Anthony Monaco

Gavin Whelan

P434 – Project 2

Report for Project 2

During this project, we tried to work as much together as possible to allow for the free exchange of ideas. This being said, we had accomplished most of this project on Gavin’s computer. This allowed us to work through implementing the new hadoop features and finding bugs quicker. Even though we did work together, we did do some tasks separately. Gavin implemented the Hadoop features and helped connect our code to the future-grid. Anthony worked on finding information about serialized code in java, as well as the write up.

Throughout this project, we were allowed some bit of freedom in implementing our code the way we wished it to work while staying within the Hadoop file structure. This gave us the ability to make several important design decisions that resulted in many different strengths and weaknesses in our code.

Within the Hadoop map reduce implementation, we were allowed the freedom to organize both file input and outputs. For our input file structure, we had to organize a number of arguments that were essential to getting out code to run. These arguments were as follows: Image id, image width and height, camera position for both the x and the y axis, camera angle for both the x and y axis, sub view x and y axis, sub view width and height, and the URL that is hosting the image to be ray traced. This is used by the map function to correctly ray trace a specific segment of the whole image. For the output of the map function, we are using a PPM file to encode the ray traced image that is passed to the reduce function. Within the PPM file a number of attributes about the image is stored. They are as follows: a number that identifies the file type, the width of the image, the height of the image, the maximum color value that can be stored, and a number of height rows that are the different pixels of the image. We were originally going to store each ray traced sub view as a series of bytes organized in an ASCII representation of the RGB format. However, this caused us problems because at the end of the main function we would have to write a separate converter from our design into an image file format. This proved to be redundant code, which is now rendered obsolete with our introduction to PPM into our code.

Our overall impression about our implementation of Hadoop with our ray tracing code is that Hadoop is actually easier to implement the server side code than our previous project. This could be because we had already created most of the code used in our first project, but we feel that the Hadoop system is a great way for an amateurs first getting into virtual machines to implement simple code without needing to think about the file management, parallelization and other complex ideas needed for a complete virtualized machine. We feel that this project has given us the groundwork for future work with distributed systems, virtualized machines, and cloud computing.

To run our project on the future grid, a user needs to first run the included build file with the target “jar”. This will create a jar file to upload onto any similar Hadoop system. The next step for the user to take is to make sure that Hadoop is running and to create an input folder on the Hadoop file system. Next, the user needs to run the included shell script with the path to the input folder and a path to a future created output folder, as the arguments. This will run the Hadoop system with the jar file to create a PPM output file called rayTrace.PPM. Open this up in an image viewer application of the users choice.