Tom Murray

+44 (0)7500 016864 | Bristol, UK | tom.paul.murray@gmail.com | linkedin.com/in/tompaulmurray | tommurray.co.uk

Summary

Software engineer with 10+ years' industry experience primarily with C++. 6+ years' compiler design & development for novel hardware platforms, design & development of high performance kernels & libraries to implement machine learning applications, and hardware/software co-design. 3+ years porting AAA Windows-based games to other platforms, contributing to several published works and leading development of one. Most recently working at Fractile across compiler, device runtime, compute kernels, and architectural development.

Work Experience

Senior Software Engineer - Compiler Mar 2025 — Present
Fractile *Bristol, UK*

- Designed & developed early compilers for a novel and evolving hardware platform aimed at machine learning workloads in a small team. Mostly MLIR/LLVM and Clang-based consuming stablehlo from TorchXLA. First a static dataflow scheduler and compiler (with spatially unrolled programs, static allocation etc.) and then a more classic compiler.
- Contributed to development of hardware architecture: proposed modifications/additions, modelled key workloads (a full target model, and specific blocks/layers) to validate and/or guide architectural decisions both from a performance and a programmability perspective.
- Developed prototype performance critical kernels & device runtime software.

Senior Software Engineer - Toolchain Apr 2024 — Feb 2025
VyperCore *Bristol, UK*

- Developed LLVM fork to support VyperCore hardware (RISC-V based). Added new instructions, builtins, target support, linker support, C/C++ language runtime support, and toolchain packaging for distribution.
- Support for numpy and other Python libraries written in C/C++.

Senior Software Engineer - Monetisation May 2023 — Dec 2023
King *Remote (Bristol, UK)*

- Contracted as a senior software engineer in the monetisation team for Candy Crush Soda Saga.
- Worked in cross-functional teams (UI, UX, production, engineering etc.) to deliver new monetisation features in the game as well as bug-fixing and maintaining existing ones. Wrote mainly in C++ and used King's in-house engine targeting Android, iOS, and multiple desktop & web platforms.
- Alongside 2 other developers and over ~4 months successfully delivered a complete overhaul of the in-game shop, a key monetisation feature in the game.
- Sole developer for 2 other minor monetisation features from pre-production to delivery.

Member of the Software Team Feb 2018 — Jan 2023
Graphcore *Bristol, UK*

- Tech lead for a new team responsible for design, planning, and implementation of an unannounced compiler project since early 2022. Project made heavy use of MLIR tooling.
- Modelled whole application and kernel-level performance/memory usage as collateral for hardware design decisions, or evidence of value of hardware features proposed by the software team.
- Designed algorithms, wrote new kernels & libraries for PopLibs (akin to CUDA libraries) e.g. Sparse fully connected layers (forward & backward passes), Top-K & Sort, Slim Convolutions (few input/output channels), Gather/Scatter, Optimised broadcast element-wise operations. See Selected Projects.
- Developed new & existing features for Poplar (low level compiler + runtime akin to CUDA) e.g. Code Overlays.
- Technical leadership & consulting for Poplar & PopLibs acting as a design consultant for proposed Poplar/PopLibs features, supporting other members of the team, and working with higher level users (e.g. ML frameworks, customers) to get optimal results out of the hardware through Poplar/PopLibs.
- Fixing bugs, optimising compiler output, reducing compilation time, CI, inducting new joiners, interviewing.

Developer Oct 2015 — Feb 2018

Feral Interactive *London, UK*

- Project Lead on macOS/Linux ports of Mad Max from original Windows source code written largely in C++ and using DirectX. Led the project from receiving first source code drop to shipping Oct 2016.
- Developer on 3 other ports of Windows games: BioShock Remastered macOS, BioShock 2 Remastered macOS, Dawn of War III macOS/Linux.
- Last 6 months focused on development of Apple Metal code paths in core libraries. Performance optimisations, debug/workaround of graphical issues, implementation of new features. Liaison with third parties for new features and driver issues.

Placement Student Jul 2013 — Aug 2014

Feral Interactive *London, UK*

- Contributions to several ports of AAA Windows games to macOS.
- Implemented first shipped version of downloadable content management in Feral games that allowed users to purchase and download downloadable content for 4 Feral titles during my placement year and several more titles since then.
- 6 months developing a macOS port of GRID 2 from Windows from first delivery of source code. Shipped one month after finishing my placement.

Education

Univeristy of Bath Bath, UK

BSc w/ Honours, Computer Science *2011 — 2015*

Skills

- **Languages**: C++ (98 through 20), C (99, 23), Python
- **Technologies & Libraries**: MLIR, LLVM, Clang, RISC-V, Intel Thread Building Blocks (TBB), GoogleTest, GoogleMock, pytest, Boost, OpenGL, OpenCL, Apple Metal
- **Tools**: Git, Bazel, CMake, gdb, lldb, Jenkins, GitHub, GitLab, Phabricator, Jira, Subversion, neovim, VS Code, Xcode

Selected Projects

Tech Lead, MLIR-based compiler project (unannounced) @Graphcore Early 2022 — Jan 2023

Acted as the technical lead for an ongoing unannounced compiler project within Graphcore for ~1 year. This project used the LLVM project's MLIR tooling to create a new intermediate representation and set of transformation passes to perform optimisations at different levels of abstraction from IPU hardware.

Responsibilities included: technical design/planning/direction, software architectural design, implementation (C++ 17/TableGen), liaison with other technical leaders, being a point of contact for technical questions from outside of the team, providing guidance to team members, and keeping up to date with MLIR/LLVM project discussions, current practices, new features, and general direction to feed into the project.

Sole Developer, Top-K & Sort @ Graphcore Code Documentation

Working alone, I created an optimised implementation of top-k & sort based on a bitonic sort over the course of 7-8 weeks. I adapted bitonic sort to implement top-k, and to the distributed memory architecture of Graphcore's IPU. I wrote kernels to run on individual processors using C++17 & assembly (custom ISA) with coordinating code in C++ 17.

The top-k operation in a key target application for the new implementation was ~48x faster than with the previous heap-sort based implementation, resulting in the application as a whole running ~2x faster.

Sole Developer, Poplar Code Overlays API @ Graphcore Documentation

Working alone, I was responsible for designing & implementing an experimental overlays feature for IPU hardware in the Poplar compiler & run-time. After meeting with internal customers for this feature, I designed and presented the "FunctionBuffer" API. I then implemented the first version of this API which allowed code to be stored off-chip and loaded during execution of a program. This functionality was successfully used to enable internal customers to run programs that did not otherwise fit into limited on-chip memory by swapping out code from off-chip memory during run-time.

Co-developer, Sparse Fully Connected Layer Implementation @ Graphcore Code Tech Note

Over the course of ~4 months, myself and a colleague were responsible for designing and implementing a library for a sparse fully connected layer in neural network where weights for the layer are sparse. We designed and implemented a custom algorithm for IPU hardware.

We split work roughly 50/50. I personally was responsible for part of the design & specification, implementation of some of the kernels in C++17 & assembly (custom ISA), some of the coordinating code in C++17, and later for writing a tech note as documentation/a kind of code companion (linked).

The aim of the project was to drive a customer engagement by providing the functionality they required while operating at or close to a pre-calculated theoretical maximum throughput for IPU hardware, which we delivered.

Project Lead, Max Max macOS Port @ Feral Interactive Ltd. Game Wiki

I was lead developer of the macOS port of Mad Max from receiving the original C++ 11 source code targeting Windows/DirectX through to the final release over the course of 1 year. For several months I worked alone, and for the remaining time additional developers joined me to work on a Linux port and bring the game to release quality.

Some of my contributions were: making the project compile and run in Xcode (and its version of Clang), adding missing features to in-house implementations of Windows/DirectX/other third party APIs using macOS/OpenGL/OpenCL APIs and more, debugging/fixing initial major issues (spanning functional, graphical, and performance issues) such that the game was playable before putting it through several cycles of QA, and performance optimisations to reach the widest possible range of target hardware.

The project successfully met QA requirements and released in October 2016.