Python HPC Tools

Python has an incredible ecosystem for scientific computing. In this chapter we provide a brief overview of some of the existing libraries before diving down much deeper in the following parts.

Contents

Jupyter Notebook

Jupyter is a key part of the Python ecosystem. It allows the creation of Jupyter Notebook documents that mix executable code, descriptions, figures and formulas in a single file that can be viewed and edited inside a web browser. Jupyter notebooks can be used either through the Jupyter Notebook tool or the more recent Jupyterlab environment.

Numpy and Scipy

Numpy and Scipy are key tools for any scientific Python installation. Numpy defines a fast array data type and provides a huge amount of operations on this type including all common linear algebra operations. Moreover, Numpy beautifully handles multi-dimensional arrays and operations on them. While Numpy provides fairly low-level routines, Scipy builds on top of Numpy to provide a collection of high-level routines, including graph algorithms, optimisation, sparse matrices, ode solvers, and many more. Both, Numpy and Scipy together are one of the main reasons for the success of Python.

Numba

Numba is a tool for the just-in-time compilation of Python functions. Python itself is a slow language. Each operation has considerable overhead from the Python interpreter, making especially time-critical for-loops inefficient in Python. Numba allows to just-in-time compile Python functions into direct machine code that needs not access the Python interpreter. Moreover, in doing so it allows to use simple loop parallelisations that cover a lot of use-cases for parallel computing on a single machine. In addition to all this, Numba has features to directly cross-compile code for use on GPU accelerators. Numba will be one of our main tools to write performant Python codes for CPUs and GPUs.

Matplotlib

Matplotlib is the standard on Python for data visualization in two dimensions. It is incredibly feature rich. While simple plots are easy to do, there is a huge underlying API that allows very fine-grained control for complex data visualization settings. Indeed, the complexity of this API has lead to the development of other libraries that build on Matplotlib and provide simplified interfaces for specific application areas.

Dask

Dask provides a powerful environment to specify complex calculations as graph that is then optimally executed on a given parallel environment. It allows scalability for algorithms from a single desktop up to a HPC systems with thousands of nodes.

Pandas

Pandas is the standard data-analysis library for Python. It has efficient data structures and tools to handle and process large-scale data sets in Python.

Tensorflow, PyTorch, and scikit-learn

Python is the preferred environment for machine learning. We will not go into the various tools as part of this module, but just mention some of them for completeness, in particular Tensorflow, PyTorch and scikit-learn.

Other tools

There are many other tools available for scientific computing in Python, such as mpi4py for data communication on clusters using the MPI standard, petsc4py, an interface to the widely used PetsC library for parallel sparse matrix operations, or FEniCS, a powerful PDE solution package using the finite element method. Many more large and small packages exist for specific application areas which are too numerous to all mention here.