High-Performance Computing By Advanced Stream Processing Using GPUs

Young Sung (Sean) LEE
seanl@cse.unsw.edu.au
Programming Languages and Systems Group
The School of Computer Science and Engineering
The University of New South Wales

Abstract

The pipelines in modern GPUs have become programmable and reconfigureable – they used to have a set of fixed funcionalities with no programmability. In addition, *graphical processing units* (GPUs) have evolved to the point where its performance / cost ratio is superior to *central processing units* (CPUs) with computations of high arithmetic intensity. The combination of the programmability and the efficiency of GPUs may result in significant processing speed-up, when GPUs are employeed as co-processors to CPUs. For this reason, there have been a number of attempts to utilise GPUs in general-purpose computing. They have mapped *flat data parallelism* (FDP), which requires manual flattening of data, on GPUs with *non-uniform approach*, in which programmers must explicitly separate operations to be executed on GPUs and on CPUs.

In order to support wider range of algorithms, *nested data parallelism* (NDP), as well as FDP, needs to be mapped on GPUs. NDP, in contrast to FDP, is capable of processing *irregular data structures* such as sparse matrices and trees, and this capability makes it possible to map algorithms such as permute and sparse matrix multiplication. In addition to NDP, the realisation of *uniform approach* in the programming language level is sought by programmers without sufficient knowledge nor experience in graphics area.

I have been working to enable NDP operations on GPUs with uniform approach. It involves a GPU library for bulk operations on unsegmented and segmented arrays, and program transformations.

As part of the GPU library, I have implemented segmented and unsegmented array operations such as scan, fold, filter, and map, and more operations are to be implemented. Input arrays are passed to GPUs as textures, the operations are performed on each elements by programmable processing units in parallel. The output arrays are taken from the framebuffer objects. Programmers can define the behaviour of these operations with C-like syntax. For example, scan operations can be used for plus-scan and for max-scan. The behaviour definition will become richer when a proper pre-processor is implemented for that. Anonymous lambda function support in the behaviour definition is also under consideration as part of the pre-processor. The executions of these operations take less time on GPUs than on CPUs; the execution times of some operations on GPUs were only about 10% of their execution times on CPUs. However, the code compilation and the data transfer between CPUs and GPUs have become issues; GPU codes are compiled at runtime, and it affects the runtime unfavourably. Techniques to reduce these overhead have to be developed.

Upon finishing the implementation of the GPU library, program transformations will be investigated and implemented.