DCGN - Message Passing on GPUs

UCDAVIS
UNIVERSITY OF CALIFORNIA

Jeff Stuart and John D. Owens
University of California, Davis

What is DCGN?

Message Passing

DCGN stands for Distributed Computing for GPU Networks. DCGN allows CUDA developers to issue message-passing commands on the GPU without explicit CPU supervision or interference.

Dynamic Communication

DCGN allows for fully-dynamic communication. Unlike many previous GPU communication libraries, DCGN allows for both flexible communication and efficient communication.

No CPU Oversight

DCGN removes the necessity for explicit oversight by the CPU. And due to the use of a polling scheme, CUDA kernels do not have to exit when communication would be required.

Slots

In order to achieve efficient runtimes, DCGN uses slots: a user-specified maximum number of simultaneous communication requests allowed. DCGN reserves enough memory for N communication requests (N = #slots). The user must explicitly index a slot when issuing a communication request. This mitigates the need for locking primitives on the GPU.

CPU Utilization

DCGN lets developers run CPU kernels simultaneously with GPU kernels. The method for writing and launching CPU kernels is nearly orthogonal to the methods for GPU kernels.

Background

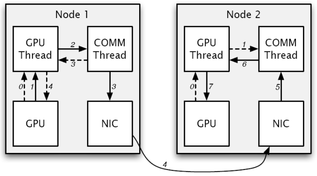
DCGN was created for two reasons; 1) because static communication libraries such as CUDASA [1] and Zippy [2] only offered communication via the partitioned global-address space (PGAS) metaphor, and 2) previous dynamic communication [3] on the GPU suffered from poor performance.

DCGN merges the positives from 1) and 2) without taking any of the negatives. Excellent performance is achieved with truly dynamic communication.

[1] M. Strengert, C. Mller, C. Dachsbacher, and T. Ertl. CUDASA: Compute Unified Device and Systems Architecture. In *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV08),* pages 49–56, 2008.
[2] Z. Fan, F. Qiu, and A. E. Kaufman. Zippy: A framework for computation and visualization on a GPU cluster. *Computer Graphics Forum, 27(2), June 2008.*
[3] Adam Moerschell and John D. Owens. Distributed texture memory in a multi-GPU environment. *Computer Graphics Forum, 27(1):130–151, March 2008.*

Communication

The route for communication via DCGN is complex, but as simple as possible. DCGN threads on the CPU monitor for communication requests and handle said requests appropriately, often times through the use of a network interface card.

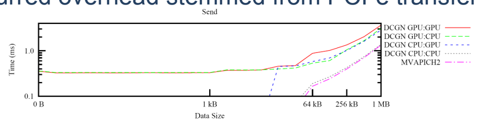


Results

DCGN was designed to aid developers by not only allowing GPUs to communicate, but to do so easily and without explicit oversight by the CPU. To this extent, we evaluated DCGN based on speed and parallel and efficiency, and also on ease of programming and the reduction in lines of code and total number of kernels.

Communication Calls

In testing communication calls, we found that DCGN typically adds very little overhead. For small data, the overhead above MVAPICH2 was less than 0.5 ms. For larger data, the overhead grew with respect to the required time for PCI-e transfers. In fact, the overwhelming majority of the incurred overhead stemmed from PCI-e transfers.



Microbenchmarks

Beyond testing basic communication, we also tested against a few applications. We implemented a version in DCGN and one using GPUs as slave (GAS) devices. DCGN performed within 10% of the speed and efficiency of GAS applications, and in some cases performed faster than the GAS application.

Code Differences

DCGN allowed each application to be written using only a single kernel and invocation, whereas the GAS versions required several kernels and numerous kernel invocations.