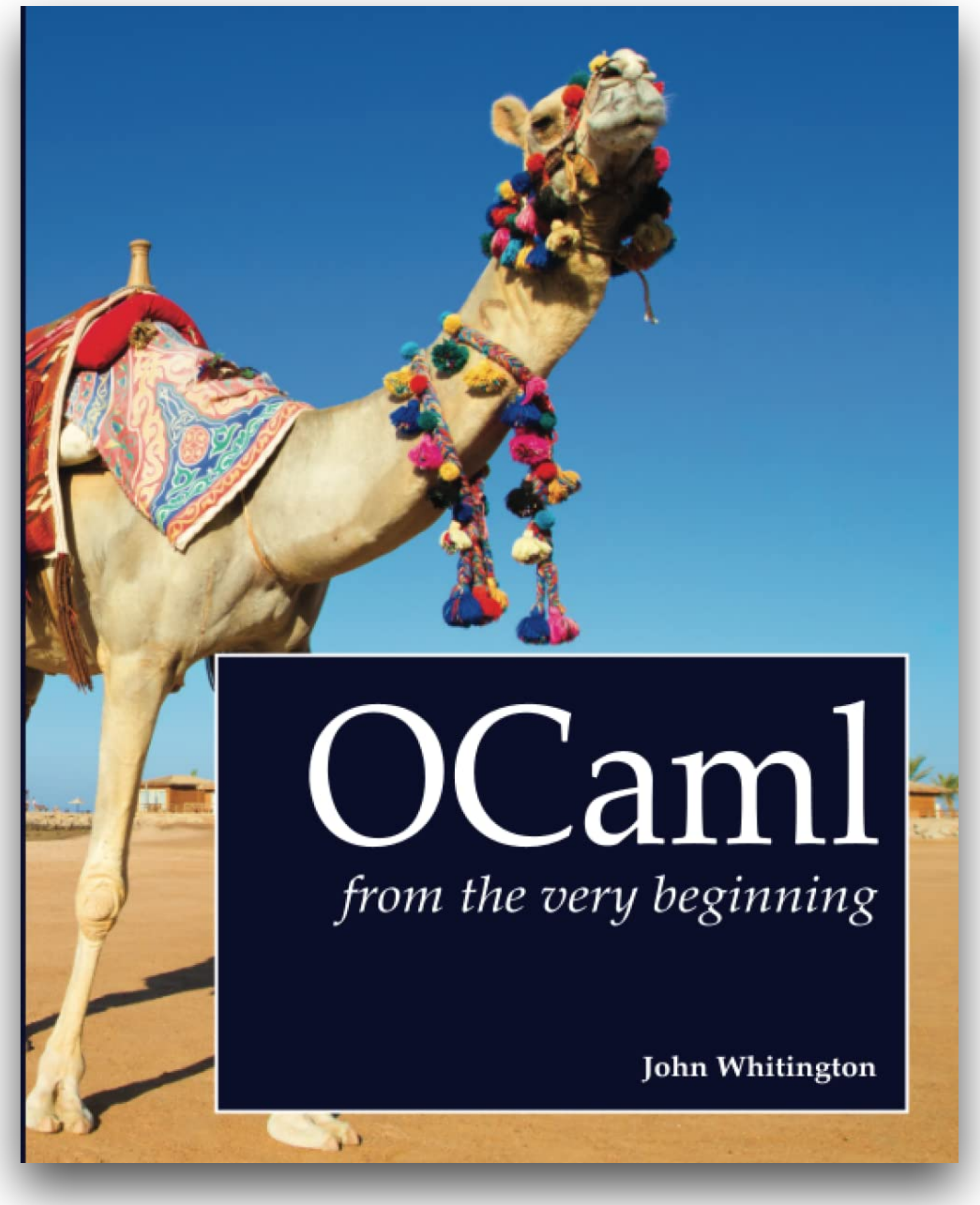
Finding my research focus

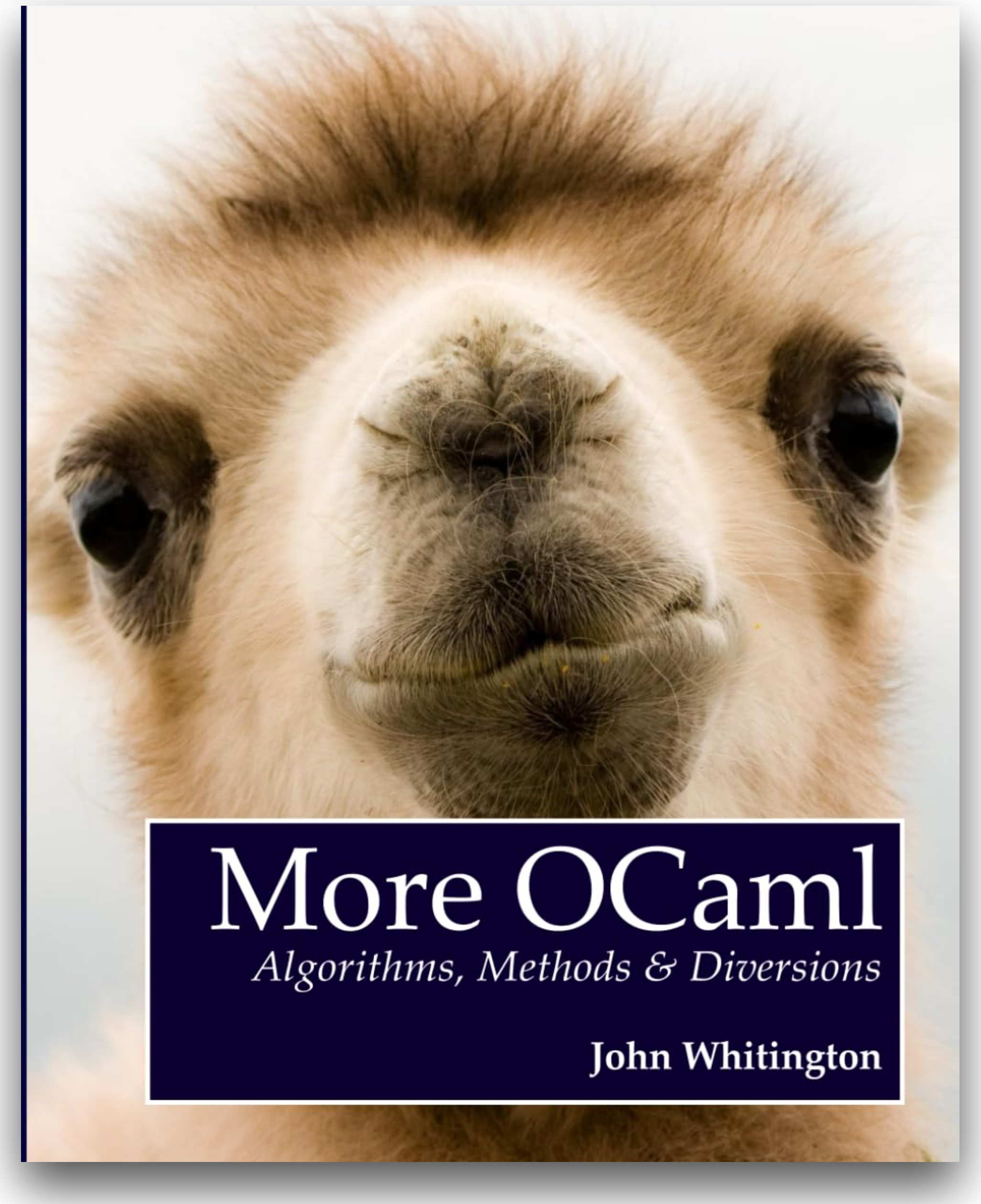


- Loved being in the area of ***translating research to practice***
  - Runtime systems are naturally amenable to this
- Have an ***impact*** outside of research papers
  - Benefit “real” users, not “imagined” ones
  - *MultiMLton hasn’t been developed since 2014*
- Like the ***academic freedom*** to move about in the spectrum

Multicore OCaml

- Post-doc @ U Cambridge
- **Multicore OCaml** — native support for concurrency and parallelism to the OCaml programming language
- OCaml
  - A functional-first programming language in the ML family
  - Projects — Rocq, Frama-C, Why3, F*
  - Industrial Users — Jane Street, Meta, ARM, SemGrep, Microsoft
  - ***Still sequential in 2014***
- *Promise of translating learning from MultiMLton to a widely-used language*

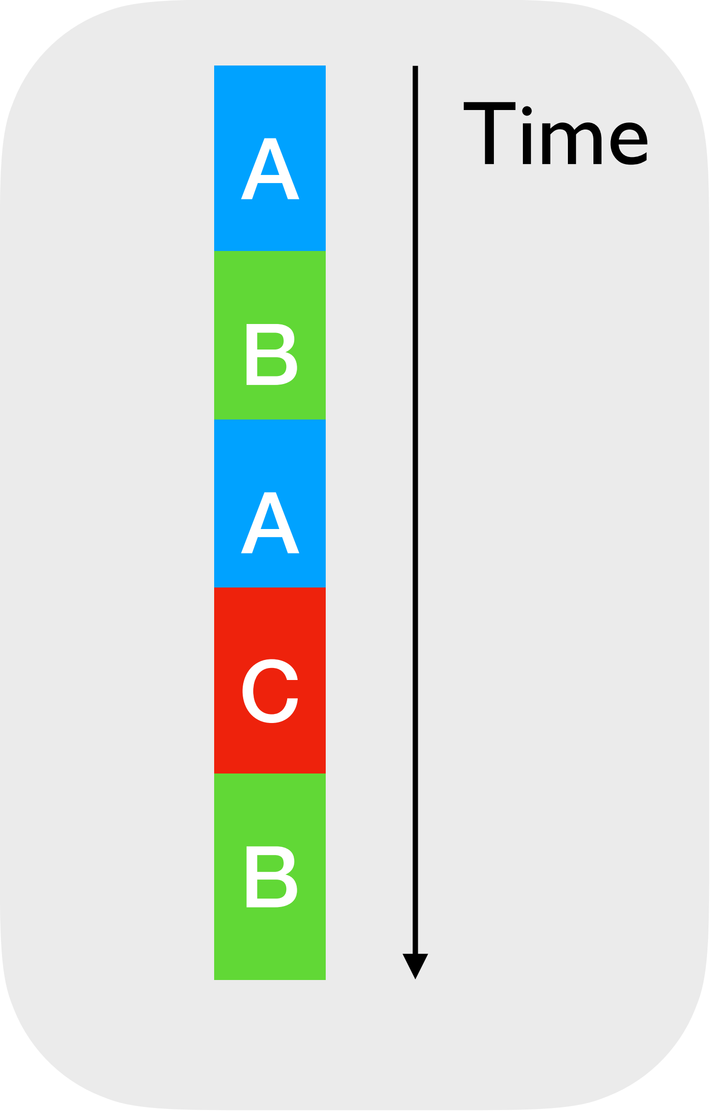




Multicore OCaml

- Native support for concurrency and parallelism to OCaml

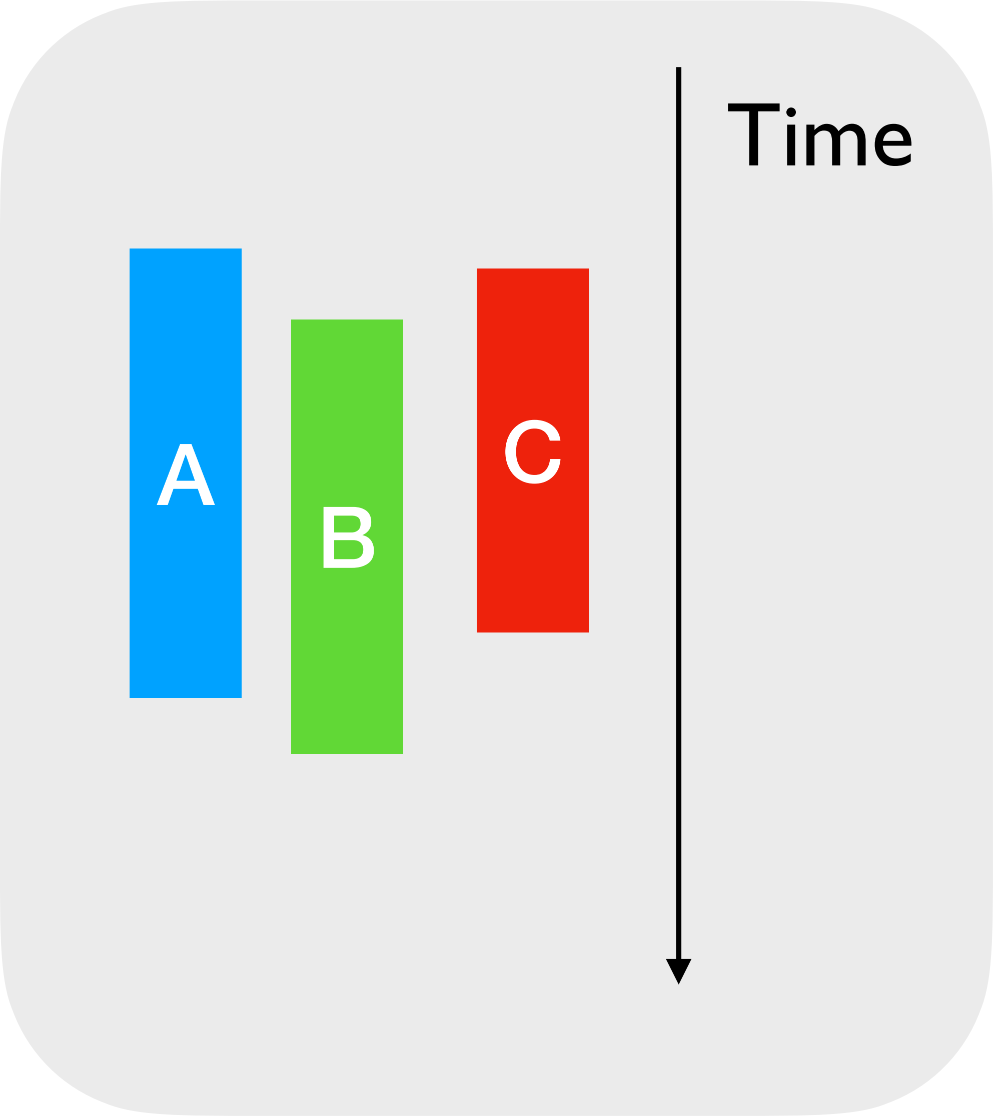
Concurrency



Interleaved execution

Effect Handlers

Parallelism



Simultaneous execution

Domains

Challenges

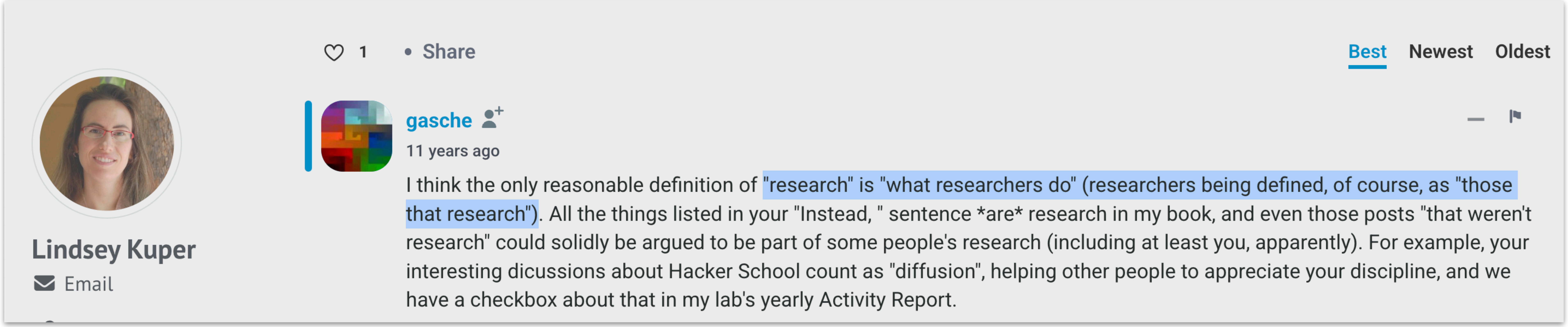
- **A new multicore garbage collector and multicore runtime system**
  - Replacing a car engine with a new one!
- **Make the language itself thread-safe**
  - OCaml is a safe language! (Unlike C/C++, Go)
- **Maintain feature and performance backwards compatibility!**
  - Most OCaml programs will continue to remain single-threaded

WHEN A USER TAKES A PHOTO, THE APP SHOULD CHECK WHETHER THEY'RE IN A NATIONAL PARK...
SURE, EASY GIS LOOKUP. GIMME A FEW HOURS.
...AND CHECK WHETHER THE PHOTO IS OF A BIRD.
I'LL NEED A RESEARCH TEAM AND FIVE YEARS.
IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

XKCD published in 2014
Today, bird recognition is a commodity ML task.

Research Focus

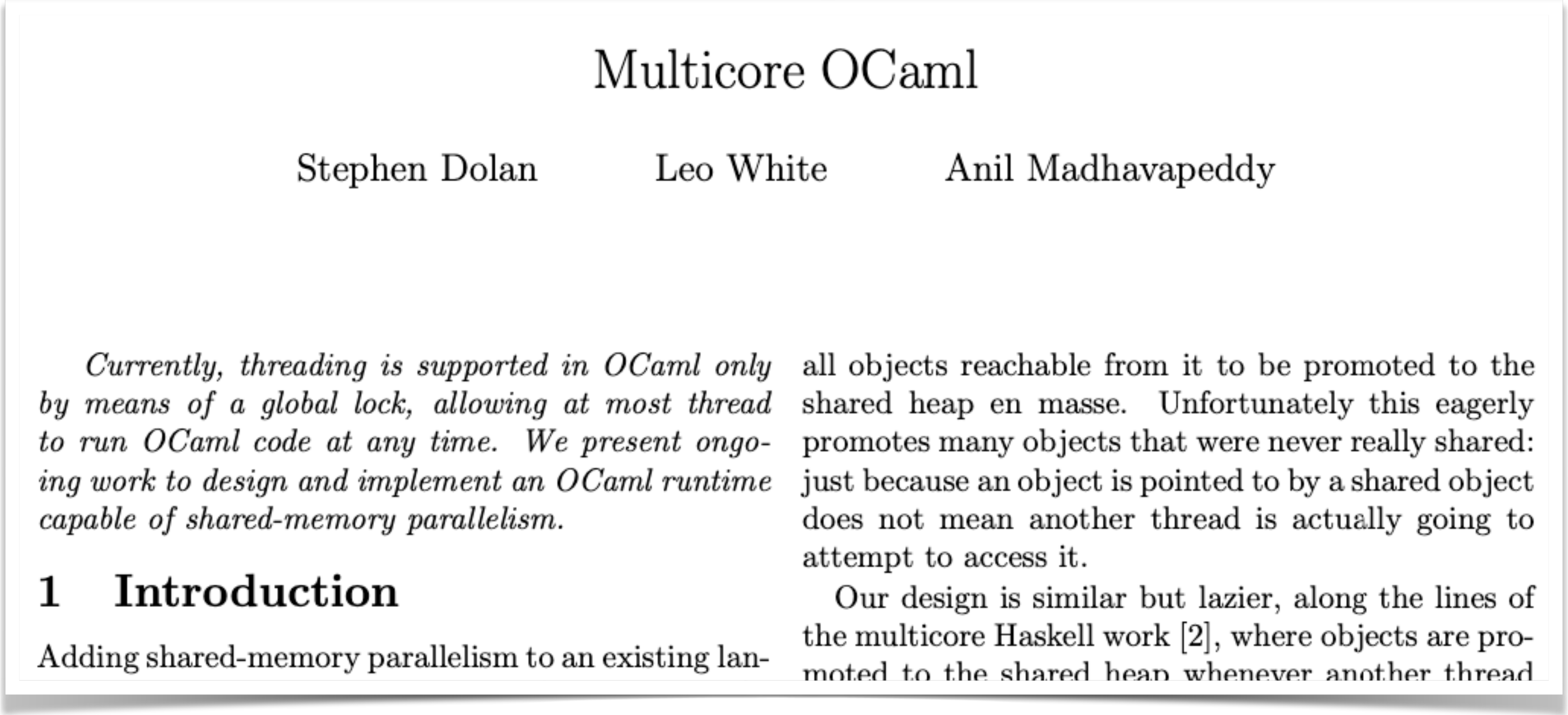
- The goal was ***upstreaming*** multicore features to OCaml
  - Publishing papers is a means to ***build credibility*** for upstreaming
  - Conscious tradeoff to have an impact beyond papers
- Building in the open
  - Liberally licensed open-source software
  - Quality >>> research-prototypes, < production (…initially)



gasche
11 years ago

I think the only reasonable definition of "research" is "what researchers do" (researchers being defined, of course, as "those that research"). All the things listed in your "Instead, " sentence *are* research in my book, and even those posts "that weren't research" could solidly be argued to be part of some people's research (including at least you, apparently). For example, your interesting dicussions about Hacker School count as "diffusion", helping other people to appreciate your discipline, and we have a checkbox about that in my lab's yearly Activity Report.

Starting out



Multicore OCaml

Stephen Dolan Leo White Anil Madhavapeddy

Currently, threading is supported in OCaml only by means of a global lock, allowing at most thread to run OCaml code at any time. We present ongoing work to design and implement an OCaml runtime capable of shared-memory parallelism.

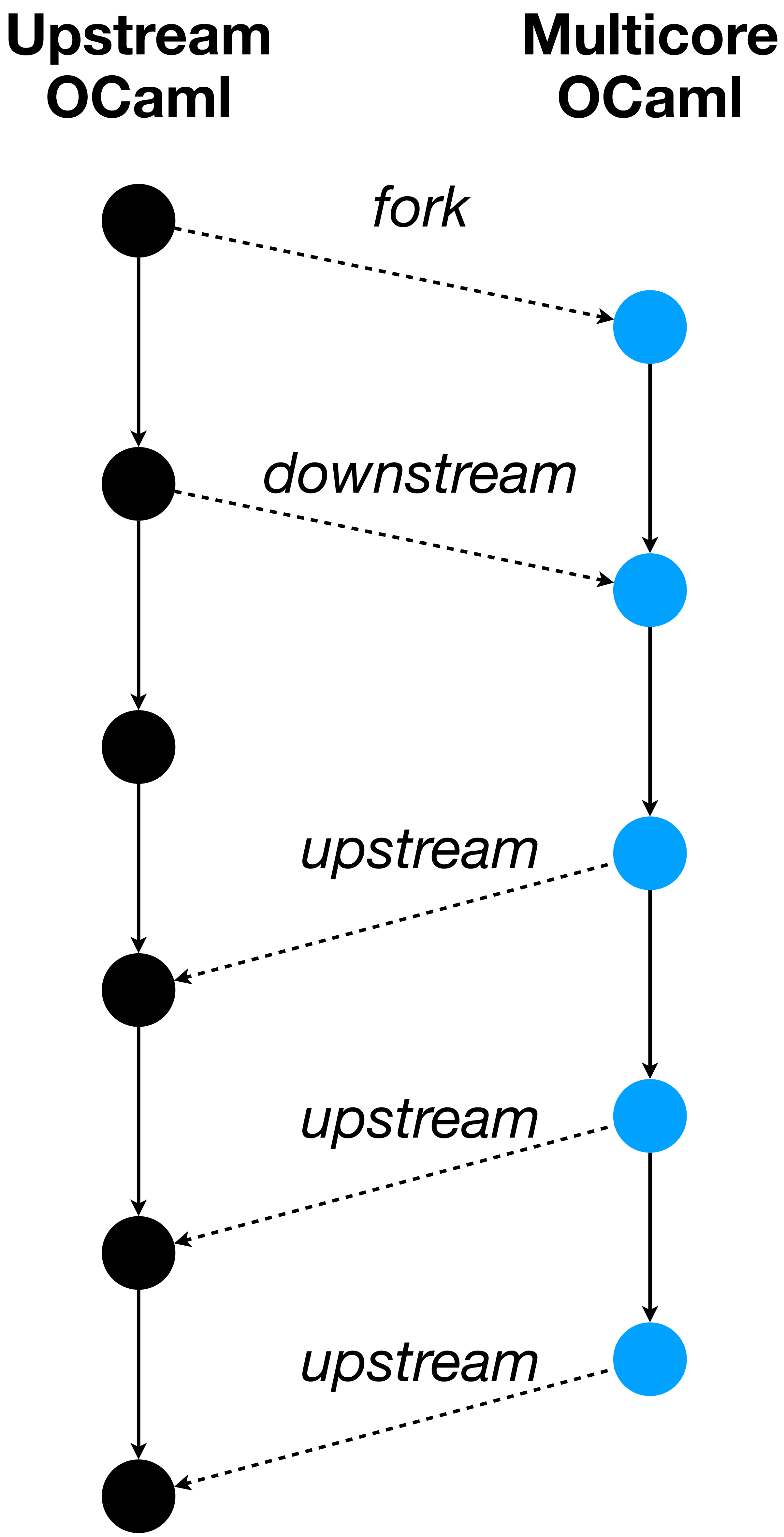
1 Introduction

Adding shared-memory parallelism to an existing lan-

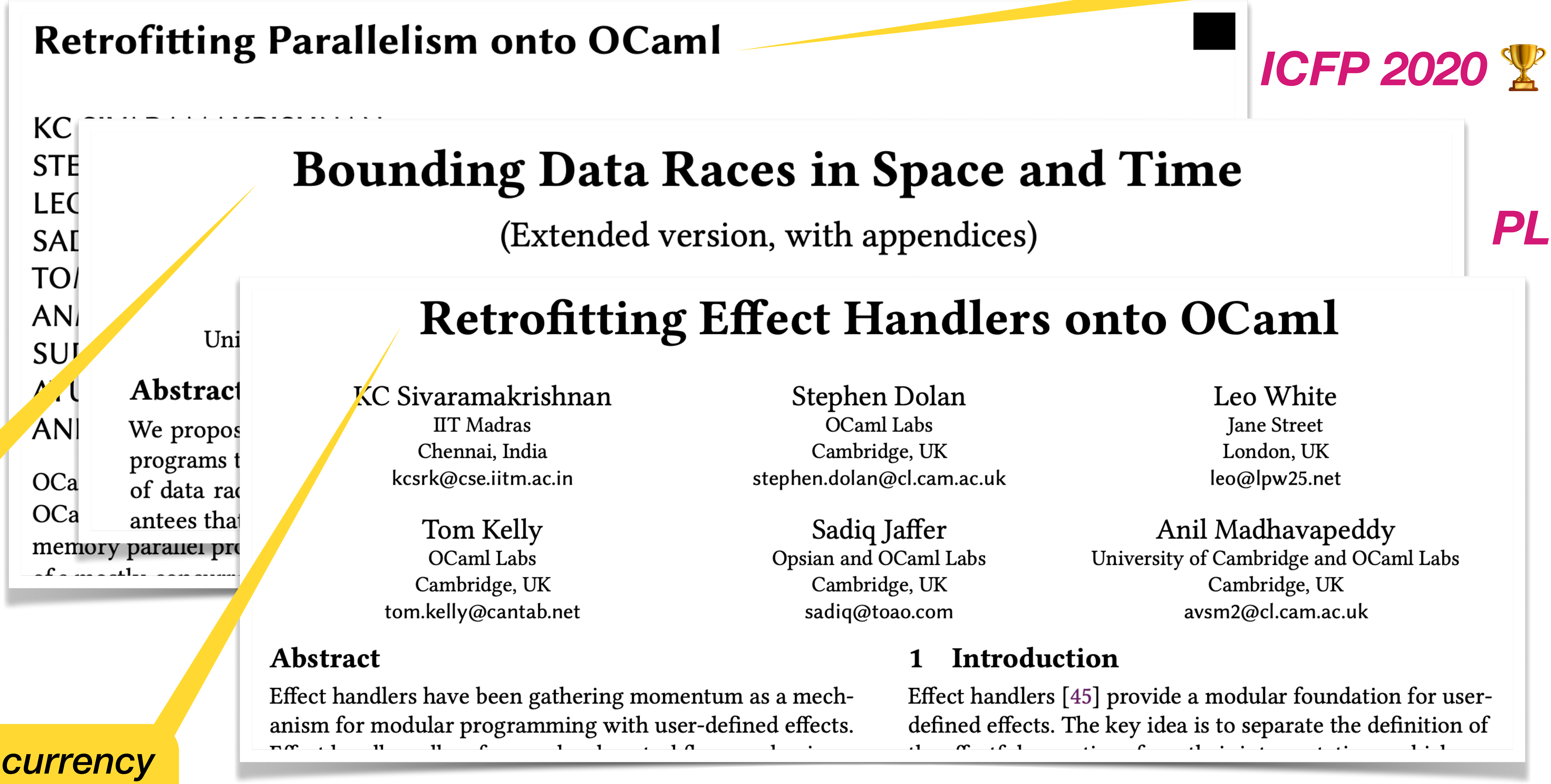
all objects reachable from it to be promoted to the shared heap en masse. Unfortunately this eagerly promotes many objects that were never really shared: just because an object is pointed to by a shared object does not mean another thread is actually going to attempt to access it.

Our design is similar but lazier, along the lines of the multicore Haskell work [2], where objects are promoted to the shared heap whenever another thread

OCaml Workshop 2014



Building confidence through papers



Retrofitting Parallelism onto OCaml

KC
STE
LEO
SAD
TO
AN
SU
AT U
AN

OCa
OCa
memory parallel pro

Bounding Data Races in Space and Time

(Extended version, with appendices)

Uni

Abstract

We propos
programs t
of data rac
antees that

Retrofitting Effect Handlers onto OCaml

KC Sivaramakrishnan
IIT Madras
Chennai, India
kcsrk@cse.iitm.ac.in

Stephen Dolan
OCaml Labs
Cambridge, UK
stephen.dolan@cl.cam.ac.uk

Leo White
Jane Street
London, UK
leo@lpw25.net

Tom Kelly
OCaml Labs
Cambridge, UK
tom.kelly@cantab.net

Sadiq Jaffer
Opsian and OCaml Labs
Cambridge, UK
sadiq@toao.com

Anil Madhavapeddy
University of Cambridge and OCaml Labs
Cambridge, UK
avsm2@cl.cam.ac.uk

Abstract

Effect handlers have been gathering momentum as a mechanism for modular programming with user-defined effects.

1 Introduction

Effect handlers [45] provide a modular foundation for user-defined effects. The key idea is to separate the definition of

Multicore GC and runtime system

ICFP 2020 🏆

PLDI 2018

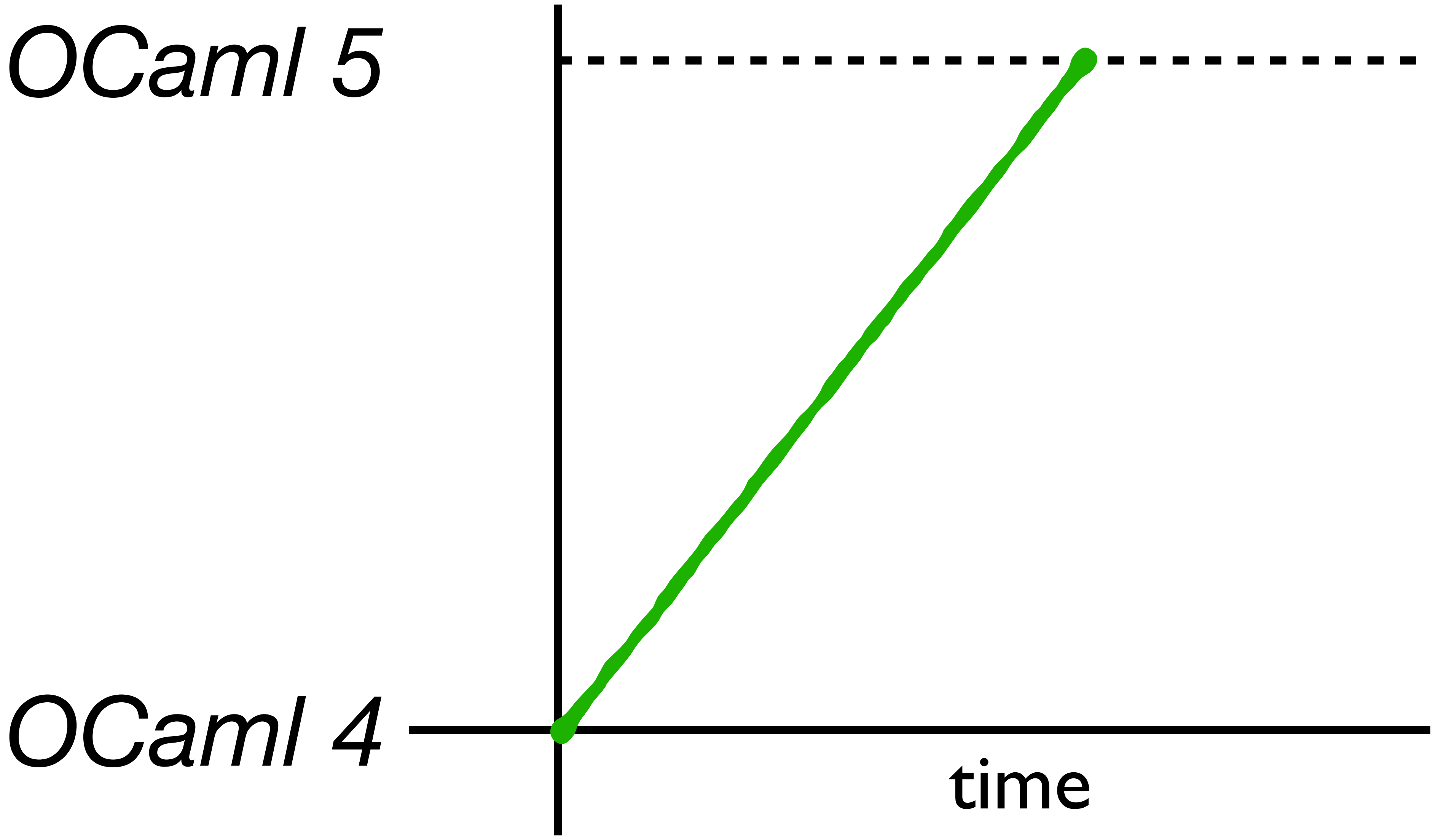
Relaxed Memory Model

PLDI 2021

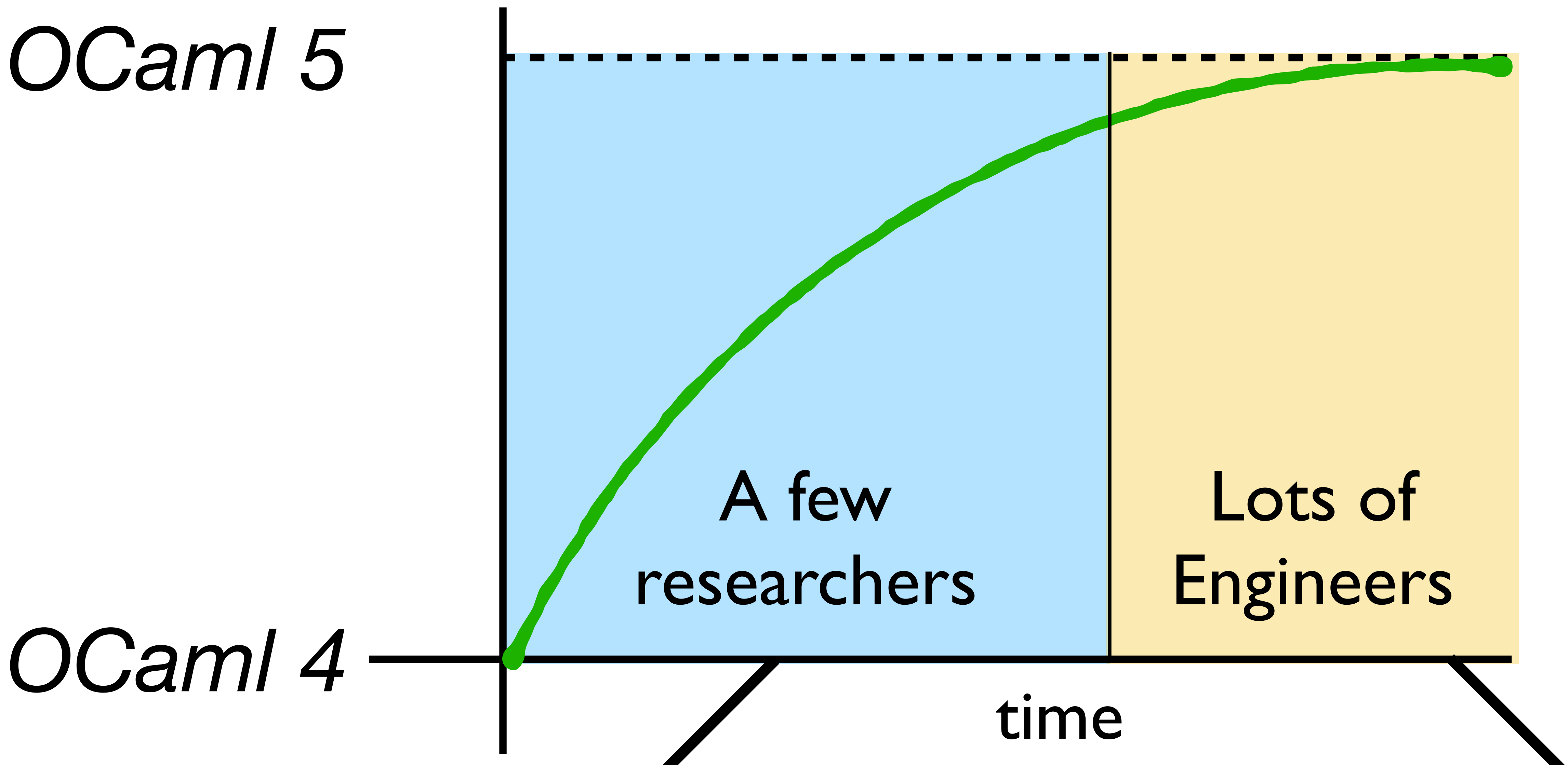
Concurrency story

Peer-reviewed ideas build confidence

Growing the language



Growing the language



UNIVERSITY OF CAMBRIDGE

Jane Street

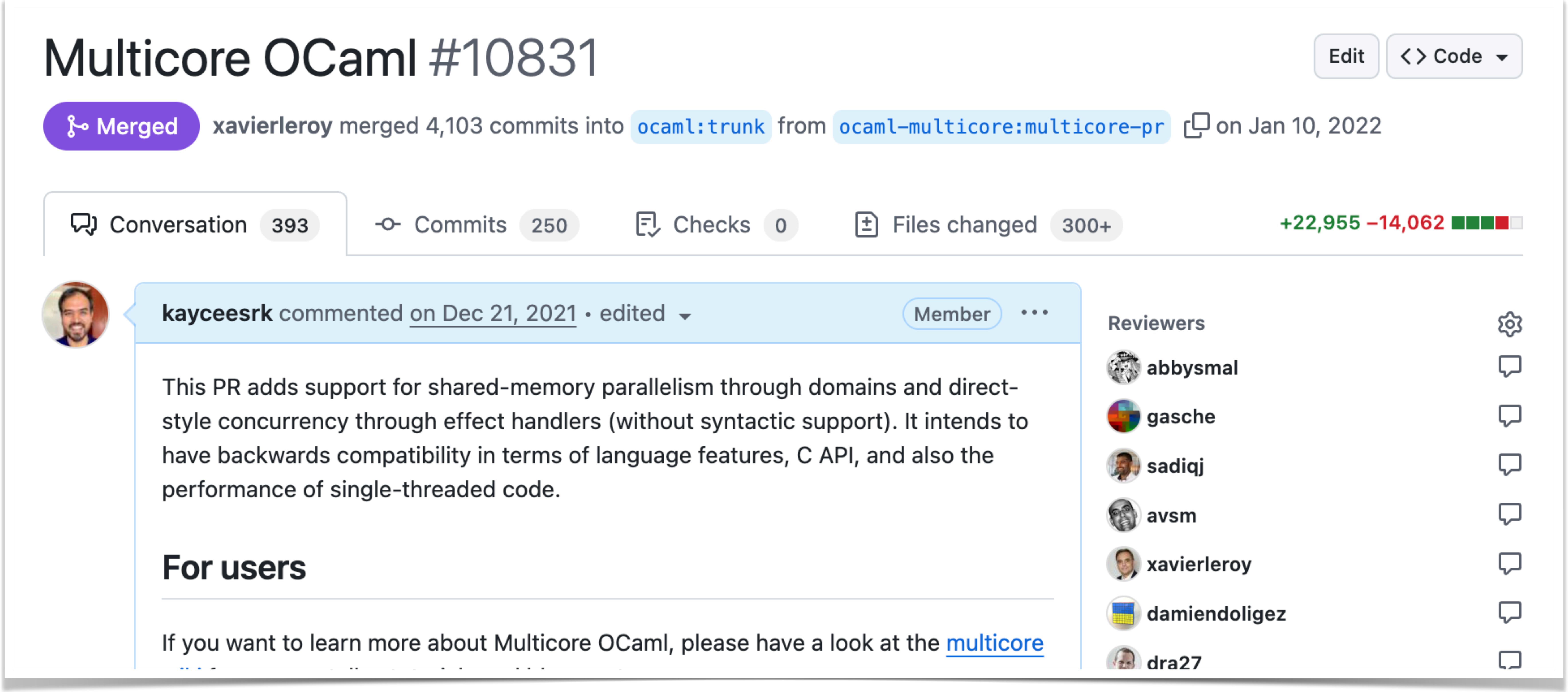
IIT MADRAS
Indian Institute of Technology Madras

OCaml

Tarides

Independent Contributors

Upstream and Release



Multicore OCaml #10831

Merged xavierleroy merged 4,103 commits into `ocaml:trunk` from `ocaml-multicore:multicore-pr` on Jan 10, 2022

kayceesrk commented on Dec 21, 2021 • edited

This PR adds support for shared-memory parallelism through domains and direct-style concurrency through effect handlers (without syntactic support). It intends to have backwards compatibility in terms of language features, C API, and also the performance of single-threaded code.

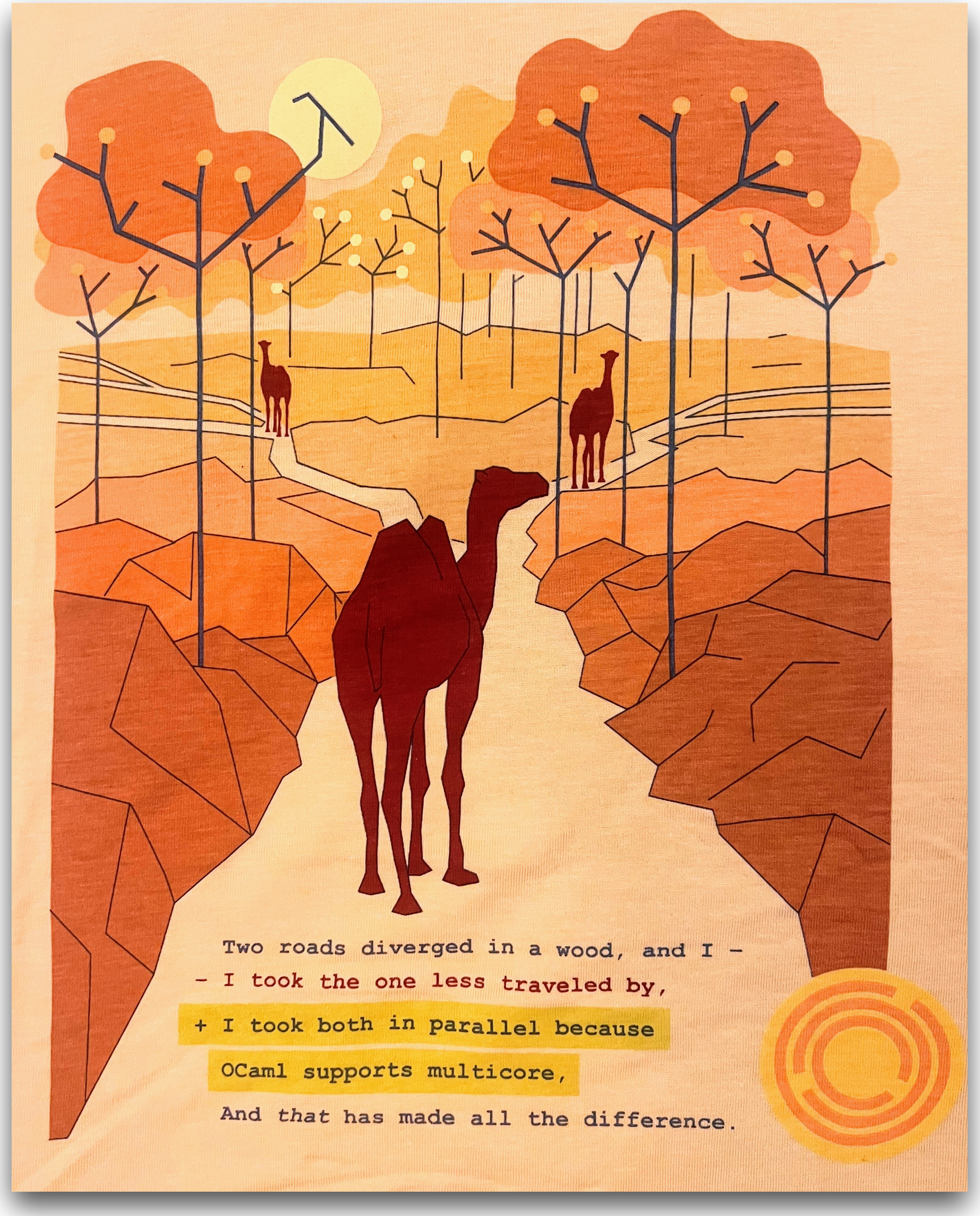
For users

If you want to learn more about Multicore OCaml, please have a look at the multicore

Started — Mar 2014, **Merged** — Jan 2022

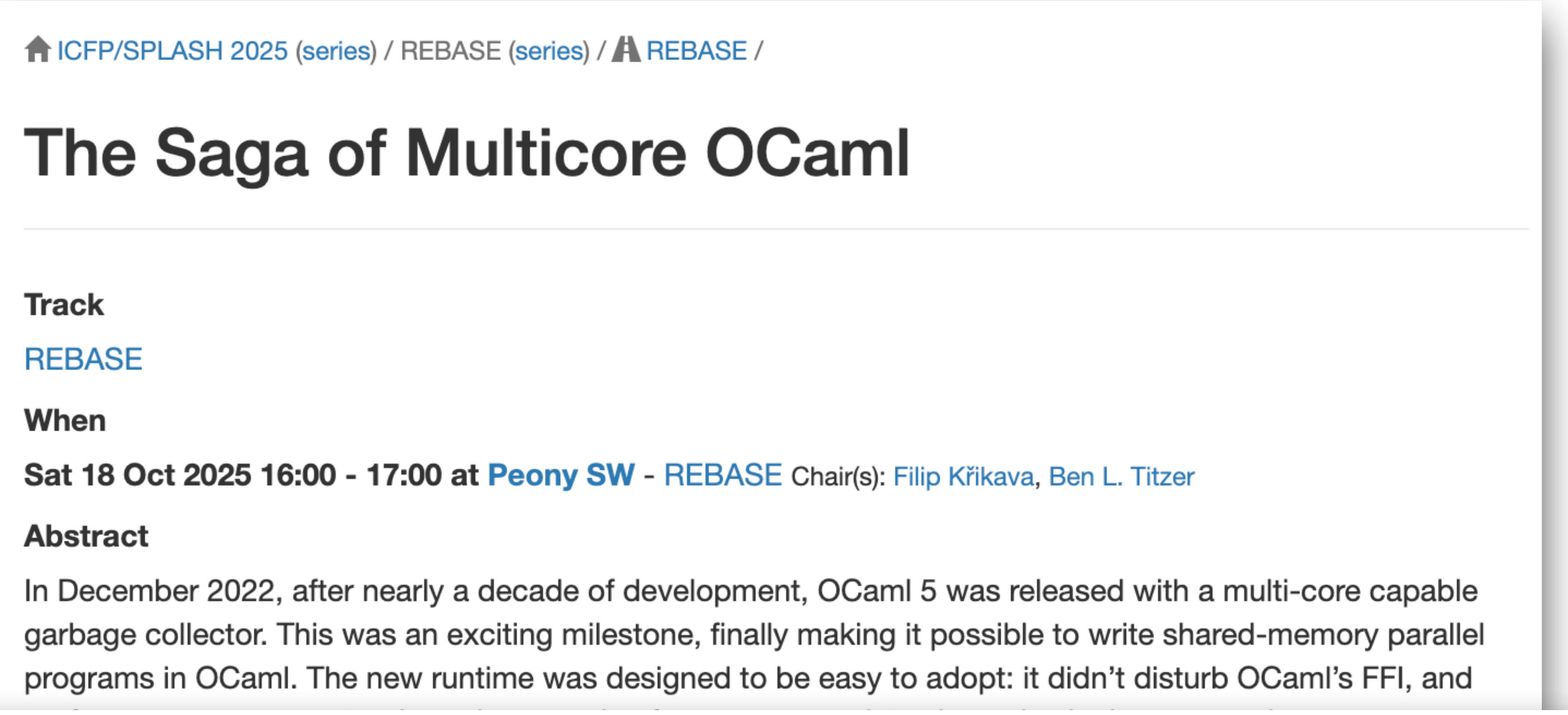
Upstream and Release

- **Released —** Dec 16 2022, as OCaml 5.0
- **Long tail** of adding missing features, bug fixes and performance improvements
  - 5.1 — Sep 2023
  - 5.2 — May 2024
  - 5.3 — Jan 2025
  - 5.4 — Sep 2025


Two roads diverged in a wood, and I -
- I took the one less traveled by,
+ I took both in parallel because
OCaml supports multicore,
And *that* has made all the difference.

Adoption

- Several ***severe performance regressions*** were observed by industrial users
- Despite our efforts around
  - ***Rigorous***, ***continuous*** benchmarking on ***real-world programs***
  - sandmark.tarides.com — Benchmark suite, Infra and runners
- Missing gap
  - Open-source workloads do not fully characterise production workloads
- Jane Street, SemGrep are running OCaml 5 in production! 🎉



ICFP/SPLASH 2025 (series) / REBASE (series) / REBASE /

The Saga of Multicore OCaml

Track

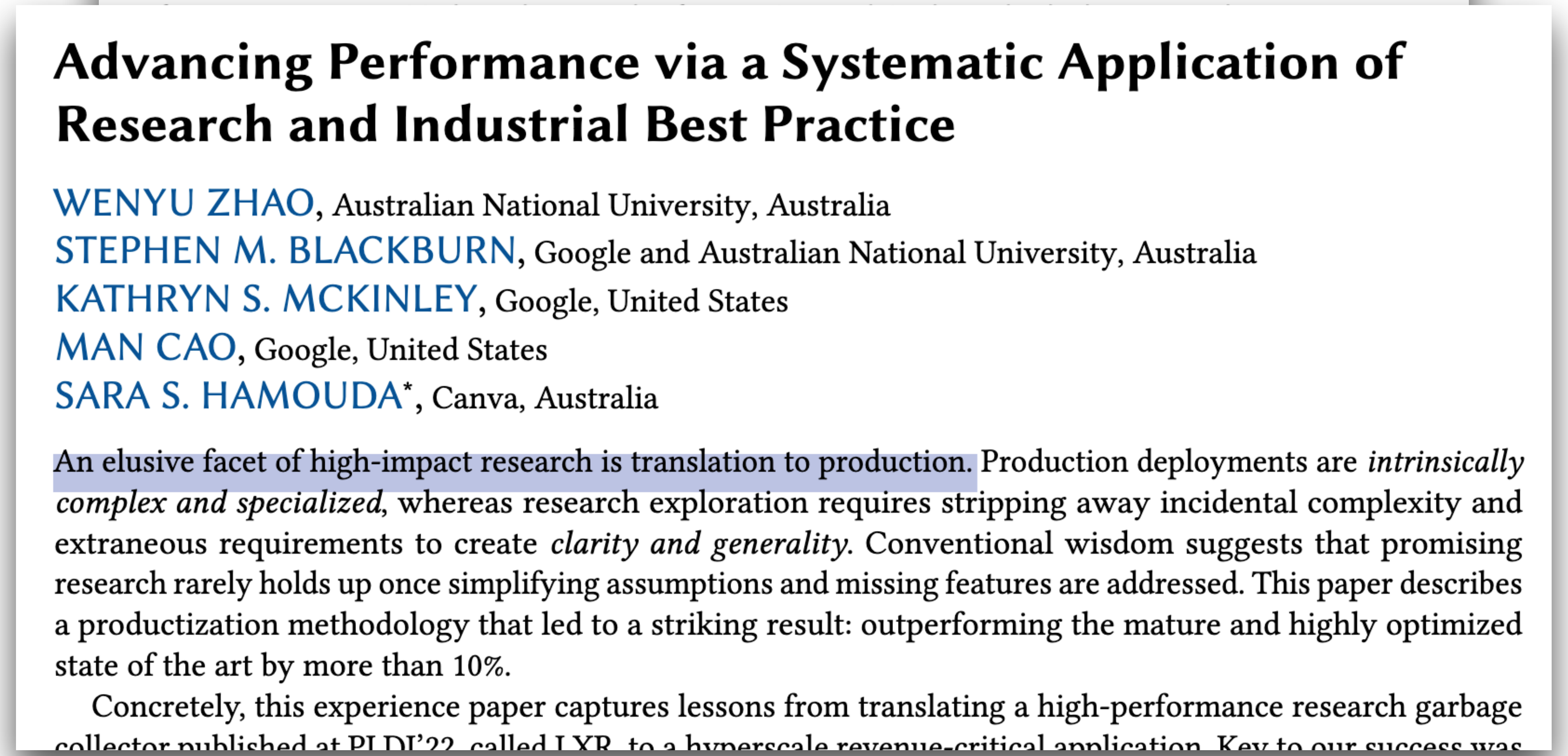
REBASE

When

Sat 18 Oct 2025 16:00 - 17:00 at Peony SW - REBASE Chair(s): Filip Křikava, Ben L. Titzer

Abstract

In December 2022, after nearly a decade of development, OCaml 5 was released with a multi-core capable garbage collector. This was an exciting milestone, finally making it possible to write shared-memory parallel programs in OCaml. The new runtime was designed to be easy to adopt: it didn't disturb OCaml's FFI, and



Advancing Performance via a Systematic Application of Research and Industrial Best Practice

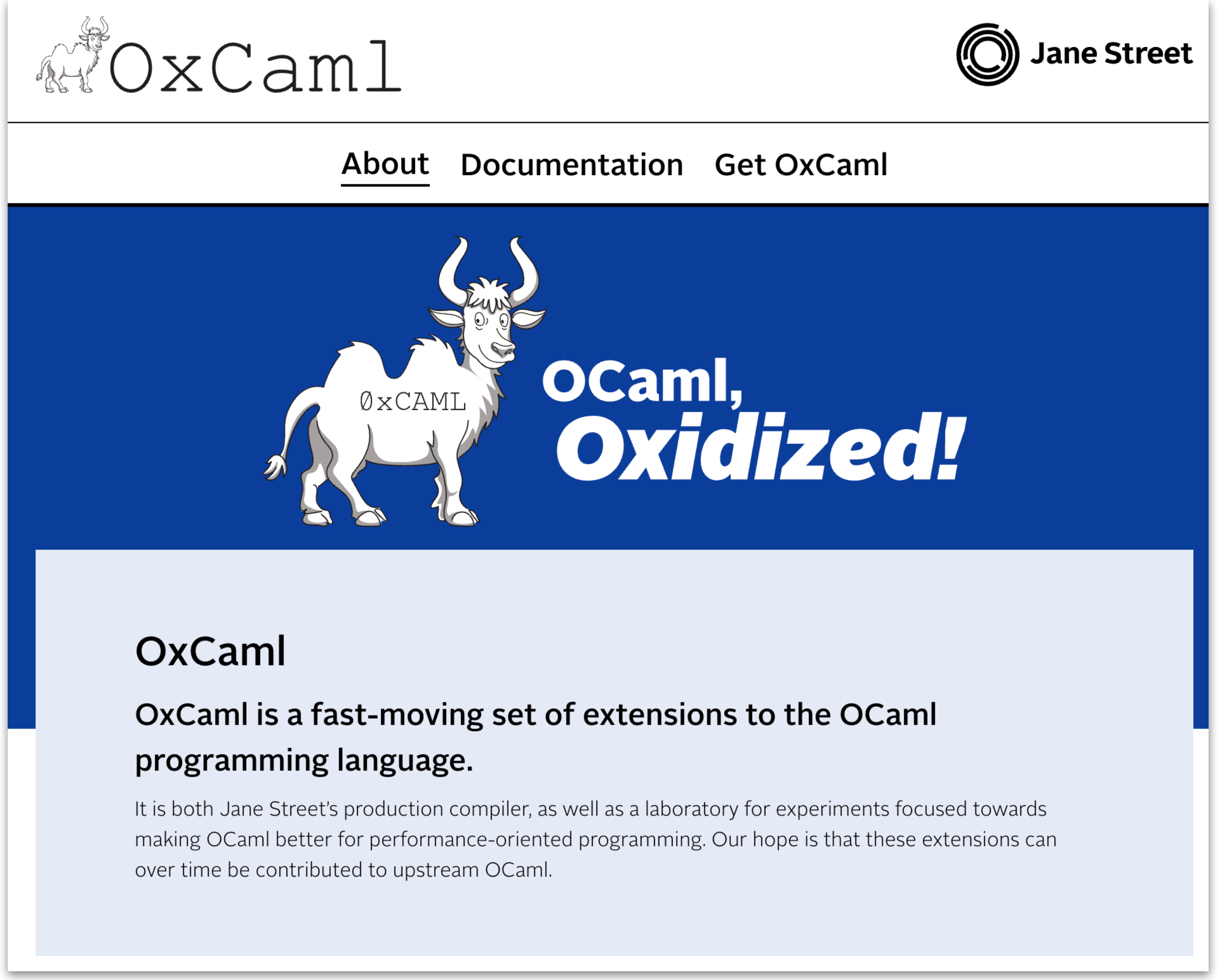
WENYU ZHAO, Australian National University, Australia
STEPHEN M. BLACKBURN, Google and Australian National University, Australia
KATHRYN S. MCKINLEY, Google, United States
MAN CAO, Google, United States
SARA S. HAMOUDA*, Canva, Australia

An elusive facet of high-impact research is translation to production. Production deployments are *intrinsically complex and specialized*, whereas research exploration requires stripping away incidental complexity and extraneous requirements to create *clarity and generality*. Conventional wisdom suggests that promising research rarely holds up once simplifying assumptions and missing features are addressed. This paper describes a productization methodology that led to a striking result: outperforming the mature and highly optimized state of the art by more than 10%.

Concretely, this experience paper captures lessons from translating a high-performance research garbage collector published at PLDI'22, called LXR, to a hyperscale revenue-critical application. Key to our success was

What’s next for OCaml?

- **OxCaml** — Bridging the performance and safety gap between OCaml and Rust
  - ***Data-race-free parallelism*** through ***modes***
  - Better control over object layout, allocations and GC
- Draws lessons from Multicore OCaml execution
  - Several award-winning papers at POPL, ICFP, OOPSLA
- But different in other ways…
  - In production at Jane Street
  - Valuable user-feedback-oriented design



OxCaml

OxCaml is a fast-moving set of extensions to the OCaml programming language.

It is both Jane Street’s production compiler, as well as a laboratory for experiments focused towards making OCaml better for performance-oriented programming. Our hope is that these extensions can over time be contributed to upstream OCaml.

https://oxcaml.org

📣 FP Launchpad @ IIT Madras 🚀

- **A Centre for Functional Systems Research and Education**
- Mission — Robust, high-performance systems using O(x)Caml
  - **PL & compilers:** language design, semantics, optimisation
  - **Pragmatic verification:** Type systems, deductive verification, testing, model checking
  - **Runtime systems:** GC, performance tooling, parallelism
  - **Hardware-software co-design:** Correct, secure, fast systems end-to-end
- Opportunities
  - Post-bacc fellowships, MS/PhD positions, Research Staff

If interested, talk to me 🙏