Scheduling DAG applications on multi-core processor packages architectures

Johnatan E. Pecero, Sebastien Varrete, and Pascal Bouvry

CSC/SnT University of Luxembourg
{firstname.lastname}@uni.lu

1 Introduction

In this paper, we deal with the problem of scheduling fine grain task graph applications on multi-core processor packages. We model this problem as a task scheduling problem with hierarchical communication delays. That is, we adopt the model in which the communications between cores on a processor package are much faster than those between cores belonging to different ones. This model incorporates the hierarchical nature of the communications using today's parallel computers, as shown by many PCs where cores are grouped into processor packages. Our main contribution is to provide a scheduling algorithm based on a genetic approach that considers this model when scheduling the fine grain applications. Simulation results demonstrate the effectiveness of the proposed approach when compared with a list scheduling algorithm for hierarchical communications [1].

We consider computing architectures with a set P of M homogeneous processor packages each package containing a set of m homogeneous cores. One example of this kind of architecture could be machines with large shared memory having two 2.5 GHz Intel Xeon E5000 processor package series of two or four cores each. As stated, the computational model we adopt here is *the hierarchical communication model* [2]. That is the communication delay ϵ_{ij} between two cores in the same package is negligible ($\epsilon_{ij} = 0$), while it takes c_{ij} units of time if the two cores belong to different packages. The rational is that in multi-core processors both computation units are integrated on the same die. Thus, communication between these computation units does not have to go outside the die, and hence is independent of the die pin overhead, making intra-die communication much faster than inter-die. The application is modeled by a precedence graph $G = (T, E)$, where the set of vertices T is a set of n tasks. Each edge $e = (t_i, t_j) \in E$ is a precedence constraint meaning that the results of task t_i must be available before t_j starts its execution. To every task t_j, there is an associated value p_j representing its processing time. In terms of classical scheduling problems, the application has to be scheduled on P, the set of mM processor packages. The l-th processor package is denoted P_{l*} whereas the k-th core of the l-th processor is denoted $P_{lk} \in P_{l*}$. A solution of the problem of scheduling an application on the multi-cores is to determine the processor and core assignment and an execution date for each task of the application respecting scheduling constraints. The associated optimization problem is to minimize C_{max}, the completion time (i.e. makespan) of the last executed task. In the 3-fields notation this problem is denoted by $P_M(P_m)|prec, p_j, C = (c_{ij}, 0)|C_{max}$ [2]. In terms of complexity, the problem is NP-hard as it contains NP-hard problems as particular cases [1].

2 Genetic-based Scheduling Algorithm on Multi-core (GSAM)

The solution we provide is based on a genetic algorithm. To encode a solution we use two arrays of integers M' and m' of length n. The array M' encodes the processor package, where the i-th location of the array corresponds to the task t_i and its content represents the processor package P_{l*} where task t_i is allocated. The array m' encodes the core P_{lk} where task t_i is executed on processor package P_{l*}. The initial population has been generated at random. The fitness function is the inverse of the makespan. To compute the makespan of any individual we implemented an algorithm based on the list scheduling principle. Informally, the principle is to schedule the application iteratively one time slot after the other on the assigned processor and core. GSAM is executed until a fixed number of iterations has been reached. We have implemented the proportionate selection scheme based on the roulette wheel principle. We used an elitist model where the global best individual is always kept and replace the worst individual in the new generation, the rest of the individuals are replaced by the new generation. We implemented single recombination operator; mutation randomly changes one task in the scheduling to a different processor package and core.

3 Experiments

We compare the performance of GSAM with a modified list scheduling algorithm [1], called *ETF* in the paper, that considers hierarchical communications. The list scheduling algorithm maintains a list of ready tasks that are greedily scheduled on available resources. To make a more accurate comparison we have decided to select at random the ready task to be executed. We have selected for our simulations task graphs from the *Standard Task Graph Set* [3]. *SPEC fpppp* composed of 334 tasks and 1196 edges, *sparse matrix solver* composed of 96 tasks and 128 edges, and *robot control application* with 88 tasks in size and 131 edges were adopted as targeted benchmark. For each graph, we have varied the *communication to computation cost ratio* (CCR). We have generated four CCRs (0.5, 1, 2, 5). The parameters of GSAM are: population size equal to 40, generation limit of 50, recombination probability of 0.85, mutation probability of 0.005. We used the makespan (in milliseconds) of the applications generated by the algorithms as the main performance measure. We have simulated the algorithms on 2 processor architectures composed of 2, 3, 4, and 6 cores. Figure 1 depicts the results for the simulations. In left we show average results for both algorithms on all the applications when the CCR increases. The figure in the right depicts results for architectures with different number of cores. In both cases GSAM produces better schedules on average than ETF. In our simulations, GSAM produces schedules 12.5% shorter than ETF.

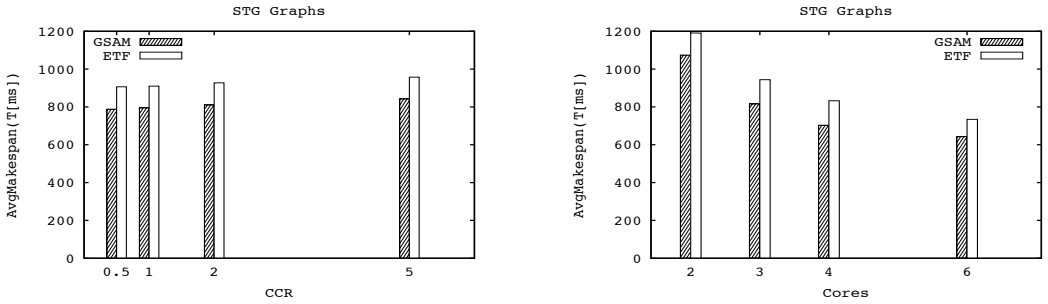

Fig. 1. Results for simulations

4 Concluding Remarks

In this work, we have provided a GA approach for scheduling DAG applications on multi-core processor packages. We have adopted the hierarchical model were communications between cores on a same package are negligible regarding communications between different packages. We compared the provided algorithm with a modified list scheduling algorithm taken from the literature. The simulation results emphasize the interest of GSAM.

We plan to extend this work, first we consider to investigate our algorithm in the context of power-aware scheduling on multi-core architectures. Our idea is first to load all the tasks and cores in the same package before distributing the load to another package, therefore putting the idle packages in a lower state, hence saving power. Furthermore, we are planning to consider memory contention issues. The idea is the opposite of the power-aware policy, that is, first to distribute the load between processor packages before distributing the load between cores on a same package.

5 Acknowledgements

This work is supported by the FNR Luxembourg through project Green-IT no. C09/IS/05 and the AFR Luxembourg through the grant no. PDR-08-010.

References

1. Blachot, F., Huard, G., Pecero, J. E., Saule, E., Trystram, D.: Scheduling instructions on hierarchical machines. IEEE IPDPS-PDSEC 2010, Atlanta, USA (2010)
2. Bampis, E., Giroudeau, R., König, J. C.: An approximation algorithm for the precedence constrained scheduling problem with hierarchical communications. Theor. Comput. Sci., 290(3): 1883–1895, 2003
3. Kasahara, H., Tobita, T., Matsuzawa, T., Sakaida, S.: Standard task graph set. Available: http://www.kasahara.elec.waseda.ac.jp/schedule/