Languages for High-Performance Computing

There exist a zoo of programming languages. In this short section we briefly want to discuss some frequently used programming languages and their suitability for High-Performance Computing.

Fortran

Fortran is one of the dinosaurs of scientific computing. Fortran originated in the 1950s and its most recent incarnation is Fortran 2018. Fortran is still actively used for a lot of HPC code, especially when it comes to legacy applications. But new projects should not be started in Fortran.

C/C++

C++ is the default language of Scientific Computing. It is mature, has a huge ecosystem and most modern heterogeneous compute environments (Cuda/Sycl, etc.) are developed for C++. Some HPC projects also still use the C programming language, in particular for library development. But C++ is the best choice for most new projects.

Julia

Julia is a relatively recent programming language that has quickly built up a sizeable support community. Even though Julia is a high-level language, it has been successfully used for simulations at Petaflops scale. It should definitely be considered for new applications.

Python

Python is the most widely used high-productivity language in Scientific Computing. Its very simple syntax and broad library support make it ideal for quickly building scalable applications. Python itself is not at all HPC capable unlike Julia. The language does not natively support the type of data structures and other features needed for fast computations. However, over the years Python interfaces to many C/C++ libraries have been developed and with the Numpy extension Python has data types and algorithms for very efficient array operations available. Python today is the number one language for machine learning and many other demanding HPC applications. In this module we will mainly use Python and deep dive into how to develop performant HPC applications in Python.

Matlab

Matlab is one of the oldest high-productivity languages and has been the defacto standard for fast numerical prototyping before Python. It is still heavily used in many numerical applications, given its excellent toolbox and huge amount of legacy code that exists. While Matlab has quite favourable licenses for academic use, it is expensive for commercial use, and if possible Python as open-source alternative is preferable for new projects.

Rust

Rust The Rust programming language is a very recent newcomer. Its first stable release occured in 2015. However, it is quickly become extremely popular as a direct competitor to C++. The main feature of Rust is the ownership model the memory safety model that allows to detect many errors at compile time that would lead to runtime crashes in C++. Rust does not yet have a wide HPC ecosystem and most numerical libraries are in their infancy. Nevertheless, it is feasible that Rust over time matures into a serious HPC language.

Other languages

Other modern programming languages include Go, Java, and C#. Java and C# are business languages that are not designed for demanding HPC applications. Go may be suitable for some HPC type applications. But this is not where its focus lies.

By Timo Betcke