**Discussion on results from Problem 5 openmp implementation**

This implementations and their result where done in a Hp-PavilionG7 with AMD A6 processor and operating Ubuntu 13.10. Graphs referenced on the discussion are in the “graphs\_implementation.pdf ” file and they are distributed comparing each of the schedule modes presented on the problem ( static , dynamic and guided). They are 4 figures for each schedule options 2 of them are using the nowait options and the other 2 are without the nowait option ( No-nowait). Between those 2 in nowait and No-nowait , are 2 different figures one for the Time Exec , or time It took to execute the whole process by thread and the graph is observing which Thread by Id where active during this execution and how many times it was activated. All the graphs are compared through what I call a “Times Delay” number which establish how many times the dummy function ran through process.

As seen in the figures A( Dynamic Imp.) , There where some difference between the implementation using nowait and those that doesnt use it. When active the nowait the division of work is more organized yet less effective and efficient than those with the no-nowait option. As seen in the figure A.1 and A.2 the Times the Thread where active where significantly bigger in comparison, this could suggest the implementation of the nowait even though it might work for longer scenarios of multiple execution of data in this example the execution was to unstable and poor in comparison to the other implementation where the average time was below the trough all the “Times Delay” point

Static implementations as presented in Figures B, present the implementation as showed in the figures B.3 and B.4 of the static schedule mode. This implementations in comparison to the Dynamic format the time of execution is a lot bigger but the most appreciative value here is the constant time of resolve. This implementation for the parallelization kept essentially the same average time through the execution of all nodes. This implementations where maximized by adding the use of nowait option but in essence the resulting time was around the same results. Additionally the Times Thread were active didnt change, meaning that the constance of the result will kept along in longer runs of data, and for system that has to wait between process this type of scheduling is perfect.

Last the Guided implementations as shown in the Figure C on the referenced file. This implementation bested both last implementations but specially the guided implementation. The time of execution present the characteristic behavior of this scheduling , the dividing the process in exponentially decreasing chunks of data that makes easier the implementation and runtime of the code making better times the longer the implementation was kept to execute. But the implementation even though can make a faster execution, the distribution of the jobs through a grid for example could be a problem since the implementation still requires some control of the result if needed to group back for more calculations yet the runtime execution was impeccable and of course in case of larger projects with constant more increasing data added to it it could be a benefit to have them all the time working more and more freely and because of that open to more process and work to execute