Performance characterization of CPU and GPU centric workloads

Wesley Lee and Sachin Mehta

May 2, 2016

Data and Desires

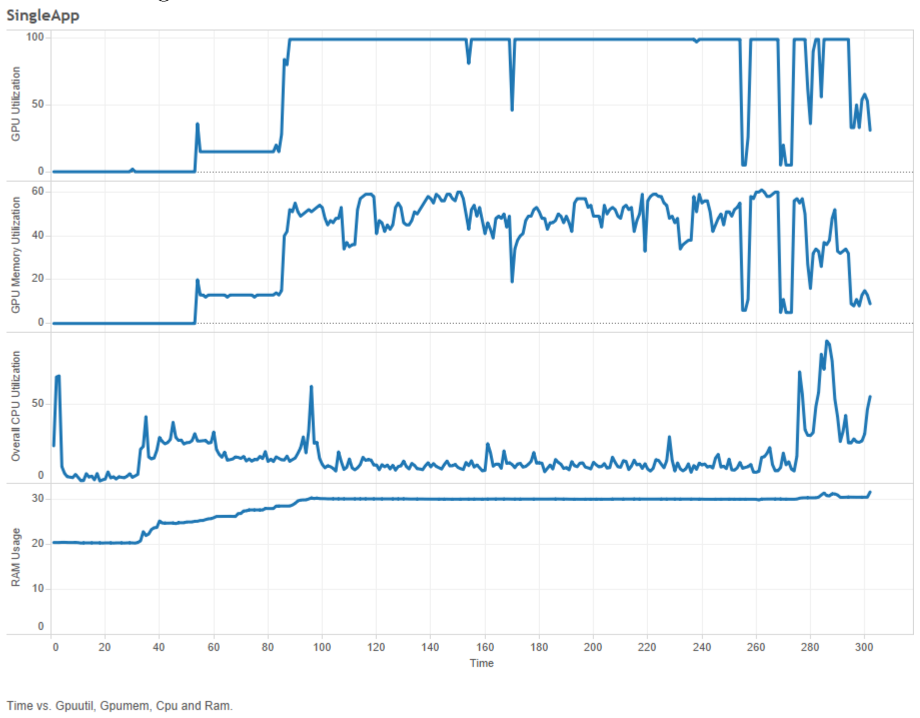
Our data consists of CPU and GPU performance metrics over time for three distinct applications. Examining these metrics can help us help us understand the overall behavior of application (CPU centric or GPU centric workload). For any given application, one might be interested in, among others, performance during initialization, identifying computer bottlenecks, and utilization of the various computer cores (i.e. single vs multi-threading). Note the three different applications run for different amounts of time and are generally incomparable. We hope the differential behavior of the applications yield a better idea of the capabilities of our visualization.

The intended audience for our visualization are computer programmers and design engineers who would be able to use the visualization to examine the performance of applications as well as identify bottlenecks of hardware.

Exploring the Data

Below, we have plotted time series of the major performance metrics for a single application.

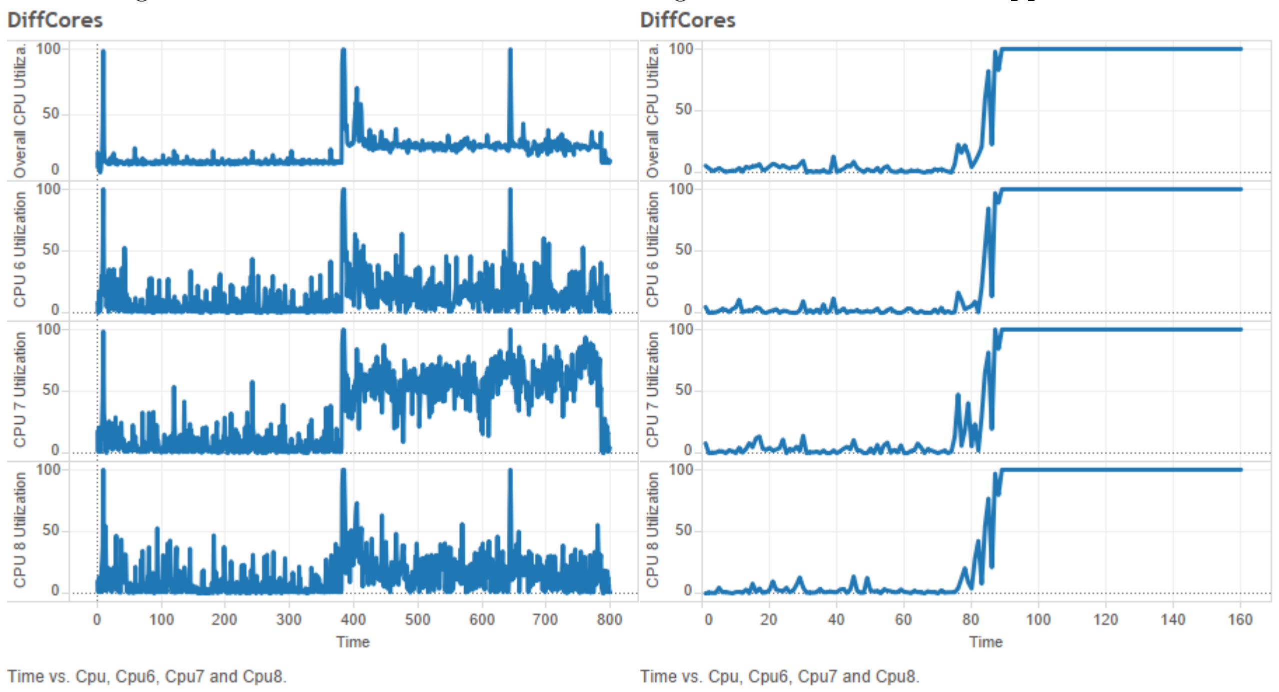
Figure 1: Performance of Heaven Benchmark 4.0



The data is characterized by a few notable properties. While all four variables are utilization percentages and thus must lie between 0 and 100, the ranges of the variables can be very different (i.e. RAM Usage mostly lying between 20-30% while GPU Utilization ranging from 0-100%). The runtime of an application is characterized by several phases of different behavior, and users may wish to examine phases separately.

Furthermore, sometimes examining the CPU behavior on different cores may reveal details about an application that one cannot see from the major performance metrics.

Figure 2: Per core CPU utilization on single- vs multi- threaded applications



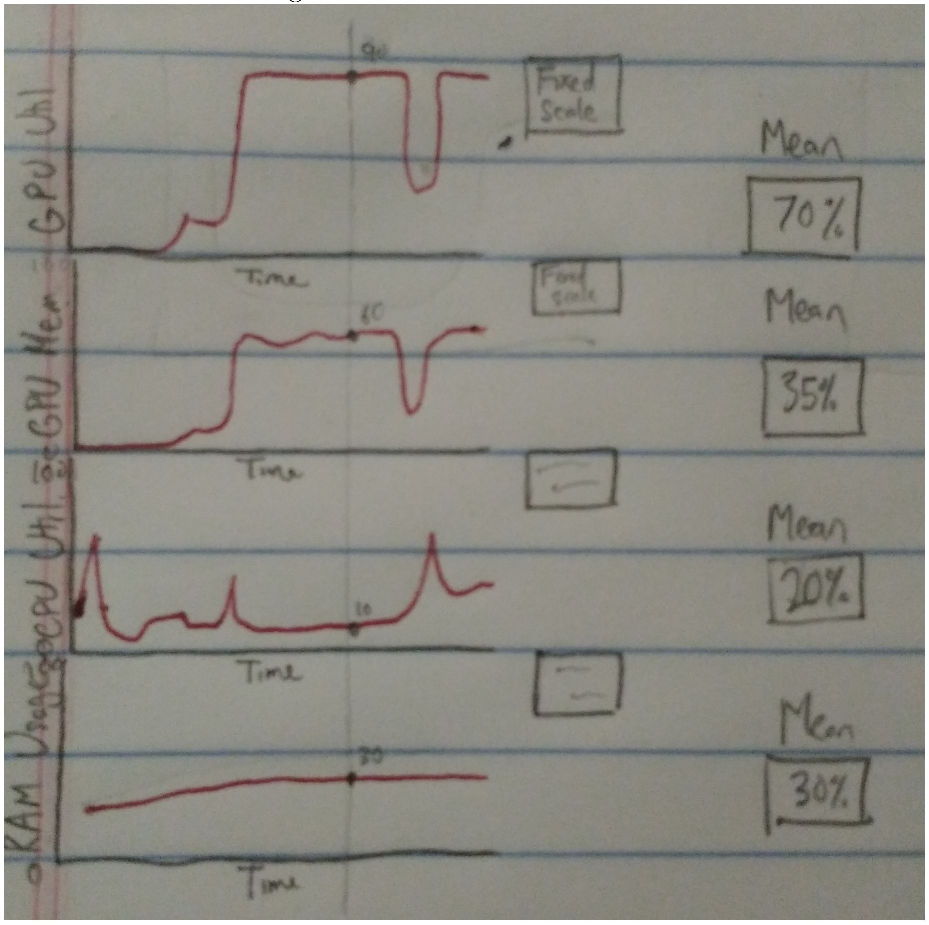
Namely, in multi-threaded applications (such as that on the right) one would expect similar utilization rates across multiple cores, while in single-threaded applications (such as that on the left) one would expect to run the application on one core, but CPU scheduler might switch the application from one core to other core giving different utilization rates across multiple cores (Please note that for single-threaded applications, the scheduler runs the application on 1 core at any given instant (but switches very often)).

Storyboard

We have two competing objectives in our visualization: First, we would like a fair amount of expressiveness. For any given application, we will want the ability to compare and contrast behavior of various performance metrics and examine performance on sub-intervals of the dataset. On the other hand, making all of this information at once may be overwhelming, and thus we have a simultaneous desire for simplicity.

We decided to have two views for each application: a basic view and a more detailed view.

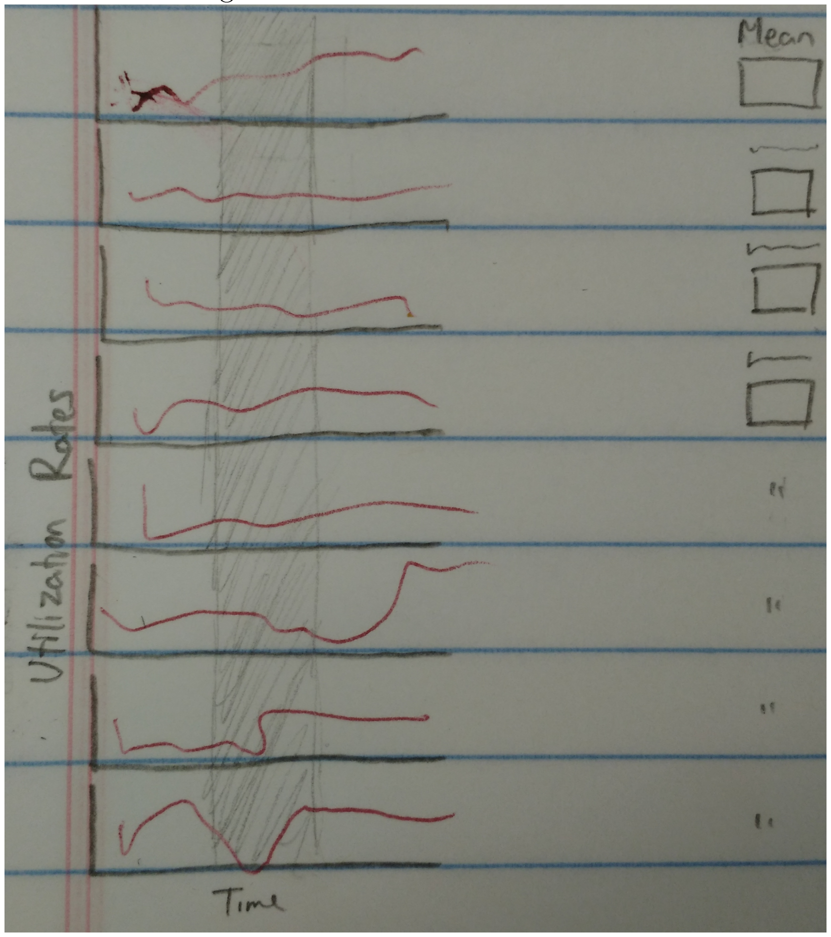
Figure 3: Sketch of basic view



The basic view will feature time series for the four major utilization percentages, along with their corresponding means. This would allow for a simple, quick overview of the data from which it would be easy to glean major trends in behavior. We hope to include two basic types of interactivity. First, there would be a scale button that allows one to choose between a fixed 0-100% scale (the default) and a variable scale that fits the data. The former scale is better for overall scope, while the latter scale allows one to pick up smaller changes in usage (for example changes in RAM usage in the figure above). Second, there would be a shared tooltip among the four graphs in the form of a vertical line that would provide the numeric values at any given time. This would be useful for lining up the behavior among the performance metrics.

If a user wanted to examine an application more closely, they would be able to bring up a detailed view.

Figure 4: Sketch of detailed view



This would feature time series for all performance metrics, including utilization percentages for individual CPU cores (which are not in the basic view). The view would be similar in the basic view in that each performance metric is represented by a graph and a mean, but would have different choices in interactivity. First, we will remove the ability to scale each of the graphs separately in order to keep the utilization percentages on the same scale (and thus comparable). Second, we will modify the tooltip from before to allow for the selection of a subset of the run time. This would cause the graphs to “filter and zoom-in” on the selected time period and update the means appropriately. Enabling this behavior in the detailed view but not the basic view is somewhat natural since we don’t expect the casual viewer to be interested in (or necessarily understand) finer details while experts would be. Such a view can allow for certain insights that are otherwise hard to see. For instance, if we look at the Cinebench single core application, we expect it to run on single core or two threads. However, the CPU scheduler switches the job from one core to other automatically at a very high rate. This might give a false insight that application is running on all the cores. However, on zooming in the application details, we can see evidence that the application is rapidly switching between cores (even on a per second basis). Unfortunately, our sampling rate is very low (1s, comparing to 16ms industry standard), so zooming-in is not quite as effective as it could be. Collecting the data at higher sampling rates would allow for even finer level details.