Using the duplicaticity of a taskgraph to select a suitable multiprocessor scheduling strategy

Frode Eika Sandnes

Dept. Computer Science, Faculty of Engineering
Oslo University College, Cort Adelers gate 30
N-0254 Oslo, Norway, frodes@iu.hio.no

Abstract

Most multiprocessor taskgraph scheduling algorithms are variations of the priority list scheduling approach where interprocessor communication-overheads are ignored. An alternative is duplication-based scheduling, where tasks are duplicated on multiple processors to reduce inter-processor traffic. However, these strategies are more suitable for taskgraphs with a certain structure exhibiting a high degree of fan-out. Therefore, not all taskgraphs can be scheduled efficiently using duplication. In this paper a new and easily computed measure is introduced - the degree of duplicaticity. The degree of duplicaticity quantifies the suitability of applying a duplication-based strategy to a taskgraph, and it can thus be used as an indicator for deciding weather to employ a traditional list-scheduling technique or a duplication based technique.

1 Task graph scheduling

The NP hard multiprocessor static taskgraph scheduling problem have received sustained attention for several decades. It involves assigning the vertices, or tasks, of an acyclic directed graph onto a set of interconnected processing elements such that the makespan, or total computing time, is minimised. Most approaches belong to the family of priority-list scheduling algorithms, differentiated by the way in which task priorities are assigned. There is a large number of strategies that are easy to implement and that produce powerful results, for example demand scheduling, critical-path [1], highest level first known execution times (HLFET), shortest co-level first known execution times (SCEFT), graph-level/communication intensity, Dynamic Highest Level First/Most Immediate Successor First (DHLF/MISF), Depth First/Heuristics (DF/H), Depth First/Implicit Heuristic Search (DF/IHS) [2, 3, 4] and the famous and frequently cited Critical Path/Most Immediate Successor First (CP/MISF) [5] proposed by Kasahara and Narita. There are also more complex and harder-to-implement strategies that produce marginally better results, for example dynamic critical path scheduling [6]. More recently, researchers have employed modern stochastic search techniques, such as evolutionary strategies, to obtain better schedules. The most cited approach was proposed by Ahmad and Dhodhi who generated close-to-optimal schedules by combing the traditional CP/MISF approach with an evolutionary search strategy [7], and variations on the theme has been carried

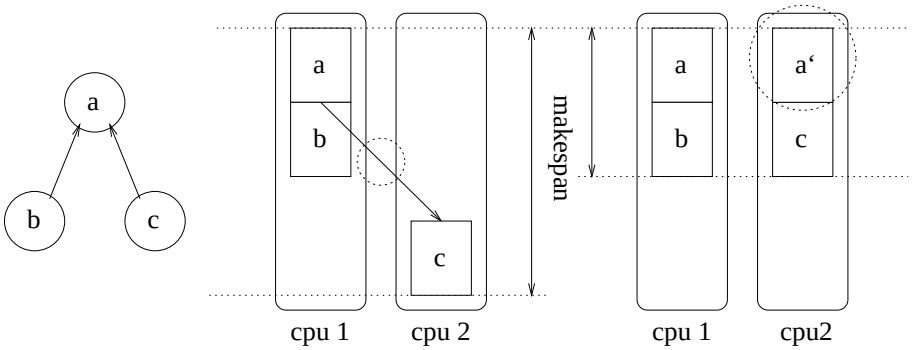

Figure 1: A three-node fan-out taskgraph, a schedule obtained using a priority-list strategy and a schedule obtained using task duplication.

out by the likes of Hou, Ansari and Ren [8], Sandnes and Megson [9, 10], Chamaret, Rebreyend and Sandnes [11] and Correa, Ferreira and Rebreyend [12]. One fundamental shortcoming haunting many of these strategies is that zero communication costs are assumed, i.e. that there is no penalty, or delay, associated with the exchange of information between processing nodes. Another shortcomming is that many approaches assume that the processing elements can operate without interruption, a problem discussed by Sinnen and Sousa [13]. In practice, the interprocessor traffic overhead is a major bottleneck in multiprocessor systems. The concept of duplication-based scheduling [14, 15] has emerged as a direct result of this difficulty.

1.1 Duplication-based scheduling

Duplication-based scheduling achieves short makespans by assigning the same task to several processors such that the results of the computation need not be transmitted to other processors. One assumes that the cost of sending the result is larger than the cost of the actual computation itself. Several researchers including Cretienne [16, 17] Park, Shirazi and Marquis [14, 15], Dabha and Agrawal [18], Munier and Hanen [19], Tsuchiya, Osada and Kikuno [20], Kwok and Ahmad [21], Bampis, Giroudeau and Konig [22], Sandnes and Megson [23], Manoharan [24], Park and Choe [25] and Lepere and Rapine [26] have incorporated task duplication into their scheduling algorithms. Several heuristics are proposed. In general, a task is placed on the same processor as its parent. Fan-out nodes are usually replicated. Park et al. [14] show that their duplication-based algorithm is optimal for fan-out trees.

Duplication-based scheduling can be illustrated by an example. Figure 1 shows a three-node fan-out taskgraph, a schedule obtained using a non-duplication-based strategy and one obtained using task duplication. The non-duplication-based schedule has the longest makespan due to the delay associated with the message sent from task a at cpu 1 to task c at cpu 2. The task duplicated schedule is optimal as its makespan is equal to the length of the critical path. The root node a is duplicated on the second processing element. It is impossible to obtain the optimal schedule for this taskgraph without task duplication in the presence of non-zero communication delays.

1.2 Taskgraph structure

Duplication-based scheduling have received moderate attention. First, duplication-based scheduling algorithms are complex and intrinsic compared to many of the non-duplication-based algorithms such as CP/MISF. Secondly, task duplication only benefit taskgraphs with a certain structure. Taskgraphs suitable for task duplication strategies contain a high degree of fan-out, i.e. many of the vertices must have more than one child. However, many real-world computations do not have such characteristics, as they contain a high degree of fan-in, which means that many of the graph the vertices have more than one parent. For example, arithmetic computations takes many arguments and computes one or a small set of results, such as computing the sum of a list of numbers. A sum takes the list of numbers as arguments and produces only one result, the sum. Taskgraphs with a high degree of fan-out conversely takes few arguments but produce many results. These observations are the motivation behind the measure of duplicaticity proposed in this paper. This measure quantifies the extent to which a given taskgraph can benefit from duplication-based scheduling in the presence of non-zero communication costs.

2 Taskgraph measures

Several measures have been proposed for evaluating how well a taskgraph can be scheduled. A precedence constrained task graph can be represented by a directed acyclic graph, $G = (V, E)$, containing a finite nonempty set of vertices, V, and a finite set of directed edges, E, connecting the vertices. The collection of vertices, $V = \{v_1, v_2, ..., v_n\}$, represents the set of n computational tasks to be executed and the directed edges, $E = \{e_{ij}\}$, define the precedence relations that exists between the tasks (e_{ij} denotes a directed edge from vertex v_i to v_j).

Jereb and Pipan [27] proposed measures for evaluating a task graph. Their measures are defined by two numbers, the degree of simultaneousness and the degree of connection. The degree of simultaneousness is defined as

$$DS(G) = \frac{n - \mathcal{H}(G)}{n} \tag{1}$$

and the degree of connection is defined as

$$DC(G) = \frac{|E|}{\frac{n}{2}(n-1)} \tag{2}$$

where n is the number of vertices in the task graph, $\mathcal{H}(G)$ is the length of the critical path of the task graph and $|E|$ is the number of edges in the task graph. The degree of simultaneousness is *zero* for chain-structured task graphs and *one* for task graphs with totally independent tasks. The degree of connection is *zero* if none of the nodes are connected, and *one* if all the nodes are connected. Jain and Rajaraman [28] criticised this model as being over-simplistic and proposed a more descriptive measurement by incorporating the width of the taskgraph levels. They define the degree of simultaneousness as $DS(G) = UF(G) \times DS(G)^{old}$, where DS^{old} is Jereb and Pipan's definition of DS. The uniformity factor $UF(G)$ is defined as

$$UF(G) = \frac{L_{avg}}{L_{max}} = \frac{n}{L_{max} \times \mathcal{H}(G)} \tag{3}$$

The value L_{avg} is the size of the average level and L_{max} is the size of the widest level in the task graph. The new degree of connection is defined as

$$DC(G)^{new} = \frac{(|E| - \mathcal{E}_{min})}{(\mathcal{E}_{max} - \mathcal{E}_{min})} \tag{4}$$

where

$$\mathcal{E}_{min} = \sum_{i=1}^{\mathcal{H}(G)-1} \sum_{j=1}^{\mathcal{H}(G)} L_i L_j \tag{5}$$

and $\mathcal{E}_{max} = n - L_1$. Jain and Rajaraman also define the degree of simultaneousness for a limited number of processors as:

$$DS(G,p) = \frac{n - \sum_{i=1}^{\mathcal{H}(G)} \left(\frac{L_i}{p}\right)}{p \times \sum_{i=1}^{\mathcal{H}(G)} \left(\frac{L_i}{p}\right)} \tag{6}$$

where p is the number of processors and L_i is the number of nodes at level i.

2.1 Degree of duplicaticity

Beneficial characteristics for a taskgraph, in the context of duplication-based scheduling, is a high degree of fan-out, or many vertices with several children. Thus, the *degree of fan out* $\mathcal{O}$ of a vertex v_i can be defined as

$$\mathcal{O}(v_i) = \begin{cases} \mathcal{C}(v_i) & ,\mathcal{C}(v_i) > 1 \\ 0 & ,\mathcal{C}(v_i) \leq 1 \end{cases} \tag{7}$$

where $\mathcal{C}(v_i)$ is the number of children at vertex v_i. Note that only quantities of two children or more are considered, as zero or one child does not contribute towards the degree of fan-out.

Disadvantageous taskgraph characteristics, in the context of duplication-based scheduling, is the degree of fan-in, or many vertices with several parents. Thus, the *degree of fan-in* $\mathcal{I}(v_i)$ at a vertex v_i can be defined as

$$\mathcal{I}(v_i) = \begin{cases} \mathcal{P}(v_i) & ,\mathcal{P}(v_i) > 1 \\ 0 & ,\mathcal{P}(v_i) \leq 1 \end{cases} \tag{8}$$

where $\mathcal{P}(v_i)$ is the number of parent at vertex v_i. The degree of duplicaticity $\mathcal{D}$ of graph G can therefore be defined as

$$\mathcal{D}(G) = \frac{1}{2}\left(\frac{1}{|E|}\sum_{I=1}^{n}[\mathcal{O}(v_i) - \mathcal{I}(v_i)] + 1\right) \tag{9}$$

A degree of duplicaticity $\mathcal{D} \in [0,1]$ close to zero indicates that a graph cannot be scheduled successfully using a duplication-based strategy. All balanced fan-in trees have a duplicaticity of zero. Similarly, a duplicaticity close to 1 indicates that a graph will benefit from being scheduled using a duplication-based strategy rather than a non-duplication-based approach. All balanced fan-out trees have a duplicaticity of 1. A chain structured task graph has a duplicaticity of 0.5 as it has no fan-in or fan-out vertices The complexity of computing the duplicaticity $\mathcal{D}$ for a graph G is linear $O(n)$ in time, where n is the number of vertices in the graph.

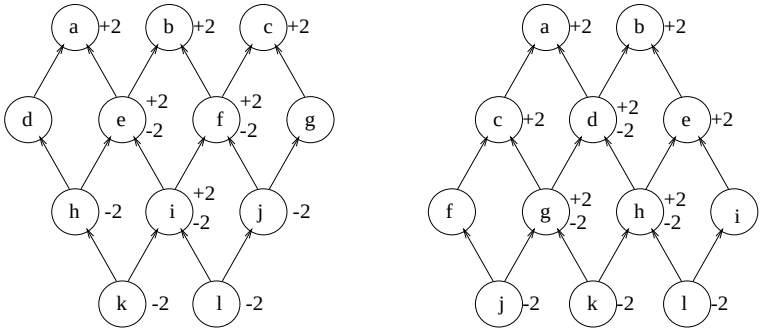

Figure 2: Two task graphs with duplicaticity of 0.4375 and 0.5625 respectively.

2.2 Examples

The duplicaticity of the systolic structure on the left in Figure 2 is 0.4375. Tasks a, b, c, e, f and i are all fan-out nodes with two children yielding a total of 12 fan-out edges. Similarly, tasks e, f, h, i, j, k and l are fan-in nodes yielding 14 fan-in edges. Notice that several nodes are both fan-in and fan-out nodes. This structure is unlikely to benefit from a duplication-based scheduling approach. However, the taskgraph on the right in Figure 2 has a duplicaticity of 0.5625, and a duplication-based technique could be the preferred choice.

3 Results

Table 1 shows the duplicaticity for the Stanford robot arm, the elbow manipulator [5] and a selection of common algorithms [12]. The first column lists the name of the taskgraph, the second column lists the makespans obtained using a scalable duplication heuristic [14] on a fully connected 16-element processor network. A scalable heuristic is able to control the number of processors used while a non-scalable heuristic requires an unrestricted number of processors. The third column shows the makespans obtained combining the scalable duplication heuristic with stochastic search [23], the fourth and the fifth columns show the makespans obtained using list-scheduling techniques, namely cp-misf [29] and a strategy based on stochastic search [7] modified for networks with non-zero communication costs. The sixth column shows the difference between the makespans obtained using stochastic list-based and stochastic duplication-based strategies in percentage. The final column lists the computed duplicaticity of the graphs.

4 Discussion

Generally, a duplication based scheduling approach performs well when there are no restrictions on the number of processors available, while the schedules shown in the table are all restricted to 16 processors. This limitation is imposed in order to compare the list-based and the duplication-based strategies. Consequently, the results obtained with the duplication based scheduling approaches are worse than the ones obtained using the list-scheduling approaches, with the exception of the cstandford graph which results in 25% shorter makespan using the duplication based strategy. The duplicaticity for most of the graphs is 0.5 or close to this – making it hard to decide weather to use a duplication- or list-based strategy. However, the prolog graph has a duplicaticity of 0.62 indicating

problem	List-scheduling		Duplication			
	one-pass	stoch.	one-pass	stoch.	% diff.	dup.
cstanford	859	858	1358	636	-25%	0.50
celbow	7170	6030	15705	7830	30%	0.51
bellford	844	864	6884	5880	580%	0.48
diamond1	1284	1380	8516	1728	25%	0.50
diamond2	2416	2368	6928	3766	59%	0.50
diamond3	1996	2228	17004	14357	544%	0.50
diamond4	1478	1692	10670	8028	374%	0.49
divconq	1615	1201	7276	3692	307%	0.50
fft-2	4851	4059	20064	12694	212%	0.50
fft2-b	477	399	1965	1243	236%	0.50
gauss-m	3499	3102	28107	22420	622%	0.27
iterative	202	202	560	455	125%	0.50
prolog	72	56	332	96	71%	0.62

Table 1: Results obtained with the DFRN heuristic

a high the degree of fan-out and thus a graph suitable for duplication-based scheduling. This is indeed true as the makespan of 96 using the stochastic duplication-based strategy is close to the makespan of 56 obtained using the stochastic list-based strategy. The effect of the duplication based strategy on the prolog graph would probably be stronger with a larger task to processor ratio (currently only 214/16). The duplicaticity of gauss-m is 0.27 suggesting a low degree of fan-out and thus recommending a list-based approach. This is evident as the makespan for the gauss-m graph using duplication-based strategy (22420) is 622%, or nearly an order of magnitude, longer than the makespan obtained suing a list-based strategy (3102). Similar effects can be seen for the other graphs with a duplicaticity less than 0.5, namely the bellford graph with a duplicaticity of 0.46 and a duplication-penalty of 580%, and the diamond4 graph with a duplicaticity of 0.49 and a duplication penalty of 374%.

A shortcoming of the small dataset presented in this paper is that they represent highly regular structures, and the study would have been more complete with a suite of irregular graphs. Further, the study is limited to 16 processors. Since the performance of the duplication based scheduling heuristics is linked to the number of available processors, it makes sense to devote some attention the number of processors.

Finally, the degree of duplicaticity should not be used as a the sole decision-making criterion, but in conjunction with other parameters such the number of processors and other graph topology characteristics. Duplicaticity is thus a partial indicator indicating how suited a graph is to a duplication based scheduling strategy. The degree of duplicaticity measures the graph and not the algorithm used to schedule the graphs.

5 Summary

A measure for evaluating the duplicaticity of a taskgraph is proposed. The degree of duplicaticity can be used as a partial indicator to predict weather a duplication-based strategy will reduce the makespan when scheduling the taskgraph or weather a conventional list-based scheduling approach should be employed. The complexity of computing

the degree of duplicaticity is linear in time.

Acknowledgement

I would like to thank G. M. Megson at the Departement of Computer Science at The University of Reading, England for useful comments and discussions.

References

[1] W. H. Kohler, A Preliminary Evaluation of the Critical Path Method for Scheduling Tasks on Multiprocessor Systems, IEEE Transactions on Computers, pp 1235-1238, 1975.

[2] J. Y. S. Luh, C. S. Lin, Scheduling of Parallel Computation for a Computer-Controlled Mechanical Manipulator, IEEE Transactions on Systems, Man and Cybernetics, vol SMC-12, no 2, pp214-234, 1982.

[3] C. S. G. Lee, C. L. chen, Efficient Mapping Algorithm for Scheduling Robot Inverse Dynamics Computation on a Multiprocessor System, IEEE Transactions on Systems, Man and Cybernetics, vol 20, no 3, pp582-595, 1990.

[4] C. L. Chen, C. S. G. Lee, E. S. H. Hou, Efficient scheduling algorithm for robot inverse dynamics computation on a Multiprocessor System, IEEE Transactions on Systems, Man and Cybernetics, vol 18, no 5, pp729-734, 1988.

[5] H. Kasahara, S. Narita, Practical Multiprocessor Scheduling Algorithms for Efficient Parallel Processing, IEEE Transactions on Computers, vol C-33, no 11, pp1023-1029, 1984.

[6] Y.-K. Kwok, I. Ahmad, Dynamic Critical Path Scheduling: An Effective Technique for Allocating Task Graphs to Multiprocessors, IEEE transactions on Parallel and Distributed Processing, vol. 7, no 5, pp506-521, 1996.

[7] I. Ahmad, M. K. Dhodhi, Multiprocessor Scheduling in a Genetic Paradigm, Parallel Computing, vol 22, pp395-406, 1996.

[8] E. S. H. Hou, N. Ansari, H. Ren, A Genetic Algorithm for Multiprocessor Scheduling, IEEE Transactions on Parallel and Distributed Systems, vol 5, no 2, pp113-120, 1994.

[9] F. E. Sandnes, G. M. Megson, A Hybrid Genetic Algorithm Applied to Automatic Parallel Controller Code Generation, IEEE Proceedings of the 8'th Euromicro Workshop on Real-Time Systems, pp70-75, 1996.

[10] F. E. Sandnes, G. M. Megson, Improved Static Multiprocessor Scheduling using Cyclic Task Graphs: A Genetic Approach, PARALLEL COMPUTING: Fundamentals, Applications and New Directions, North-Holland, no. 12, 703-710, 1998.

[11] B. Chamaret, P. Rebreyend, F. E. Sandnes, Scheduling problems: A comparison of hybrid genetic algorithms, Proceedings of the 2nd IASTED International Conference on Parallel and Distributed Computing and Networks, Brisbane, Australia, pp 210-213, December, 1998.

[12] R. C. Correa, A. Ferreira, P. Rebreyend, Scheduling multiprocessor tasks with genetic algorithms, IEEE Transactions on Parallel and Distributed Systems, vol. 10, no. 8, pp825-837, 1999.

[13] O. Sinnen, L. Sousa, Experimental evaluation of task scheduling accuracy, Proceedings of the International Conference on Parallel and Distributed Computing, Application and Technologies, PDCAT2002, Kanazawa, Japan, pp378-383, 2002.

[14] G.-L. Park, B. Shirazi, J. Marquis, DFRN: A New Approach for Duplication Based Scheduling for Distributed Memory Multiprocessor Systems, IEEE Proceedings of IPPS'97, pp157-166, 1997.

[15] G.-L. Park, B. Shirazi, J. Marquis, A Scalable Scheduling Algorithm for Parallel and Distributed Systems, EuroPDS'97, 1997.

[16] P. Cretienne, A polynomial time to optimally schedule tasks over an ideal distributed system under tree like precedence constraints, European Journal of operational research, vol. 2, pp225-230, 1989.

[17] J.-Y. Colin, P. Cretienne, C. p. m. scheduling with small communication delays and task duplication, Operations Research, vol. 39, pp681-684, 1991.

[18] S. Darbha, D. Agrawal, A task duplication based scalable scheduling algorithm for distributed memory systems, Journal of Parallel and Distributed Computing, vol. 46, no. 1, pp, pp15-27, 1997.

[19] A. Munier, C. Hanen, Using duplication for scheduling unitary task onto m processors with communication delays, Theoretical Computer Science, vol. 178, pp119-127, 1997.

[20] T. Tsuchiya, T. Osada, T. Kikuno, Genetics-based multiprocessor scheduling using task duplication, Microprocessors and Microsystems, vol. 22, no. 3-4, pp197-207, 1998.

[21] Y.-K. Kwok, I. Ahmad, On Exploiting Task Duplication in Parallel Program Scheduling, IEEE Transactions on Parallel and Distributed Systems, vol. 9, no. 9, pp872-892, 1998.

[22] J. K. E. Bampis, R. Giroudeau, Using duplication for the multiprocessor scheduling problem with hierarchical communications, in: EURO-PAR'99: PARALLEL PROCESSING, Lecture notes in Computer Science, no. 1685, pp. 369-372, 1999.

[23] F. E. Sandnes, G. M. Megson, An evolutionary approach to static taskgraph scheduling with task duplication for minimised interprocessor traffic, Proceedings of the PDCAT'2001 International Conference on Parallel and Distributed Computing Applications and Techniques, Taipei, Taiwan, 9-11 July, pp 101-108, 2001.

[24] S. Manoharan, Effect of task duplication on the assignment of dependency graphs, Parallel Computing, vol. 27, no. 3, pp257-268, 2001.

[25] C. Park, T.Y.Choe, An optimal scheduling algorithm based on task duplication, IEEE Transactions on Computers, vol. 51, no. 4, pp444-448, 2002.

[26] L. Repere, C. Rapine, An asymptotic o(lnr/lnlnr)-approximation algorithm for the scheduling algorithm with duplication on large communication delay graphs, in: 19th International Symposium on Theoretical aspects of Computer Science, In-STACTS, Antibes, France, 2002.

[27] B. Jereb, L. Pipan, Measuring Parallelism in algorithms, Microprosessing and Microprogramming, The Euromicro Journal, vol 34, pp49-52, 1992.

[28] K. K. Jain, V. Rajaraman, Parallelism measures of task graphs for multiprocessors, Microprocessing and Microprogramming, vol 40, pp249-259, 1994.

[29] H. Kasahara, S. Narita, Parallel Processing of Robot-Arm Control Computation on a Multiprocessor System, IEEE Journal of Robotics and Automation, vol RA-1, no 2, pp104-113, 1985.