Scheduling nested loops with the least number of processors

T. Andronikos, M. Kalathas, F.M. Ciorba, P. Theodoropoulos, G. Papakonstantinou

Computing Systems Laboratory
Department of Electrical and Computer Engineering
National Technical University of Athens
Zografou Campus, 15773, Athens, Greece

Abstract

Frequently the most computationally intensive part of a program is attributed to the nested loops it contains. It is therefore of interest to try to parallelize the nested loops in order to reduce the overall computation time. A special category of FOR(DO) nested loops are the uniform dependence loops, which yield efficient parallelization techniques and are the focus of this paper. The primary goals in this area of research are achieving the optimal parallel time and minimizing the number of processing elements required for the execution of the parallel program. In this paper we present a new dynamic lower bound on the number of processors needed for scheduling and we use a decision algorithm to verify that all uniform dependence loops can achieve this bound.

Keywords: UET model, lower bound, upper bound, optimal execution time, scheduling.

1 Introduction

Experience has shown that all programs that require a significant amount of computing time spend most of it executing one or more loops. For a large class of programs these loops are nested FOR loops, one of the code structures that describe a sequence of computations, whose size can be exponential relative to the code size. In this sense, a lot of research has been done over the past years concerning the parallelization of these loops. Practical results were obtained for the specific category of nested FOR loops containing uniform data dependencies, i.e., the values of the dependence vectors are constant throughout the index space.

When parallelizing loops for shared memory parallel computers, the two tasks that need to be done are to specify when the different computations are carried out (computation scheduling) and where they are carried out (computation mapping). We focus on computation scheduling, which implies finding a schedule of computations in time, while preserving the data dependencies of the initial lexicographic loop order. Many methods have focused on accomplishing the optimal time scheduling, the first of which is the hyperplane method [8].

The computations are organized into well defined distinct groups, called wavefronts or hyperplanes, using a linear transformation. The hyperplane method is applicable to all loops with uniform dependencies and all the points belonging to the same hyperplane can be concurrently executed. Darte [3] proved that this method is nearly optimal. Moldovan, Shang, Darte and others applied the hyperplane method to find a linear optimal execution schedule using diophantine equations [10], linear programming in subspaces [13] and integer programming [3]. The problem of finding a hypersurface that results in the minimum makespan was solved more efficiently in [12]. All these approaches consider unit execution time for each iteration and zero communication for each communication step (UET model). Each loop iteration requires exactly the same processing time and there is no communication penalty for the transmission of data among the processors. Previous attempts of scheduling uniform dependence loops include the free scheduling method introduced in [7]. Our goal is to schedule uniform dependence loops so that the optimal parallel processing time is achieved using the minimum number of processors.

The problem of scheduling uniform dependence loops is a very special case of scheduling Directed Acyclic Graphs (DAGs). The general DAG multiprocessor scheduling with precedence constraints is known to be NP-complete [6, 14] even when the number of processors is unbounded [11]. Many researchers have tackled the special cases of DAG scheduling [5, 7] hoping to come up with efficient polynomial algorithms. The computational scheduling problem in our approach is a special case of the general scheduling problem in the sense that the index space is a subset of the cartesian $\mathbb{N}^n$ space. The index points are not just vertices of the graph, but rather have coordinates in this cartesian space.

$^1n$ represents the number of nested loops.
Furthermore, the precedence constraints are represented by a fixed and uniform set of dependence vectors, which is the same throughout the index space.

**Problem statement.** Given a Uniform Dependence Loop (UDL), we want to find a legal schedule\(^2\) that results in the minimum makespan while achieving the optimal makespan. Our work is based on the unit execution-zero communication model (UET) of a fixed and uniform set of dependence vectors, which is the UDLs. However, as it has been shown in [12], a UET-UCT sense our approach also applies to UDLs that consider the UET-UCT model. By ignoring the communication delays the criteria of optimality are restricted on the total number of processing elements.

We use the Decision Algorithm (DA), first introduced in [1] and briefly presented in this paper, that efficiently schedules \(n\)-dimensional UDLs by setting strict criteria for the number of processors used. The index space is organized into two sequences of disjoint time-subsets \(ECT\_s\) and \(LCT\_s\), \(i \geq 1\), such that \(ECT\_i\) contains the points whose Earliest Computation Time is \(i\), and \(LCT\_i\) contains those points whose Latest Computation Time is \(i\). The \(LCT\) of an index point is computed using (1) given in section 3. If the \(ECT\) and the \(LCT\) of an index point are both \(i\), then we say that the point belongs to \(ECT\_i\) and has no delay. On the other hand, if the \(ECT\) of an index point is \(i\) but its \(LCT\) is \(i + 1\), we say that this point belongs to \(ECT\_i\) but has a delay of one moment. Following this direction, we form partitions of every \(ECT\) set according to the delay of each point, i.e., those that have no delay in \(D\_i^0\), those that have delay 1 in \(D\_i^1\), and so on. The intersection of each \(ECT\_i\) with the corresponding \(LCT\_i\) is \(D\_i\) and we call these points crucial.

**Contribution.** We establish a clever lower bound on the number of processors that takes into account the propagation of index points with delays from one time step to another and a simple and intuitive upper bound on the optimal number of processors required to achieve the minimum makespan. The proposed lower and upper bounds are estimated by the (non-exhaustive) calculation of the \(ECT\) and \(LCT\) of the index points (this is achieved using methods of computational geometry presented in [4, 12]). Moreover, we identify an important subclass of UDLs called SGRIDs, for which the proposed lower bound equals the upper bound of processors. The SGRIDs have the following property: every point of the index space is a crucial point, hence no delays are possible for any index point without violating the precedence constraints. This means that the optimal number of processors for this subclass equals the lower bound.

The contribution of this paper is the following:

- the introduction of the new dynamic lower bound on the number of processors
- the identification of SGRIDs for which the lower bound equals the upper bound
- the fact that for all UDLs the optimal number of processors coincides with the proposed lower bound (proven experimentally)

The remainder of the paper is organized as follows. The terminology used throughout the paper is given in Section 2. In Section 3 we show how to compute the \(ECT\) and \(LCT\) sets, whereas in Section 4 we present the new dynamic lower bound and a trivial upper bound on the number of processors. Section 5 gives the scheduling policy and describes the decision algorithm. Section 6 shows the experimental results along with illustrative charts. Finally, we summarize our results and propose further work.

**2 Terminology**

As algorithmic model consider the nested FOR-loop structure given in [9, p.71]. \(J\) is the index space of such a loop. \(L = \{l_1, \ldots, l_n\}\) is the initial point and \(U = \{u_1, \ldots, u_n\}\) is the final point of the index space \(J\), where \(l_i, u_i \in \mathbb{Z}\), \(1 \leq i \leq n\) are integer valued constants that represent the lower and upper limits for loop variables. \(DS = \{d_1, d_2, \ldots, d_m\}\) is the set of dependence vectors of the algorithm, which are constant all over the index space. The cardinality of \(J\), denoted \(|J|\) is \(\prod_{i=1}^{n}(u_i - l_i + 1)\).

Throughout the paper we use the following terminology:

- All the points of \(ECT\_i\) are grouped according to their delay resulting in the sequence \(D\_i^k\) of points having delay \(k\). We call crucial all the points with delay zero (denoted \(D\_i^0\)). These points must be executed at moment \(i\).
- The Optimal Execution Time (OET) is the least parallel execution time (assuming that an unbounded number of processors is available). Without any loss of generality we assume that the computation always begins at moment 1, hence \(OET = ECT(U)\).
- The Optimal Number of Processors (ONP) is the least number of processors required to achieve the \(OET\).

**3 Computing the \(ECT\) and \(LCT\) sets**

Given a UDL, the \(ECT\_i\) set contains those points \(j\) of the index space that do not depend on other points of the index space. Every other \(ECT\_i+1\) set, \(1 \leq i \leq OET - 1\) contains those points \(j\) of the index space that depend on points belonging to one of the previously computed \(ECT\) sets, at least

---

\(^2\)A schedule is legal iff it includes all index points and does not violate any dependence constraint.
one of which must be $ECT_i$. After computing the $ECT$ subsets, we can compute also the $LCT$ of any index point $j$ using the formula:

$$LCT(j) = OET - ECT(U - j + L) + 1$$

without having to go through the entire index space. This formula holds because of the geometric properties of the index space. The geometric shape of $LCT_i$ is a mirror image of the $ECT_{OET-i+1}$ if we take as origin the final point $U$ and we reverse the direction of the axes.

4 Computing the Lower and Upper Bound of Processors

For any given UDL, we may have three major Lower Bounds (LBs) depending upon the cardinality of $ECT_1$ set. The most trivial one is $LB_1 = \left\lceil \frac{|ECT_1|}{OET} \right\rceil$. In case that $|ECT_1| > LB_1$, in the first moment we utilize all free processors and for most UDLs we achieve the OET using $ONP = LB_1$ due to the geometry of the index space that allows to obtain maximum processors utilization at each time step.

It may be the case, however, that $LB_1$ processors do not suffice for a legal schedule in $OET$, as in $|ECT_1| < LB_1$, for instance. In this case we cannot obtain maximum processor utilization. On the other hand, at each moment we have the sequence of crucial points $D^0_i$. Hence, another lower bound is $LB_2 = \max(|D^0_i|)$, since with fewer than $LB_2$ processors we cannot produce a schedule in $OET$. Unfortunately, in many cases even this does not suffice. This $LB_2$ is static in the sense that it regards the number of crucial points as constant for each time step regardless the number of processors, which is not accurate. For instance, if some of the points of $D^0_i$ are not scheduled at moment $i$, they become crucial at moment $i+1$, so the correct number of crucial points at moment $i+1$ is greater than the initially computed $D^0_{i+1}$ points.

Therefore, it is clear that we need a better and improved lower bound. The proposed lower bound takes into consideration the propagation of points with delays that remain unexecuted, from one time step to another. We solve a sequence of linear inequalities of the form:

$$\min P_r \in \mathbb{N} \text{ such that } P_r \geq E_r \text{ where } 1 \leq r \leq OET$$

$$\text{and } LB_3 = \max\{P_r\}$$

For the first hyperplane, the candidate number of processors to schedule the crucial points of $ECT_1$ must satisfy the inequality:

$$P_1 \geq E_1 = D^0_1$$

For every subsequent hyperplane $r$, we first check if the number of processors we have found so far $P_{r-1}$ is sufficient. We do that by computing the total number of points that must be scheduled at this moment, using the formula:

$$E_r = D^0_r + \sum_{k=1}^{r-1} \sum_{i=0}^{r-k} D^i_k - P_{r-1}$$

(3)

where $r$, $1 \leq r \leq OET$, is the current hyperplane. If $E_r \leq P_{r-1}$ then $P_{r-1}$ is sufficient for the scheduling of the crucial points of the first $r$ $ECT$ sets at this moment also, consequently there is no need for more processors. In this case, we set $P_r = P_{r-1}$ and continue with the next hyperplane.

On the other hand, if $E_r > P_{r-1}$ then we clearly need more processors and we compute the exact number of processors, $P_r$, necessary for the current hyperplane as follows. We impose the condition:

$$P_r \geq E_r = D^0_r + \sum_{k=1}^{r-1} \sum_{i=0}^{r-k} D^i_k - P_r$$

(4)

In order to solve meaningfully the above inequality we must examine $r$ cases. Clearly $P_r \geq D^0_r$ and we have to consider whether $P_r \leq \sum_{i=0}^{r-k} D^i_k$ or $P_r > \sum_{i=0}^{r-k} D^i_k$, where $1 \leq k \leq r - 1$. So we compute the $r-1$ terms $\sum_{i=0}^{r-k} D^i_k$, we order them in increasing order along with $D^0_i$ (so we have a sequence of $r$ terms $t_1 \leq \ldots \leq t_r$) and we try to solve the inequality (4) initially assuming that $t_1 \leq P_r \leq t_2$, and if we do not find a solution we assume that $t_2 < P_r \leq t_3$, and so on, until we find the first solution for (4). The worst case scenario is that when $P_r > t_r$.

We must stress the fact that in solving (4) we take into consideration the physical meaning of $E_r$. For instance if $\sum_{i=0}^{r-k} D^i_k < P_r$, then in $E_r$ the term $\sum_{i=0}^{r-k} D^i_k - P_r$ is not taken as a negative integer, which would have no meaning, but is taken instead as zero, meaning that the index points belonging to this set were all scheduled and none remained.

In the end, $LB_3$ is the last dynamically computed $P_r$ that sufficed for a legal schedule of all the index points of the nested loops. We take as the final $LB = \max\{LB_1, LB_2, LB_3\}$.

We define $UB = \max\{|ECT_1|, \ldots, |ECT_{OET}|\}$ as the trivial Upper Bound of processors and we say that with $UB$ processors we can legally schedule this UDL achieving $OET$.

Example 4.1 As an example, consider a two-dimensional index space representing the following two-level loop nest in unit-increment steps:

```c
for (i=18; i<=21; i++) {
    for (j=18; j<=21; j++) {
    }
}
```

The $ECT$ sets partitioned according to the delay value are depicted in Figure 1. For the first hyperplane we have the in-
In order to solve the above inequality we must examine the case 

\[ E_1 \]

which satisfies the condition. Therefore we must examine the

case where \( 38 < P_4 \leq 45 \) and we solve the same equation as above, taking the first parenthesis as zero instead of a negative integer:

\[ P_4 = 36 + (38 - P_4) + (45 - P_4) \Rightarrow P_4 = 43 \]

which does not satisfy the condition. Next we go to the fifth hyperplane and check if \( P_5 \) processors suffice.

\[ E_5 = D_5^0 + (D_5^0 + D_5^1 + D_5^1 + D_5^1 + D_5^1 - P_4) + (D_5^0 + D_5^1 + D_5^2 - P_4) + (D_5^0 + D_5^1 + D_5^2 - P_4) + (D_5^0 + D_5^1 + D_5^2 - P_4) = 49 \]

hence \( E_5 > P_5 \) and again we need more processors. We impose the condition:

\[ P_5 \geq D_5^0 + (D_5^0 + D_5^1 + D_5^1 + D_5^1 + D_5^1 - P_4) + (D_5^0 + D_5^1 + D_5^2 - P_4) + (D_5^0 + D_5^1 + D_5^2 - P_4) \Rightarrow P_5 \geq 27 + (40 - P_5) + (47 - P_5) + (54 - P_5) + (61 - P_5) \]

and we try to find a solution by solving the equation:

\[ P_5 = 27 + (40 - P_5) + (47 - P_5) + (54 - P_5) + (61 - P_5) \Rightarrow P_5 = 46 \]

which does not satisfy the condition. Therefore we must examine the case where \( 40 < P_5 \leq 47 \) and we solve the same equation as above, taking the first two parentheses as zero:

\[ P_5 = 27 + (47 - P_5) + (54 - P_5) + (61 - P_5) \Rightarrow P_5 = 48 \]

which does not satisfy the condition. And again we have to examine the next case which is \( 47 < P_5 \leq 54 \) and we solve the same equation as above, taking the first two parentheses as zero:

\[ P_5 = 27 + (54 - P_5) + (61 - P_5) \Rightarrow P_5 = 48 \]

which time satisfies the condition. We proceed with the sixth hyperplane and check whether \( P_6 \) processors suffice.

\[ E_6 = D_6^0 + (D_6^0 + D_6^1 + D_6^1 + D_6^1 + D_6^1 - P_5) + (D_6^0 + D_6^1 + D_6^2 - P_5) + (D_6^0 + D_6^1 + D_6^2 - P_5) + (D_6^0 + D_6^1 + D_6^2 - P_5) + (D_6^0 + D_6^1 + D_6^2 - P_5) = 40 \]

hence \( E_6 < P_6 \). Since we can schedule all the points of this hyperplane with \( P_6 \) processors, we do not need to estimate a new value for the number of processors and we consider \( P_6 = P_5 = 48 \). On the last hyperplane, we check if \( P_7 \) processors suffice.

\[ E_7 = \]

hence \( E_7 < P_7 \). Since we can schedule all the points of this hyperplane with \( P_7 \) processors we consider \( P_7 = P_6 = 48 \) and \( ONP \) required for the optimal schedule is the last computed \( P_7 \), hence \( LB = P_7 = 48 \)
5 The Decision Algorithm (DA)

In order to verify if all UDLs can achieve the proposed dynamic LB we use the scheduling algorithm of uniform dependence loops that was introduced in [1]. For a better understanding, we give a short description of the algorithm below.

Scheduling policy. The algorithm always schedules the crucial points at their appropriate moments, i.e., it schedules the points of $D^0$ at moment $i$. If there are not enough processors to schedule all the crucial points, then the algorithm will return NO indicating that it is impossible to achieve $OET$ using this number of processors and to produce a legal schedule. In the case where we have sufficient processors to schedule all the crucial points, and still remain with free processors, we begin to schedule points with delay 1. If the points with delay 1 are more than the free processors, then they are sorted in increasing order of their $ECT$ value, and those with the same $ECT$ value are sorted in decreasing order of their out degree. If after the assignment of all points with delay 1 there still remain available processors, we proceed with scheduling the points with delay 2, and so on. For every candidate value $P_r$ we call the DA to verify whether there exists a legal schedule with $P_r$ processors.

6 Results

We have generated random UDLs and determined the optimal schedules. The results are depicted in the above figures indicating that all UDLs are scheduled using $LB$ processors, in comparison with the results before introducing $LB_3$.

The statistical analysis of the experimental results lead us to the conclusion that we can schedule every random UDL using the proposed lower bound. We also identified a subclass of UDLs, called SGRIDS, for which $UB$ actually coincides with $LB_3$.

- SGRIDS (Scaled GRIDS): every UDL in this subclass has the general structure of a GRID, i.e., all dependence vectors reside on the axes and the value of the non-zero coordinate is an arbitrary positive integer. GRIDs are a

---

\(^3\)The out degree of an index point is the number of dependence vectors leaving it, that end up to points within the index space.
particular case of SGRIDs, where the dependence vectors are the unitary vectors along the axes.

Moreover, if the size of the index space in each dimension of an SGRID-UDL is divisible with the corresponding coordinate then every index point is crucial and $LB_2 = UB$.

7 Conclusion

We have proposed a new improved lower bound on the number of processors necessary to schedule $n$-dimensional nested loops with uniform dependencies in $OET$. The experimental results have shown that all UDLs can be optimally scheduled using $LB$ processors. This means we can know apriori the necessary number of processors and we can start building the schedule using the appropriate lower bound, also reducing the scheduling costs from the hardware point of view. Future work will focus on improving the time complexity of the algorithm by utilizing methods of computational geometry presented in [4, 12] to compute the ECT sets. Finally, we believe that further theoretical investigation is required in order to give a satisfactory explanation regarding the aforementioned experimental results.

References


