CSCI 6360: Parallel Computing Lecture Summary - 8

Anirban Das (dasa2@rpi.edu)

February 9, 2018

Summary on the Archer/CT Dose Modeling Simulation System

According to the paper 'Heterogenous Concurrent Execution of Monte Carlo Photon Transport on CPU, GPU and MIC' by Wolfe et. al. ARCHER is an application for computing radiation dose for XRay CT imaging. The goal of the system is to take full advantage of available CPU, GPU and MIC simultaneously to boost performance of the simulation of whole body phantom of patients.

The basic underlay is a heterogenous computing framework that makes use of CPU, as well as GPU and MICs. CPU has been the usual choice of computing hardware till now , but with the advent of GPU, with their high compute density (large number of cores) and deep pipelines are inherently built for parallel operations. GPUs are very efficient for tasks where most of the threads are performing almost same instructions (SIMT) like in graphical processing tasks, but cannot handle complex logic at the same time. MIC on the other hand is a sweet spot among the two. They can handle both complex task logic and massive parallel operations at the same time, but have medium compute density (medium number of cores per system).

Archer brings together all three platforms under a simulation tool. The hurdle is the fact that the Monte Carlo simulation for the random walks, where each particle moves through millions of the patient body voxels can be arbitrary, and hence the conditional logic becomes extremely complex due to the stochasticity . This makes designing SIMD systems for this very challenging.

Each computing component of ARCHER is connected via PCIe bus. The main component is the CPU, the GPU and MICs though present , is only used for acceleration. One dedicated thread of the cpu, acts as the data and instruction bridge between the CPU and GPU. The 'self service' technique according to the authors, allows separate devices to divide the computational workload in a load balanced way. The system will keep on checking for computation, divide, compute until the simulation is complete.

The CPU component of ARCHER in various benchmark systems ranged from Intel Xeon to i5 desktop processors and the code is written in C/C++ using MPI for inter node parallelization and OpenMP for intra node parallelization. The GPU systems ranged from high end NVIDIA K40 to powerful desktop graphics cards like GTX Titan to low end GK20A. ARCHER$_{GPU}$Coding is done in C/C++/CUDA and Streams are used for intra node parallelization. The MIC (Intel Xeon 5110P etc.) specific code is almost the same as CPU's except that an additional -mmic flag is set during compilation to specify the linkage of the executable for the MIC architecture.

According to the paper, the execution model is as follows: the data is loaded from the file, a separate program written in OpenCL and CUDA detects the underlying architectures and their compute specifics. Based on the specifics, the system will choose single or all or combination of available architectures to optimally place data and execution load. Then separate specific executables will run on specific devices.

It is seen that this heteregenous compute architecture coupled with the load balancing system, achieves massive performance improvement compared to execution on homogenous architectures. Inspite of the fact that the auto vectorization of the C++ compiler is far from efficient, the

GPUs provided massive average improvement with the TITAN at 10.1X speed up on the base performance on i5 CPU. The MICs provided around 4X speedup. The composite systems including combinations of the MICs , CPU and GPU gave impressive 12-44X speedup. Thus ARCHER is able to harvest the heterogenous parallel computational power to provide an almost real time analysis of simulation heavy jobs as described above.