Towards a SYCL API for Approximate Computing

Lorenzo Carpentieri, University of Salerno (UNISA)
Biagio Cosenza, University of Salerno (UNISA)
Overview

- Approximate computing introduction
- Software techniques
- SYprox: a SYCL API for approximate computing
  - Perforation Schema
  - Reconstruction schema
  - Host and device perforation
- Experimental evaluation
Introduction to Approximate Computing

▷ Data/computation can be inaccurate and still produce acceptable results
▷ Trade accuracy for higher speedup or smaller energy consumption
▷ Many applications:
  ○ machine learning, neural networks
  ○ computer vision, image processing
  ○ signal processing
▷ Many techniques:
  ○ Hardware: approximate and faulty hardware, memoization etc.;
  ○ Software: perforation, mixed precision, synchronization elision;
Mixed precision methods combine the use of different numerical formats in one computational workload.

Lower-precision pros:
▷ Faster computation and less memory footprint
▷ Transmit more numbers
▷ Use less energy

Cons:
▷ Limits the range of values we can represent
▷ Introduce quantization error
Loop and code perforation

Loop perforation (coarse grained approach):
- skips loop iterations to reduce computation;
- \( k \) is the skip factor used for tuning accuracy vs. performance

Code perforation (fine grained approach):
- skips loop instructions to reduce computation

Many applications are memory-bound
  - computation is cheap, memory access is expensive

Data often contains redundancy
  - many applications can deal with some amount of error

Data perforation:
  - skip the loading of redundant parts in input data
  - exploit data locality to reconstruct perforated data, reducing the final error
Input and output reconstruction

A reconstruction phase is needed in order to reduce the error introduced by the data perforation:

- output reconstruction\(^1\) (high error);
- input reconstruction\(^2\) (less error).

---


SYprox: a SYCL API for Approximate Computing

SYprox a portable SYCL API for developing approximate computing techniques:

- New perforation approach:
  - Host perforation
  - Device perforation

- Different perforation schemes

- Input and output reconstruction

- Data perforation + mixed precision

```cpp
pbuffer<half, 2, pcol::lerp> buf_a(a, range<2>{N,N});
// output reconstruction with lerp
pbuffer<half, 2, pcol::lerp> out_buf(out, range<2>{N,N});
// global size and work group size
range<2> gl{N,N/2}, ws{32, 32};
q.submit([&](handler &h){

    paccessor<float, 2, pcol::lerp> perf_acc{buf_a, h, read};

    h.parallel_for(nd_range<2>{gl, ws},
    [&, &](nd_item<2> it){
        id<2> id = it.get_global_id();

        // acc_a data are perforated host side
        out_acc[id*2] = acc_a[id] * 2;

        // perf_acc data are perforated device side
        out_acc[id*2] = perf_acc[id] * 2;
    });
});
```
SYprox approach

a) Accurate execution

b) SYprox approach
Perforation schemes define which data should be not computed.

SYprox provides 4 built-in perforation schemes.

The **skip factor** for the 1D/Row/Column schemes defines the number of elements/rows/columns to skip.

- 1D skip factor 2
- 1D skip factor 3
- Row skip factor 2
- Column skip factor 2
- Chess
- Row skip factor 3
- Column skip factor 3
Data perforation happens before sending data on the device;

Less data transfer between host and device;

All the perforation schemes adopted are aligned with memory architecture
Data perforation happens on the device;

Send all data from host to device;

Perforation schemes can be affected by the array memory layout (e.g. row-major, column major)

```
buffer<half, 2> buf_a(a, range<2>{N,N});
buffer<half, 2>
    out_buf(out, range<2>{N,N});
range<2> gl{N/2,N}, ws {32, 32};
q.submit([&](handler &h){
    paccessor<half, 2, float>
    perf_acc(buf_a, h, read);
    h.parallel_for(nd_range<2>{gl,ws},
    [&] (nd_item<2> it){
        id<2> id = it.get_global_id();
        out_acc[id*2] = perf_acc[id] * 2;
    });
});
```
SYprox provides two reconstruction strategies:

- **Output reconstruction** approximates perforated data after computation;
- **Input reconstruction**: perforated elements are reconstructed in local memory before computation.

SYprox provides 3 built-in way to reconstruct data:

- Nearest neighbour;
- Basic linear interpolation;
- Stencil interpolation;
Evaluation

- Implemented 3 image processing applications using SYprox;
- Combined different approximate computing techniques:
  - Data perforation;
  - Input and output reconstruction;
  - Mixed precision using half precision floating point.
- Run applications on NVIDIA V100 GPU;
- Measure run time and error:
  - Speedup 1.2x to 6x;
  - Average error less than 10%.

Applications

- Box blur
- Median filter
- Sobel filter
Results

Box blur

Median

Sobel

better
SYprox: a SYCL API for approximate computing

- Combined different approximate computing techniques:
  - Data perforation;
  - Input and output reconstruction;
  - Mixed precision using half precision floating point.

- Speedup 1.2x to 6x;
- Average error less than 10%.

Contacts:
Lorenzo Carpentieri: lcarpentieri@unisa.it
Biagio Cosenza: bcosenza@unisa.it