Eugene Burmako
PhD, Computer Science
+1-631-464-1469
eugene.burmako@gmail.com
San Francisco, CA
US Permanent Resident

EXPERTISE

Compilers,
Programming languages,
Developer tools.

HIGHLIGHTS

Cofounded and championed Swift MLIR, a toolchain that compiles ML programs written in Swift into MLIR. Building on top of frontend metaprogramming via Swift quasiquotes and backend metaprogramming via Swift Intermediate Language, we delivered two prototypes that can compile small computational kernels and machine learning models written in Swift to a combination of standard and XLA HLO dialects of MLIR.

Created Rsc, an experimental Scala compiler focused on compilation speed. Through outlining - really quick computation of type signatures - we've been able to extract additional parallelism from Scala builds and obtain significant compilation speedups. On Twitter Util, one of the foundational libraries of the Twitter stack, our compilation pipeline is ∼2.7 times faster than the traditional compilation pipeline on powerful hardware.

Cofounded and spearheaded Scalameta, an opensource toolchain that raises the bar of what is possible for effective Scala development environment. Together with Denys Shabalin and Ólafur Páll Geirsson, we turned Scalameta into a strategically important community initiative, which nowadays encompasses more than 10 projects that add up to more than 13000 commits made by more than 200 contributors.

Designed and implemented reflection and macros for Scala together with Martin Odersky, enabling a number of new patterns used in popular community libraries: Akka, Play, Sbt, Scala.js, ScalaTest, Shapeless, Slick, Spark, Spire and others.

Bootstrapped the Scala macros community from zero to self-sustaining, gave more than 20 talks about metaprogramming, managed more than 10 successful student projects, supervised the work of more than 40 contributors, provided help with more than 30 opensource projects.

Contributed ∼750 commits, ∼750 tests and ∼150k lines of code to the Scala compiler, fixed ∼350 issues, being the 2nd top contributor to the Scala 2.10 release and the 3rd top contributor to the Scala 2.11 release.

PROJECTS

Swift MLIR. Having learned about MLIR, I've been looking for opportunities to integrate it into Swift, and in May 2019, I found a plausible plan of attack and proceeded to work on what has become two distinct prototypes - one based on Swift quasiquotes and another one based on Swift Intermediate Language.

- Proposed a design and a roadmap for integration between Swift and MLIR, broke it down into bite-sized work items, evolved as necessary to adjust for intermediate research findings and changes in the MLIR landscape.
- Together with Alex Suhan, worked on the first prototype of Swift MLIR, splitting the work between frontend (mostly myself) and backend (mostly Alex).
- Explored lightweight modular staging, implemented virtualized if and for, inspired by Scala virtualized, in the Swift compiler.
- Implemented quasiquotes for the majority of Swift expressions (∼2kloc compiler patch, ∼3kloc libQuote library, ∼3kloc libQuote tests).
- Together with Adam Paszke, kickstarted libSIL - a Swift-based data model and parser for Swift Intermediate Language.
- Implemented the SIL dialect of MLIR and relevant components of a Swift compilation pipeline that enable ingesting the intermediate representation of the Swift compiler into the MLIR ecosystem.
- Designed a prototype programming model and corresponding MLIR compiler passes for the second prototype to compile small computational kernels and machine learning models written in Swift to a combination of standard and XLA HLO dialects of MLIR.

Swift for TensorFlow. In Feb 2019, I joined the Swift for TensorFlow team at Google Brain to help build a next-generation platform for machine learning founded by Chris Lattner et al. During my time at the Swift for TensorFlow team, I was mostly focused on Swift MLIR, but also made a few smaller contributions to Swift tooling and internal use cases for our product.

- Documented the state of internal tooling landscape for Swift for TensorFlow, met with partner teams, proposed a roadmap for future work.
- Investigated compilation performance of Swift for TensorFlow targets in the Google monorepo, identified and documented areas of improvement.
- Performed the initial investigation into SourceKit, implemented an MVP of resolving cross-target references in the Google monorepo.
- Figured out and documented how to enable Python interop and TPU support in Swift for TensorFlow targets in the Google monorepo.

Rsc. In Aug 2017, I started developing Rsc, an experimental Scala compiler focused on compilation speed. Shortly after its inception, the project got internal funding at Twitter. At the moment, Rsc is focused on outlining - really quick computation of type signatures for an almost full subset of Scala. Once outlines are computed, it becomes possible to compile all files of the program in parallel, which leads to significant speedups on powerful hardware.

- Owned internal and external roadmaps of Rsc, ensuring continuous delivery on commitments in the face of significant uncertainty.
- Worked together with partners within Advanced Scala Tools and Build teams towards integration of Rsc into the Twitter toolchain and developer workflow.
- Co-designed and implemented Rsc outlining, achieving significant coverage of Twitter internal codebase.
- Co-designed and prototyped RscCompat, an automated code rewrite that transforms Scala code into a subset of Scala supported by Rsc.
- Put together Warp - a simplistic build tool specialized for experiments with Rsc-based toolchains.

Scalameta. Based on the success of scala.reflect and learning from user feedback, in Oct 2013 I set out to build a new metaprogramming platform for Scala. Together with my colleague Denys Shabalin, we've laid out the axioms: 1) user convenience comes first, 2) never lose a token or irreversibly desugar a tree, 3) always save typed ASTs into the resulting bytecode. After my colleague Ólafur Páll Geirsson joined the project, we shifted focus to developer tools which significantly increased the popularity of Scalameta.

- Co-designed and implemented lexical, syntactic and semantic models for the Scala programming language, tested them out in practice using the experience of working on the Scala compiler.
- Co-designed and prototyped other aspects of the Scalameta vision (lightweight and portable reflection API, quasiquoting and hygiene, AST interpreter, AST persistence, integration with build tools, IDEs, etc).
- Worked together with tool developers from the community to make sure that the new metaprogramming platform adequately addresses their needs.
- Evangelized and applied the research behind Scalameta in an industrial setting by designing and building next-generation developer tools at Twitter.
- Planned and managed the timeline of the project, organized opensource contributors and their contributions.

Scala is a statically-typed object-functional programming language. Since Sep 2011, I'm a committer to scalac, the official Scala compiler. During my first two years, I was a very active contributor to the compiler before moving my research into separate repositories.

- Contributed and maintained a novel language feature: def macros, along with the underlying metaprogramming API.
- Implemented Macro Paradise, a popular compiler plugin with additional macro flavors, which was clocking 20k+ monthly downloads and ultimately got merged into the official Scala compiler.
- Discovered and codified new tricks and idioms on the intersection of macros and vanilla Scala features together with prominent community members.
- Worked together with tool developers to enable support for the newly introduced language features and patterns.
- Fixed hundreds of issues in scalac, patching various subsystems of the compiler including the parser and the typechecker.

Scala Macros. After joining the PhD program at EPFL in Sep 2011, I've come up with an idea to implement macros in Scala. I realized the initial vision in the official Scala compiler under the supervision of Martin Odersky and then proceeded to research and preach new flavors of macros, gradually gaining support in the community. Very soon, macros spread like wildfire, and now they are used in many popular libraries, including major components of the Lightbend platform.

- Co-designed and implemented scala.reflect, the library that has become the foundation for compile-time and runtime metaprogramming in Scala 2.10.
- Co-designed and implemented def macros for Scala 2.10.
- Designed and realized a number of other macro flavors (dynamic macros, interpolation macros, implicit macros, fundep materializers, type macros, type providers and macro annotations).
- Promoted, documented and maintained Scala macros, communicating with the adopters and contributors from the Scala community.
- Provided consulting and support for a number of production and research users, ramping up adoption of the new technologies.

Pickling is an experimental persistence library for Scala that uses implicits and macros to derive extremely fast and customizable serializers, matching and surpassing the performance of similar Java libraries. Pickling has gained popularity and was eventually included in sbt, the most popular build tool for Scala.

- Co-designed object-oriented pickler combinators.
- Implemented and optimized the initial version of compile-time pickler generation facilities that stood the test of time for more than two years.

Conflux is an experimental parallel programming platform for C# that I developed in 2010-2011. Conflux features a domain-specific language and a high-level programming model that transparently distributes computational algorithms to GPUs both in single-machine and cluster environments:

- Designed and developed a high-level programming model in the form of an embedded DSL for C#.
- Implemented a C# frontend: a C# AST and a decompiler library.
- Implemented a CUDA backend: a code generator for the PTX assembler of NVIDIA graphics processors and a JIT compilation library.
- Implemented a CPU backend: a code generator for multicore CPUs.
- Provided integration with the Visual Studio debugger.

Relinq is a library for transforming C# expression trees into JavaScript code and vice versa. Relinq can be used to implement such scenarios as: querying LINQ data sources from JavaScript, invoking cross-process LINQ queries and constructing LINQ queries dynamically.

- Designed and developed a JavaScript-based query language.
- Extracted and documented a coherent subset of the C# 3.0 specification.
- Implemented an expression compiler for the aforementioned subset of C# that supports generics, lambdas and local type inference.

Truesight is an experimental decompiler for the bytecode of the .NET virtual machine. The library provides intermediate representation for both low-level (bytecode) and high-level (abstract syntax tree) views of code. Truesight supports basic constructs of C# and is capable of decompiling structured control flow: conditionals, loops, and restricted gotos (i.e. returns, breaks and continues).

- Implemented a parser for CIL that reifies the bytecode and preserves debug information.
- Designed and documented an intermediate tree-based representation for decompiled C# code.
- Implemented visualization for the IR: graphviz dumps, integration with the Visual Studio debugger.
- Created and implemented an algorithm for decompilation of structured programs.

Tiller is a C# software suite for designing and generating WYSIWYG interactive reports. It features a programming language that can be used to specify formulae and to script report generation process. The programming language comes with a JIT compiler that speeds up execution:

- Designed and developed a script language.
- Implemented a JIT compiler for the language.
- Implemented a thread-safe ZIP-based engine for storing report components.

WORK

Google
Mountain View, CA
Staff Software Engineer, Tech Lead / Manager
Feb 2020 - present
Activities: Our project exists at the intersection of Machine Learning and hardware, but is still under wraps so we can't share much more detail.

Google
Mountain View, CA
Staff Software Engineer
Feb 2019 - Feb 2020
Activities: Working on Swift for TensorFlow - a new way to develop machine learning models which gives the power of TensorFlow directly integrated into Swift.

Twitter
San Francisco, CA
Staff Software Engineer, Tech Lead of the Advanced Scala Tools team
Nov 2016 - Feb 2019
Activities: Worked on my opensource Scala projects including Rsc, Scalameta and others to make Scala tooling better - both internally inside the company and publicly in the Scala community.

École Polytechnique Fédérale de Lausanne
Lausanne, Switzerland
Doctoral Assistant
Sep 2011 - Nov 2016
Activities: Did a PhD at EPFL, on the academic side of the Scala programming language team. Along with the opensource work on putting my research in practice into Scala, I was also writing occasional papers and doing teaching for the faculty.

ScienceSoft
Minsk, Belarus
Software Engineer
Nov 2007 - Aug 2011
Activities: Enterprise software development with C# and a little bit of JavaScript, with integration into Microsoft Office and Microsoft SharePoint.

Itransition
Minsk, Belarus
Software Engineer
Oct 2005 - Sep 2007
Activities: Enterprise software development with C#.

Omega Software
Minsk, Belarus
Software Engineer
Oct 2004 - Sep 2005
Activities: Enterprise software development with C++.

EDUCATION

École Polytechnique Fédérale de Lausanne
Lausanne, Switzerland
Sep 2011 - Mar 2017
Degree: PhD in Computer Science
Thesis: Unification of Compile-Time and Runtime Metaprogramming in Scala

United Institute of Informatics Problems
Minsk, Belarus
Nov 2009 - Oct 2012
Degree: Researcher in Computer Science

Belarusian State University
Minsk, Belarus
Sep 2003 - Jun 2008
Degree: Specialist in Computer science

MEMBERSHIPS

- Scala Center Advisory Board (Mar 2018 - Feb 2019)
- Scala Language Server Protocol Working Group (Feb 2018 - Feb 2019)
- GPCE 2018 Program Committee (Feb 2018 - Oct 2018)
- Scala Symposium 2017 Program Committee (Apr 2017 - Oct 2017)
- GPCE 2017 Program Committee (Dec 2016 - Oct 2017)
- Scala Improvement Process Committee (Aug 2016 - Feb 2019)

TALKS

1. E. Burmako, A. Suhan "Swift as syntactic sugar for MLIR", Swift for TensorFlow meetup, San Francisco, September 9, 2019
2. E. Burmako, A. Suhan "Swift as syntactic sugar for MLIR", Google ML Languages Summit, San Francisco, August 13, 2019
3. E. Burmako "Swift for TensorFlow: A new way of doing machine learning", Google Developers ML Summit, Tokyo, July 11, 2019
4. E. Burmako "Towards Parallelizing Scala Compilations", Scale by the Bay, San Francisco, November 15, 2018
5. E. Burmako "SemanticDB: a Common Data Model for Scala Developer Tools", Workshop on Meta-Programming Techniques and Reflection, Boston, November 5, 2018
6. E. Burmako "SemanticDB: a Common Data Model for Scala Developer Tools", Scala Symposium, September 28, 2018
7. E. Burmako "How We Built Tools That Scale to Millions of Lines of Code", Scala Days, New York City, June 20, 2018
8. E. Burmako "Reasonable Scala Compiler", Scale by the Bay, San Francisco, November 16, 2017
9. E. Burmako "Concrete Next Steps for Scala Macros", SF Scala, San Francisco, June 7, 2017
10. E. Burmako, S. Hood "Semantic Tooling at Twitter", Scala Days, Copenhagen, June 1, 2017
11. E. Burmako, S. Hood "Semantic Tooling at Twitter", Scala Days, Chicago, Apr 21, 2017
12. E. Burmako "Scala.Meta Semantic API", ScalaSphere DevTools Summit, Kraków, Mar 2-3, 2017
13. E. Burmako "A New Macro System for Scala", Scala eXchange, London, Dec 8, 2016
14. E. Burmako "Scala.Meta: the Past, the Present and the Future", Scala by the Bay, San Francisco, Nov 12, 2016
15. E. Burmako "To Macros and Beyond!: How macros changed Scala, and what's coming next", Curry On, Rome, Jul 19, 2016

16. E. Burmako “Metaprogramming 2.0: No, Macros Are Not Going Away”, Scala Days, Berlin, Jun 17, 2016
17. E. Burmako “Metaprogramming 2.0”, Scala Days, New York City, May 11, 2016
18. E. Burmako “What Did We Learn In Scala Meta?”, ScalaSphere DevTools Summit, Kraków, Feb 11, 2016
19. E. Burmako “State of the Meta, Fall 2015”, fby.by, Minsk, Nov 28, 2015
20. E. Burmako “Hands on Scala.Meta”, Scala World, Lake District, Sep 21, 2015
21. E. Burmako “State of the Meta, Summer 2015”, Scala Days, Amsterdam, Jun 09, 2015
22. E. Burmako “State of the Meta, Spring 2015”, Scala Days, San Francisco, Mar 17, 2015
23. E. Burmako “State of the Meta, Fall 2014”, f(by) - Conference on Functional Programming, Minsk, Nov 22, 2014
24. E. Burmako, D. Shabalin “Easy Metaprogramming For Everyone!”, Scala Days, Berlin, Jun 16-18, 2014
25. E. Burmako, D. Shabalin “Macrology 201”, flatMap, Oslo, May 12-13, 2014
26. E. Burmako, T. Brown “Macro-Based Type Providers in Scala”, Scalar, Warsaw, Apr 5, 2014
27. E. Burmako “Rethinking Scala Macros”, Northeast Scala Symposium, New York, March 1-2, 2014
28. E. Burmako, L. Hupel “Macros vs Types”, NE Scala, New York, March 1-2, 2014
29. E. Burmako “What Are Macros Good For?”, Scala eXchange, London, December 2-3, 2013
30. E. Burmako, D. Shabalin “Scala Macros”, Google PhD Student Summit on Compiler & Programming Technology, Munich, November 25, 2013
31. E. Burmako “Philosophy of Scala Macros”, Strange Loop, St. Louis, September 18-19, 2013
32. E. Burmako “What Are Macros Good For?”, Scalapeño, Tel Aviv, July 17, 2013
33. E. Burmako “Applied Materialization”, Bay Area Scala Enthusiasts, San Francisco, June 13, 2013
34. E. Burmako “Half a Year in Macro Paradise”, Scala Days, New York City, June 10-12, 2013
35. E. Burmako “Macros in Scala”, Codefest, Novosibirsk, March 30-31, 2013
36. E. Burmako “Metaprogramming in Scala 2.10”, Minsk, April 28, 2012
37. E. Burmako, J. C. Vogt “Scala Macros”, Scala Days, London, April 17-18, 2012
38. E. Burmako “Project Kepler: Compile-Time Metaprogramming for Scala”, Lausanne, September 27, 2011
39. E. Burmako “Conflux: GPGPU for .NET”, 1st International Conference, Application Developer Days, Yaroslavl, 2010
40. A. Varanovich, V. Tulchinsky, E. Burmako, V. Falfushinsky, R. Sadykhov “Automated Software Development in Heterogeneous GPU/CPU Environments for Seismic Modeling”, EAGE, Barcelona, 2010

PAPERS

1. E. Burmako, M. Odersky “Unification of Compile-Time and Runtime Metaprogramming in Scala”, PhD thesis, Lausanne, March, 2017
2. V. Ureche, E. Burmako, M. Odersky “Late Data Layout: Unifying Data Representation Transformations”, Object Oriented Programming Systems Languages & Applications, Portland, October 19-21 2014
3. A. Biboudis, E. Burmako “MorphScala: Safe Class Morphing with Macros”, Scala Workshop, Uppsala, Jul 28-29, 2014

4. H. Miller, P. Haller, E. Burmako, M. Odersky. “Instant Pickles: Generating Object-Oriented Pickler Combinators for Fast and Extensible Serialization”, Object-Oriented Programming, Systems, Languages & Applications, Indianapolis, October 26-31, 2013
5. E. Burmako “Scala Macros: Let Our Powers Combine!”, Scala Workshop, Montpellier, July 2, 2013
6. D. Shabalin, E. Burmako, M. Odersky “Quasiquotes for Scala”, Technical report, Lausanne, March, 2013
7. E. Burmako, M. Odersky “Scala Macros”, Valentin Turchin Workshop on Metacomputation, Pereslavl-Zalessky, July 5-9, 2012
8. E. Burmako, R. Sadykhov “Conflux: Embedding Massively Parallel Semantics in a High-Level Programming Language”, Pattern Recognition and Information Processing, Minsk, 2011
9. E. Burmako, R. Sadykhov “Compilation of high-level programming languages for execution on graphical processor units”, Supercomputer Systems and Applications, Minsk, 2010
10. A. Varanovich, E. Burmako, R. Sadykhov “Implementation of CUDA kernels for execution in multiprocessor systems with the use of dynamic runtime”, Technologies of high-performance computing and computer-aided modeling, St. Petersburg, 2010