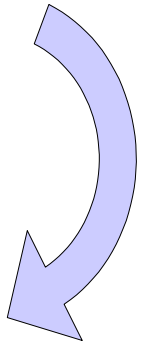
An Automated Approach to Improving Communication-Computation Overlap in Clusters

Lewis Fishgold
Anthony Danalis
Lori Pollock
Martin Swany

University of Delaware

Problem

CPU cycles in large clusters are not fully utilized



- Communication delays impede scalability
- Scientific applications generate heavy communication
- Many scientific codes are written using MPI
- Domain scientists are not expert systems programmers
- Maintainability & simplicity preferred over performance

State of the Art

Parallelizing Compilers

HPF, UPC, Co-Array Fortran, Fortran-D, Split-C, ...

- ✔ No user expertise in PP required
- ✔ Good performance
- ✔ Portable
- ✗ Not ubiquitous
- ✗ Cannot handle MPI
- ✗ Do not fine-tune for cluster
- ✗ User must learn the new language

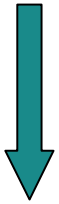
Code Transformation Solutions

Polaris, PARAMAT, PAP

- ✔ No user expertise in PP required
- ✔ Improve performance
- ✔ No new language
- ✗ Cannot handle MPI
- ✗ Limited applicability
- ✗ Sub-optimal performance

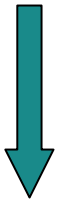
Approach

Automatically Transform scientific source codes



Many domain scientists use small subset of MPI's API and collective communication.

Maximize communication/computation overlapping



Many scientific codes have such data dependencies that asynchronous I/O can be overlapped with computation (pre-pushing).

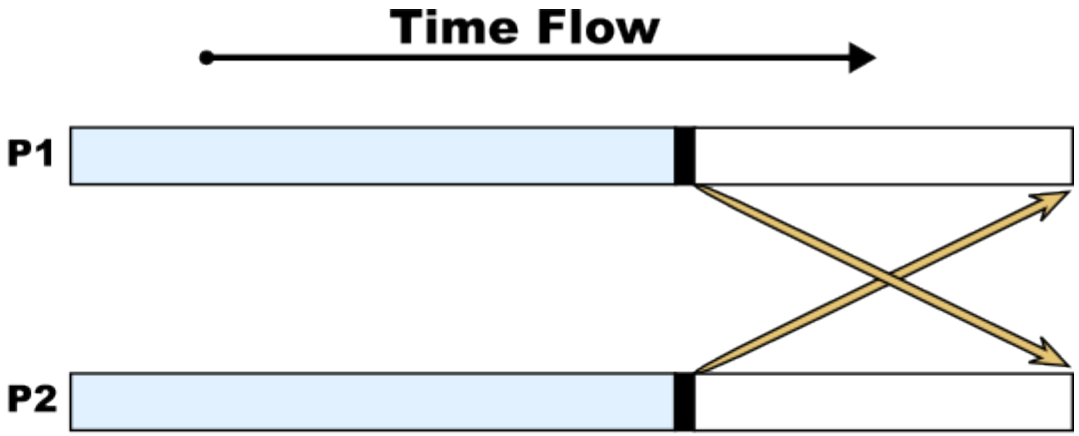
Hide communication latency

Many modern clusters have specialized networks that can overlap communication and computation very efficiently.

"Prepushing" Transformation

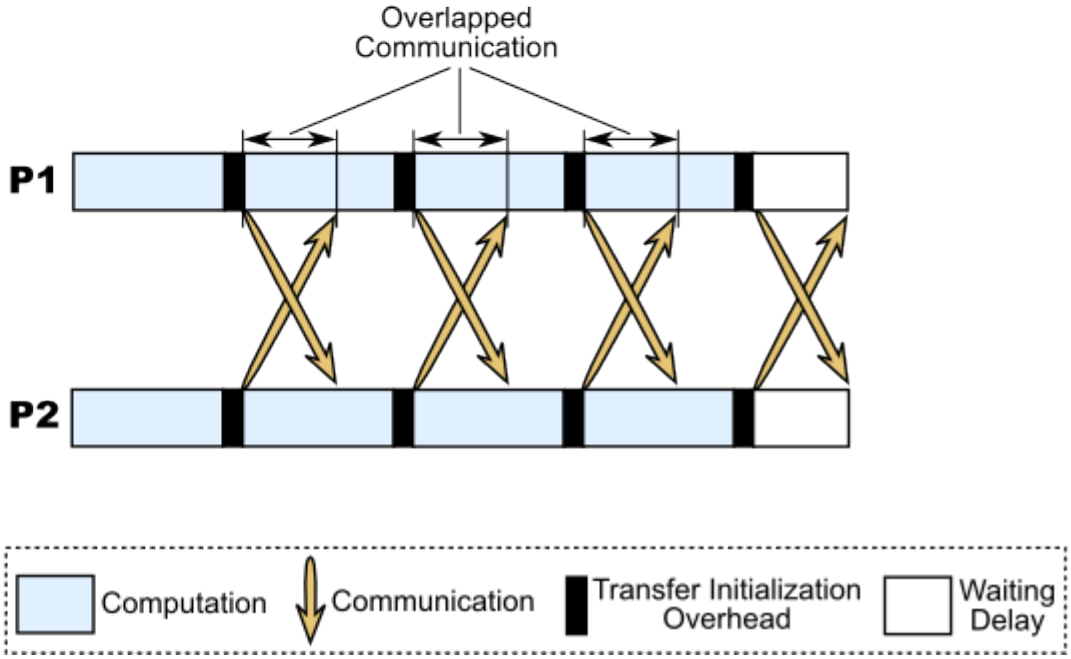
Original code

```
integer, dimension(M,N):: array

do i = 1, N
   /* computation kernel */
   subroutine( array(1,i) )
enddo

size = M*N
DataTransferCall( array(1,1), size, ... )

Other_Computation()
```



Tiled code

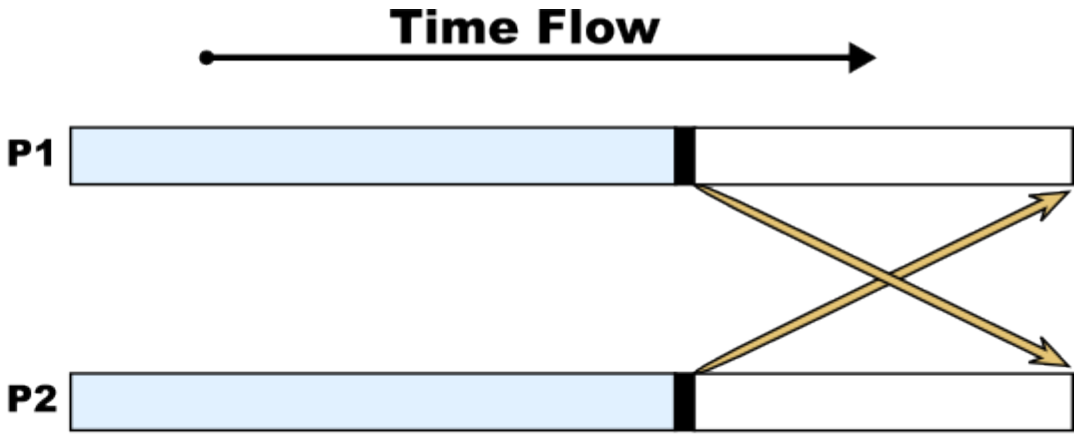
```
integer, dimension(M,N):: array

do i = 1, N, K

   do j = i, i+K-1
      /* computation kernel */
      subroutine( array(1,j) )
   enddo
   if( i > K ) then
      /* block for the arrival of the data  */
      MPI_WAITALL( request( i - K) )
   endif

   size = M*K
   /* asynchronous network transfer */
   MPI_ISEND( array(1,i), size, ... )
   MPI_IRECV( destn(...), request(i), ... )
enddo

MPI_WAITALL( request( i - K) )
```



"Prepushing" Transformation

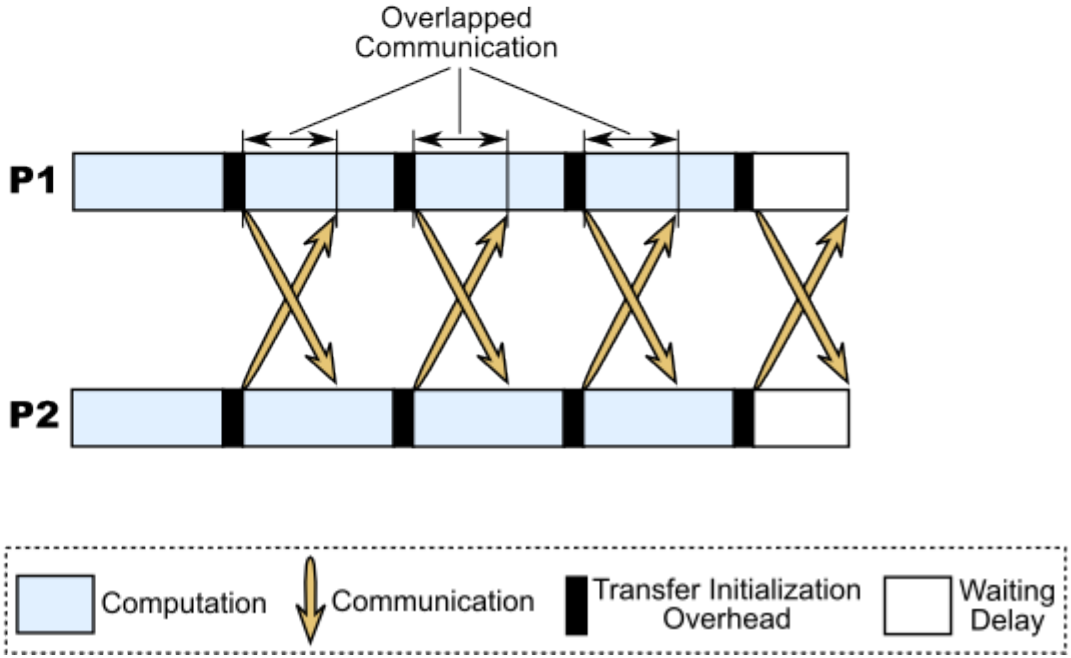
Original code

```
integer, dimension(M,N):: array

do i = 1, N
   /* computation kernel */
   subroutine( array(1,i) )
enddo

size = M*N
DataTransferCall( array(1,1), size, ... )

Other_Computation()
```



Tiled code

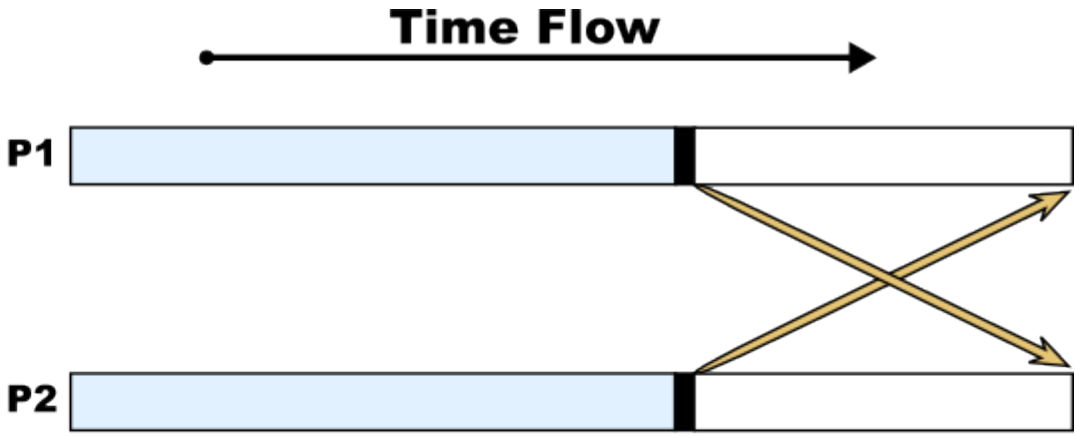
```
integer, dimension(M,N):: array

do i = 1, N, K

   do j = i, i+K-1
      /* computation kernel */
      subroutine( array(1,j) )
   enddo
   if( i > K ) then
      /* block for the arrival of the data  */
      MPI_WAITALL( request( i - K) )
   endif

   size = M*K
   /* asynchronous network transfer */
   MPI_ISEND( array(1,i), size, ... )
   MPI_IRECV( destn(...), request(i), ... )
enddo

MPI_WAITALL( request( i - K) )
```



"Prepushing" Transformation

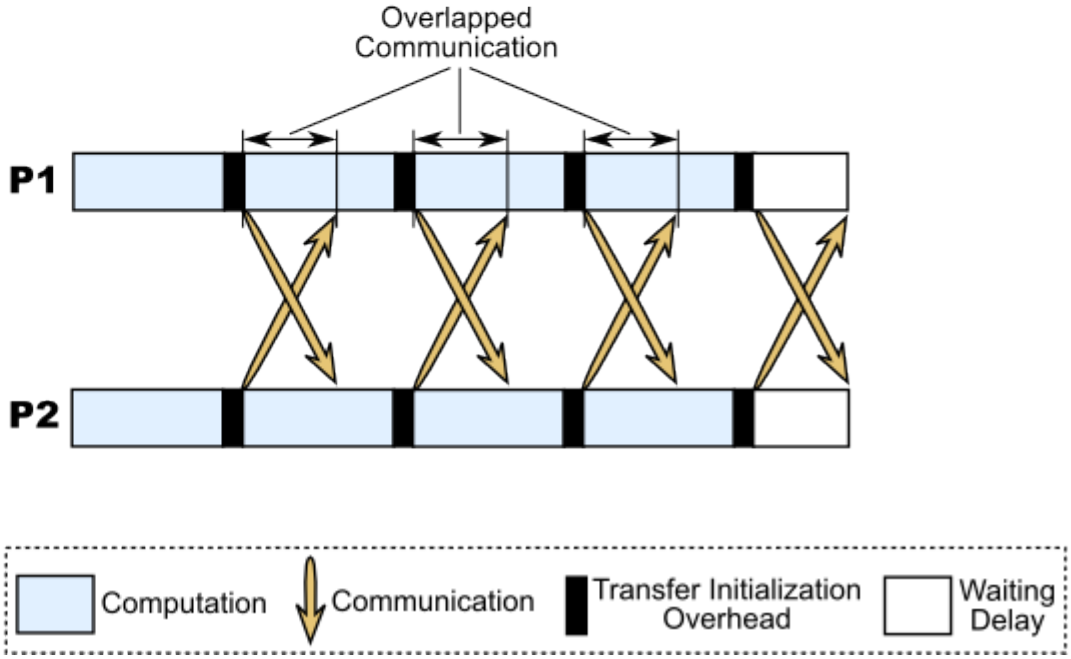
Original code

```
integer, dimension(M,N):: array

do i = 1, N
   /* computation kernel */
   subroutine( array(1,i) )
enddo

size = M*N
DataTransferCall( array(1,1), size, ... )

Other_Computation()
```



Tiled code

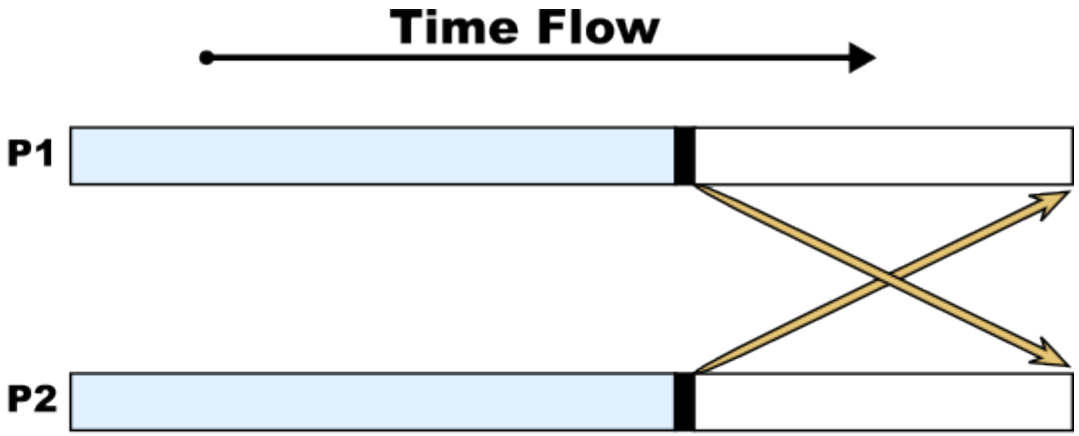
```
integer, dimension(M,N):: array

do i = 1, N, K

   do j = i, i+K-1
      /* computation kernel */
      subroutine( array(1,j) )
   enddo
   if( i > K ) then
      /* block for the arrival of the data  */
      MPI_WAITALL( request( i - K) )
   endif

   size = M*K
   /* asynchronous network transfer */
   MPI_ISEND( array(1,i), size, ... )
   MPI_IRECV( destn(...), request(i), ... )
enddo

MPI_WAITALL( request( i - K) )
```



"Prepushing" Transformation

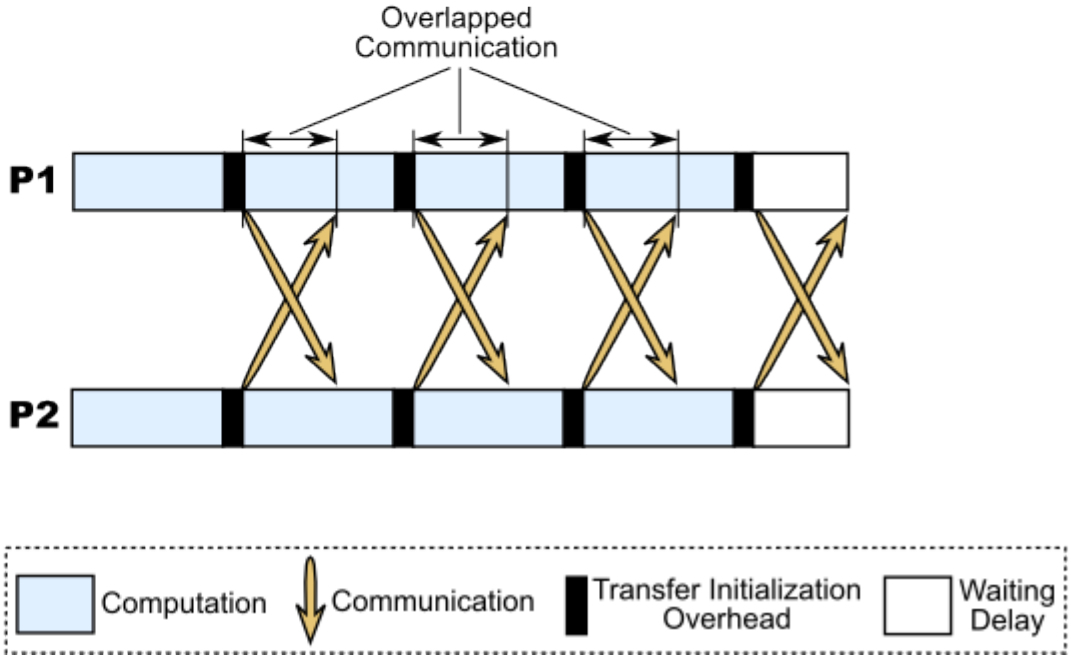
Original code

```
integer, dimension(M,N):: array

do i = 1, N
   /* computation kernel */
   subroutine( array(1,i) )
enddo

size = M*N
DataTransferCall( array(1,1), size, ... )

Other_Computation()
```



Tiled code

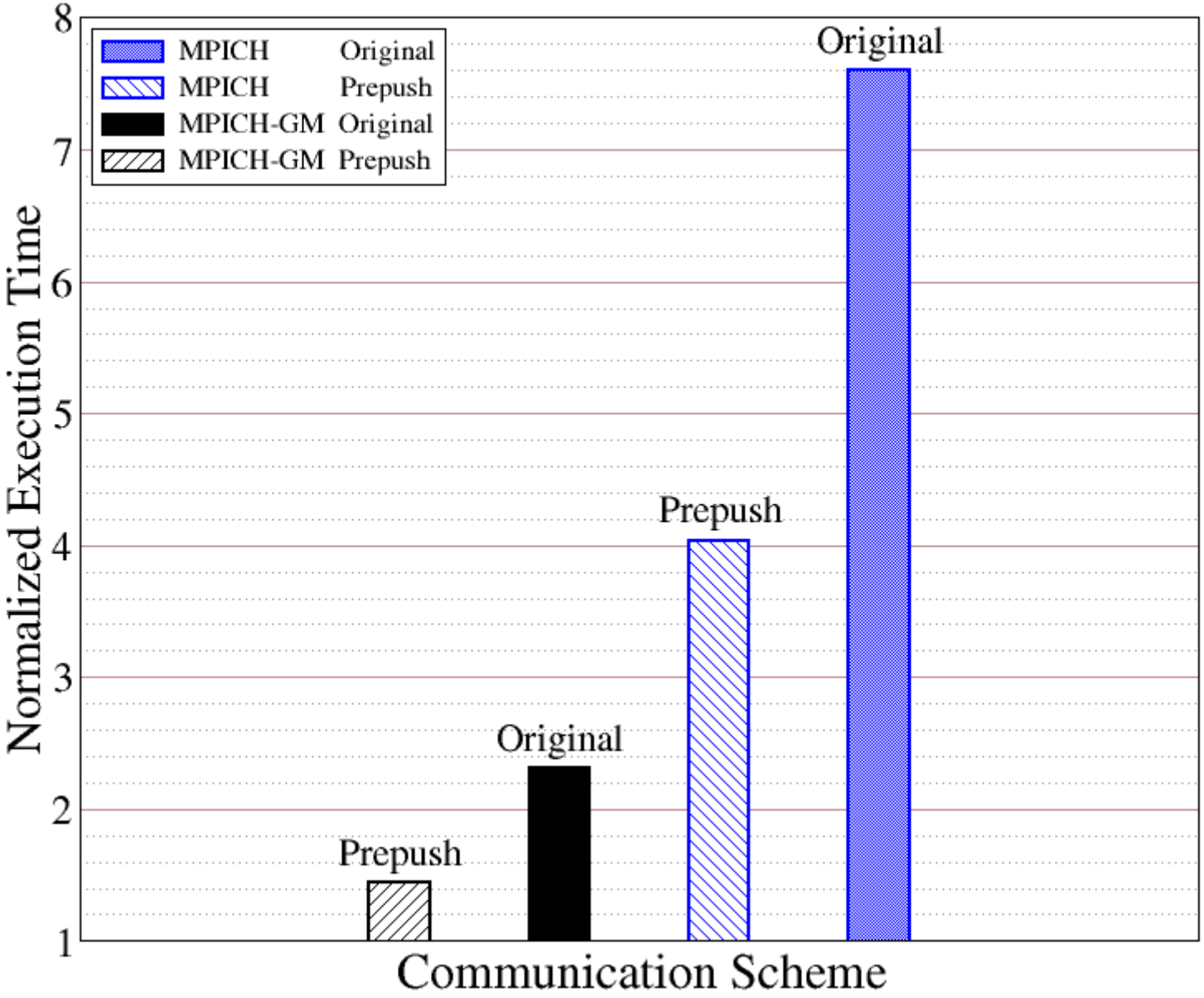
```
integer, dimension(M,N):: array

do i = 1, N, K

   do j = i, i+K-1
      /* computation kernel */
      subroutine( array(1,j) )
   enddo
   if( i > K ) then
      /* block for the arrival of the data  */
      MPI_WAITALL( request( i - K) )
   endif

   size = M*K
   /* asynchronous network transfer */
   MPI_ISEND( array(1,i), size, ... )
   MPI_IRECV( destn(...), request(i), ... )
enddo

MPI_WAITALL( request( i - K) )
```



Empirical Study Setup

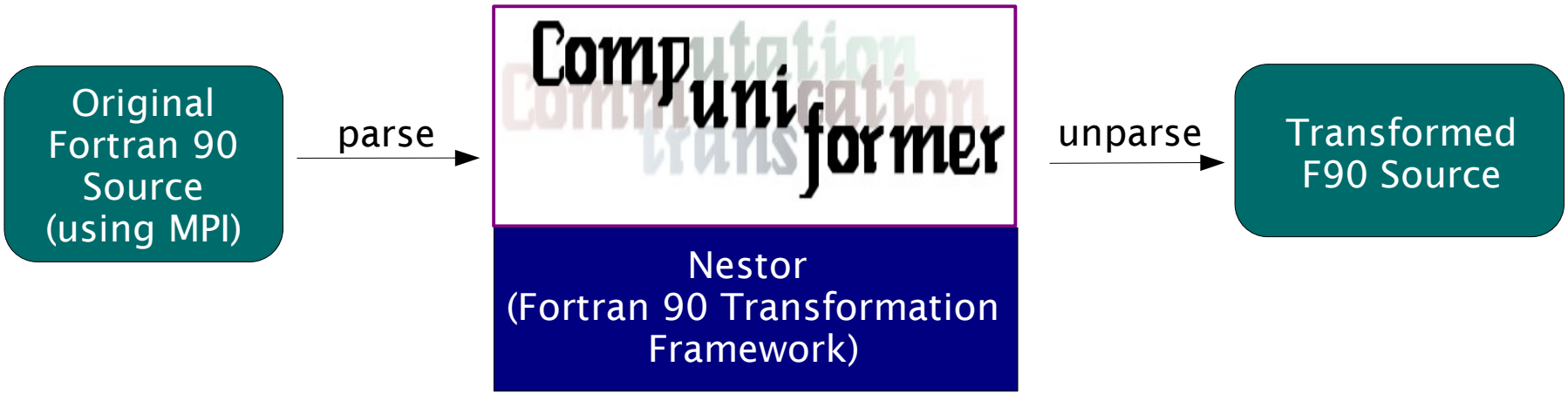
- Two scientific applications as targets
  - Viscoelastic turbulent flow in a channel
  - Magneto-hydrodynamic turbulence through spectral methods
- Transformations performed manually
- All experiments on the same cluster
- Whole parameter space was scanned
- Best result for each scenario is presented

Empirical Study Results



Implementation

Transformer prototype - compuniformer



```
PROGRAM MAIN
!     USE AA
!     include "mpif.h"

      integer I, ix,iy, iz, ip, ixeff, iproc, iproc2
      integer NT, err
      integer input(1:n)
      integer sendBuf(1:n,1:n,1:n,1:np)
      integer result(1:n,1:n,1:n,1:np)
      integer temp(1:n)

      CALL MPI_INIT( ERR )
      call MPI_COMM_RANK(MPI_COMM_WORLD, mynum, ERR)
      CALL MPI_BARRIER(MPI_COMM_WORLD, ERR)

      do I = 1,N
        input(i) = 1
      enddo

      do iz = 1, N
        do iy = 1, N
          call compute(input, temp, iy)
            do ix = 1, N
              IPROC  = 1+(ix-1)/(n/np)
              IXEFF  = ix - (iproc-1)*(n/np)
              sendBuf(ixeff,iy,iz,iproc) = temp(ix)
          enddo
        enddo
      enddo

      do I = 1,N
        input(i) = 1
      enddo

      NT = n*n*(n/np)*np
      call MPI_ALLTOALL(sendBuf, NT, MPI_INTEGER, result, NT, &
                        MPI_INTEGER, MPI_COMM_WORLD,
ERR)
      CALL MPI_BARRIER(MPI_COMM_WORLD, ERR)
      CALL MPI_FINALIZE( ERR )
      STOP
      END PROGRAM MAIN
```

```
integer*4 TEMP(1:N,1:2)
integer*4 requests(1:2*(1*1*1*(1+(N-1)/(N/NP)-(1+(1-1)/(N/NP))+1)))
integer*4 stats(1:MPI_STATUS_SIZE,1:2*(1*1*1*(1+(N-1)/(N/NP)-(1+(1-1)/&
                                                        (N/NP))+1)))

do IZ = 1,N
  do IY = 1,N
    call COMPUTE(INPUT,TEMP(1,MOD(IY,2)+1),IY)
    tempOffset = 1
    reqNum = 1
    k = 1
    if (MOD(IY,k) .eq. 0 .or. N .eq. IY) then
      if (MOD(IY,k) .ne. 0) then
        k = MOD(IY,k)
      end if
      if (IY .gt. K) then
        call MPI_WAITALL(2*(1*1*1*(1+(N-1)/(N/NP)-(1+(1-1)/(N/NP))+1)), &
                                                        requests(1),stats,err)
      end if
      do IPROCIter = 1+(1-1)/(N/NP),1+(N-1)/(N/NP),1
        call MPI_ISEND(TEMP(tempOffset,MOD(IY,2)+1),1*1*(N-(1+(N-1)/(N/NP)- &
          1)*(N/NP)-(1-(1+(1-1)/(N/NP)-1)*(N/NP))+1),MPI_INTEGER,(IPROCIter- &
          1)/(NP/np),1, MPI_COMM_WORLD, requests(reqNum),ERR)

        reqNum = reqNum+1
        if ((IPROCIter-1)/(NP/np) .eq. MYNUM) then
          do from = 0,np-1,1
            call MPI_IRECV(RESULT(1-(1+(1-1)/(N/NP)-1)*(N/NP),IY-k+1,IZ, from &
              *(NP/np)+MOD(IPROCIter,NP/np)+1),1*1*(N-(1+(N-1)/(N/NP)-1)* &
              (N/NP)-(1-(1+(1-1)/(N/NP)-1)*(N/NP))+1),MPI_INTEGER,from,1,  &
              MPI_COMM_WORLD, requests(reqNum), ERR)

            reqNum = reqNum+1
          end do
        end if
        tempOffset = tempOffset+1*1*(N-(1+(N-1)/(N/NP)-1)*(N/NP)-(1-(1+(1-1)/ &
                                                        (N/NP) -1)*(N/NP))+1)
      end do
    end if
  end do
end do
call MPI_WAITALL(2*(1*1*1*(1+(N-1)/(N/NP)-(1+(1-1)/(N/NP))+1)),requests(1), &
                                                        stats,err)
```

```
PROGRAM MAIN
!     USE AA
!     include "mpif.h"

      integer I, ix,iy, iz, ip, ixeff, iproc, iproc2
      integer NT, err
      integer input(1:n)
      integer sendBuf(1:n,1:n,1:n,1:np)
      integer result(1:n,1:n,1:n,1:np)
      integer temp(1:n)

      CALL MPI_INIT( ERR )
      call MPI_COMM_RANK(MPI_COMM_WORLD, mynum, ERR)
      CALL MPI_BARRIER(MPI_COMM_WORLD, ERR)

      do I = 1,N
        input(i) = 1
      enddo

      do iz = 1, N
        do iy = 1, N
          call compute(input, temp, iy)
            do ix = 1, N
              IPROC  = 1+(ix-1)/(n/np)
              IXEFF  = ix - (iproc-1)*(n/np)
              sendBuf(ixeff,iy,iz,iproc) = temp(ix)
          enddo
        enddo
      enddo

      do I = 1,N
        input(i) = 1
      enddo

      NT = n*n*(n/np)*np
      call MPI_ALLTOALL(sendBuf, NT, MPI_INTEGER, result, NT, &
                        MPI_INTEGER, MPI_COMM_WORLD,
ERR)
      CALL MPI_BARRIER(MPI_COMM_WORLD, ERR)
      CALL MPI_FINALIZE( ERR )
      STOP
      END PROGRAM MAIN
```

```
integer*4 TEMP(1:N,1:2)
integer*4 requests(1:2*(1*1*1*(1+(N-1)/(N/NP)-(1+(1-1)/(N/NP))+1)))
integer*4 stats(1:MPI_STATUS_SIZE,1:2*(1*1*1*(1+(N-1)/(N/NP)-(1+(1-1)/&
                                                        (N/NP))+1)))

do IZ = 1,N
  do IY = 1,N
    call COMPUTE(INPUT,TEMP(1,MOD(IY,2)+1),IY)
    tempOffset = 1
    reqNum = 1
    k = 1
    if (MOD(IY,k) .eq. 0 .or. N .eq. IY) then
      if (MOD(IY,k) .ne. 0) then
        k = MOD(IY,k)
      end if
      if (IY .gt. K) then
        call MPI_WAITALL(2*(1*1*1*(1+(N-1)/(N/NP)-(1+(1-1)/(N/NP))+1)), &
                                                        requests(1),stats,err)
      end if
      do IPROCIter = 1+(1-1)/(N/NP),1+(N-1)/(N/NP),1
        call MPI_ISEND(TEMP(tempOffset,MOD(IY,2)+1),1*1*(N-(1+(N-1)/(N/NP)- &
          1)*(N/NP)-(1-(1+(1-1)/(N/NP)-1)*(N/NP))+1),MPI_INTEGER,(IPROCIter- &
          1)/(NP/np),1, MPI_COMM_WORLD, requests(reqNum),ERR)

        reqNum = reqNum+1
        if ((IPROCIter-1)/(NP/np) .eq. MYNUM) then
          do from = 0,np-1,1
            call MPI_IRECV(RESULT(1-(1+(1-1)/(N/NP)-1)*(N/NP),IY-k+1,IZ, from &
              *(NP/np)+MOD(IPROCIter,NP/np)+1),1*1*(N-(1+(N-1)/(N/NP)-1)* &
              (N/NP)-(1-(1+(1-1)/(N/NP)-1)*(N/NP))+1),MPI_INTEGER,from,1,  &
              MPI_COMM_WORLD, requests(reqNum), ERR)

            reqNum = reqNum+1
          end do
        end if
        tempOffset = tempOffset+1*1*(N-(1+(N-1)/(N/NP)-1)*(N/NP)-(1-(1+(1-1)/ &
                                                        (N/NP) -1)*(N/NP))+1)
      end do
    end if
  end do
end do
call MPI_WAITALL(2*(1*1*1*(1+(N-1)/(N/NP)-(1+(1-1)/(N/NP))+1)),requests(1), &
                                                        stats,err)
```

```
PROGRAM MAIN
!     USE AA
!     include "mpif.h"

      integer I, ix,iy, iz, ip, ixeff, iproc, iproc2
      integer NT, err
      integer input(1:n)
      integer sendBuf(1:n,1:n,1:n,1:np)
      integer result(1:n,1:n,1:n,1:np)
      integer temp(1:n)

      CALL MPI_INIT( ERR )
      call MPI_COMM_RANK(MPI_COMM_WORLD, mynum, ERR)
      CALL MPI_BARRIER(MPI_COMM_WORLD, ERR)

      do I = 1,N
        input(i) = 1
      enddo

      do iz = 1, N
        do iy = 1, N
          call compute(input, temp, iy)
            do ix = 1, N
              IPROC  = 1+(ix-1)/(n/np)
              IXEFF  = ix - (iproc-1)*(n/np)
              sendBuf(ixeff,iy,iz,iproc) = temp(ix)
          enddo
        enddo
      enddo

      do I = 1,N
        input(i) = 1
      enddo

      NT = n*n*(n/np)*np
      call MPI_ALLTOALL(sendBuf, NT, MPI_INTEGER, result, NT, &
                        MPI_INTEGER, MPI_COMM_WORLD,
ERR)
      CALL MPI_BARRIER(MPI_COMM_WORLD, ERR)
      CALL MPI_FINALIZE( ERR )
      STOP
      END PROGRAM MAIN
```

```
integer*4 TEMP(1:N,1:2)
integer*4 requests(1:2*(1*1*1*(1+(N-1)/(N/NP)-(1+(1-1)/(N/NP))+1)))
integer*4 stats(1:MPI_STATUS_SIZE,1:2*(1*1*1*(1+(N-1)/(N/NP)-(1+(1-1)/&
                                                        (N/NP))+1)))

do IZ = 1,N
  do IY = 1,N
    call COMPUTE(INPUT,TEMP(1,MOD(IY,2)+1),IY)
    tempOffset = 1
    reqNum = 1
    k = 1
    if (MOD(IY,k) .eq. 0 .or. N .eq. IY) then
      if (MOD(IY,k) .ne. 0) then
        k = MOD(IY,k)
      end if
      if (IY .gt. K) then
        call MPI_WAITALL(2*(1*1*1*(1+(N-1)/(N/NP)-(1+(1-1)/(N/NP))+1)), &
                                                        requests(1),stats,err)
      end if
      do IPROCIter = 1+(1-1)/(N/NP),1+(N-1)/(N/NP),1
        call MPI_ISEND(TEMP(tempOffset,MOD(IY,2)+1),1*1*(N-(1+(N-1)/(N/NP)- &
          1)*(N/NP)-(1-(1+(1-1)/(N/NP)-1)*(N/NP))+1),MPI_INTEGER,(IPROCIter- &
          1)/(NP/np),1, MPI_COMM_WORLD, requests(reqNum),ERR)

        reqNum = reqNum+1
        if ((IPROCIter-1)/(NP/np) .eq. MYNUM) then
          do from = 0,np-1,1
            call MPI_IRECV(RESULT(1-(1+(1-1)/(N/NP)-1)*(N/NP),IY-k+1,IZ, from &
              *(NP/np)+MOD(IPROCIter,NP/np)+1),1*1*(N-(1+(N-1)/(N/NP)-1)* &
              (N/NP)-(1-(1+(1-1)/(N/NP)-1)*(N/NP))+1),MPI_INTEGER,from,1,  &
              MPI_COMM_WORLD, requests(reqNum), ERR)

            reqNum = reqNum+1
          end do
        end if
        tempOffset = tempOffset+1*1*(N-(1+(N-1)/(N/NP)-1)*(N/NP)-(1-(1+(1-1)/ &
                                                        (N/NP) -1)*(N/NP))+1)
      end do
    end if
  end do
end do
call MPI_WAITALL(2*(1*1*1*(1+(N-1)/(N/NP)-(1+(1-1)/(N/NP))+1)),requests(1), &
                                                        stats,err)
```

```
PROGRAM MAIN
!     USE AA
!     include "mpif.h"

      integer I, ix,iy, iz, ip, ixeff, iproc, iproc2
      integer NT, err
      integer input(1:n)
      integer sendBuf(1:n,1:n,1:n,1:np)
      integer result(1:n,1:n,1:n,1:np)
      integer temp(1:n)

      CALL MPI_INIT( ERR )
      call MPI_COMM_RANK(MPI_COMM_WORLD, mynum, ERR)
      CALL MPI_BARRIER(MPI_COMM_WORLD, ERR)

      do I = 1,N
        input(i) = 1
      enddo

      do iz = 1, N
        do iy = 1, N
          call compute(input, temp, iy)
            do ix = 1, N
              IPROC  = 1+(ix-1)/(n/np)
              IXEFF  = ix - (iproc-1)*(n/np)
              sendBuf(ixeff,iy,iz,iproc) = temp(ix)
          enddo
        enddo
      enddo

      do I = 1,N
        input(i) = 1
      enddo

      NT = n*n*(n/np)*np
      call MPI_ALLTOALL(sendBuf, NT, MPI_INTEGER, result, NT, &
                        MPI_INTEGER, MPI_COMM_WORLD,
ERR)
      CALL MPI_BARRIER(MPI_COMM_WORLD, ERR)
      CALL MPI_FINALIZE( ERR )
      STOP
      END PROGRAM MAIN
```

```
integer*4 TEMP(1:N,1:2)
integer*4 requests(1:2*(1*1*1*(1+(N-1)/(N/NP)-(1+(1-1)/(N/NP))+1)))
integer*4 stats(1:MPI_STATUS_SIZE,1:2*(1*1*1*(1+(N-1)/(N/NP)-(1+(1-1)/&
                                                        (N/NP))+1)))

do IZ = 1,N
  do IY = 1,N
    call COMPUTE(INPUT,TEMP(1,MOD(IY,2)+1),IY)
    tempOffset = 1
    reqNum = 1
    k = 1
    if (MOD(IY,k) .eq. 0 .or. N .eq. IY) then
      if (MOD(IY,k) .ne. 0) then
        k = MOD(IY,k)
      end if
      if (IY .gt. K) then
        call MPI_WAITALL(2*(1*1*1*(1+(N-1)/(N/NP)-(1+(1-1)/(N/NP))+1)), &
                                                        requests(1),stats,err)
      end if
      do IPROCIter = 1+(1-1)/(N/NP),1+(N-1)/(N/NP),1
        call MPI_ISEND(TEMP(tempOffset,MOD(IY,2)+1),1*1*(N-(1+(N-1)/(N/NP)- &
          1)*(N/NP)-(1-(1+(1-1)/(N/NP)-1)*(N/NP))+1),MPI_INTEGER,(IPROCIter- &
          1)/(NP/np),1, MPI_COMM_WORLD, requests(reqNum),ERR)

        reqNum = reqNum+1
        if ((IPROCIter-1)/(NP/np) .eq. MYNUM) then
          do from = 0,np-1,1
            call MPI_IRECV(RESULT(1-(1+(1-1)/(N/NP)-1)*(N/NP),IY-k+1,IZ, from &
              *(NP/np)+MOD(IPROCIter,NP/np)+1),1*1*(N-(1+(N-1)/(N/NP)-1)* &
              (N/NP)-(1-(1+(1-1)/(N/NP)-1)*(N/NP))+1),MPI_INTEGER,from,1,  &
              MPI_COMM_WORLD, requests(reqNum), ERR)

            reqNum = reqNum+1
          end do
        end if
        tempOffset = tempOffset+1*1*(N-(1+(N-1)/(N/NP)-1)*(N/NP)-(1-(1+(1-1)/ &
                                                        (N/NP) -1)*(N/NP))+1)
      end do
    end if
  end do
end do
call MPI_WAITALL(2*(1*1*1*(1+(N-1)/(N/NP)-(1+(1-1)/(N/NP))+1)),requests(1), &
                                                        stats,err)
```

```
PROGRAM MAIN
!     USE AA
!     include "mpif.h"

      integer I, ix,iy, iz, ip, ixeff, iproc, iproc2
      integer NT, err
      integer input(1:n)
      integer sendBuf(1:n,1:n,1:n,1:np)
      integer result(1:n,1:n,1:n,1:np)
      integer temp(1:n)

      CALL MPI_INIT( ERR )
      call MPI_COMM_RANK(MPI_COMM_WORLD, mynum, ERR)
      CALL MPI_BARRIER(MPI_COMM_WORLD, ERR)

      do I = 1,N
        input(i) = 1
      enddo

      do iz = 1, N
        do iy = 1, N
          call compute(input, temp, iy)
            do ix = 1, N
              IPROC  = 1+(ix-1)/(n/np)
              IXEFF  = ix - (iproc-1)*(n/np)
              sendBuf(ixeff,iy,iz,iproc) = temp(ix)
          enddo
        enddo
      enddo

      do I = 1,N
        input(i) = 1
      enddo

      NT = n*n*(n/np)*np
      call MPI_ALLTOALL(sendBuf, NT, MPI_INTEGER, result, NT, &
                        MPI_INTEGER, MPI_COMM_WORLD,
ERR)
      CALL MPI_BARRIER(MPI_COMM_WORLD, ERR)
      CALL MPI_FINALIZE( ERR )
      STOP
      END PROGRAM MAIN
```

```
integer*4 TEMP(1:N,1:2)
integer*4 requests(1:2*(1*1*1*(1+(N-1)/(N/NP)-(1+(1-1)/(N/NP))+1)))
integer*4 stats(1:MPI_STATUS_SIZE,1:2*(1*1*1*(1+(N-1)/(N/NP)-(1+(1-1)/&
                                                        (N/NP))+1)))

do IZ = 1,N
  do IY = 1,N
    call COMPUTE(INPUT,TEMP(1,MOD(IY,2)+1),IY)
    tempOffset = 1
    reqNum = 1
    k = 1
    if (MOD(IY,k) .eq. 0 .or. N .eq. IY) then
      if (MOD(IY,k) .ne. 0) then
        k = MOD(IY,k)
      end if
      if (IY .gt. K) then
        call MPI_WAITALL(2*(1*1*1*(1+(N-1)/(N/NP)-(1+(1-1)/(N/NP))+1)), &
                                                        requests(1),stats,err)
      end if
      do IPROCIter = 1+(1-1)/(N/NP),1+(N-1)/(N/NP),1
        call MPI_ISEND(TEMP(tempOffset,MOD(IY,2)+1),1*1*(N-(1+(N-1)/(N/NP)- &
          1)*(N/NP)-(1-(1+(1-1)/(N/NP)-1)*(N/NP))+1),MPI_INTEGER,(IPROCIter- &
          1)/(NP/np),1, MPI_COMM_WORLD, requests(reqNum),ERR)

        reqNum = reqNum+1
        if ((IPROCIter-1)/(NP/np) .eq. MYNUM) then
          do from = 0,np-1,1
            call MPI_IRECV(RESULT(1-(1+(1-1)/(N/NP)-1)*(N/NP),IY-k+1,IZ, from &
              *(NP/np)+MOD(IPROCIter,NP/np)+1),1*1*(N-(1+(N-1)/(N/NP)-1)* &
              (N/NP)-(1-(1+(1-1)/(N/NP)-1)*(N/NP))+1),MPI_INTEGER,from,1,  &
              MPI_COMM_WORLD, requests(reqNum), ERR)

            reqNum = reqNum+1
          end do
        end if
        tempOffset = tempOffset+1*1*(N-(1+(N-1)/(N/NP)-1)*(N/NP)-(1-(1+(1-1)/ &
                                                        (N/NP) -1)*(N/NP))+1)
      end do
    end if
  end do
end do
call MPI_WAITALL(2*(1*1*1*(1+(N-1)/(N/NP)-(1+(1-1)/(N/NP))+1)),requests(1), &
                                                        stats,err)
```

Future Work

- Generalize acceptable input code patterns
- Accept more collective calls
- Investigate matching send() + recv()
- Develop remaining components
- Perform extensive evaluation

Contribution

- Performance
  - Communication overlapping improves performance
- Maintainability
  - The original, simple code is preseved
- Usability
  - No knowledge of the transformation required