UT Southwestern Medical Center
Lyda Hill Department of Bioinformatics

BioHPC

# Accelerated Scientific Computing with Python

[web] portal.biohpc.swmed.edu
[email] biohpc-help@utsouthwestern.edu

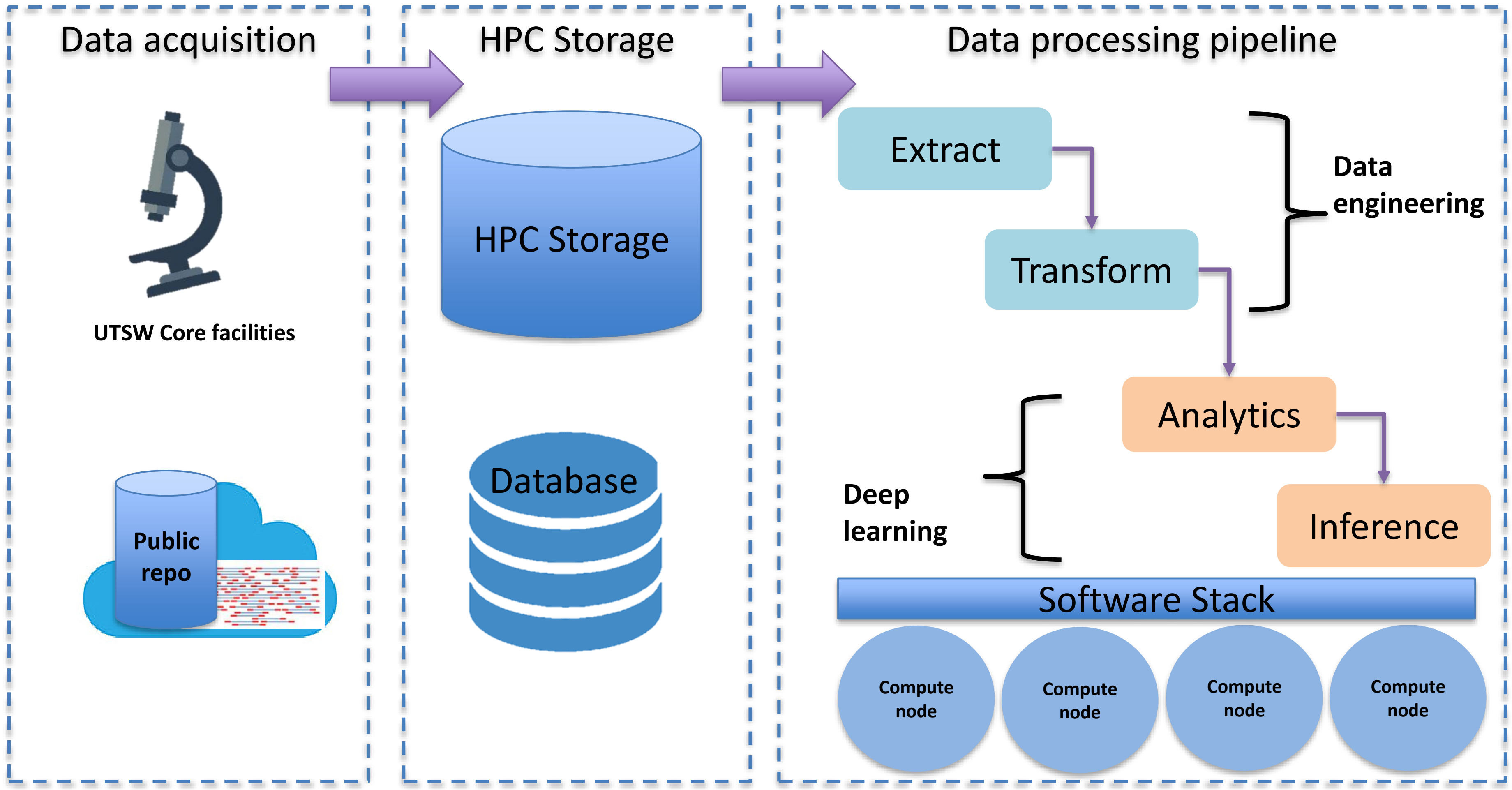
## Welcome! A quick note before getting started…

- Most future BioHPC training sessions will be hybrid!
- Choose to join us online, or **In-Person.**
- **Classroom Location: G9.102**
- Users are encouraged to attend in-person.

## Outline

- General overview of scientific Python-based workflows
- *Numpy* and optimizations
- *Numba* and optimizations
- *JAX* and optimizations
- Live Demo

## Modern Python Workflows on HPC cluster: Data engineering to data analysis



## Modern Python Workflows on HPC cluster

- Python performance overview:
    - According to the creator of Python, the focus of the language was not meant to be fast, but to be *expressive* and *quick to prototype*
    - Regardless of the original focus of Python, it has become very popular in HPC workflows (AI/ML, numerical analysis, etc.)
    - So, how exactly Python works in HPC?
        - We can achieve high performance by skipping the Python layer!

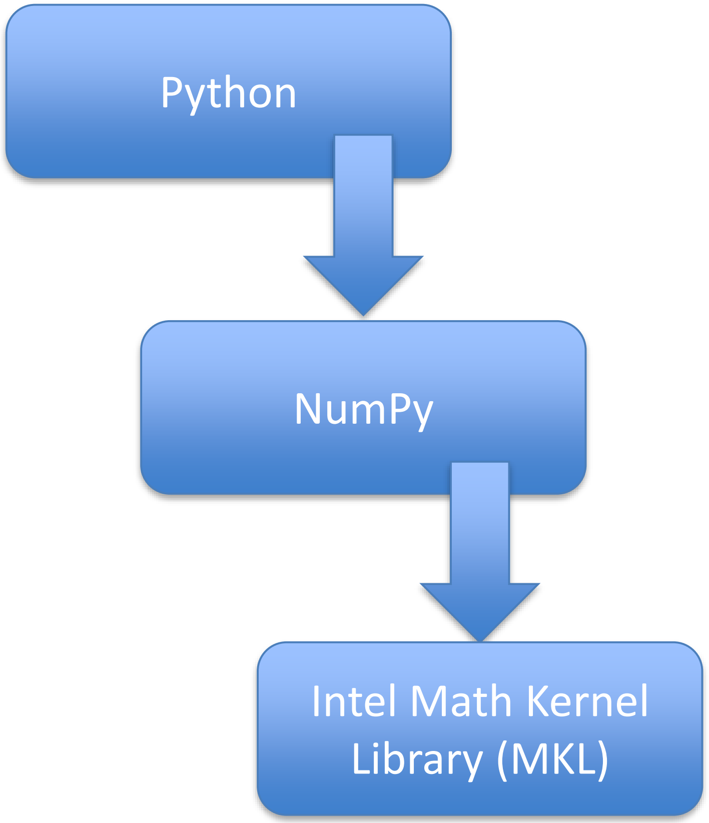
## Modern Python Workflows on HPC cluster (cont.)

A typical Python-based numerical workflow:

Enforces Global Interpreter Lock (GIL) [Global Interpreter Lock] and is single threaded.

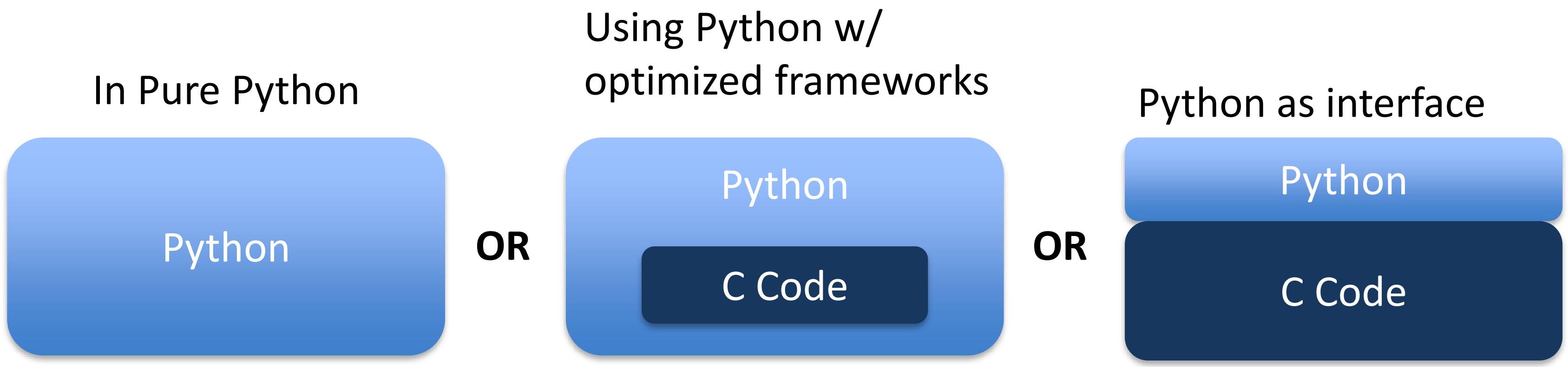
Gets around the GIL
(multi-thread and multi-core).
BLAS can be bottleneck.

Gets around BLAS API bottleneck.
Fastest performance level
Dispatches to hardware vectorization



## Modern Python Workflows on HPC cluster (cont.)

**Different types of Python-based scientific workflows**



## Modern Python Workflows on HPC cluster: NumPy

NumPy

**N**umerical **P**ython: Provides a data structure called **`ndarray`**

- Provides efficient multi-dimensional data structures for storing numerical data
- Provides a large number of functions that do useful things to array
- Delegates most of the operations on such arrays to optimized, pre-compiled C code under the hood.
- Applies **vectorization** on certain operations
  - A vectorized function is applied simultaneously over many values instead of a single value, which is how it looks from the python code (e.g. Summation of two matrices or array element-wise multiplication)

```
def multiply_lists(list_a, list_b):
    for i in range(len(list_a)):
        list_a[i] * list_b[i]
```

Non-vectorized array multiplication

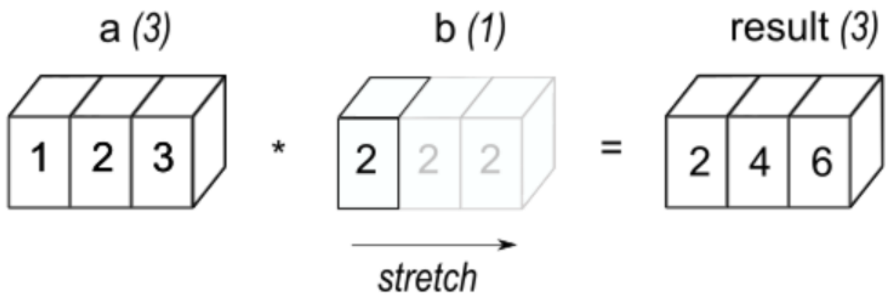
```
arr_a = np.array(list_a)
arr_b = np.array(list_b)

def multiply_arrays(arr_a, arr_b):
        arr_a * arr_b
```

Vectorized array multiplication

## Modern Python Workflows on HPC cluster: NumPy (contd.)

**Broadcasting in Numpy:**

- The term broadcasting describes how NumPy treats arrays with different shapes during arithmetic operations.
- Subject to certain constraints, the smaller array is "broadcast" across the larger array so that they have compatible shapes
- In order to broadcast, the size of the **trailing axes** for both arrays in an operation must either be the same size or one of them must be one.



https://numpy.org/doc/stable/user/basics.broadcasting.html

UTSouthwestern Medical Center
Lyda Hill Department of Bioinformatics
BioHPC

## Modern Python Workflows on HPC cluster: NumPy (contd.)

- NumPy doesn’t depend on any other Python packages, however, it does depend on an accelerated linear algebra library - typically Intel MKL or OpenBLAS.
- The NumPy wheels on PyPI, which is what pip installs, are built with OpenBLAS.
- In the conda defaults channel, NumPy is built against Intel MKL. MKL is a separate package that will be installed in the users' environment when they install NumPy.

Intel and BioHPC co-hosted a workshop on March 1st 2022 with the topic of "Intel AI Analytics Toolkit“ – If you are interested in the training material (video recording + PDF ), then email BioHPC-help@UTSouthwestern.edu

## Modern Python Workflows on HPC cluster: NumPy (contd.)

**Parallelization on multiple CPUs with NumPy:**

- Many functions in NumPy will try to take advantage of multi-core parallelism in your machine.
- The NumPy library uses multithreading by default.
- If your Python code uses multiprocessing module, you should set these four environment variables in your job script:

```
export OMP_NUM_THREADS=1
export MPI_NUM_THREADS=1
export MKL_NUM_THREADS=1
export OPENBLAS_NUM_THREADS=1
```

UTSouthwestern Medical Center
Lyda Hill Department of Bioinformatics
BioHPC

## Modern Python Workflows on HPC cluster: SciPy

- **This is what SciPy provides;** built on top of NumPy, it interfaces with a wide range of C (and Fortran) libraries to provide a core of essential algorithms in scientific computing.
- Some SciPy libraries:
  - *cluster*: hierarchical clustering, vector quantization, K-means
  - *interpolate*: interpolation tools
  - *linalg*: linear algebra routines
  - *ndimage*: various functions for multi-dimensional image processing
- When to use SciPy?
  - Always check to see if the algorithm you need exists in SciPy; it will probably be faster than your own implementation

https://scipy.org/

## Modern Python Workflows on HPC cluster: Numba

- Numba translates Python functions to optimized machine code at runtime (just-in-time compilation) using the industry-standard LLVM compiler library.
- Easy-to-use: No need to replace Python interpreter or make fundamental changes to your Python code, just apply one of the Numba decorators to your Python function.
- Numba is designed to be used with NumPy arrays and functions.
- Provides several options to parallelize your code either on CPU or GPU.

Numba

https://numba.pydata.org/

## Modern Python Workflows on HPC cluster: JAX

- JAX provides a simple and powerful API for writing accelerated numerical code.
- JAX provides a NumPy-inspired interface for convenience.
- JAX arrays can often be used as drop-in replacements of NumPy arrays.
  - e.g: `numpy.sin(x)` <--> `jax.numpy.sin(x)`
- All JAX operations are implemented in terms of operations in **XLA** – the Accelerated Linear Algebra compiler.
- Using a just-in-time (JIT) compilation decorator, sequences of operations can be optimized together and run at once.
- Not all JAX code can be JIT compiled, as it requires array shapes to be static & known at compile time.

https://github.com/google/jax

## Live Demo

- How to access the code?

Code samples uploaded to
https://portal.biohpc.swmed.edu/content/training/training-slides-and-handouts/

- Packages used and installation procedure:
  - Numpy https://numpy.org/install/
  - Numba https://numba.pydata.org/numba-doc/latest/user/installing.html
  - JAX https://github.com/google/jax#installation

UTSouthwestern Medical Center
Lyda Hill Department of Bioinformatics
BioHPC

## Which one to use: Numpy, Numba, or JAX

- Due to the architectural differences, it is not meaningful to compute these techniques.
- JAX is shown to be a promising numerical computing, so give it a shot!
    - It also support GPU as a backend computation device.