Solving the Constant-Degree Parallelism Alignment Problem

Claude G. Diderich[1*] and Marc Gengler[2]

[1] Swiss Federal Institute of Technology – Lausanne, Computer Science Department, CH-1015 Lausanne, Switzerland, E-mail: `diderich@di.epfl.ch`
[2] Ecole Normale Supérieure de Lyon, Laboratoire de l'Informatique du Parallélisme, F-69364 Lyon, France, E-mail: `Marc.Gengler@lip.ens-lyon.fr`

Abstract. We describe an exact algorithm for finding a computation mapping and data distributions that minimize, for a given degree of parallelism, the number of remote data accesses in a distributed memory parallel computer (DMPC). This problem is shown to be NP-hard.

1 The alignment problem

An important problem when compiling nested loops towards DMPCs is how to map the computation and the data onto processors. This problem can be subdivided into two subproblems: 1) the *alignment problem* which assigns computation and data to a set of virtual processors, and 2) the *mapping problem* which folds the set of virtual processors onto the physical ones. In this paper we address the alignment problem. Following the linear algebra formulation of the alignment problem by Huang and Sadayappan [6] in 1991, researchers have primarily focused on finding linear or affine computation and data alignment functions requiring no remote data accesses [3] or on developing heuristics for minimizing communication [2].

The alignment problem is the problem of finding an alignment of loop iterations with the array elements accessed, that is, mappings of the loop iterations and array elements to a set of virtual processors. The alignment should address the two needs: i) maximize the degree of parallelism, i.e. use as many processors as possible, ii) minimize the number of non local data accesses, i.e. distribute the array elements such that a processor owns a maximal number of the elements it accesses. Depending on how the needs i) and ii) are verified, various subproblems can be defined. When allowing only local data accesses, we talk about the *communication-free alignment problem*. Another subproblem is defined by minimizing the number of remote data accesses for a given degree of parallelism. This subproblem is called the *constant-degree parallelism alignment problem*. We consider array access functions that are linear or affine and use the approach presented by Bau *et al.* [3] for expressing the alignment problem. Access l to array k is described by a function F_k^l. The unknown computation mappings C_j and data mappings D_k can also be written as matrix functions. $\mathcal{I}$ represents the index domain defined by the loop bounds, $\mathcal{D}_k$ the array access domain and $\mathcal{P}$ the virtual multi-dimensional grid of processors.

[*] Supported by a grant from the Swiss Federal Institute of Technology – Lausanne.

$$\begin{array}{llll} F^l_k\colon \mathcal{I} & \longrightarrow \mathcal{D}_k : & \mathbf{i} \longmapsto F^l_k(\mathbf{i}) = \mathbf{F}^l_k\,\mathbf{i} + \mathbf{f}^l_k \\ C_j\colon \mathcal{I} & \longrightarrow \mathcal{P} : & \mathbf{i} \longmapsto C_j(\mathbf{i}) = \mathbf{C}_j\,\mathbf{i} + \mathbf{c}_j \\ D_k\colon \mathcal{D}_k & \longrightarrow \mathcal{P} : & \mathbf{a} \longmapsto D_k(\mathbf{a}) = \mathbf{D}_k\,\mathbf{a} + \mathbf{d}_k \end{array}$$

$C_j(\mathbf{i})$ represents the processor on which iteration $\mathbf{i}$ of assignment instruction j is executed. Similarly, the function D_k indicates on which processors the elements of array k are located. The requirements i) and ii) can be formulated as follows. Eqns. (1) are called alignment or locality constraints.

$$\begin{array}{l} \max_{\mathbf{C}_j,\mathbf{c}_j}\left(\min_j\left(\operatorname{rank}\left(\mathbf{C}_j^{\mathsf{T}}\right)\right)\right). \\ \forall\, \mathbf{i} \in \mathcal{I}\colon\ \mathbf{C}_j\,\mathbf{i} + \mathbf{c}_j = \mathbf{D}_k\left(\mathbf{F}^l_k\,\mathbf{i} + \mathbf{f}^l_k\right) + \mathbf{d}_k. \end{array} \tag{1}$$

The algorithm for solving the communication-free alignment problem presented by Bau *et al.* [3], called LINEAR-ALIGNMENT, is defined as follows. The set of eqns. (1) is rewritten in the following equivalent form.

$$\forall \mathbf{i} \in \mathcal{I}\colon \left(\hat{\mathbf{C}}_j\ \hat{\mathbf{D}}_k\right)\begin{pmatrix} \mathbf{I} \\ -\hat{\mathbf{F}}^l_k \end{pmatrix}\begin{pmatrix} \mathbf{i} \\ 1 \end{pmatrix} = 0 \tag{2}$$

$$\text{with}\quad \hat{\mathbf{C}}_j = \left(\,\mathbf{C}_j\ \ \mathbf{c}_j\,\right), \hat{\mathbf{D}}_k = \left(\,\mathbf{D}_k\ \ \mathbf{d}_k\,\right), \hat{\mathbf{F}}^l_k = \begin{pmatrix} \mathbf{F}^l_k & \mathbf{f}^l_k \\ 0 & 1 \end{pmatrix}$$

To simplify the problem we require that eqns. (2) hold for any vector $\mathbf{i}$, regardless of whether or not it belongs to the iteration domain $\mathcal{I}$. The alignment constraints then become equations of the form (3). Allowing no communication imposes that all locality constraints (3) are verified simultaneously, leading to (4).

$$\left(\hat{\mathbf{C}}_j\ \hat{\mathbf{D}}_k\right)\begin{pmatrix} \mathbf{I} \\ -\hat{\mathbf{F}}^l_k \end{pmatrix} = 0. \tag{3}$$

$$\hat{\mathbf{U}}\,\hat{\mathbf{V}} = 0 \tag{4}$$

$$\begin{array}{ll} \text{where} & \hat{\mathbf{U}} = \left(\,\hat{\mathbf{C}}_1 \cdots \hat{\mathbf{C}}_t\ \ \hat{\mathbf{D}}_1 \cdots \hat{\mathbf{D}}_s\,\right), \hat{\mathbf{V}} = \left(\,\hat{\mathbf{V}}_{u,v,w} \cdots \hat{\mathbf{V}}_{x,y,z}\,\right) \\ & \hat{\mathbf{V}}_{j,k,l} = \left(0 \cdots 0\ \mathbf{I}\ 0 \cdots 0\ \ -\hat{\mathbf{F}}^l_k\ \ 0 \cdots 0\right)^{\mathsf{T}} \end{array}$$

The sub-matrix $\hat{\mathbf{V}}_{j,k,l}$, where $\mathbf{I}$ is the jth block and $-\hat{\mathbf{F}}^l_k$ the $(t+k)$th block, represents the alignment constraint of data access l of array k in statement j and the processor using that data. Eqn. (4) is equivalent to $\hat{\mathbf{V}}^{\mathsf{T}}\,\hat{\mathbf{U}}^{\mathsf{T}} = \mathbf{0}$. Therefore, the column vectors of the unknown matrix $\hat{\mathbf{U}}^{\mathsf{T}}$ are in the null space of the known $\hat{\mathbf{V}}^{\mathsf{T}}$.

2 The constant-degree parallelism alignment problem

Often, the degree of parallelism of a communication-free alignment is non-existing. This leads us to define the *constant-degree parallelism alignment problem* (CDPAP), which consists of finding communication and data mappings such that the degree of parallelism obtained is at least equal to the input parameter d and the communications are minimized. The *constant-degree parallelism alignment algorithm* (CDPAA) solves the CDPAP.

Assume that it is possible to find a communication-free alignment of parallelism degree d' for a given problem $\hat{\mathbf{V}}$. We simplify $\hat{\mathbf{V}}$ by finding a minimal set of alignment constraints from (1) to be left unsatisfied such that the simplified problem has a solution of degree of parallelism d when solved by the LINEAR-ALIGNMENT algorithm. To increase the parallelism introduced by the LINEAR-ALIGNMENT algorithm by $d'' = d - d'$, we have to construct a modified problem $\hat{\mathbf{V}}'$ such that the size of the basis of the null space of that problem is increased by d'' compared to the size of the null-space of the original problem.

We will use the notation of $\tilde{\mathbf{V}}$ to represent the vector space spanned by the column vectors of the matrix $\hat{\mathbf{V}}$. $\tilde{\mathbf{V}}$ represents the space of all the alignment constraints. In order to increase the degree of parallelism by at least d'', we need to find a subspace $\tilde{\mathbf{V}}'$ of $\tilde{\mathbf{V}}$ such that $\dim(\tilde{\mathbf{V}}) - \dim(\tilde{\mathbf{V}}') \geq d''$. Let $\tilde{d} = \dim(\tilde{\mathbf{V}}) - d''$. There exist an infinite number of such subspaces $\tilde{\mathbf{V}}'$ of dimension $\tilde{d}$, but only finitely many are of interest to us. In fact, all subspaces of $\tilde{\mathbf{V}}$ that contain less than $\tilde{d}$ vector columns of $\hat{\mathbf{V}}$ are uninteresting, because we know that there exists at least one subspace containing at least $\tilde{d}$ column vectors of $\hat{\mathbf{V}}$. Furthermore, the set of all the subspaces of degree at most $\tilde{d}$ containing at least $\tilde{d}$ column vectors can be easily enumerated. To do so, we select $\tilde{d}$ column vectors of $\hat{\mathbf{V}}$. Then, for each valid subset of column vectors of $\hat{\mathbf{V}}$, we compute a basis and count the number of alignment constraints that can be expressed in that basis. Finally, we select a subspace $\hat{\mathbf{V}}^*$ that contains the largest number of alignment constraints.

2.1 Some important aspects

Data dependences. As long as all alignment constraints are verified, data dependences are as well. As soon as alignment constraints are dropped this may no longer be true. Removing alignment constraints that represent part of data dependences may increase the degree of parallelism without reducing the execution time. In [4] we characterize the relation between the computed alignment and the iteration scheduling, that is, the relation between processor and time parallelism. Essentially, we show that a sufficient, but not necessary, condition to get a non constant number of active processors during each time step consists of imposing that there be at least two fulfilled alignment constraints that correspond to a data dependence.
Cost function. In the CDPAA we use a counting argument based on the number of non local data accesses as optimization function for computing efficient alignment functions. Another possibility would be to assign different weights to the different alignment constraints depending on their importance and then minimize the weighted sum of remote data accesses. Such a cost function even allows the user of the algorithm to require some alignment constraints to be verified by assigning to them an infinite weight. For example, the owner computes rule can be imposed by assigning a weight of $+\infty$ to the alignment constraint $\hat{\mathbf{C}}_j = \hat{\mathbf{D}}_k\hat{\mathbf{F}}_k^1$, where $\hat{\mathbf{F}}_k^1$ represents the element of array k being modified by instruction j. Furthermore, a weighting function may also be used, in principle, in order to ignore constant offsets or not, prefer non local constant accesses to non-local linear ones, etc.
Complexity and optimality. The following results are extracted from [4]. The input size of any CDPAP is characterized by four parameters, which are the number of data accesses n, the larger of the maximal dimension of any array and the maximal

loop nest depth e, the number of assignment statements c and the number of arrays a. When considering all these parameters variable, we have the following theorem, obtained by reduction from the homogeneous, bipolar MAX FLS$^=$ problem [1].

Theorem 1 *The CDPAP is NP-hard.*

Theorem 2 *The CDPAA finds communication and data alignments that need a minimal number of non local data accesses for a given degree of parallelism.*

Theorem 3 *The CDPAA needs $O(n^{4+e\,(c+a)})$ time to find an optimal alignment.*

2.2 Experimental results and related work

To show the performance of the CDPAA, we apply the algorithm to various loop nests of different sizes extracted from various programs and benchmarks. In each example, which does not have a communication-free alignment, we search for alignment functions having at least one degree of parallelism. Results are given in [4].

Many techniques for solving the alignment problem proposed by different research teams [2, 3, 5] are related to the approach taken in this paper. Anderson and Lam [2] define necessary conditions for the data being local. These conditions admit a direct translation into the framework defined by Bau *et al.* [3] and are particular cases. The problem considered by Dion and Robert in [5] is also a particular case of [3]. Dion and Robert find an optimal solution to their problem. Our technique for constructing a maximal set of alignment constraints that can be verified while providing a given degree of parallelism is more general than [5], as we allow any access function, rather than restricting ourselves to access functions of full rank.

3 Conclusion

In this paper we presented an extension to the communication-free alignment algorithm of Bau *et al.* to remove a minimal number of unsatisfiable constraints to increase the degree of parallelism up to a given constant. In our future work, we are investigating the possibility to incorporate into our framework the notion of scheduling vector so as to be able to optimize a single function when solving both the scheduling and the alignment problem.

References

1. E. Amaldi and V. Kann. The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theoret. Comput. Sci.*, 147(1-2):181–210, 1995.
2. J. M. Anderson and M. S. Lam. Global optimizations for parallelism and locality on scalable parallel machines. In *Proc. PLDI '93*, pages 112–125, 1993.
3. D. Bau, I. Kodukula, V. Kotlyar, K. Pingali, and P. Stodghill. Solving alignment using elementary linear algebra. In *Proc. LCPC '94*, LNCS 892, pages 46–60, Springer, 1994.
4. C. G. Diderich and M. Gengler. Solving the constant-degree parallelism alignment problem. Research Rep. DI-96/195, Swiss Fed. Inst. of Tech. – Lausanne, Switzerland, 1996.
5. Michèle Dion and Yves Robert. Mapping affine loop nests: New results. In *Proc. HPCN '95*, LNCS 919, pages 184–189, Springer, 1995.
6. C.-H. Huang and P. Sadayappan. Communication-free hyperplane partitioning of nested loops. In *Proc. LCPC '91*, LNCS 589, pages 186–200, Springer, 1991.