

AE 597: HW #3

Determining Communication speed using *HPF*

Anirudh Modi

March 20, 1998

1 Distribution of Array

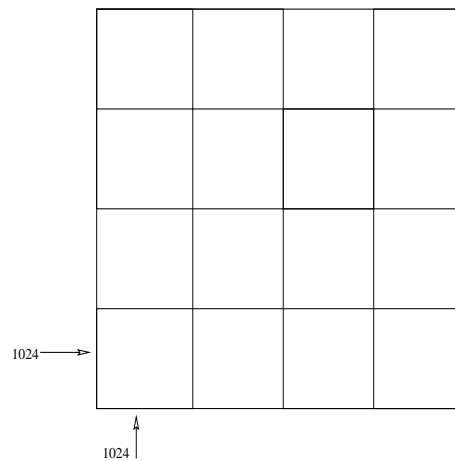


Figure 1: (block,block) distribution of 4096×4096 array among 16 processors. $16 \times 1024 = 16384$ elements being communicated using $\text{CSHIFT}(A, 1, 1)$.

1. As can be easily seen by the figure above, the (block,block) distribution leads to uniform distribution of the array elements among the 16 processors. The elements communicated by using the $\text{CSHIFT}(A, 1, 1)$ in this case is $16 \times 1024 = 16384$.

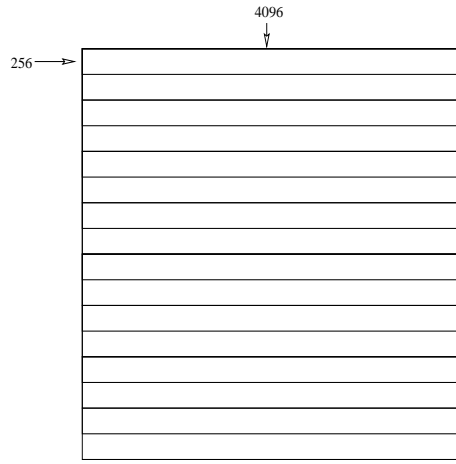


Figure 2: $(*,\text{block})$ distribution of 4096×4096 array among 16 processors. No communication required using $\text{CSHIFT}(A, 1, 1)$.

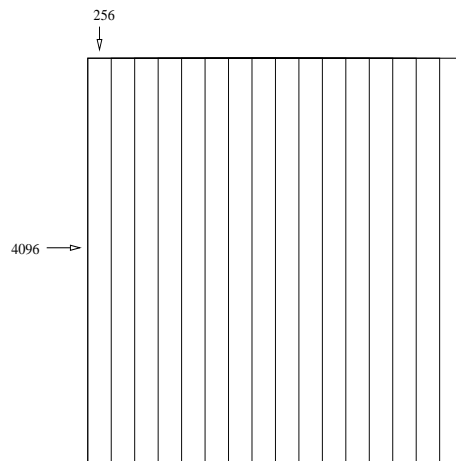


Figure 3: $(\text{block},*)$ distribution of 4096×4096 array among 16 processors. $16 \times 4096 = 65536$ elements being communicated using $\text{CSHIFT}(A, 1, 1)$.

2. For the $(*,\text{block})$ distribution, the elements communicated by using $\text{CSHIFT}(A, 1, 1)$ is 0 as there is no inter-processor communication needed due to the row-wise (or dimension 1) distribution of the array.
3. For the $(\text{block},*)$ distribution, the elements communicated by using $\text{CSHIFT}(A, 1, 1)$ is $16 \times 4096 = 65536$ which is 4 times as in the previous

case.

2 Program listing

The compile command used for the SP2 is:

```
xlhpf comm.f -O3 -qstrict -qarch=pwr2 -qtune=pwr2 -qhot -o comm
```

```
Program Commtime

implicit none

integer n, k
parameter ( n = 4096 )
parameter ( k = 500 )

real*8, dimension(n,n) :: A, B
real*8 time0,time0i,time1,time1i
integer i,j,mclock

!hpf$ processors procs(4,4)
!hpf$ distribute(block,block) onto procs::A
!hpf$ align with A :: B

! fill the array with random numbers between 0. and 1.
  do i = 1, n
    do j = 1, n
      call random_number(A(i,j))
    enddo
  enddo

print *, "Matrix A initialized."

time0 = time1 = 0.0

time0i = real(mclock())/100.0
do i = 1,k
  B = cshift( A, 1, 1 )
!   B = cshift( A, 3, 1 )           ! Case 4
```

```

!      B(1:n-1:1,:) = A(2:n:1,:)          ! Triplet Notation (line 1)
!      B(n,:) = A(1,:)                    ! Triplet Notation (line 2)
      B = A*B
    enddo
    time0 = time0 + real(mclock())/100.0 - time0i
    time0 = time0/(k*1.0)

    time1i = real(mclock())/100.0
    do i = 1,k
      B = A*B
    enddo
    time1 = time1 + real(mclock())/100.0 - time1i
    time1 = time1/(k*1.0)

    time0 = time0 - time1

    print *, ' N :' , n
    print *, ' k :' , k
    print *, ' Time taken per CSHIFT for 16 processors (sec) :' ,time0
    print *, ' Finished.'

    stop
  end

```

3 Results

Here is the timing for 16 processor runs on the NPACI IBM-SP2 (*sp.npaci.edu*).

Case	Parallel run time (sec)	Elements comm.	Bytes comm.
1	0.03526	16384	131072
2	0.03004	0	0
3	0.05720	65536	524288
4	0.03524	16384	131072
5	0.03516	16384	131072

The NPACI SP2 has 128 thin node POWER2 Super Chip (P2SC) processors with 256 MBytes of memory on each processor running at 160 Mhz and are capable of a peak performance of 640 MFLOPS each. It is capable of a peak bi-directional data transfer rate of 110 MB/second between each node pair. Cases 1, 2 and 3 are for CSHIFT($A, 1, 1$) with (block,block), (*,block) and (block,*) distribution respectively. Case 4 is for CSHIFT($A, 3, 1$) with (block,block) distribution and case 5 is for *triplet notation* with (block,block) distribution. The program uses double precision elements in the array each of 8 bytes, hence the bytes communicated is simply the number of array elements communicated multiplied by 8.

Case	Time (msec)	Bytes	MB/sec	% of peak MB/s
1	0.522	131072	25.11	22.8
2	0.000	0	-	-
3	2.716	524288	19.30	17.5
4	0.520	131072	25.21	22.9
5	0.512	131072	25.60	23.3

The communication time is calculated by the difference between the timings of cases 1, 3, 4 and 5 with case 2 as it is the case involving no communication.

4 Conclusion

The communication bandwidth from the above result is seen to be around 25 MB/s which is approximately 23% of the theoretical communication bandwidth of 110 MB/s for the NPACI SP-2. The above exercise also highlights the importance of the optimal distribution of data among the processors such that the communication between them is minimized. It is clear that case 2 is the best case to run for this problem as it requires no interprocessor communication and is hence the fastest and case 3 is the worst as it requires the maximum communication time. The cases 1, 4 and 5 seem to take almost the same time here but may be different on different machines depending how the compiler handles the corresponding HPF commands (i.e. the other CSHIFT command and the triplet notation).