accelpy

Release 0.3.1

Youngsung Kim

Jan 24, 2022

CONTENTS:

accelpy is a light-weight Python accelerator interface that allows a gradual migration of time-consuming code to various accelerators such as GPU, through multiple programming models including Cuda, Hip, OpenAcc, OpenMp, C++, and Fortran.

Conceptually, user defines what an accelerator does by providing **accelpy** with an "order", computational code in multiple native programming models and inputs & outputs. And the user executes the "order" to get results.

Practically, **accelpy** generates and compiles a source code based on the "order" and inputs & outputs to build a shared library. Once the shared library is built, **accelpy** sends the input data to accelerator, runs the "order" in the generated shared library, and finally receives the result from executing the "order" to the output variable(s). In other words, **accelpy** takes the responsibility of native code interface, data movement between host and accelerator, and accelerator execution control.

accelpy is not for production use yet.

An example of adding two vectors in Cuda, Hip, OpenAcc, or OpenMp:

```python
import numpy as np
from accelpy import Accel, Order

N = 100
a = np.arange(N)                # input a
b = np.arange(N)                # input b
c = np.zeros(N, dtype=np.int64) # output c

# define acceleration task in one or more programming models in either a string or a file
vecadd = """
set_argnames(("a", "b"), "c")

[hip, cuda]
    int id = blockIdx.x * blockDim.x + threadIdx.x;
    if(id < a.size) c(id) = a(id) + b(id);

[openacc_cpp]
    #pragma acc loop gang worker vector
    for (int id = 0; id < a.shape[0]; id++) {
        c(id) = a(id) + b(id);
    }

[openmp_fortran]
    INTEGER id

    !$omp do
    DO id=1, a_attr%shape(1)
        c(id) = a(id) + b(id)
    END DO
    !$omp end do
"""

# create a task to be offloaded to an accelerator
# with an order, inputs(a, b), and an output(c)
accel = Accel(a, b, Order(vecadd), c)

# asynchronously launch N-parallel work
accel.run(N)
```

(continues on next page)

(continued from previous page)

```

# do Python work here while accelerator is working

# implicitly copy the calculation result to the output array "c"
accel.stop()

assert np.array_equal(c, a + b)
```

Assuming that at least one compiler of the programming models (and a hardware) is available, the “vecadd order” will be compiled and executed on either a GPU or a CPU.

The easiest way to install **accelpy** is to use the pip python package manager.

```
>>> pip install accelpy
```

Source code: https://github.com/grnydawn/accelpy

CHAPTER

ONE

INDICES AND TABLES

- genindex
- modindex
- search