Fundamental Algorithm Techniques
For Many-Core Processors: GPUs

Hamidreza Zaboli & Frank Dehne
School of Computer Science
Carleton University, Ottawa, Canada

ABSTRACT

The field of multi-core and many-core processors has been pushed to new levels by the advent of general purpose GPU (GPGPU) architectures such as Nvidia Tesla [12] and accessible programming environments such as CUDA [11]. An example of an Nvidia GeForce GPU is shown in Figure 1 in comparison to a Cell processor architecture shown in Figure 2. Major differences between a Cell processor and a GPGPU include

- A large number of processor cores (200+)
- A large global memory with high bandwidth
- Internal shared memories for each group of 8 processors
- The need for SIMD (SIMT) style programming

The large number of cores and large global memory make GPGPUs interesting architectures for high performance computing applications. Indeed, GPGPUs have already been deployed for various such applications, including:

- Computational Geometry
- Computer Graphics
- Computer Vision
- Image Processing
- Linear Algebra
- Physics Simulation
- Ray Tracing
- Scientific Computing
- Signal Processing

In this presentation, we discuss fundamental parallel algorithm techniques for GPGPUs in comparison to the Cell processor. In particular, we review the impact of the SIMD control for GPGPUs on the parallel algorithms best suited for GPGPUs. We will look in particular at fundamental problems such as sorting, prefix sum, matrix operations, etc. Recently, a variety of new sorting methods for GPGPUs have been introduced. Many are based on well known SIMD paradigms for sorting such as bitonic sort (see Figure 3). Others are adaptations of well-known sorting algorithms such as merge sort, quick sort, radix sort, etc. In some cases, they use a combination of these sorting methods. We discuss time complexity and
communication complexity of these sorting methods and general lessons learned from this for efficient GPGPU algorithm design.

Figure 1: Layout of a Nvidia GeForce 8 GPU[1].

Figure 2: The architecture of a Cell processor with 1 PPE and 8 SPEs[2].

Figure 3: Phases of bitonic merging process for sorting 8 numbers
References


