David Leopoldseder, Dr.

Researcher | Production Compiler Engineer | Architect

Vienna, Austria | +43 676 926 8779 | contact@davidleopoldseder.com | Website

GitHub | LinkedIn | Google Scholar | ORCID

PROFILE

Researcher, production compiler engineer and architect with more than ten years' experience in the JVM compiler ecosystem, focused on GraalVM. Principal architect and primary implementer of multiple major Graal and Native Image optimisations across public and substantial proprietary Oracle code. Owns the compiler lifecycle from opportunity identification and IR/algorithm design through implementation, correctness, benchmarking, profiling, generated-code analysis, product integration and long-term maintenance. Current work includes Ristretto, the Native Image JIT for Crema, which remains under active development.

CORE EXPERTISE

Architecture: Sea-of-Nodes and SSA IR | compiler phases and profitability | JIT and AOT | runtime/compiler integration | lowering | deoptimisation | on-stack replacement

Optimisation and performance: graph and loop transformations | control-flow and profile analysis | benchmarking and profiling | generated-code and IR analysis | regression and root-cause diagnosis

Production engineering: correctness and security hardening | deployment and maintenance | AMD64 and AArch64 | `gdb` | `perf`

EXPERIENCE

Selected highlights from a broader compiler-engineering portfolio, not an exhaustive account.

Oracle

Austria

Consulting Member of Technical Staff

February 2024-present

Principal Researcher, 2020-January 2024 | Senior Researcher, 2019-2020 | Member of Technical Staff, 2016-2017

COMPILER OPTIMISATION, NATIVE IMAGE AND PRODUCTION DELIVERY

- **Architected and implemented proprietary production graph and loop optimisations in Graal**, including simulation-based code duplication, loop inversion and rotation, partial loop unrolling and counted-loop strip mining; owned algorithm and IR design, applicability and profitability, phase integration, correctness, benchmarking and product integration. Related research
- **Designed and implemented Graal control-flow-graph construction and profile-propagation algorithms**, including inverted-loop support for counted-loop detection that exposes additional control-flow shapes to downstream loop optimisations. Public implementation
- **Introduced Native Image's `-O3` mode and made major contributions across `-O2`, `-O3` and `-Os`.** In internal evaluations, `-O3` with profile-guided optimisation matched Graal JIT peak performance within measurement noise; suite-specific runtimes or scores were aggregated independently for Java DaCapo, Scala DaCapo, Renaissance and SPECjvm2008. Public `-O3` implementation
- **Designed, implemented and shipped Native Image support for Java FFM shared arenas (`Arena.ofShared()`)**, combining safepoint-synchronised close semantics with dominance and loop analysis that inserts minimal lazy session-validity checks across scoped native-memory accesses; hardened the implementation for image layers and Vector API signatures. Implementation
- **Implemented target-specific lowering and LIR/code-generation fixes for AMD64 and AArch64**. Extensive generated-code analysis using `gdb` and `perf`, with deep x86-64 microarchitectural and cache-performance expertise. Public examples

RISTRETTO - ACTIVE DEVELOPMENT

- **Principal architect and main author of Ristretto**, the actively developed Native Image JIT for Java classes loaded at run time through Crema; designed the compiler/runtime boundary and runtime bytecode-to-Graal-IR path. Initial architecture
- **Owns the system end to end** across metadata and profiling, priority scheduling, machine-code installation, copied and runtime-adapted Graal/Native Image optimisation suites, speculation, deoptimisation, on-stack replacement, hierarchy assumptions, inlining, diagnostics, watchdogs and JCK-driven correctness hardening. Subsystem evidence

EXPERIENCE (CONTINUED)

Oracle Austria

PRODUCTION QUALITY AND TECHNICAL LEADERSHIP

- **Personally fixed and closed an average of 100+ defects per year over at least eight years** across Graal, GraalVM, Native Image and SubstrateVM, including an average of 15+ CVSS-assessed security issues annually; reviewed and coordinated additional fixes.
- **Provided technical guidance** to four interns, two doctoral researchers and at least two engineers; conducted 75+ technical interviews over eight years and submitted a hiring recommendation after each interview.

RESEARCH AND TEACHING

Johannes Kepler University Linz Institute for System Software

External Lecturer and Academic Collaborator 2017-2025

Researcher / Doctoral Candidate, 2017-2019 | Student Researcher, 2014-2016

- **Researched and implemented aggressive Graal JIT optimisations**, including dominance-based duplication simulation and fast-path loop unrolling, and collaborated on benchmarking, compiler diagnostics and runtime optimisation. DBDS paper | Fast-path unrolling paper
- **Published research evaluation of simulation-based duplication** reported a 5.89% mean gain in peak performance, with gains up to 40%, while measuring compilation-time and code-size costs; fast-path unrolling reduced run time by up to 35% on targeted microbenchmarks. These are published research results, not performance claims for proprietary production implementations.
- **Taught compiler construction and dynamic compilation/runtime optimisation** and advised student work in compilers, managed runtimes and software analysis.
- **At JKU, co-supervised one completed doctorate and advised or co-advised eight completed master's and bachelor's theses** in compiler and runtime systems. Official thesis records

EDUCATION

Doctorate (Dr. techn.), Computer Science - Software Engineering 2019

Johannes Kepler University Linz | Passed with distinction | *Simulation-Based Code Duplication in a Dynamic Compiler*

MSc, Computer Science - Software Engineering 2016

Johannes Kepler University Linz | Passed with distinction | *Graal AOT JS: A Java Bytecode-to-JavaScript Compiler*

BSc, Computer Science 2014

Johannes Kepler University Linz

PATENTS & INVENTIONS

- *Simulation-based code duplication* - US 10,338,903 B2, granted 2019.
- *Online machine learning based compilation* - US 11,392,356 B1, granted 2022.
- *Compiler generation for partial evaluation* - US 2023/0071160 A1, published application, 2023.

SELECTED PUBLIC ENGINEERING EVIDENCE

Graal IR and loop analysis: StructuredGraph | CFG and profile propagation | counted loops

Native Image: `-O3` introduction | shared arenas | `-Os` implementation

Back end and generated code: AMD64 lowering | AMD64/AArch64 code paths

Ristretto: initial architecture | runtime graph builder | optimisation pipeline

RECOGNITION AND PROFESSIONAL SERVICE

JKU Young Researchers' Award (2021) | First place, Graduate Category, ACM Student Research Competition at SPLASH 2017 | CGO 2018 Best Paper Finalist

Programme and artifact-evaluation service for VMIL, ICOOOLPS, ECOOP, the Graal Workshop and VMM; Renaissance benchmark-suite committee; TACO sub-reviewer; reviewer for the Journal of Computer Languages.

COMPLETE PEER-REVIEWED PUBLICATION RECORD

Google Scholar: 648 citations | h-index 11 (13 August 2026).

2026

Partial-Evaluation Templates: Accelerating Partial Evaluation with Pre-compiled Templates. **CGO 2026, pp. 109-122.**
Florian Huemer; Aleksandar Prokopec; **David Leopoldseder**; Raphael Mosaner; Hanspeter Mössenböck

2025

Automated Profile-Guided Replacement of Data Structures to Reduce Memory Allocation. **The Art, Science, and Engineering of Programming 10(1), Article 3.**
Lukas Makor; Sebastian Kloibhofer; Peter Hofer; **David Leopoldseder**; Hanspeter Mössenböck

Profile-Guided Field Externalization in an Ahead-of-Time Compiler. **ECOOP 2025, 19:1-19:32.**
Sebastian Kloibhofer; Lukas Makor; Peter Hofer; **David Leopoldseder**; Hanspeter Mössenböck

Reusing Highly Optimized IR in Dynamic Compilation. **ECOOP 2025, 25:1-25:25.**
Andrej Pecimuth; **David Leopoldseder**; Petr Tůma

A Pragmatic Approach to Replay Compilation (Extended Abstract). **Programming 2025, 3:1-3:4.**
Andrej Pecimuth; **David Leopoldseder**; Petr Tůma

2024

Taking a Closer Look: An Outlier-Driven Approach to Compilation-Time Optimization. **ECOOP 2024, 20:1-20:28.**
Florian Huemer; **David Leopoldseder**; Aleksandar Prokopec; Raphael Mosaner; Hanspeter Mössenböck

An Analysis of Compiled Code Reusability in Dynamic Compilation. **VMIL 2024, pp. 43-53.**
Andrej Pecimuth; **David Leopoldseder**; Petr Tůma

2023

Compilation Forking: A Fast and Flexible Way of Generating Data for Compiler-Internal Machine Learning Tasks. **The Art, Science, and Engineering of Programming 7(1), Article 3.**
Raphael Mosaner; **David Leopoldseder**; Wolfgang Kisling; Lukas Stadler; Hanspeter Mössenböck

Control Flow Duplication for Columnar Arrays in a Dynamic Compiler. **The Art, Science, and Engineering of Programming 7(3), Article 9.**
Sebastian Kloibhofer; Lukas Makor; **David Leopoldseder**; Daniele Bonetta; Lukas Stadler; Hanspeter Mössenböck

Diagnosing Compiler Performance by Comparing Optimization Decisions. **MPLR 2023, pp. 47-61.**
Andrej Pecimuth; **David Leopoldseder**; Petr Tůma

GraalVM Compiler Benchmark Results Dataset (Data Artifact). **ICPE Companion 2023, pp. 65-69.**
Lubomír Bulej; Vojtěch Horký; Michele Tucci; Petr Tůma; François Farquet; **David Leopoldseder**; Aleksandar Prokopec

2022

Improving Vectorization Heuristics in a Dynamic Compiler with Machine Learning Models. **VMIL 2022, pp. 36-47.**
Raphael Mosaner; Gergö Barany; **David Leopoldseder**; Hanspeter Mössenböck

Automatic Array Transformation to Columnar Storage at Run Time. **MPLR 2022, pp. 16-28.**
Lukas Makor; Sebastian Kloibhofer; **David Leopoldseder**; Daniele Bonetta; Lukas Stadler; Hanspeter Mössenböck

Dynamic Taint Analysis with Label-Defined Semantics. **MPLR 2022, pp. 64-84.**
Jacob Kreindl; Daniele Bonetta; Lukas Stadler; **David Leopoldseder**; Hanspeter Mössenböck

Machine-Learning-Based Self-Optimizing Compiler Heuristics. **MPLR 2022, pp. 98-111.**
Raphael Mosaner; **David Leopoldseder**; Wolfgang Kisling; Lukas Stadler; Hanspeter Mössenböck

COMPLETE PUBLICATION RECORD (CONTINUED)

2021

CompGen: Generation of Fast JIT Compilers in a Multi-Language VM. DLS 2021, pp. 35-47.
Florian Latifi; **David Leopoldseder**; Christian Wimmer; Hanspeter Mössenböck

Low-Overhead Multi-Language Dynamic Taint Analysis on Managed Runtimes through Speculative Optimization. MPLR 2021, pp. 70-87.
Jacob Kreindl; Daniele Bonetta; Lukas Stadler; **David Leopoldseder**; Hanspeter Mössenböck

Using Machine Learning to Predict the Code Size Impact of Duplication Heuristics in a Dynamic Compiler. MPLR 2021, pp. 127-135.
Raphael Mosaner; **David Leopoldseder**; Lukas Stadler; Hanspeter Mössenböck

2020

Multi-Language Dynamic Taint Analysis in a Polyglot Virtual Machine. MPLR 2020, pp. 15-29.
Jacob Kreindl; Daniele Bonetta; Lukas Stadler; **David Leopoldseder**; Hanspeter Mössenböck

SymJEx: Symbolic Execution on the GraalVM. MPLR 2020, pp. 63-72.
Sebastian Kloibhofer; Thomas Pointhuber; Maximilian Heisinger; Hanspeter Mössenböck; Lukas Stadler; **David Leopoldseder**

2019

Supporting On-Stack Replacement in Unstructured Languages by Loop Reconstruction and Extraction. MPLR 2019, pp. 1-13.
Raphael Mosaner; **David Leopoldseder**; Manuel Rigger; Roland Schatz; Hanspeter Mössenböck

Renaissance: Benchmarking Suite for Parallel Applications on the JVM. PLDI 2019.
Aleksandar Prokopec; Andrea Rosà; **David Leopoldseder**; Gilles Duboscq; Petr Tůma; Martin Studener; Lubomír Bulej; Yudi Zheng; Alex Villazón; Doug Simon; Thomas Würthinger; Walter Binder

An Optimization-Driven Incremental Inline Substitution Algorithm for Just-in-Time Compilers. CGO 2019, pp. 164-179.
Aleksandar Prokopec; Gilles Duboscq; **David Leopoldseder**; Thomas Würthinger

2018

Dominance-Based Duplication Simulation (DBDS): Code Duplication to Enable Compiler Optimizations. CGO 2018, pp. 126-137. Best Paper Finalist.
David Leopoldseder; Lukas Stadler; Thomas Würthinger; Josef Eisl; Doug Simon; Hanspeter Mössenböck

A Cost Model for a Graph-Based Intermediate Representation in a Dynamic Compiler. VMIL 2018, pp. 26-35.
David Leopoldseder; Lukas Stadler; Manuel Rigger; Thomas Würthinger; Hanspeter Mössenböck

Fast-Path Loop Unrolling of Non-Counted Loops to Enable Subsequent Compiler Optimizations. ManLang 2018, Article 2.
David Leopoldseder; Roland Schatz; Lukas Stadler; Manuel Rigger; Thomas Würthinger; Hanspeter Mössenböck

Parallel Trace Register Allocation. ManLang 2018, Article 7.
Josef Eisl; **David Leopoldseder**; Hanspeter Mössenböck

An Analysis of x86-64 Inline Assembly in C Programs. VEE 2018, pp. 84-99.
Manuel Rigger; Stefan Marr; Stephen Kell; **David Leopoldseder**; Hanspeter Mössenböck

2017

Simulation-Based Code Duplication for Enhancing Compiler Optimizations. SPLASH Companion 2017, pp. 10-12.
David Leopoldseder

Making Collection Operations Optimal with Aggressive JIT Compilation. SCALA at SPLASH 2017, pp. 29-40.
Aleksandar Prokopec; **David Leopoldseder**; Gilles Duboscq; Thomas Würthinger

2015

Java-to-JavaScript Translation via Structured Control Flow Reconstruction of Compiler IR. DLS 2015, pp. 91-103.
David Leopoldseder; Lukas Stadler; Christian Wimmer; Hanspeter Mössenböck