**Abstract**

**Scheduling of DAG with data placement on multicore NUMA platform**

Directed acyclic graphs (DAG) based scheduling algorithms in traditional computing paradigms focus more on computational tasks and less than on the data placement during scheduling mapping. In the hierarchical memories context, this behavior will have negative impact on performance of system. With the introduction of non-uniform memory access (NUMA) multicore architectures, simultaneous optimization of computation and data placement in DAG based scheduling and mapping become the most challenging topics in scheduling related research. Such architecture exposes multilevel hierarchical memories with different characteristics which make the cost of the communication different in each level. From the other side, some interdependent tasks are communication intensive tasks that need to load and store their data frequently from the memory depending on data location, the performance of such system and its scheduler depends not only on its threads scheduling decision but also on its data locality decision (the mapping of the thread data).

In this thesis, we tackle the problem of the scheduling parallel applications described by DAG in target platforms by exploring case where not just computation and communication are considered in scheduling decision but also data placement on hierarchical memories platforms. The current scheduling and mapping policies try to reduce the overall penalties of the remote access by taking into account the data locality in the scheduling decisions. But most of the work done is for independent tasks context with static initialization. To do this for DAG application execution on the hierarchical platforms, it is necessary to find a way to combine both policies to get the most appropriate decision about when and where to schedule thread and where to place its data. In this work, we will:

1. Include the platform topology information (supplied by run-time environment at start time of application).

2. Use the application pattern (application structure provided by developers).

3. Divide the tasks set to a number of disjoint sets based on this structure and on task state.

4. Wide the tasks visibility by exploring a large horizon in order to gather more information about the state of current process running DAG.

5. Balance the load using distance based work stealing strategy at run-time.

These are the main ideas of the proposed scheduling and mapping policy of this work integrated as heuristics in this process to guide it and to reduce the impact of NUMA penalties on the completion total time of the DAG and preserve the system performance.

Keywords: DAG based scheduling, mapping, data locality, Multicore machines, NUMA architectures, Hierarchical platforms.