Brian Rodriguez
U0853593

Quiz 7

Saeed Lecture:

TAU Performance System software is a portable profiling and tracing tool kit that analyzes parallel programs written in a variety of languages. However, TAU is not as intuitive as one might think. In Saeed's lecture, he mentioned that it took him a year to fully realize the TAU architecture. Luckily for Saeed, TAU commander is a tool kit that integrates the TAU software with a more intuitive user interface. Essentially, TAU commander makes TAU more productive so that developers creating parallel programs can utilize the power profiling tools that TAU offers.

During the lecture Saeed showed-off the powerful tools that TAU has to offer. He ran a simple MPI program that printed "Hello from Thread …" where ellipse is the thread number. The program may not have been a meaningful program but TAU commander analyzed the program and showed where the program was spending the most time and in this case MPI_bcast was the call of most importance. The data was beautifully displayed on charts and easily readable. TAU, gathers statistical data during the programs life-span but also adds a considerable amount of overhead to the program as demonstrated in class.

It was speculated that TAU commander adds a 5x overhead for a parallel program wanting to use its profiling tools. On a smaller scale a parallel program running for 5 seconds would have to run for 25 seconds, this is still quite a miniscule increase but on a larger scale for a program running for a day it would actually take 5 days to run. There are pros and cons to using the TAU commander but in either case this tool is still a power visual profiler.

SLURM scripts are a way of posting jobs/tasks to high-performance computing clusters that contain hundreds of nodes with multiple GPUs and CPUs. Saeed demonstrated the different types of configurations options you can post for a task; the number of compute nodes that you want the cluster to use when running your program, GPU and CPU options, the amount of time that you wish the program to run before terminating, etc.

From personal experience, SLURM scripts are easy to configure but take some practice to get the compute node to-do everything that you ask of it. By this I mean that you must explicitly tell the SLURM script all of the tasks you want it to run in sequential order. The only thing a compute node does it run your program and spit out some data depending on your configurations, after that the node is essentially a transparent worker that does not know what to do other than compute.

Working with SLURM scripts sounds like a daunting task at first and is one but once you know how to navigate the inner workings of SLURM then you can make the compute node do a lot of really neat things like notify you when the job is done by sending you an email, return profiling data using tools like TAU, and a number of other really cool configurations.

Vinu Lecture:

K-means clustering is a way of taking data points with some sort of N-Dimensionality and finding the centroid of a given cluster. When you are given data with N-dimensionality, the objective of k-means is to take this data, and partition it into k clusters, where the centroid of these clusters is the mean of all data points that were assigned to the centroids cluster. Moreover, centroids are randomly generated or randomly selected from the data set that we provide to the algorithm.

K-means is an iterative algorithm that consists of two steps:
Step 1 Assignment Step: This step assigns a data point to the nearest centroid that was randomly assigned or generated from the data given. The nearest centroid is defined by the squared Euclidean distance between a centroid and a point.

Step 2 Centroid update: This step serves to update the centroids of the clusters. This is done by taking the mean of all the data points assigned to a centroids cluster.

This algorithm performs these two steps iteratively until either no data points change the clusters, the some of the distances computed is minimized, or some max number of iterations has been reached. K-means is said to converge to a result always. Although, it is notable that the result may not always be the most optimized results. It is advised to rerun this algorithm multiple times to compare each result.

K is the number of clusters to be computed by the algorithm. K is an arbitrary number that must be optimized and chosen wisely given any dataset. In practice, K is a number that must be chosen and compared against other values of K to find the most optimal K value for the dataset.

Vinu gave us a tutorial on writing our first GPU codes in C++. I personally, have never wrote GPU code but I have seen a similar syntax in a lot of source code. I now know that when I see __global__ before a function, it means that this function is useable by the GPU. It was interesting to see how fast GPU code can be written but I imagine production grade GPU code is a lot more complex and requires a lot more algorithmic complexity to create.

We also wrote a squaring function that is used by the GPU. We took the number of Block Units and used the block index to be used as an index in the array and squared the value stored inside. It was interesting to think that if the cost of doing the computation on a CPU is 2 nanoseconds and the cost on a GPU is 10 nanoseconds, then the GPU would only take 10 nanoseconds and the CPU would take 128 nanoseconds. Since the GPU is doing the operations in parallel, it would have to do 64 squares but all the operations would be done in parallel so it would only take 10 nanoseconds. In the CPU, the operations are done sequentially; even if the operation takes 2 nanoseconds the overall time would still be 128 nanoseconds.