CUPL : A Compile-time Uncoalesced Memory Access Pattern Locator for CUDA

Madhur Amilkanthwar, Shankar Balachandran

Department of Computer Science and Engineering, Indian Institute of Technology, Madras.

{ madhur, shankar } @ cse.iitm.ac.in

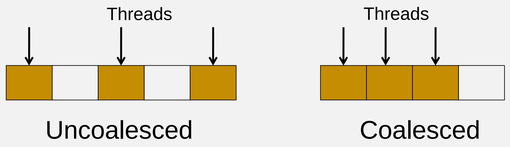
ACM SRC, ICS 2013

1. The problem

How to locate uncoalesced memory access patterns in CUDA programs at compile-time?

2. Background

- **Uncoalesced vs. Coalesced**



3. Motivation

- **Better DRAM bandwidth utilization.**
- **Less memory transactions.**

4. CUPL

- **Input :**
  A valid CUDA kernel + kernel launch configuration.
- **Output :**
  Warnings if an array is accessed in an uncoalesced manner.

5. An example

```
__global__ void kernel(int *A)
{
  for(int i=0;i<32;i++)
    A[32*threadIdx.x + i] *= 10;
}
```

Kernel launch configuration

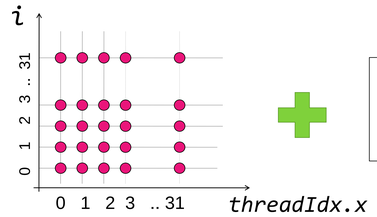
<<< 1, 32 >>>

Phase 1

Covert to a valid C code

PET

Polyhedral representation

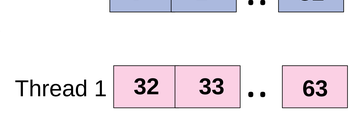


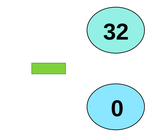
Iteration domain

Access Relation :
S_0[i] → A[32*threadIdx.x + i]

Get sets of locations accessed by any two consecutive threads

Difference between lowest memory addresses





D >0

Phase 2

Warning

Optimized code runs 3.5X faster!!

6. Benefits

CUPL has two-fold use :

- **It can help the programmer – to locate the regions of the code to optimize.**
- **It can help a compiler – to locate an opportunity to perform efficient data layout transformations.**

7. Results

Suite	Benchmark name	Kernel name	Array name
Rodinia	Kmeans	Invert_mapping	Input
Rodinia	Stream cluster	kernel_compute_cost	work_mem_d
NVIDIA	Box_filter	d_box_filter_x	Id, od
NVIDIA	Histogram	merge_histogram	d_partialHist

8. Future work

- **Handle all cases of uncoalesced accesses e.g. cases where difference, D, is not constant.**
- **Use CUPL as a preprocessor to perform data layout transformation.**
- **Integrate CUPL in state-of-the-art C to CUDA code translators.**

9. References

[1] S. Che et al. *A Characterization of the Rodinia Benchmark Suite with Comparison to Contemporary CMP Workloads. In IISWC, pages 1-11, 2010.*

[2] S. Verdoolaege. isl: *An Integer Set Library for the Polyhedral Model. In ICMS, pages 299-302, 2010.*

[3] S. Verdoolaege and G. Tobias. *Polyhedral Extraction Tool. In Second Int. Workshop on Polyhedral Compilation Techniques (IMPACT 2012), Jan. 2012*

[4] CUDA programming guide. Version 5.