Integrating Automatic Data Alignment and Array Operation Synthesis to Optimize Data Parallel Programs

Gwan-Hwan Hwang[1], Jenq Kuen Lee[1] and Dz-Ching R. Ju[2]

[1] Department of Computer Science, National Tsing-Hua University, Hsinchu, Taiwan
[2] Hewlett-Packard Company, Cupertino, CA 95014,USA

Abstract. Both automatic data alignment and array operation synthesis have been shown to be very important and effective schemes to optimize data parallel programs. However, they were considered separately so far by the research community. In this paper, we address the issue how to integrate the array operation synthesis scheme into the automatic alignment process. We propose a new array alignment concept, called segmented alignment, to help incorporate array operation synthesis scheme into automatic alignment process. Experiments done on an 8-node DEC Farm show that the automatic alignment process with the help of array operation synthesis and segmented alignment concepts can significantly outperform the one without these mechanisms.

1 Introduction

In data parallel languages (such as Fortran 90), the automatic alignment problem is to find an alignment relationship to align arrays so that incurred data communications will be minimized in distributed memory architectures. On the other hand, array operation synthesis is to compose consecutive array operations or array expressions into a composite access function of the source arrays at compile time to optimize data parallel programs[1]. However, they were considered separately so far by the research community despite of their successes. An important question remains open as how we can integrate array operation synthesis into the automatic alignment process. We address this open issue by proposing a new array alignment concept, called segmented alignment, to help incorporate array operation synthesis scheme into automatic alignment process. In addition, we will show through examples that neither application order between array operation synthesis and automatic alignment schemes will guarantee the optimal solutions. Our strategy to solve this problem is using a two-step scheme. In the first step, we test the possibility to obtain a communication-free code by integrating array operation synthesis and segmented alignments. In this process, we first apply operation synthesis scheme, and then we perform automatic alignments with the help of segmented alignments for the codes produced in the previous process. If a communication-free code can be obtained this way, the output code is the optimal solution by integrating array operation synthesis scheme into automatic alignment process. If we can not get a communication free code in the first step, we will use our methods in the second step, where we derive a cost model and present an optimal solution based on the cost model to integrate array synthesis into the automatic alignment framework. We also show that the problem of finding an optimal solution is NP-hard. Therefore, we develop a practical heuristic algorithm for compilers to incorporate synthesis strategy into automatic alignment framework. Due to space limitations, we will only explain the concept of segmented alignment below, and leave the detailed of the process in a technical note [2].

2 Segmented Alignment

Consider the following code fragment which is obtained by applying the array operation synthesis to array expression A=CSHIFT((TRANSPOSE(EOSHIFT(B,$\frac{N}{2}$,-6.6,1))+C),$\frac{N}{2}$,1). If we use conventional alignment methods to specify the data distribution of array A, B, and C, it is impossible to construct a communication-free code fo the four parallel loops. For example, consider the **Loop 1** and **Loop 3**, A(I,J), B(J+$\frac{N}{2}$,I+$\frac{N}{2}$) and C(I+$\frac{N}{2}$,J) should be properly aligned to insure communication-free in **Loop 1**. However, to insure communication-free in **Loop 3**, A(I,J), B(J+$\frac{N}{2}$,I-$\frac{N}{2}$) and C(I-$\frac{N}{2}$,J) should be aligned.

```
        REAL A(N,N), B(N,N), C(N,N)
        DO ITER=1, Number_Iteration
!Loop 1
           FORALL (I=1:N/2,J=1:N/2)
              A(I,J)=B(J+N/2,I+N/2)+C(I+N/2,J)
           END FORALL
!Loop 2
           FORALL (I=1:N/2,J=N/2 + 1:N)
              A(I,J)=-6.6+C(I+N/2,J)
           END FORALL
!Loop 3
           FORALL (I=N/2 + 1:N,J=1:N/2)
              A(I,J)=B(J+N/2,I-N/2)+C(I-N/2,J)
           END FORALL
!Loop 4
           FORALL (I=N/2 + 1:N,J=N/2 + 1:N)
              A(I,J)=-6.6+C(I-N/2,J)
           END FORALL
        ENDDO
```

In order to help incorporate synthesized codes into automatic alignment framework, we propose a new concept, called segmented alignment, to solve the above problem. The idea of segmented alignment is to allow alignment relations within delimited index domains. We add **WHEN** and **IN** keywords to target codes for HPF-style languages. See the following HPF compiler directives which can obtain a communication-free code for the above loops.

```
        REAL A(N,N), B(N,N), C(N,N)
!HPF$   TEMPLATE TEMP1(N/2,N/2), TEMP2(N/2,N/2), TEMP3(N/2,N/2), TEMP4(N/2,N/2)
!HPF$   ALIGN A(I,J) WITH TEMP1(I,J) WHEN (I,J) IN (1:N/2,1:N/2)
!HPF$   ALIGN B(I,J) WITH TEMP1(J-N/2,I-N/2) WHEN (I,J) IN (N/2 + 1:N,N/2 + 1:N)
!HPF$   ALIGN C(I,J) WITH TEMP1(I-N/2,J) WHEN (I,J) IN (N/2 + 1:N,1:N/2)
!HPF$   ALIGN A(I,J) WITH TEMP2(I,J) WHEN (I,J) IN (1:N/2,N/2 + 1:N)
!HPF$   ALIGN C(I,J) WITH TEMP2(I-N/2,J) WHEN (I,J) IN (N/2 + 1:N,N/2 + 1:N)
!HPF$   ALIGN A(I,J) WITH TEMP3(I,J) WHEN (I,J) IN (N/2 + 1:N,1:N/2)
!HPF$   ALIGN B(I,J) WITH TEMP3(J+N/2,I-N/2) WHEN (I,J) IN (1:N/2,N/2 + 1:N)
!HPF$   ALIGN C(I,J) WITH TEMP3(I+N/2,J) WHEN (I,J) IN (1:N/2,1:N/2)
!HPF$   ALIGN A(I,J) WITH TEMP4(I,J) WHEN (I,J) IN (N/2 + 1:N,N/2 + 1:N)
```

```
!HPF$   ALIGN C(I,J) WITH TEMP4(I+N/2,J) WHEN (I,J) IN (1:N/2, N/2 + 1:N)
!HPF$   DISTRIBUTE TEMP1(BLOCK,BLOCK),TEMP2(BLOCK,BLOCK)
!HPF$   DISTRIBUTE TEMP3(BLOCK,BLOCK),TEMP4(BLOCK,BLOCK)
```

Note that the extended segmented alignment directives can be emulated by standard HPF programs by splitting an array into several subarrays according to the disjointed index domains resulted from the array operation synthesis. Then, we specify those subarrays with proper alignment relations to produce communication-free code. The details of the emulation can also be seen in [2].

We list the general form for generating the segmented alignment compiler directives as follows. For the following loops,

FORALL $(I = L : U : S)$
  $A_1(\alpha_1 * I + \beta_1) = \mathcal{F}(A_2(\alpha_2 * I + \beta_2), A_3(\alpha_3 * I + \beta_3), \ldots, A_n(\alpha_n * I + \beta_n))$
END FORALL

we generate the following HPF-like compiler directives to obtain a communication-free execution.

!HPF$ TEMPLATE TEMP(X1:X2)

!HPF$ ALIGN A_1(I) WITH TEMP($\frac{LCM(\alpha_1,\alpha_2,\ldots,\alpha_n)}{\alpha_1} * I$) WHEN (I) IN $(\alpha_1 * L + \beta_1 : \alpha_1 * U + \beta_1 : \alpha_1 * S)$

!HPF$ ALIGN A_2(I) WITH TEMP($\frac{LCM(\alpha_1,\ldots,\alpha_n)}{\alpha_2} * I + \beta_1 * \frac{LCM(\alpha_1,\ldots,\alpha_n)}{\alpha_1} - \beta_2 * \frac{LCM(\alpha_1,\ldots,\alpha_n)}{\alpha_2}$) WHEN (I) IN $(\alpha_2 * L + \beta_2 : \alpha_2 * U + \beta_2 : \alpha_2 * S)$

...

!HPF$ ALIGN A_n(I) WITH TEMP($\frac{LCM(\alpha_1,\ldots,\alpha_n)}{\alpha_n} * I + \beta_1 * \frac{LCM(\alpha_1,\ldots,\alpha_n)}{\alpha_1} - \beta_n * \frac{LCM(\alpha_1,\ldots,\alpha_n)}{\alpha_n}$) WHEN (I) IN $(\alpha_n * L + \beta_n : \alpha_n * U + \beta_n : \alpha_n * S)$

3 Discussion

The segmented alignment helps to generate communication-free codes, but does not always guarantee to generate communication free codes. In our work, we prove the segmented alignment is always better than the conventional alignment of HPF. However, the optimal solution is NP-hard. Thus, we also develop a practical heuristic algorithm for compilers to incorporate synthesis strategy into automatic alignment framework. Experiments done on an 8-node DEC Farm have shown that the automatic alignment process with the help of our proposed array operation synthesis and segmented alignment concepts can significantly outperform the one without these two mechanisms [2].

References

1. G. H. Hwang, J. K. Lee, and D. C. Ju. *Array Operation Synthesis to Optimize HPF Programs*, ICPP '96, USA.
2. G. H. Hwang, J. K. Lee, and D. C. Ju. *Integrating Automatic Data Alignment and Array Operation Synthesis to Optimize Data Parallel Programs*, Technical Report, CS Dept., Tsing-Hua Univ., Taiwan, 1997, also in `http://falcon.cs.nthu.edu.tw/ ~ghhwang/papers.html`.
3. S. Chatterjee, J.R. Gilbert, R.Schreiber, and S.-H. Teng. *Automatic Array Alignment in Data-Parallel Programs*, ACM POPL, Jan. 1993.