Rodney Beede  
Neelam Agrawal

Yogesh Virkar

CSCI5576 – High Perf Sci Computing

Due: 2011-4-04

**Project Checkpoint – Parallel Zip Archive Password Recovery (P-ZAPR)**

*Rodney Beede*

I’ve written the framework code that handles arguments and calls methods for getting the range of passwords for each process. This has enabled my group members to write their code in separate cpp files. This allows testing of the brute force or dictionary code even when all parts haven’t been completed yet.

I’ve devised an algorithm for the brute force method that can determine the maximum number of possible passwords and divide it into a range for each process. In addition the algorithm can take an arbitrary password position in the range and factor it into an actual string password. I suspect that for actually iterating through the range this amount of factoring will be too slow so I will look into writing code to do a simple character increment on the current password string as well.

*Neelam Agrawal*

I have written the code for decrypting the zip file with a given password. The file is first read into a global variable. Then each block of 128 bits of the zip file is decrypted using AES-256. The AES-256 implementation is taken from <http://www.efgh.com/software/rijndael.htm> as given by Philip J. Erdelsky <http://alumnus.caltech.edu/~pje/> . Here we need to check if a decrypted file is actually what the original file was. I plan to do so by checking the frequency of letters in English language with the decrypted text. I will also consider the parallelization of the AES algorithm if time permits.

*Yogesh Virkar*

I have written the code for dictionary attack. The dictionary.cpp contains a function called initializePasswordGenerator\_dictionary which distributes the password file over all the processes. The distribution of words in the dictionary to check is balanced according to the number of processes with the possibility of an uneven distribution handled as well. Since this type of attack is limited by the passwords in the password file I will be looking into various ways in which the existing passwords can be transformed into different words (a simple example would be string reversal) so that the probability of success increases. The advantage would be that the size of the dictionary input data file won’t change since more password guesses would be made on the fly.

*Next Steps*

We will finish the coding this week and run tests on our local machines. Once we are satisfied that the code is in a usable state we will run several jobs on supercomputer clusters such as Frost, Trestles, and Janus. We intend on running on a varying number of process counts such as 128, 1024, and 2048. Once we have collected data we can then being to develop our report and do any follow up runs for further data collection.