Predicting the Performance of Machine Learning Algorithms running on Heterogeneous Computing Platforms

Accelerators such as Graphic Processing Units (GPU's) produce high fidelity images and work rapidly with huge datasets. They are useful to speed-up the performance of various unsupervised algorithms of machine learning. As clustering algorithms often have their applications in many fields like image processing and artificial intelligence, it is imperative to improve its performance by parallel implementation. Moreover, running applications with huge datasets as input requires expensive resources without a guarantee of performance improvement, so using modeling techniques to predict the performance can be a great asset. To address this issue, this work attempts to develop performance models for estimating the expected results before actual deployment of resources. Furthermore, we aim to provide time-scaled prediction (e.g., end-to-end execution times) of an application running on a given hardware platform.

We are investigating K-means clustering as the representative of unsupervised learning algorithms to process images with many pixels. Our clustering implementation reads image files with the color and position information for each pixel, and then uses the K-means clustering algorithm to predict the number of clusters by grouping pixels based on their color differentiation and x – y coordinates. Initially, the K-means clustering randomly chooses a set of pixels as cluster centroids, then it computes new clusters in parallel to update the new centroids till it converges. We consider two types of processing elements: CPU and a heterogeneous system containing both CPU and GPU. We also run on multiple processors each with multiple threads. The comparison is made between the sequential version in CPU C code, and the heterogeneous versions implemented using OpenMP, MPI and CUDA. We use several sample images with different features of varying objects, image sizes and resolutions to evaluate the performance of all these implementations.

Our next step proceeds towards the accurate prediction of quantified expected improvement in performance of these algorithms on heterogeneous platform of CPU and GPU. We built the heterogeneous versions of the Computed Tomography (CT) Image Reconstruction application running on different computer platforms, which does not have looping behavior, and build a performance prediction model to represent it. By successfully capturing the characteristics of heterogeneous processing elements, our model predicts individual processing times for each stage (e.g., image weighting, filtering and reconstruction) in CT image reconstruction. Specifically, we built an underlying Markov model to predict the performance using low level thread activity for its execution. The states in which one thread can exists at any given time is either active or passive. The model estimates probabilities for jumping between these two states or staying in the same state. These estimates the total time that stage may take. We further extend our model to support the prediction of an application running with a large number of threads. By comparing the predicted results with the actual ones, we found that our Markov-based prediction models are able to accurately estimate the execution times for each stage as well as the entire CT image reconstruction process. In the future we plan to apply our performance prediction to more sophisticated machine learning algorithms including Gaussian Mixture Models and Latent Dirichlet Allocation. We are working to model the K-means application and plan to discuss our results at the workshop.