Efficient Mapping of Interdependent Scans

Michel Barreteau and Paul Feautrier

Laboratoire PRiSM, Université de Versailles - S^{t}Quentin
45, avenue des États-Unis, 78 035 Versailles, FRANCE

Abstract. Distributed memory multiprocessors are extremely sensitive to communication costs. Some global communications such as scans and reductions are of special interest since their cost is much lower than for point to point communications. Our paper focuses on an algorithm which efficiently takes the mapping of scans into account.

1 Introduction

Communications remain the most critical aspect of performance in efficiently programming distributed memory multiprocessors. Hence minimizing communications is an indispensable task. A static placement may be obtained in two different ways: one may ask the user to insert annotations to specify data mapping à la HPF. Our approach is to leave this work to the compiler (see 2). However such a placement will not be sufficient if some particularities of the target machine are not taken into account, for instance if communication primitives with low overhead are not used. In this paper we propose a method to compute a placement which efficiently exploits interprocessor data movements such as scans and reductions. Especially we detail the Cholesky example to make the reader sensitive to the interdependences of scans. Neglecting them yields a mapping which may be incompatible with the minimization of communications.

2 Automatic parallelization

Automatic parallelization consists of extracting from the source program all restrictions on its potential parallelism. The Data Flow Graph (DFG) depicts the results of this analysis, i.e. the data movements between sets of operations [2]. A System of Affine Recurrence Equations (SARE) is built from the DFG. It is suitable for detecting recurrences [4]. We will consider the following SARE of three equations, which is extracted from a Cholesky decomposition program:

$$Ins\_2[i,k] = \begin{cases} \mathbf{Scan}(\{\mathbf{i_2},\mathbf{k_2} \mid (\mathbf{3} \le \mathbf{i_2} \le \mathbf{n}) \wedge (\mathbf{1} \le \mathbf{k_2}) \wedge (\mathbf{1} \le \mathbf{i_2} - \mathbf{k_2})\}, \\ \quad ([\mathbf{0}\ \mathbf{1}]),\ +,\ -\mathbf{Ins\_6}[\mathbf{k_2},\mathbf{i_2}]^{\mathbf{2}},\ \mathbf{InitS_2})[\mathbf{i},\mathbf{k}] \\ \quad if\ (i \le n) \wedge (2 \le k) \wedge (1 \le i-k) \\ a[i,i] - (a[k,i]/\sqrt{a[k,k]})^2 \quad if\ (2 \le i \le n) \wedge (k = 1) \end{cases}$$

$$Ins\_5[i,j,k] = \begin{cases} \mathbf{Scan}(\{\mathbf{i_5},\mathbf{j_5},\mathbf{k_5} \mid (\mathbf{3} \le \mathbf{i_5}) \wedge (\mathbf{j_5} \le \mathbf{n}) \wedge (\mathbf{1} \le \mathbf{k_5}) \\ \quad \wedge (\mathbf{1} \le \mathbf{i_5} - \mathbf{k_5}) \wedge (\mathbf{1} \le \mathbf{j_5} - \mathbf{i_5})\},\ ([\mathbf{0}\ \mathbf{0}\ \mathbf{1}]),\ +, \\ \quad -(\mathbf{Ins\_6}[\mathbf{k_5},\mathbf{j_5}] * \mathbf{Ins\_6}[\mathbf{k_5},\mathbf{i_5}]),\ \mathbf{InitS_5})[\mathbf{i},\mathbf{j},\mathbf{k}] \\ \quad if\ (j \le n) \wedge (2 \le k) \wedge (1 \le i-k) \wedge (1 \le j-i) \\ a[i,j] - ((a[k,j]/\sqrt{a[k,k]}) * (a[k,i]/\sqrt{a[k,k]})) \\ \quad if\ (2 \le i) \wedge (j \le n) \wedge (k = 1) \wedge (1 \le j-i) \end{cases}$$

$$Ins\_6[i,j] = \begin{cases} Ins\_5[i,j,i-1]/Ins\_2[i,i-1] \\ \quad if\ (2 \leq i) \wedge (j \leq n) \wedge (1 \leq j-i) \end{cases}$$

3 Scans & Reductions

Since scans and reductions are particular recurrences due to their associative operator, they may be computed in a parallel way using a binary tree. A reduction is a special scan in which only the final result is used. We only consider unidirectional scans, i.e. scans whose accumulation follows a unique direction in their iteration space. Let us remember the semantics of the unidirectional operator $\texttt{Scan}(\mathcal{D}_S, \mathbf{v}_S, \odot, \mathtt{D}_S, \mathtt{G}_S)$: $\mathcal{D}_S$ describes the accumulation space, $\mathbf{v}_S$ is the direction vector, $\odot$ a binary and associative operator, $\mathtt{D}_S$ the expression of the data to be reduced and $\mathtt{G}_S$ are initial values. `Scan` computes a value for each integral point of $\mathcal{D}_S$, i.e. a multidimensional array `S`. Some so-called *virtual* processors are in charge of these computations. They define the virtual processors space.

> Two scans S_2 and S_5 whose direction vectors are $\mathbf{v}_{S_2} = (0,1)$ and $\mathbf{v}_{S_5} = (0,0,1)$ have been detected in our SARE. S_2 computes the array S2: $\forall\ (3 \leq i_2 \leq n) \wedge (1 \leq k_2) \wedge (1 \leq i_2 - k_2)$, $\mathtt{S2}[i_2,k_2] = \mathtt{S2}[i_2,k_2-1] + (-Ins\_6[k_2,i_2]^2)$.

Practical implementations of the `Scan` operator allow only direction vectors which are parallel to the canonical axes of the virtual processors grid. A correct determination of the orientation of scans is necessary to avoid some additional communications related to non local data. But most of the time it depends on interactions between scans. Hence, the problem is to find out an efficient orientation of all the scans when placement binds them together. To this end we recall the successive stages of the mapping algorithm introduced in [3], which will be modified in order to solve our problem.

4 Placement algorithm

A placement function π_C, which is applied to a set of operations defined by an equation C of iteration vector $\mathbf{i}_C$, gives the name of the virtual processor on which C has to be executed. As regards geometry, a grid of virtual processors whose dimension g is given ($g = 2$ for Cholesky), has been elected. For each equation and for each scan, an affine placement function with g components has to be computed: for all dimension d $(1 \leq d \leq g)$, $\pi^d_C(\mathbf{i}_C) = H^d_C.\mathbf{i}_C + K^d_C.\mathbf{n} + \mathbf{l}^d_C$ where unknowns are the coefficients μ_C of the $g \times |\mathbf{i}_C|$ matrix H, ν_C of the $g \times |\mathbf{n}|$ matrix K and ξ_C of the vector $\mathbf{l}$. The vector of structure parameters is $\mathbf{n}$ (n belongs to it). $|\mathbf{v}|$ gives the number of components of a vector $\mathbf{v}$.

Ex: $\pi^d_{Ins\_2}(i,k) = \mu^{d,1}_{Ins\_2}.i + \mu^{d,2}_{Ins\_2}.k + \nu^{d,1}_{Ins\_2}.n + \xi^d_{Ins\_2}$

4.1 Placement conditions

First let us consider usual placement conditions that are given by the DFG:

$$\pi^d_{Ins\_6}(i,j) = \pi^d_{Ins\_5}(i,j,i-1) \Leftrightarrow \{\mu^{d,1}_{Ins\_6} = \mu^{d,1}_{Ins\_5} + \mu^{d,3}_{Ins\_5}, \mu^{d,2}_{Ins\_6} = \mu^{d,2}_{Ins\_5}\} \quad (1)$$

$$\pi^d_{Ins\_6}(i,j) = \pi^d_{Ins\_2}(i,i-1) \Leftrightarrow \{\mu^{d,1}_{Ins\_6} = \mu^{d,1}_{Ins\_2} + \mu^{d,2}_{Ins\_2}, \mu^{d,2}_{Ins\_6} = 0\} \quad (2)$$

Then let us add the scan placement conditions which are developed in [1]:

1. Data alignment along the scan: $\forall\ \mathtt{C}(\mathbf{i}_C) \in \mathtt{D}_S(\mathbf{i}_S),\ \pi_S(\mathbf{i}_S) = \pi_C(\mathbf{i}_C)$
   $\pi^d_{S_2}(i_2,k_2) = \pi^d_{Ins\_6}(k_2,i_2) \Leftrightarrow \{\mu^{d,1}_{S_2} = \mu^{d,2}_{Ins\_6}, \mu^{d,2}_{S_2} = \mu^{d,1}_{Ins\_6}\}$ (3)
   $\pi^d_{S_5}(i_5,j_5,k_5) = \pi^d_{Ins\_6}(k_5,j_5) \Leftrightarrow \{\mu^{d,1}_{S_5} = 0, \mu^{d,2}_{S_5} = \mu^{d,2}_{Ins\_6}, \mu^{d,3}_{S_5} = \mu^{d,1}_{Ins\_6}\}$ (4)
   $\pi^d_{S_5}(i_5,j_5,k_5) = \pi^d_{Ins\_6}(k_5,i_5) \Leftrightarrow \{\mu^{d,1}_{S_5} = \mu^{d,2}_{Ins\_6}, \mu^{d,2}_{S_5} = 0, \mu^{d,3}_{S_5} = \mu^{d,1}_{Ins\_6}\}$ (5)
2. Results collection from the scan to any variable $\mathtt{X}$ such as $\mathtt{X}[\mathbf{i}_X] = \mathtt{S}[l(\mathbf{i}_X)]$ with $l(\mathbf{i}_X) = P_S.\mathbf{i}_X + Q_S :\ \pi_X(\mathbf{i}_X) = \pi_S(l(\mathbf{i}_X))$
   $\pi^d_{Ins\_2}(i,k) = \pi^d_{S_2}(i,k) \Leftrightarrow \{\mu^{d,1}_{Ins\_2} = \mu^{d,1}_{S_2}, \mu^{d,2}_{Ins\_2} = \mu^{d,2}_{S_2}\}$ (6)
   $\pi^d_{Ins\_5}(i,j,k) = \pi^d_{S_5}(i,j,k) \Leftrightarrow \{\mu^{d,1}_{Ins\_5} = \mu^{d,1}_{S_5}, \mu^{d,2}_{Ins\_5} = \mu^{d,2}_{S_5}, \mu^{d,3}_{Ins\_5} = \mu^{d,3}_{S_5}\}$ (7)
3. As regards the placement condition on computations which consists of finding for each scan an orientation following a j direction in the virtual processors grid: $\exists\ j\ (1 \leq j \leq g)$ *such that* $H^j_S.\mathbf{v}_S \neq 0$ *and* $\forall\ i \neq j,\ H^i_S.\mathbf{v}_S = 0$ (8), it should have priority to ensure an efficient computation. For each scan, g conditions $H^d_S.\mathbf{v}_S = e^d_S$ are at the top of the system $\mathcal{P}$ of placement equations in order to satisfy the strategy of the solution algorithm described below. The projection of $\mathbf{v}_S$ by H_S in the virtual processors grid is $\mathbf{e}_S$. The components of $\mathbf{e}_S$ are orientation parameters. All of them are equal to zero except one:
   $\forall\ S,\ \mathbf{e}_S = k.\mathbf{u}_S$ where $k \in \mathbb{Z}^*$ and $\mathbf{u}_S$ is a canonical vector (9)
   $H_{S_2}.\mathbf{v}_{S_2} = \mathbf{e}_{S_2} \Leftrightarrow \{\mu^{1,2}_{S_2} = e^1_{S_2}, \mu^{2,2}_{S_2} = e^2_{S_2}\}$ (10)
   $H_{S_5}.\mathbf{v}_{S_5} = \mathbf{e}_{S_5} \Leftrightarrow \{\mu^{1,3}_{S_5} = e^1_{S_5}, \mu^{2,3}_{S_5} = e^2_{S_5}\}$ (11)

Most of the placement conditions have to be satisfied in order to minimize communications. A greedy algorithm based on the Gauss-Jordan elimination takes the edges that transfer the most important data volume first into account. This is their *weight*. In this way placement equations which are associated to these edges have a high probability to be cut and taken part in the solution of $\mathcal{P}$. (The order of edges to be considered according to their weight - as computed by our prototype - is: (7),(4),(5),(1),(3),(6) and (2).)

In order to avoid the mapping of every operation on a unique processor (experience shows that it often happens when all the equations are satisfied), a heuristic is adopted. This is the *triviality test*: it only accepts an equation if the current solution of $\mathcal{P}$ (this equation included) is still able to generate g linearly independent solutions for each placement function.

The triviality test accepts for instance $\mu^{d,1}_{S_5} = 0$ from (4). But $\mu^{d,2}_{S_5} = 0$ is rejected because one cannot build for Ins_5 g (i.e. 2) linearly independent solutions any more ($\mu^{d,2}_{Ins\_5} = \mu^{d,2}_{S_5}$ from (7) and $\mu^{d,1}_{Ins\_5} = 0$ from (7) and (4)).

4.2 Interactions between scans

The pivot element of each equation is chosen to find out interactions between scans. As no assumption about relations on the coefficients μ_S can be made, we reject the problem to the relations $\mathcal{S}$ between the orientation parameters e_S.

$\mu^{d,2}_{Ins\_6} = \mu^{d,2}_{S_5}$, $\mu^{d,1}_{Ins\_6} = \mu^{d,3}_{S_5}$ according to (4) and (3) implies $\mu^{d,2}_{Ins\_6} = \mu^{d,1}_{S_2}$, $\mu^{d,1}_{Ins\_6} = \mu^{d,2}_{S_2}$. Thus we get $\mu^{d,1}_{S_2} = \mu^{d,2}_{S_5}$ and $\mu^{d,2}_{S_2} = \mu^{d,3}_{S_5}$. Finally $e^d_{S_2} = e^d_{S_5}$ from (10) and (11).

The constraint to be satisfied for building $\mathcal{S}$ comes straightforward from (8): let $(\mathbf{u}^1, \ldots, \mathbf{u}^g)$ be the canonical vectors which describe the g directions of the virtual processors space, then $\mathcal{S}$ must guarantee that each $\mathbf{e}_S$ follows any canonical axis of the processors grid, i.e. each $\mathbf{e}_S$ must be colinear to one $\mathbf{u}^j$.

Let us consider the equation $\mathcal{E}$: at first $e^1_{S_2} = e^1_{S_5}$ since equations are processed dimension per dimension. Then a combination of canonical vectors $\mathbf{u}^j_S$ which represents the direction of reorientation vectors $\mathbf{e}_S$ is chosen. For instance $j = 1$ (cf (8)) for S_2 and S_5, which means that both scans are supposed to lie in the direction of $\mathbf{u}^1$. Rather than expressing this colinearity, it is easier to express that each $\mathbf{e}_S$ has to be orthogonal to the $g-1$ canonical vectors $\mathbf{u}^{\bar{j}}$ which describe the supplementary subspaces of the subspace defined by $\mathbf{u}^j_S$. In this way, one has to compute $g-1$ dot products per scan: $\mathbf{e}_{S_2} \cdot \mathbf{u}^1 = 0$ and $\mathbf{e}_{S_5} \cdot \mathbf{u}^1 = 0$. Thus we get the following conditions $\{e^2_{S_2} = 0, e^2_{S_5} = 0\}$ which complete the subsystem $\mathcal{S}' = \mathcal{S} \cup \mathcal{E}$. (9) must be confirmed by the solution of $\mathcal{S}'$. $e^1_{S_5} \neq 0$ is acceptable. This combination suits us. Otherwise we should choose another combination such as $(\mathbf{u}^1_{S_2}, \mathbf{u}^2_{S_5})$. The second equation is: $e^2_{S_2} = e^2_{S_5}$. The first combination $(\mathbf{u}^1_{S_2}, \mathbf{u}^1_{S_5})$ yields: $e^2_{S_5} = 0$. (9) is still verified. We deduce $\mathcal{S} = \mathcal{S} \cup \mathcal{E} = \{e^d_{S_2} = e^d_{S_5}\}$. On the other hand the second mentioned combination $(\mathbf{u}^1_{S_2}, \mathbf{u}^2_{S_5})$ would yield: $\{e^1_{S_5} = 0, e^2_{S_5} = 0\}$. It would have been rejected because (9) is not satisfied any more. In this case $\mathcal{S}$ would remain unchanged.

When every equation has been processed, one has to replace the parameters e^d_S by their own values into the global system $\mathcal{P}$ using the last combination which satisfies $\mathcal{S}$. $\mathcal{P}$ is solved and the d^{th} solution is applied to π^d_C. Thus we get:

$$\pi^1_{S_2}(i_2, k_2) = k_2 \qquad \pi^1_{S_5}(i_5, j_5, k_5) = k_5 \qquad \pi^1_{Ins\_6}(i, j) = i$$
$$\pi^2_{S_2}(i_2, k_2) = i_2 \qquad \pi^2_{S_5}(i_5, j_5, k_5) = j_5 \qquad \pi^2_{Ins\_6}(i, j) = j$$

5 Conclusion

We presented in this paper an algorithm which computes multidimensional placement functions. It relies on data locality but also consider some special data movements such as scans and reductions and their interdependences. It enables to optimize mapping but also to produce an efficient target code. In this way hardwired communication primitives that are provided by many distributed memory multiprocessors will be efficiently exploited.

References

1. Barreteau, Feautrier: *Automatic Mapping of Scans and Reductions.* HPCS'95. 1995.
2. Feautrier: *Dataflow Analysis of Scalar and Array References.* Int. J. of Parallel Programming, 20(1):23–53. 1991.
3. Feautrier: *Toward Automatic Partitioning of Arrays on Distributed Memory Computers.* In ACM ICS'93 Tokyo pp.175–184. 1993.
4. Redon, Feautrier: *Detection of Reductions in Sequentials Programs with Loops.* PARLE, LNCS 694. Ed. Arndt Bode and Mike Reeve and Gottfried Wolf. 1993.