K-model: A New Computational Model for Stream Processors

Gabriele Capannini - email capannin@di.unipi.it

Given the demand of massive computing power in modern rendering applications, GPUs (as well as Sony's Cell BE processor, and next-generation CPUs) are designed to be extremely fast at rendering large graphics datasets. Indeed, inspired by the attractive performance/cost ratio, several studies adopt such type of processors also for carrying out data-intensive and general-purpose tasks. Their architecture consists of various single-instruction-multiple-data (SIMD) array processors that are able to compute an huge number of arithmetic operations at the same time.

Related to SIMD, there is the stream processing. It is a computer programming paradigm that allows some applications to more easily use multiple computational units by restricting the parallel synchronization, or communication that can be performed among those units. Given a set of data (a *stream*), a series of operations (*kernel* functions) are independently applied to each element in the stream. Furthermore, due to both the performance constraints and the absence of some architectural "facilities", e.g. jump prediction and re-order buffer, the design of efficient algorithms for these processors requires more efforts than for standard CPUs.

To face this issue, we propose K-model that is a computational model aimed to capture all the distinguishing features of GPUs, and alike architectures. Given a K-algorithm, two functions evaluate the amount of work performed inside the processor in term of latency and length of the instructions. The first function can be thought as a measure of the parallel complexity, whereas the other one can be thought as the complexity assuming we are on a serial RAM, and it contributes to measure the efficiency of the algorithm. However, the evaluation of the "internal-work" induced by a stream element is not the unique purpose of K-model. A relevant aspect to take into consideration is the pattern used to access the memory storing both input data and results.

Two case studies that focus on these two different type of performance constraints are presented: (i) a new function to map Bitonic Sorting Network that improves GPU exploitation and maximizes the bandwidth with which the data is transferred. It is worth noticing that being an in-place sorting based on bitonic networks our solution uses less memory that is an important aspect when sorting large volume of data, as it is required by large-scale distributed system for information retrieval; (ii) a novel and efficient method to compute another important building block for parallel algorithms: the parallel prefix sum.