Performance Evaluation of the NVIDIA Tesla V100: Block Level Pipelining vs. Kernel Level Pipelining

Xuewen Cui, Thomas R. W. Scogland , Bronis R. de Supinski and Wu-chun Feng

SyNeRGy Lab @ Virginia Tech

Abstract

Heterogeneity continues to increase in all kinds of computing applications, with the rise of accelerators such as GPUs, FPGAs, APUs, and other co-processors.
Programming models, such as CUDA, OpenACC and OpenCL are designed to offload compute-intensive workloads to co-processors efficiently.

Motivation and Goal

Drawbacks of current popular Kernel Level Pipelining

i. End user must manually partition the task to multiple sub kernel chunks and then launch by multiple GPU streams.
ii. Splitting to multiple chunks may cause extra function call overhead.
iii. Parameters (#chunks ,#streams, etc.) must be well tuned to provide optimal performance

We proposed a new block-level pipelining extension for OpenMP that:

✓ Handle data transfer and computation inside one kernel using different streaming multiprocessors.
✓ Trigger the computation using atomic operations as long as the data is ready thus pipeline the data transfer and computation.

Environment Setup

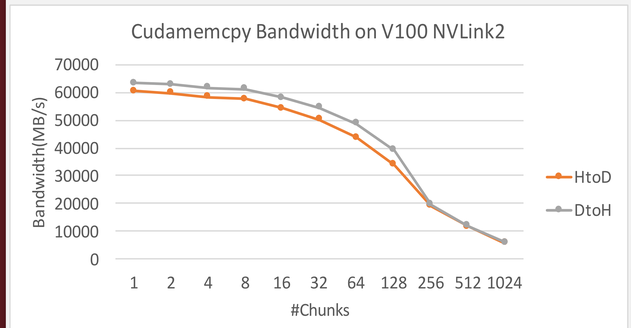
Machine1:
CPU: IBM Power9 processors
GPU: NVIDIA TESLA V100 with NVLink2
Machine2:
CPU: Intel Xeon Gold 6136
GPU: NVIDIA TESLA V100 with PCI-E
Benchmarks:
1. 2D Convolution
2. Generic Matrix-Matrix Multiplication
CUDA Version: 9.0.176

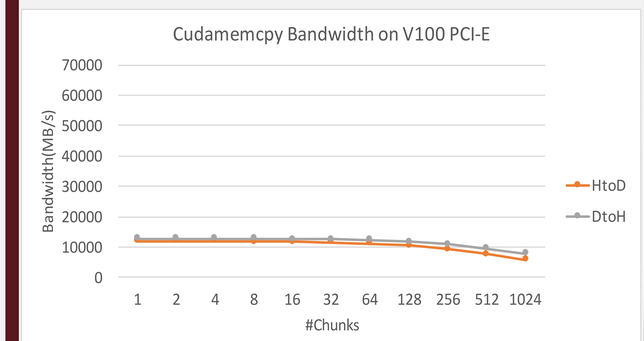
Data Transfer Bandwidth Evaluation

CUDA API

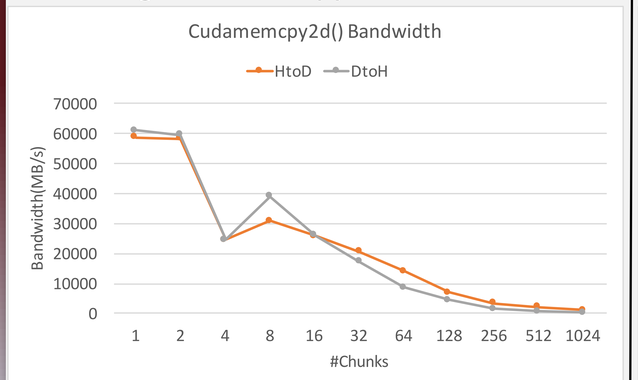
- usually provides the maximum bandwidth
- no copy order guarantee
- need to be partitioned to multiple separate calls
- does not notify concurrently running kernels

Contiguous Data Copy



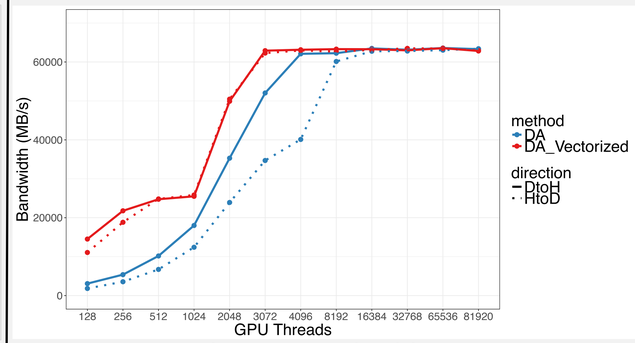


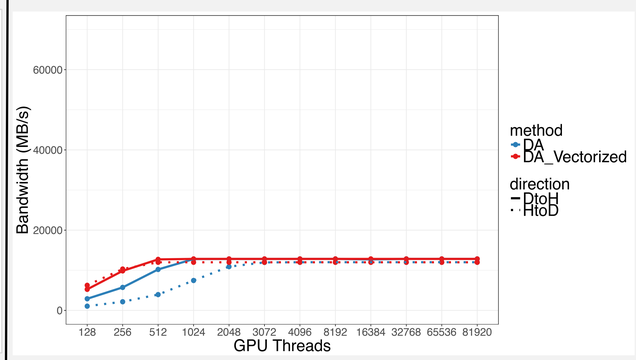
Non-contiguous Data Copy

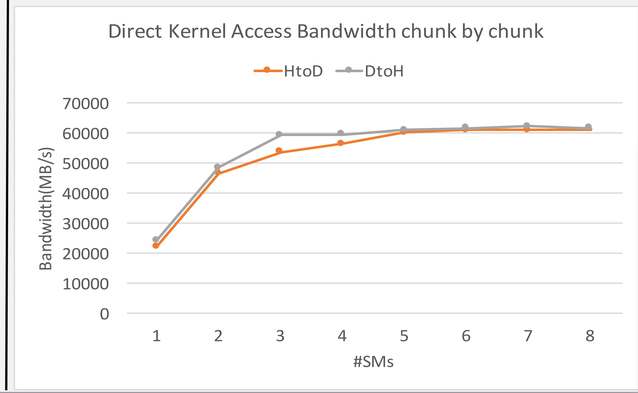


Direct Kernel Access

- require stream multiprocessors to handle the copy
- can control the copy order
- only one kernel launched
- able to notify concurrently running kernels

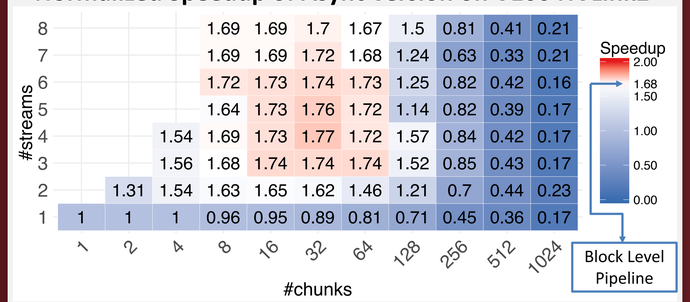


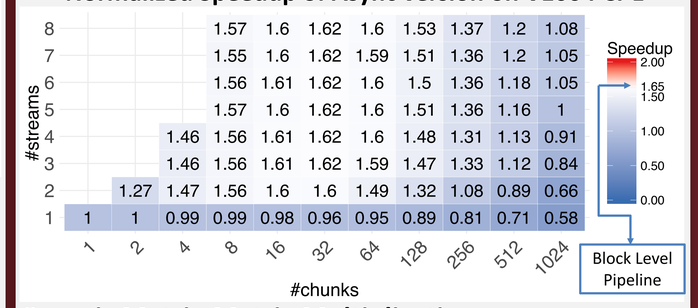




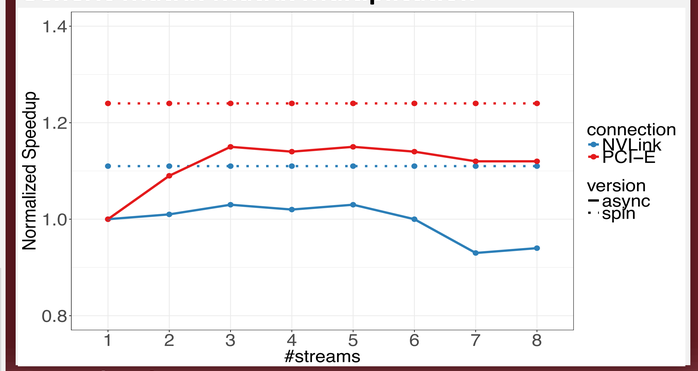
Performance Evaluation

2D Convolution





Generic Matrix-Matrix Multiplication



Conclusions

- Direct Kernel Access achieves the bandwidth limit as long as enough GPU threads are engaged.
- Bandwidth using traditional CUDA copy APIs may drop dramatically if splitting to multiple chunks, especially for non-contiguous data transfer.
- Our approach provides 95% to 108% stable performance compared to the best tuned results with traditional kernel level pipeline on V100 GPUs.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.