CRAY
THE SUPERCOMPUTER COMPANY

Overlapping All-to-All Communication and Computation using non-Blocking MPI and Coarray Fortran

June 24, 2013

Robert Fiedler
Cray, Inc.

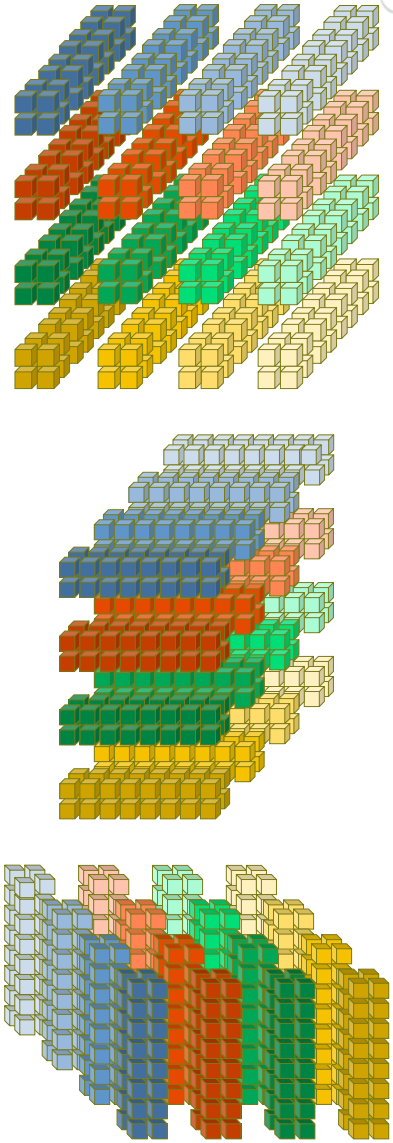
Outline

- **PSDNS turbulence code**
- **Test harness**
  - How performance is measured
- **All-to-All implementations**
  - Blocking
  - Non-blocking
- **Experiments and results**
  - What helps/hurts ability to overlap
- **Conclusions**

PSDNS Algorithm & Performance Model

CFD Using Pseudo-Spectral Method

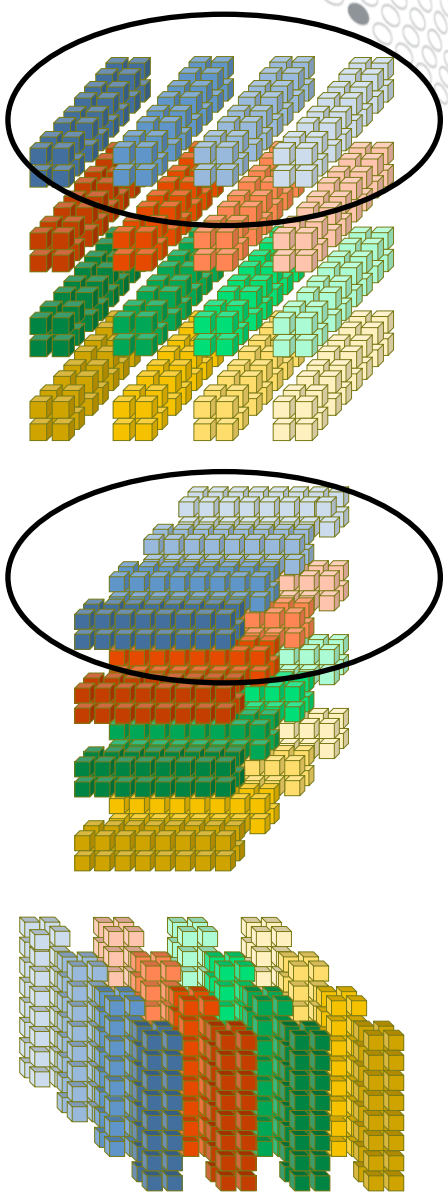
- **Uses 3D FFTs of fluid variables to compute spatial derivatives**
- **Implementation uses 2D pencil decomposition**
- **For 3D FFT, must transpose full 3D arrays twice:**
  - Begin with partitions spanning domain in x
  - 1D FFTs along x
  - Transpose within xy planes so each partition spans domain in y
  - 1D FFTs along y
  - Transpose within yz planes so each partition spans domain in z
  - 1D FFTs along z
- **After some calculations requiring no communication, inverse 3D FFTs are performed in similar fashion**
  - Dozens of forward and inverse 3D FFTs per time step
- **Transposes comprise 50-75% of run time**
  - Compute time includes local field variable updates, packing/unpacking communication buffers, 1D FFTs



Communication Optimizations

Minimize off-node communication

- **Transposes require All-to-All communication within each row (column) of pencils**
  - Multiple concurrent All-to-Alls on all rows (columns), not global All-to-All
- **Eliminate inter-nodal communication for xy transposes**
  - Place 1 or more full xy planes of domain per node
  - Each node has an entire row (16 or 32) of pencils
- **In benchmark runs with a 6k^3 grid on 3072 nodes, this strategy reduced the overall run time by up to 1.72X!**
- **Real science problem for Blue Waters**
  - 8192^3 grid on 8192 XE nodes
  - 4 kB message sizes per variable
  - 5 to 9 independent variables to transform



Test Harness

Purpose & Features

- **Compare different All-to-All implementations**
  - Include some fake computational work
    - Amount is based on measured blocking MPI communication time
    - Same work for all tests with same message size
  - Some implementations attempt overlap
    - Some use non-blocking calls, but put work after synchronization as baseline
- **Alternates row and column communicators like PSDNS**
- **Performs 2 warm-up iterations before keeping score**

Limitations

- **Tests are done using only 1 message size at a time**
  - PSDNS has a range of message sizes, but largest ones dominate

Test Harness

Overall Harness Design

- **Two separate iteration loops for 2 different data structures**
  - Not enough memory for both to exist at same time
- **Important to run all tests for given node count in same batch job**

```
Loop over message sizes
        Deallocate coarrays/Allocate arrays
        Loop over implementations ("ALG") for MPI-like ALGs
                Loop over iterations
                        All-to-All on row communicators
                        All-to-All on column communicators
                End loop on iterations
        End loop over implementations
        Deallocate/Allocate coarrays
        Loop over implementations for CAF-like ALGs
                Loop over iterations
                        All-to-All on row communicators
                        All-to-All on column communicators
                End loop on iterations
        End loop over implementations
End loop over message sizes
```

Blocking MPI and CAF Implementations

- These two methods are implemented in the current PSDNS

```
! MPI
 call mpi_alltoall(sendbuf,items,mpi_byte,
    &                recvbuf,items,mpi_byte,mpi_comm_col,ierr)

! CoM
 call compi_alltoall(sendbuf,recvbuf,items,mpi_comm_col)
```

- CoM uses an internal coarray buffer
- Breaks messages into 512 B chunks, randomizes get order
- Has internal calls to MPI_Barrier, so is blocking
- NEW: overlaps communication for one chunk with copy for next chunk

Simplified CAF All-to-All ("CoM") Pseudo-Code

```
! My image is my_im

Do i=1,n_chunks  !  Number of 512 Byte chunks in messages
        i_start = 1 + (i-1)*64  ! 8 Bytes per word
        Do j=1,n_images  ! Number of images
                co_bucket(1:64, j) = sendbuf(i_start:i_start-1+64, j)
        End do ! images
        MPI barrier (communicator)  ! Finish copy before sharing
        Do j=1,n_images
                Set k = random_order ( j )
                recvbuf(i_start:i_start-1+64, k) =
                        co_bucket(1:64,my_im)[k] ! Get from remote img.
        End do ! images
        Sync memory  ! Ensures compiled code gives correct results
        MPI barrier (communicator)
End do ! chunks
```

NEW: Overlapping CAF All-to-All ("CoM")

```
Do i=2,n_chunks-1  !  Most of the 512 Byte chunks in message
        i_sm = 1 + (i-1)*64  ! Current  chunk
        i_sp = 1+i*64        ! Next  chunk
        Do j=1,n_images    ! Number of images
            Set k = random_order ( j )
            If (i is odd) then
                recvbuf(i_sm:i_sm-1+64, k) = co_bucketm(1:64,my_im)[k]
                co_bucketp(1:64, j) = sendbuf(i_sp:i_sp-1+64, j)
            else
                recvbuf(i_sm:i_sm-1+64, k) = co_bucketp(1:64,my_im)[k]
                co_bucketm(1:64, j) = sendbuf(i_sp:i_sp-1+64, j)
            endif
        End do ! images
        MPI barrier (communicator)  ! Finish copy of chunk before sharing
End do ! chunks
```

Non-Blocking MPI Implementations

MPI code “MIB”: Work is done AFTER the Wait to get baseline timings

```
call mpix_lalltoall (sendbuf, size, mpi_double, &
     recvbuf, size, mpi_double, mpi_comm_col, req, ierr)
call mpi_wait(req, MPI_STATUS_IGNORE, ierr)
call do_work(t_comm_col,t_comp_col,i_comp_col,sum_work,size,ncol)
```

MPI code “MIN”: Work is done BEFORE the Wait to get overlap

```
call mpix_lalltoall (sendbuf, size, mpi_double, &
     recvbuf, size, mpi_double, mpi_comm_col, req, ierr)
call do_work(t_comm_col,t_comp_col,i_comp_col,sum_work,size,ncol)
call mpi_wait(req, MPI_STATUS_IGNORE, ierr)
```

CAF Implementations w/large Coarray Buffer

CAF code "CrN"

```
! Array cols(j) is a different random ordering on each image

            sync all

            do j=1,ncol
              j1 = cols(j)
              j_st = 1 + (j1-1)*size
              j_en = j_st + size - 1
              i_st = 1 + (mycol-1)*size
              i_en = i_st + size - 1
!dir$ pgas defer_sync
              co_rbuf (j_st:j_en) =  &
              co_sbuf (i_st:i_en) [myrow,j1]
            enddo

            call do_work(t_comm_row,t_comp_row,i_comp_row,sum_work,size,nrow)

            sync all
```

- Note: "CrB" method calls do_work AFTER last sync all, to get baseline

CAF Implementation w/ In-lined Work ("CsN")

```
sync all
            sum_work = 0.

             do j=1,ncol  ! Images in column communicator
                j1 = cols(j)
                j_st = 1 + (j1-1)*size
                j_en = j_st + size - 1
                i_st = 1 + (mycol-1)*size
                i_en = i_st + size - 1
! Work
                do iw=1,i_comp_col
                  do kw = 1,125*max(1,size/1024)
                    sum_work = sum_work + sin(j*(pi/kw)) + cos(kw*(pi/j))
                  enddo
                enddo

!dir$ pgas defer_sync
                co_rbuf (j_st:j_en) =  co_sbuf (i_st:i_en) [myrow,j1]
             enddo

sync all
```

- Note: "CsB" method does work loop separately AFTER sync all

Sample Results on 8k Nodes

NODES = 8192, PPN=16
module load craype-hugepages8M
setenv XT_SYMMETRIC_HEAP_SIZE 800M
setenv MPICH_GNI_MAX_EAGER_MSG_SIZE 1536
Setenv PGAS_OFFLOAD_THRESHOLD 1536
setenv MPICH_NEMESIS_ASYNC_PROGRESS 1
setenv MPICH_MAX_THREAD_SAFETY "multiple"

aprun -n 131072 -N 16 -cc 0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30 -r 1 ./overlap.exe

ALG	Bytes	iters	t_total_row	t_comp_row	t_total_col	t_comp_col
MPI	4096	1	0.393295E-02	0.948906E-03	23.3572	7.47749
CoM	4096	1	0.106461E-01	0.109911E-02	16.1125	7.66926
MIB	4096	1	0.290799E-02	0.108695E-02	15.8076	7.68819
MIN	4096	2	0.232315E-02	0.107300E-02	8.33958	7.93174
CrB	4096	1	0.361204E-02	0.116587E-02	14.6526	7.68633
CrN	4096	1	0.298595E-02	0.100684E-02	13.8591	7.68942
CsB	4096	1	0.843406E-02	0.110912E-02	14.4864	7.68456
CsN	4096	2	0.401115E-02	0.00000	7.71479	0.00000

Sample Results on 4k Nodes

NODES = 4096, PPN=16
module load craype-hugepages8M
setenv XT_SYMMETRIC_HEAP_SIZE 800M
setenv MPICH_GNI_MAX_EAGER_MSG_SIZE 1536
Setenv PGAS_OFFLOAD_THRESHOLD 1536
setenv MPICH_NEMESIS_ASYNC_PROGRESS 1
setenv MPICH_MAX_THREAD_SAFETY "multiple"

aprun -n 65536 -N 16 -cc 0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30 -r 1 ./overlap.exe

ALG	Bytes	iters	t_total_row	t_comp_row	t_total_col	t_comp_col
MPI	4096	2	0.123143E-02	0.249028E-03	6.70842	2.12199
CoM	4096	3	0.125106E-02	0.255187E-03	4.12184	2.12414
MIB	4096	2	0.697970E-03	0.245929E-03	5.95609	2.15250
MIN	4096	3	0.448926E-02	0.245333E-03	3.92301	2.56996
CrB	4096	3	0.116738E-02	0.228564E-03	4.39653	2.09289
CrN	4096	4	0.104654E-02	0.317812E-03	2.50166	2.21423

Sample Results on 4k Nodes, Very Little Work

NODES = 4096, PPN=16
module load craype-hugepages8M
setenv XT_SYMMETRIC_HEAP_SIZE 800M
setenv MPICH_GNI_MAX_EAGER_MSG_SIZE 8192
Setenv PGAS_OFFLOAD_THRESHOLD 4096
setenv MPICH_NEMESIS_ASYNC_PROGRESS 1
setenv MPICH_MAX_THREAD_SAFETY "multiple"

aprun -n 65536 -N 16 -cc 0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30 -r 1 ./overlap.exe

ALG	Bytes	iters	t_total_row	t_comp_row	t_total_col	t_comp_col
MPI	4096	3	0.156434E-02	0.709693E-04	4.81714	0.161830E-01
CoM	4096	5	0.116262E-02	0.678062E-04	2.09461	0.191511E-01
MIB	4096	4	0.309426E-02	0.672936E-04	2.92928	0.162450E-01
MIN	4096	4	0.142521E-02	0.700355E-04	2.97274	0.155610E-01
CrB	4096	5	0.204535E-02	0.658035E-04	2.40521	0.155405E-01
CrN	4096	5	0.162406E-02	0.627995E-04	2.37622	0.187273E-01
CsB	4096	5	0.249982E-02	0.673771E-04	2.39900	0.162328E-01
CsN	4096	5	0.194736E-02	0.00000	2.34259	0.00000

Conclusions and Next Steps

- **Non-blocking MPI All-to-All improves total run times compared to standard blocking CAF for node counts and message sizes of interest to the Turbulence science team**
  - Must reduce eager limit to utilize Block Transfer Engine
- **Non-blocking CAF implementation CrN best on 4k nodes**
  - Does not get good overlap on 8k nodes
  - Setting PGAS_OFFLOAD_THRESHOLD gives overlap for 2 kB msg.
- **CAF with in-lined work is best of all**
  - Not practical for use with PSDNS
  - Similar modifications would enable use of non-blocking CAF
- **New CAF library with overlap of extra copy is promising**
  - Direct comparison with prior version needed
- **Next: Evaluate non-blocking implementations in PSDNS**


CRAY
THE SUPERCOMPUTER COMPANY