Introduction

Efficient off-chip memory access is crucial to high performance; the applications are often limited by off-chip memory bandwidth. Transforming data layout is an effective way to reshape the access patterns to improve off-chip memory access behavior, but several challenges had limited the use of automated data layout transformation systems on GPUs, namely:

- How to efficiently handle arrays of aggregates?
- NVidia GTX480
- How to convert layouts on the GPU without 2X space overhead?
- How to hide the converted layout from the rest of the application?

Aim

A layout that is good across GPU architectures.

Fast conversion between layouts in GPU global memory. Need to be fast, parallelized, and in-place.

Run-time marshaling at transformation boundaries; the rest of application should work without recompilation.

Method

ASTA: Array-of-Tiled-Structure-of-Arrays

Transparency run-time marshaling for OpenCL

OCL buffers and kernel invocation tracked through library interposition:

Host

DL Runtime

Transparent run-time marshaling

Conclusions

The Array-of-Structure-of-Tiled-Array (ASTA) layout is a promising alternative to common discrete array or SoA transformation for GPU applications that access data in Array-of-Structure layout.

ASTA not only provides better performance to discrete arrays but also enables in-place marshaling on GPUs, which is crucial for accelerators relying on high-throughput access to capacity-constrained private DRAMs. ASTA allows much faster dynamic in-place marshaling from AoS than discrete arrays.

To allow developers to leverage the benefits of ASTA with minimal effort, this work decouples host and device layout needs through a user-friendly automatic transformation framework that is designed and implemented in a transparent way to host code, even allowing user to keep host code unchanged while enjoying the benefit provided by the system.