④ Partial Array Contraction and Contraction-Oriented Loop Fusion

Akira HOSOI[1,2], Toshihiro OZAWA[1,2]

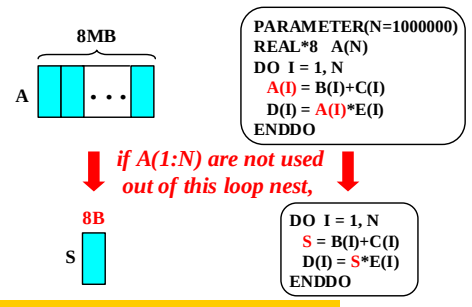
([1] APC Technology Group [2] Fujitsu Limited)

Partial Array Contraction and Contraction-Oriented Loop Fusion

1. Introduction
2. Problems of the previous array contraction and its solutions
3. Evaluation
4. Conclusion

Array Contraction

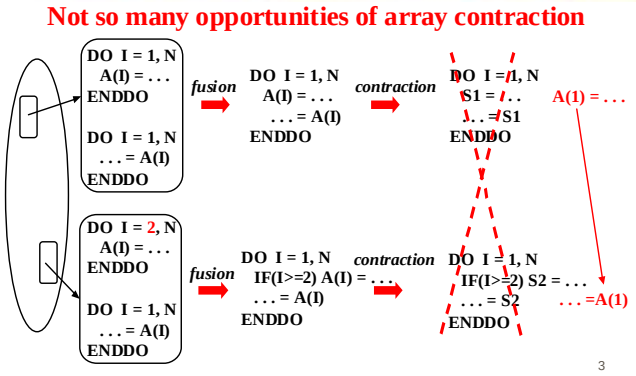
shrinks the size of an array and improves data locality



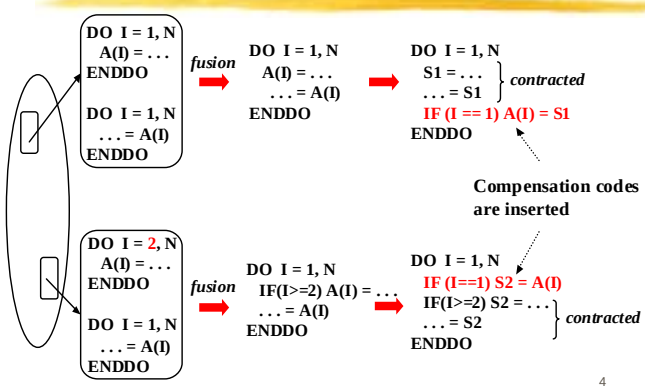
usually combined with *loop fusion*

Problems of the previous array contraction(1)

Not so many opportunities of array contraction



Partial Array Contraction with Save & Restore



Problems of the previous array contraction(2)

It is difficult to decide which loop nests should be fused

```
DO J = 1, N2
  DO I = 1, N1
    A(I, J) = . . .
  ENDDO
ENDDO

DO J = 1, N2
  DO I = 1, N1
    . . . = f(A(I, J), A(N1-I+1, J))
  ENDDO
ENDDO

DO J = 1, N2
  DO I = 1, N1
    A(I, J) = . . .
  ENDDO
ENDDO

DO J = 1, N2
  DO I = 1, N1
    . . . = g(A(I, J), A(I, N2-J+1))
  ENDDO
ENDDO
```

How we should split a loop nest sequence for better contraction ?

Definition: definition / use-dominated

- An array *A* is *definition-dominated* in a loop nest *L*: if no element is used before defined in *L*.

ex.

```
DO I = 1, N
  A(I) = . . .
ENDDO
```

```
DO I = 1, N
  A(I) = . . .
  . . . =A(I)
ENDDO
```

- An array *A* is *use-dominated* in a loop nest *L*: if no element is defined before used in *L*.

ex.

```
DO I = 1, N
  . . . = A(I)
ENDDO
```

```
DO I = 1, N
  . . . = A(I)
  A(I) = . . .
ENDDO
```


How to split a loop nest sequence

If *A* is definition-dominated in *L*, *L* becomes the first loop nest of a new group

```
DO J = 1, N2
  DO I = 1, N1
    A(I, J) = . . .
  ENDDO
ENDDO

DO J = 1, N2
  DO I = 1, N1
    . . . = f(A(I, J), A(N1-I+1, J))
  ENDDO
ENDDO

DO J = 1, N2
  DO I = 1, N1
    A(I, J) = . . .
  ENDDO
ENDDO

DO J = 1, N2
  DO I = 1, N1
    . . . = g(A(I, J), A(I, N2-J+1))
  ENDDO
ENDDO
```


Benchmark Programs

- SPEC CFP2000/applu
- NPB2.3-serial/SP CLASS A
- NPB2.3-serial/BT CLASS A

How to apply Our Method to SP

1. split
2. renaming
   save & restore code are added

```
do 100 k = 1, n3-2
  do 100 j = 1, n2-2
    do 100 i = 1, n1-2
100       lhs(i, j, k, 1:15) = . . .

  do 110 k = 1, n3-2
    do 110 j = 1, n2-2
      do 110 i = 1, n1-2
110       lhs(i, j, k, 1:15)=lhs(i, j, k, 1:15) + . . .

  do 120 k = 1, n3-2
    do 120 j = 1, n2-2
      do 120 i = 0, n1-3
120       lhs(i, j, k, 1:15)= f0(lhs(i, j, k, 1:15),
          lhs(i+1, j, k, 1:15), lhs(i+2, j, k, 1:15))
          . . .
    do 130 k = 1, n3-1
      do 130 j = 1, n2-1
        do 130 i = n1-3, 0, -1
130       . . . = g0(lhs(i, j, k, 1:15),
                  lhs(i+1, j, k, 1:15),
                  lhs(i+2, j, k, 1:15))
```

```
do k = 1, n3-2
  do j = 1, n2-2
    lhs1(0, j, k, 1:15)=lhs(0, j, k, 1:15)
    lhs1(n1-1, j, k, 1:15)=lhs(n1-1, j, k, 1:15)
  enddo
enddo

do k = 1, n3-2
  do j = 1, n2-2
    lhs(0, j, k, 1:15)=lhs1(0, j, k, 1:15)
    lhs(n1-1, j, k, 1:15)=lhs1(n1-1, j, k, 1:15)
  enddo
enddo
```


How to apply Our Method to SP (con't)

3. fusion & 4. partial contraction

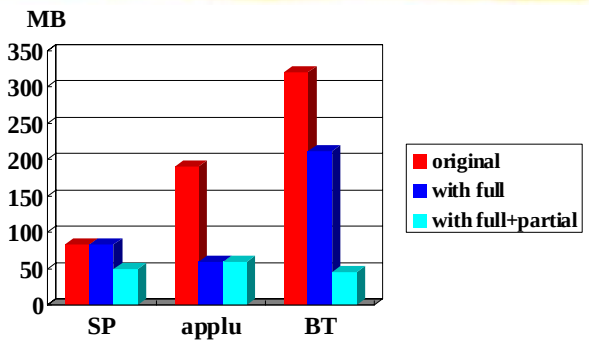
```
do k = 1, n3-2
  do j = 1, n2-2
    lhs1(0, 1:15)=lhs(0, j, k, 1:15)
    lhs1(n1-1, 1:15)=lhs(n1-1, j, k, 1:15)
    do i = 1, n1-2
      lhs1(i, 1:15) = . . .
    enddo
    do i = 1, n1-2
      lhs1(i, 1:15)=lhs1(i, 1:15) + . . .
    enddo
    do i = 0, n1-3
      lhs1(i, 1:15)=f0(lhs1(i, 1:15),
      lhs1(i+1, 1:15), lhs1(i+2, 1:15))
    enddo
    do i = n1-3, 0, -1
      . . . = g0(lhs1(i, 1:15),
                 lhs1(i+1, 1:15),
                 lhs1(i+2, 1:15))
    enddo
    lhs(0, j, k, 1:15)=lhs1(0, 1:15)
    lhs(n1-1, j, k, 1:15)=lhs1(n1-1, 1:15)
  enddo
enddo
```

} *partially contracted*

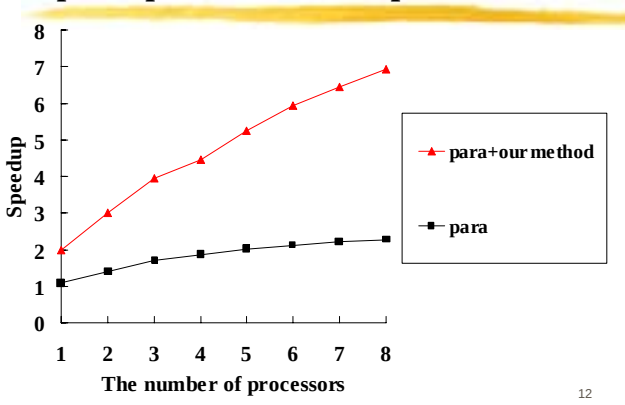
lhs(65, 65, 65, 15) / 33MB

lhs1(65, 15) / 8KB

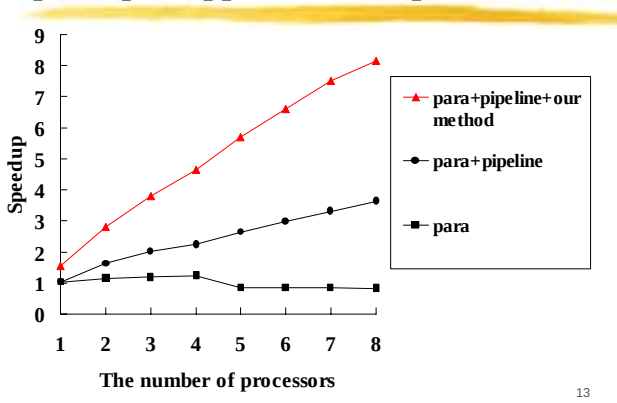
Array data size without/with contraction



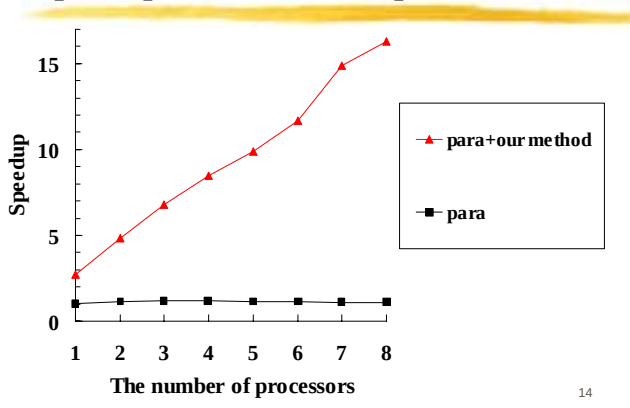
Speedup of SP on the Alpha Server



Speedup of applu on the Alpha Server



Speedup of BT on the Alpha Server



Conclusion

- We generalize array contraction and introduce *partial array contraction with save & restore*
  ➡ We have more opportunities of array contraction
- We split a loop nest sequence using *definition / use-dominated*
  ➡ Array contraction can be applied more effectively
- The performance of SP, applu, and BT are drastically improved