A Runtime optimization for OpenMP

Mihai Burcea and Michael J. Voss

# Introduction:

During compile time, it is not necessary that it contains the necessary information for optimization. Rum time optimization has become more critical for achieving the highest performance. This paper introduces to stomp: a specializing thread-library for OpenMP. The authors have stated that stOMP specializes the OpenMP parallel regions for frequently seen values. The stOMP system is built on the Omni OpenMP compiler and library, and provides a system for runtime optimization of parallel regions. The motivation for the runtime optimization of parallel regions is discussed in the paper. This paper refers to the relevant work done in the area by mentioning that how does the run time optimization is performed in object oriented languages such as Java. Then the authors went on discussing about the runtime optimizations in OpenMp applications. A run time optimization can occur by paralleling the program execution. The authors have shown results how this can be done by providing means for shared variables between threads. In addition to this the authors, discussed that Static optimization would be useless, as loop bounds and conditionals are unknown at compile-time, but if acquired during run-time much better optimization could be achieved.

# Conclusion:

How stOMP compiler works is that it creates separate files for each of the parallel region and takes care of the global and static variables. These separate files are used by the optimizer later at run time. As the optimizer creates several versions, each of the file can be used at run-time based on the requirements. This is the combination of compile time and run-time which further helps in a better optimization. The only disadvantage is that the values are changed during the run-time and a way has to be devised for updating the variables whenever there is a change. They used this technique on the EPCC scheduling micro-benchmark and the results seems to be promising.