Sanajeh:
A DSL for GPGPU programming with Python objects

東京工業大学 ChenXin JiZhe, 増原 英彦, Matthias Springer

Proposal

- Provide SMMO programming model in python
- Make it easier to write the program compared to using DynaSOAr APIs

Previous work

***DynaSOAr*[1]**

- A dynamic SOA(Structure Of Array) data layout memory allocator written in **C++/CUDA**
- Based on the **SMMO (*Single-Method Multiple-Objects*)** model, which realizes parallelism by running one single method on all objects of a class.

```
Class A{
  int index = 0:
  __device__ void addIndex() {
    index += 1;
  }
}
Int main(){
  //Let A has 3 objects with index 0, 1 ,2
  allocator_handle->parallel_do<A, &addIndex>();
  //A will have 3 objects with index 1, 2, 3
}
```

class name

function name

Fig 1. Example of SMMO

Problems in *DynaSOAr*

- Requires programmers to write codes in an extended syntax.

For example the "__device__" keyword marked as red in Fig 1.

- Difficult for programmers to write complex benchmark.

For example the Barnes-Hut simulation.

Methods

1. A core language based on SMMO

- Written in normal python syntax.
- Create objects of a class on device by ***new_all*** (or ***new_*** when parameters is needed) function
- Detect and mark device data automatically, they will be compiled into CUDA code which will be run by DynaSOAr

2. Provide a library for high-level parallel programming patterns

- Including parallel tree and graph algorithms
- Hide difficult implementation for example the race among different threads

Fig 3 is a step in Barnes-Hut simulation, in which the Race Condition(concurrent data modifying among threads) will happen

```
a = 0

def f1():
  global a
  a += 1

def f2():
  f1()

class A:
    f2()
    pass # Omit other codes

# Create 3 A objects on GPU using Sanajeh API
__pyallocator__.new_all(A, 3)
```

a is used in f1 so it will be marked

f1 is called in f2 so it will be marked

f2 is called in A so it will be marked

A is instantiated in *new_all* so it will be marked

Fig 2. Device data marking in Sanajeh

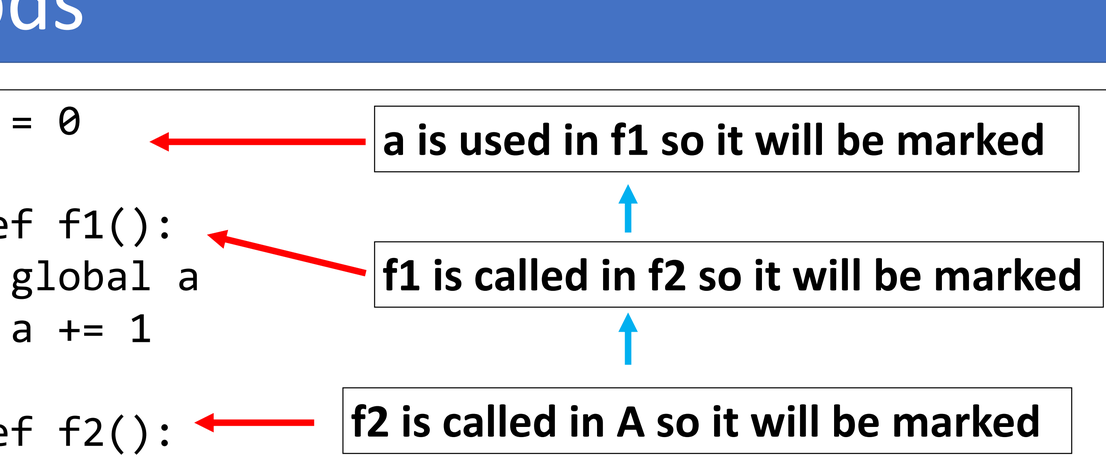

Fig 3. BodyNode inserting in Barnes-Hut

Reference

[1] Springer, Matthias. "Memory-Efficient Object-Oriented Programming on GPUs." *arXiv preprint arXiv:1908.05845* (2019).