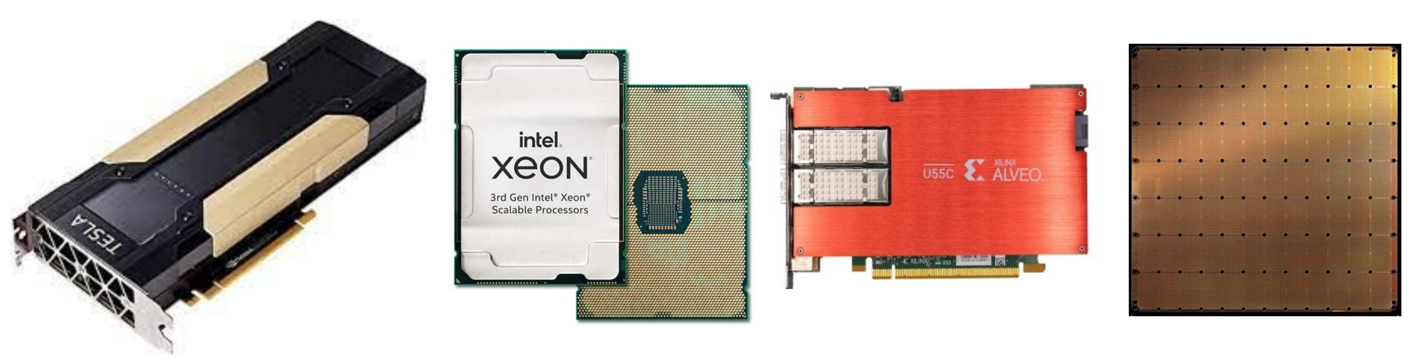
xDSL A common compiler ecosystem for domain specific languages

Nick Brown (EPCC, n.brown@epcc.ed.ac.uk), Tobias Grosser, Mathieu Fehr, Michel Steuwer (University of Edinburgh), Paul Kelly (Imperial College London)

epcc | Imperial College London | The University of Edinburgh informatics

What are Domain Specific Languages (DSLs)?

There is an explosion of hardware architectures for HPC, and as we move into the exascale era a key challenge is how to fully exploit such complex, highly parallel supercomputers.



DSLs provide a separation of concerns, where domain specific abstractions enable programmers to concentrate on their application logic whilst providing a rich amount of information upon which the toolchain can determine tricky low-level decisions around parallelism.

The term *language* is a misnomer, tools and frameworks that provide domain specific ***abstractions*** would be better

Arguably, DSLs are the only way in which we can effectively program and exploit exascale supercomputers

Why are DSLs not ubiquitous in HPC?

A major disadvantage is in the siloing of compiler infrastructure, where DSLs tend to share very little or no infrastructure between them at the toolchain level.

This siloing of toolchains results in:

- ✗ Uncertainty around long term maintenance
- ✗ Considerable effort needed to develop a DSL
- ✗ Limited third-party tools
- ✗ Reinventing the wheel
- ✗ Limited opportunities for DSLs to target multiple domains
- ✗ Considerable effort for new architectures

We must solve the challenges around siloing in order for DSLs to become more widespread

Hence our vision: A DSL being a thin layer atop an existing, mature ecosystem with a wealth of third party tools

30 second summary

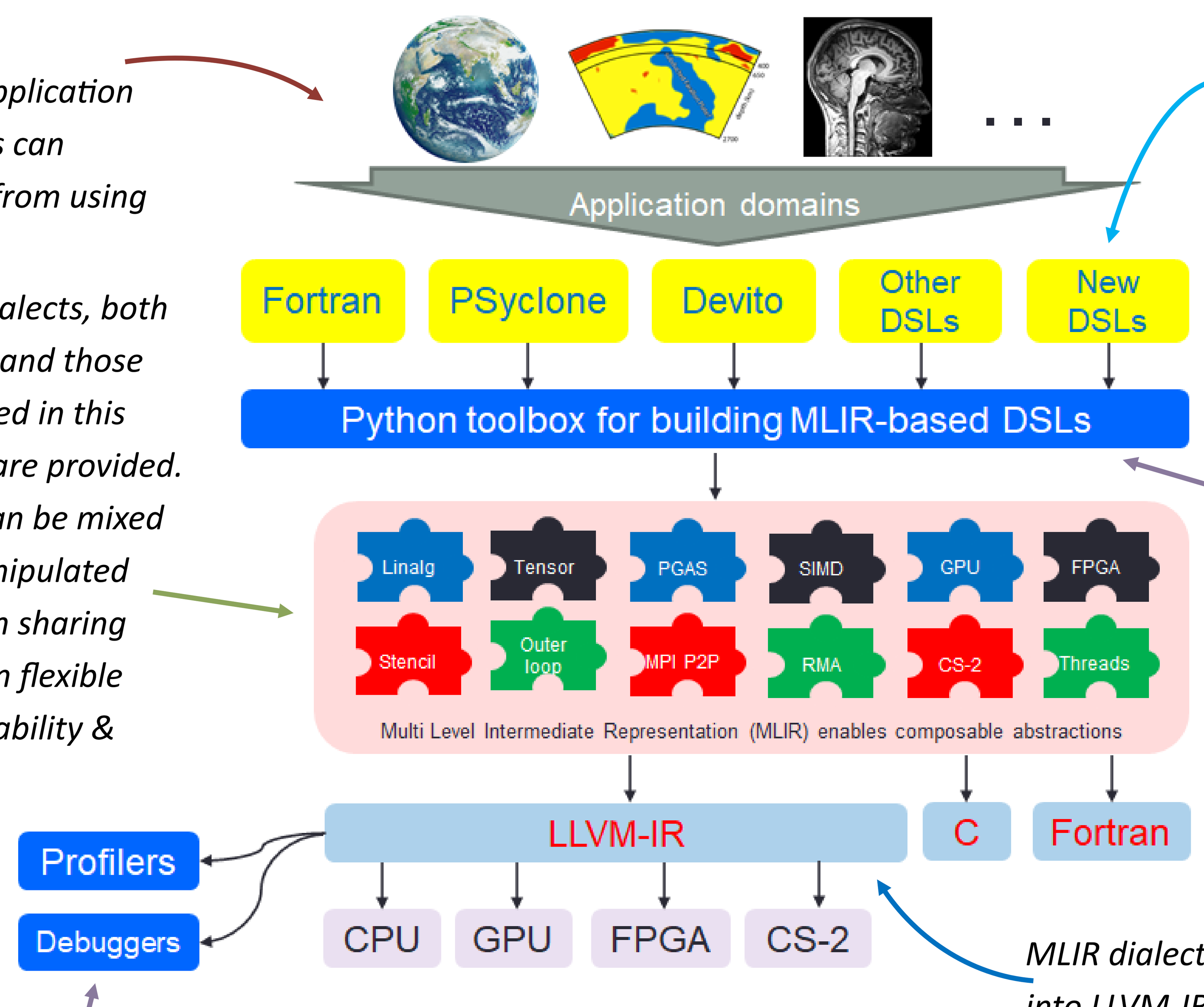
- Domain Specific Languages (DSLs) have great potential.
- Underlying toolchains are often siloed and share very little or no infrastructure.
- We are developing a Python ecosystem so DSL developers can write a thin-layer on-top of existing, well supported, MLIR/LLVM compiler infrastructure.

What is our xDSL ecosystem?

Python toolbox based upon MLIR, integration with MLIR & series of HPC dialects

Many application domains can benefit from using

Many dialects, both existing and those developed in this project are provided. These can be mixed and manipulated and such sharing results in flexible composability & reuse



Variety of existing or new DSLs can build upon our technology

Our Python toolbox exposes MLIR dialects and utilities in a high productivity manner

There are numerous third party tools for LLVM-IR, such as profilers and debuggers

MLIR dialects are translated into LLVM-IR which provides backends for many hardware architectures, or alternatively C or Fortran

Our ecosystem provides:

- ***Composability***, where DSL owners choose what parts of our ecosystem to leverage
- ***Interoperability*** between DSLs
- ***Code reuse*** of toolchain infrastructure
- ***Longevity*** of DSL compiler technology as we build upon LLVM and MLIR
- ***Performance*** because dialects and backends are developed by experts
- ***Productivity*** for both the application and DSL developers due to code reuse
- ***Portability*** for application codes and DSLs across architectures.

Funded by ExCALIBUR

ExCALIBUR

https://excalibur.ac.uk/

The UK ExCALIBUR program address the challenges and opportunities offered by computing at the exascale and aims to deliver the next generation of HPC simulation software and tooling

Building on LLVM and MLIR

LLVM is a collection of common compilation tools and infrastructure. There are numerous LLVM backends available for different hardware, targeted via LLVM-IR.

MLIR

MLIR enables representing and mixing dialects of intermediate representations and abstractions, thus providing easier integration, reuse, and optimisation. However, currently written in C++ there is a fairly steep learning curve that our Python toolbox looks to address.

Get Involved

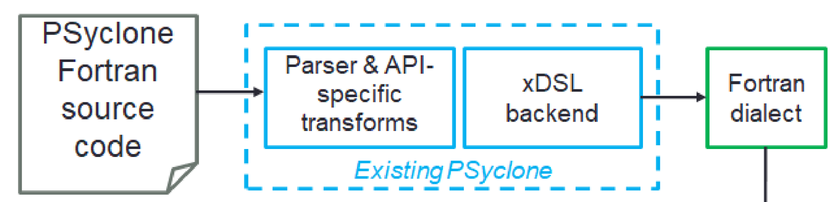
xDSL is open source and we welcome contributions

www.xdsl.dev

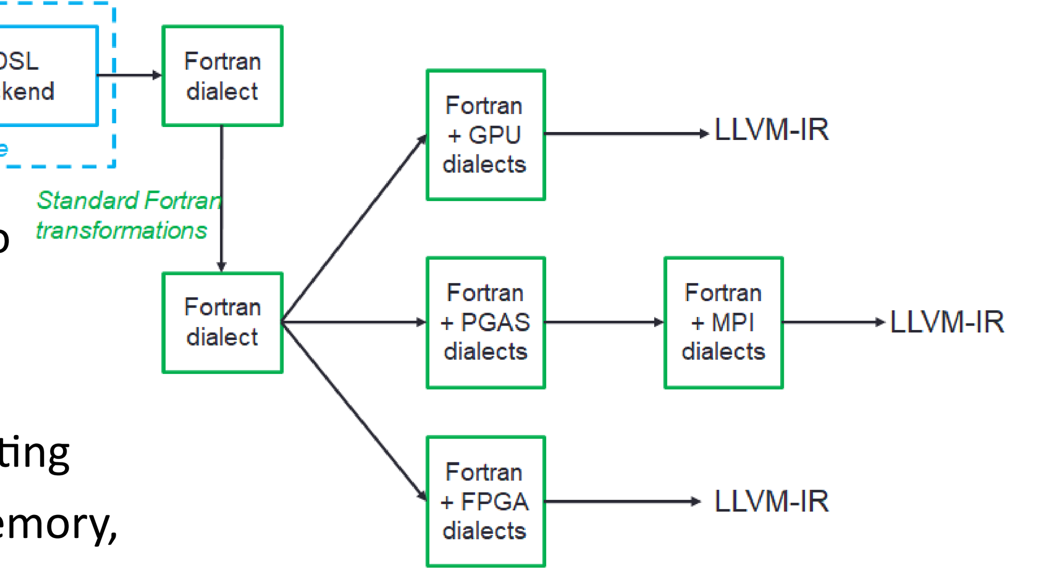
github.com/xdslproject

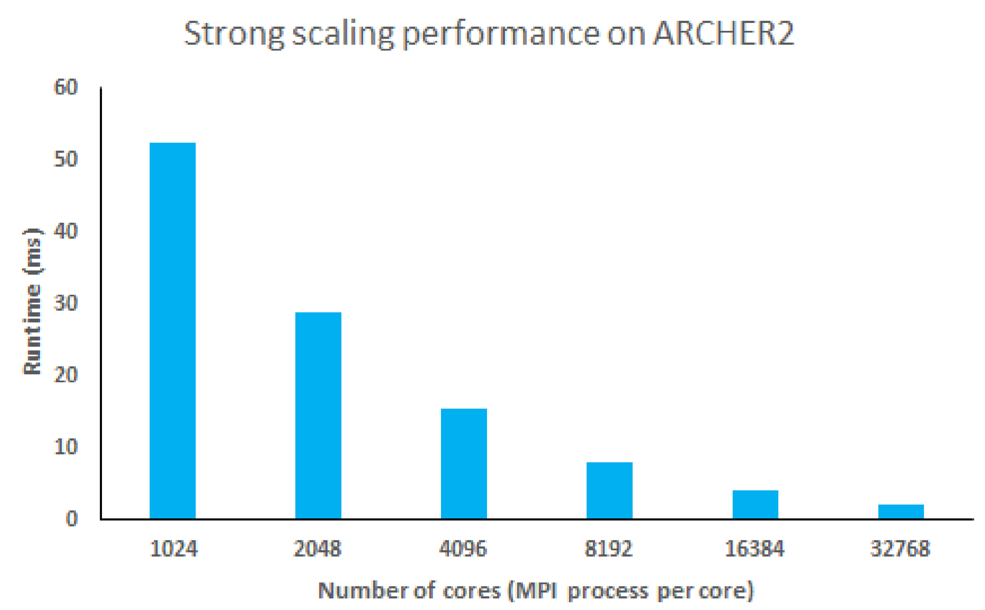
Case study: Integrating xDSL with a Fortran DSL

PSyclone is Fortran-based DSL developed by STFC & the Met Office and used for weather and climate codes including LFric & NEMO



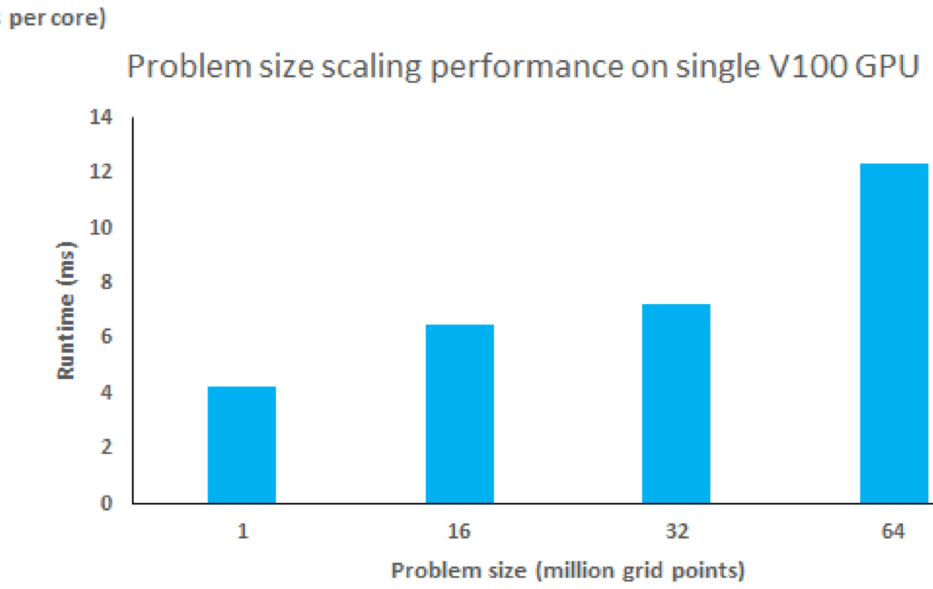
User code is transformed into the Fortran MLIR dialect, combining other dialects as appropriate and then generating code to target distributed memory, GPUs, and/or FPGAs.





This distributed memory or GPU parallelism can be automatically extracted from the scientific codes in using existing, sharable transformations. Once a DSL is integrated with xDSL, aspects such as this come largely out of the box.

The use of Fortran here is an example, xDSL can be used with many different languages, for instance the Devito DSL which is also integrated with xDSL and Python-based.



Using PW advection scheme with 137 billion global grid points on ARCHER2 with 128-core AMD Rome CPU nodes. Running on Cirrus V100s for GPUs.