

Home > CUDA ZONE > Forums > Accelerated Computing > CUDA Programming and Performance > View Topic

## Data transfer between kernels



kay21s



I have read a paper that claims to transfer data between GPU kernels. <http://pdcc.ntu.edu.sg/xtra/paper/2016Pub/GPL.pdf>

In the Appendix A, it claims that nvidia GPUs is able to "pass data directly from one kernel to another via Direct Data Transfer (DDT)".

I am not able to find such a technique described on Nvidia website. Is there an official document for this technique?

Thanks!

Posted 09/03/2016 09:06 AM

#1



njuffa



I assume the relevant quote from the paper is:

"As in case of the AMD GPU, the latest GPUs from NVIDIA (Fermi or Kepler architectures) have also been enabled with concurrent kernel execution capability so that multiple kernels can be executed on the same GPU simultaneously. They also have the ability to pass data directly from one kernel to another via Direct Data Transfer (DDT) [5]"

I have no idea what the authors could mean by DDT here, it is their own private nomenclature best I can tell. The normal way for kernels to "communicate" is that first kernel deposits data into global memory, second kernel picks up the data from global memory, pointers are passed to each kernel indicating where that data is located. Have you tried looking up the cite reference [5], which presumably introduces the term DDT?

[5] Z. Chen, J. Xu, J. Tang, K. Kwiat, and C. Kamhoua, "G-storm: GPU-enabled high-throughput online data processing in storm." In *Big Data (Big Data)*, 2015 IEEE International Conference on, pages 307 - 312, Oct 2015.

Posted 09/03/2016 01:25 PM

#2



said:

I assume the relevant quote from the paper is:

"As in case of the AMD GPU, the latest GPUs from NVIDIA (Fermi or Kepler architectures) have also been enabled with concurrent kernel execution"

Scroll To Top

kay21s



capability so that multiple kernels can be executed on the same GPU simultaneously. They also have the ability to pass data directly from one kernel to another via Direct Data Transfer DDT) [5]"

I have no idea what the authors could mean by DDT here, it is their own private nomenclature best I can tell. The normal way for kernels to "communicate" is that first kernel deposits data into global memory, second kernel picks up the data from global memory, pointers are passed to each kernel indicating where that data is located. Have you tried looking up the cite reference [5], which presumably introduces the term DDT?

[5] Z. Chen, J. Xu, J. Tang, K. Kwiat, and C. Kamhoua, "G-storm: GPU-enabled high-throughput online data processing in storm." In *Big Data (Big Data)*, 2015 IEEE International Conference on, pages 307 - 312, Oct 2015.

Thanks for your reply. I have searched the "DDT" and "Direct Data Transfer" in [5], but I cannot find any information about it. Maybe the reference is wrong, and in the Appendix A.1 the author has also performed the experiments on kernel communication on nvidia GPUs.

---

Posted 09/03/2016 02:01 PM

#3



njuffa



As far as I could tell by scanning through, the paper is mostly concerned with communication between *concurrently running* kernels, with a focus on AMD GPUs.

---

Posted 09/03/2016 02:10 PM

#4



kay21s



said:

As far as I could tell by scanning through, the paper is mostly concerned with communication between *concurrently running* kernels, with a focus on AMD GPUs.

Yes, I have carefully read the paper, which focuses on the AMD GPUs. But in the appendix, it tries to show that its design also fits for Nvidia GPUs.

As the paper mentions the DDT technique, I thought nvidia also supports kernel communication. I will ask the authors about it.

---

Posted 09/03/2016 02:30 PM

#5

[Scroll To Top](#)



txbob



OpenCL 2.0 introduces pipes, which are designed for kernel-to-kernel data transfer without otherwise explicitly managing a buffer for such purposes:

<http://developer.amd.com/community/blog/2014/10/31/opencl-2-0-pipes/>

DDT is not any official part of OpenCL terminology that I am aware of, but since the originally linked paper mentions pipes, they may be referring to that on the AMD GPU side.

Since NVIDIA GPUs don't officially support OpenCL 2.0 (and therefore pipes), the underlying kernel data transfer mechanism proposed must be different, and this is supported by a comment in the appendix:

"Unlike the AMD GPU, the NVIDIA GPU do not need users to set the packet size."

The packet size is a pipe characteristic.

Therefore the kernel data transfer being proposed ("DDT") on NVIDIA GPUs is almost certainly using an explicitly managed global buffer technique. Unfortunately further detail here probably depends on reference [5] in the paper, which seems to be an IEEE paper that I cannot find publicly. As a guess, DDT may be nomenclature devised within that paper on "G storm", to cover kernel data transfer, whose underlying implementation varies somewhat between AMD and NVIDIA GPUs.

---

Posted 09/03/2016 03:37 PM

[#6](#)

njuffa



@txbob: You should have access to all IEEE publications through your employer (site license). Just visit the IEEE digital library from a computer at work. I have a limited digital subscription myself and will check whether it covers reference [5]. Note that OP stated that DDT is not actually defined in reference [5] which they already consulted.











[Later:] Turns out the G-storm paper is covered by my limited personal subscription to IEEE's digital library. According to the paper, G-Storm is a stream processing system built on top of JCuda. As OP already mentioned, there doesn't seem to be anything like Direct Data Transfer (by name or concept) described in this paper. In fact, the system involves host/device copies:

After user kernel execution is complete, output data are stored in a pre-allocated output memory space, which is then copied back to the host at once to minimize transfer overhead.

---

Posted 09/03/2016 03:45 PM

[#7](#)[Add Reply](#)[Scroll To Top](#)

**B** **I** **U** **S** **T** |          

Please [Login](#) | [Register](#) to add a comment.

☒ Receive email notifications when someone replies to this topic

 Preview

 Reply

## GET STARTED

COMPUTEWORKS

GAMEWORKS

JETPACK

DESIGNWORKS

About CUDA

Parallel Computing

CUDA Toolkit

CUDACasts

## LEARN MORE

Training and Courseware

Tools and Ecosystem

Academic Collaboration

Documentation

## GET INVOLVED

Forums

Parallel Forall Blog

Developer Program

Contact Us