Evaluating CAF with the CGPOP Miniapp

Andrew Stone

Colorado State University

John Dennis (NCAR)

Michelle Strout (Colorado State)

NCAR

Colorado State University

Outline

Using PGAS in POP?

The CGPOP Miniapp

Evaluating CAF

Outline

Using PGAS in POP?

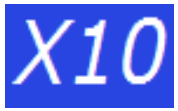
The CGPOP Miniapp

Evaluating CAF

Motivating Example

Ct

Cilk



STAPL

Map-Reduce

- POP is an Ocean simulation code.
- Currently Fortran + MPI
- About 70 KLOC
- Should PGAS be used in POP to improve maintainability?

Adoption affected by:

- Need to work with existing POP code
- Need to maintain high performance
- Whether a model is shown to improve similar code
- The experience developers have with the model

Examining Existing Benchmarks

Benchmarks are small (in terms of SLOC) programs that perform some common type of computation.

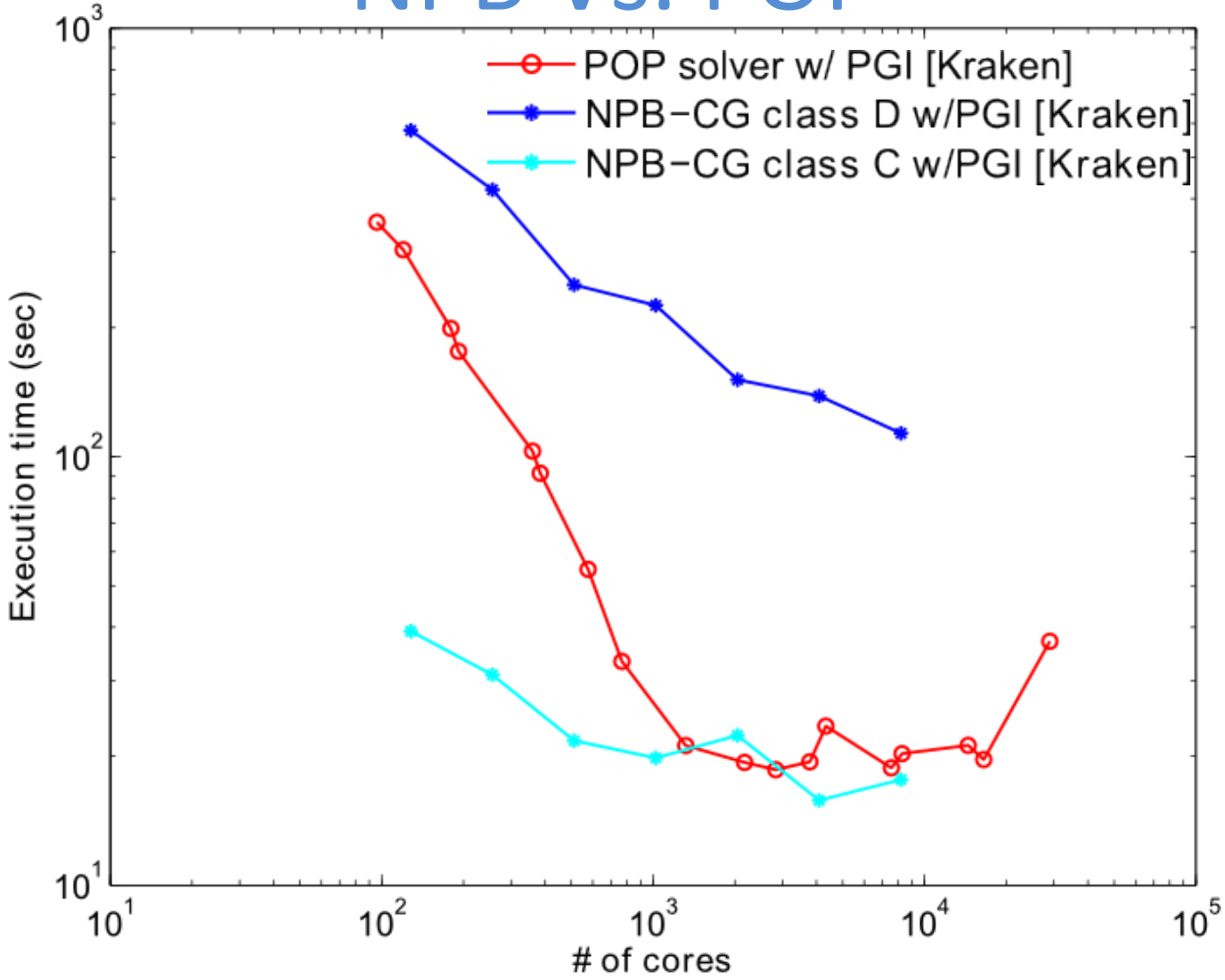
High-performance LINPACK
NAS parallel benchmarks
HPC Challenge benchmarks

The main computation in POP is its conjugate gradient solver

So maybe we should see how well programming models implement NAS CG?

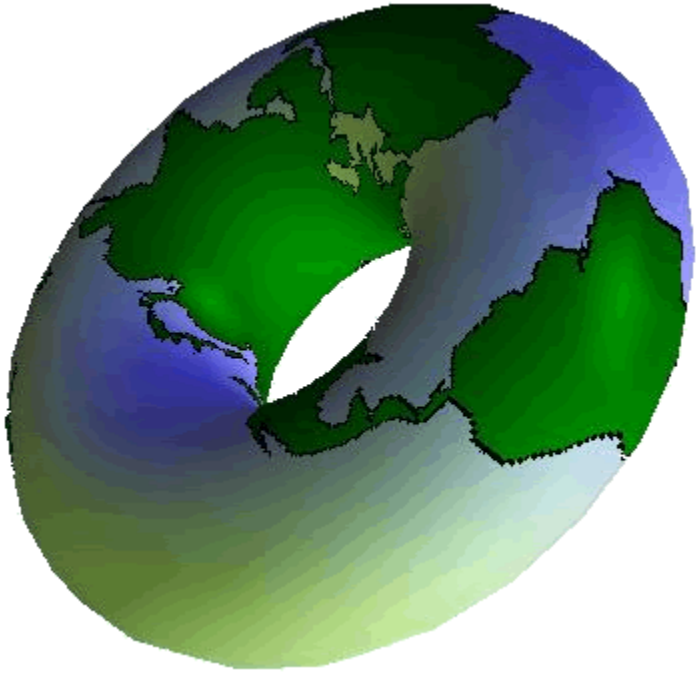
But does NAS CG accurately model POP?

NPB Vs. POP



NAS does not model the performance of POP's CG solver

How NAS and POP differ



POP has:

Specific boundary conditions

Specific data-distribution and data-structures to represent distribution

Application specific optimizations (don't store/compute land points for an Ocean simulation)

Solution: A Miniapp!

Small pieces of code (1k - 10k SLOC) extracted from POP.

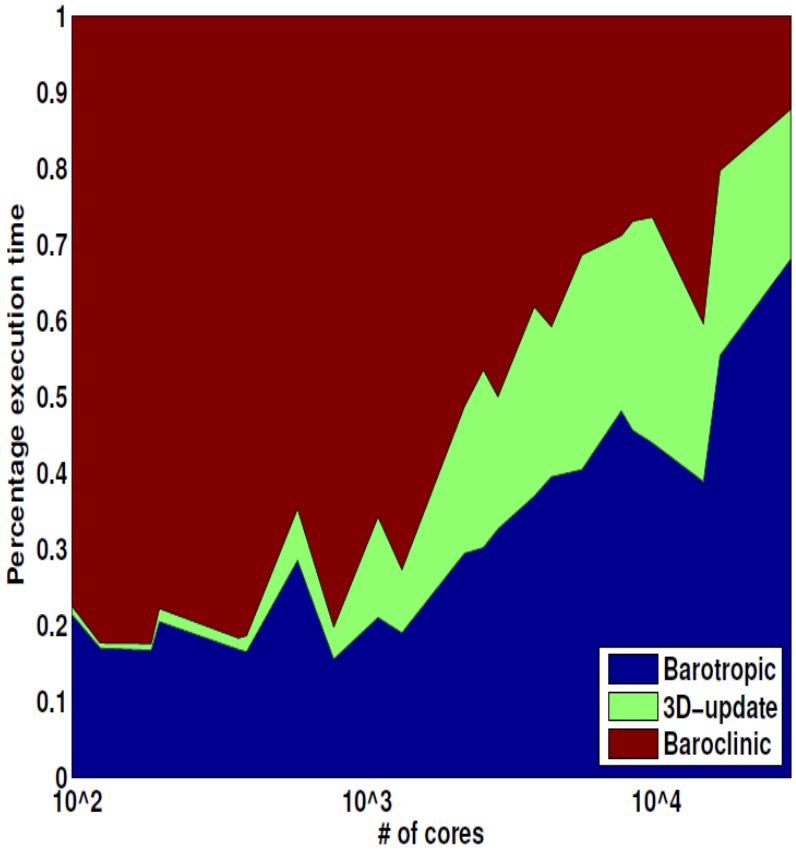
Outline

Evaluating Programming Models

The CGPOP Miniapp

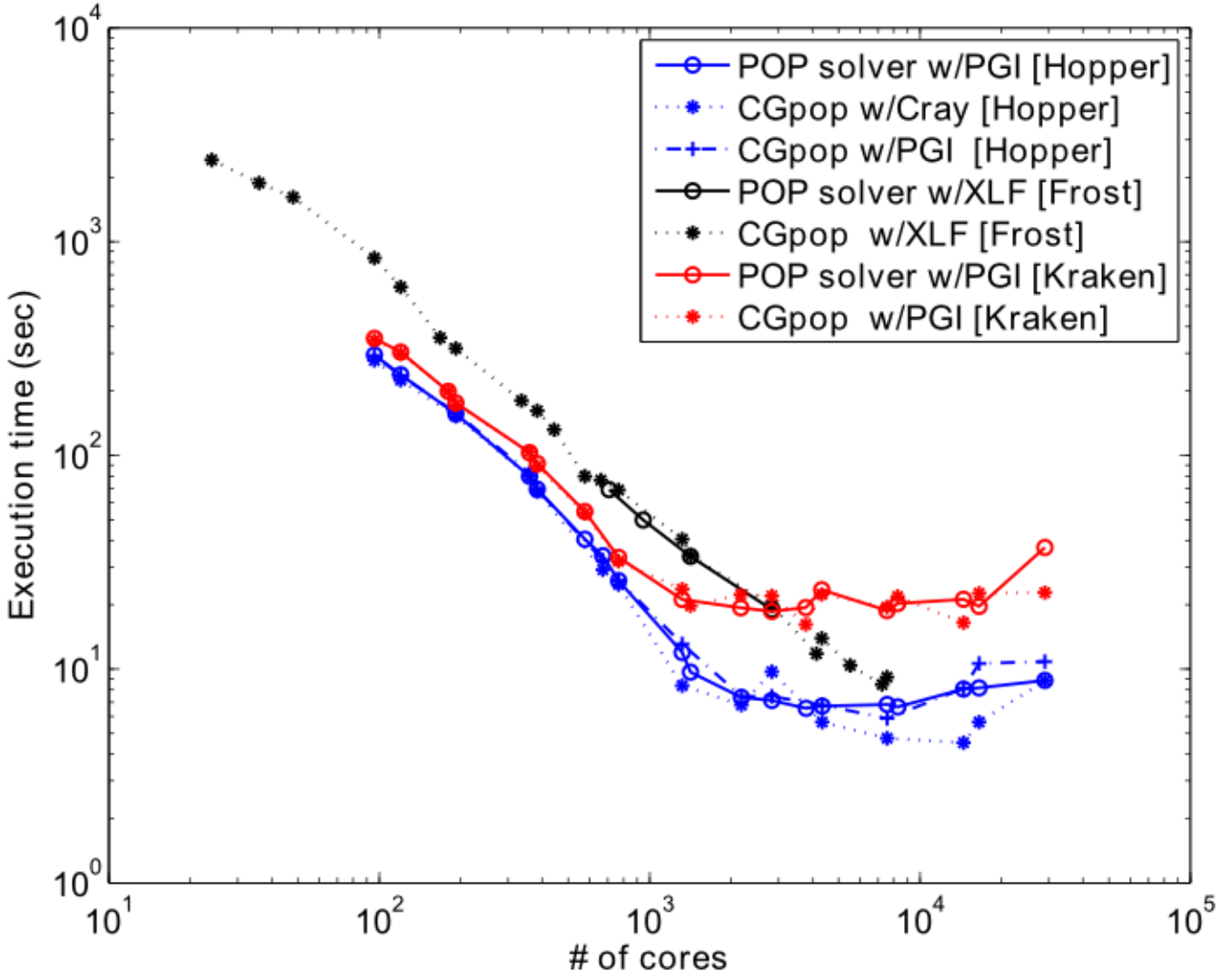
Evaluating CAF

Developing a Miniapp for POP



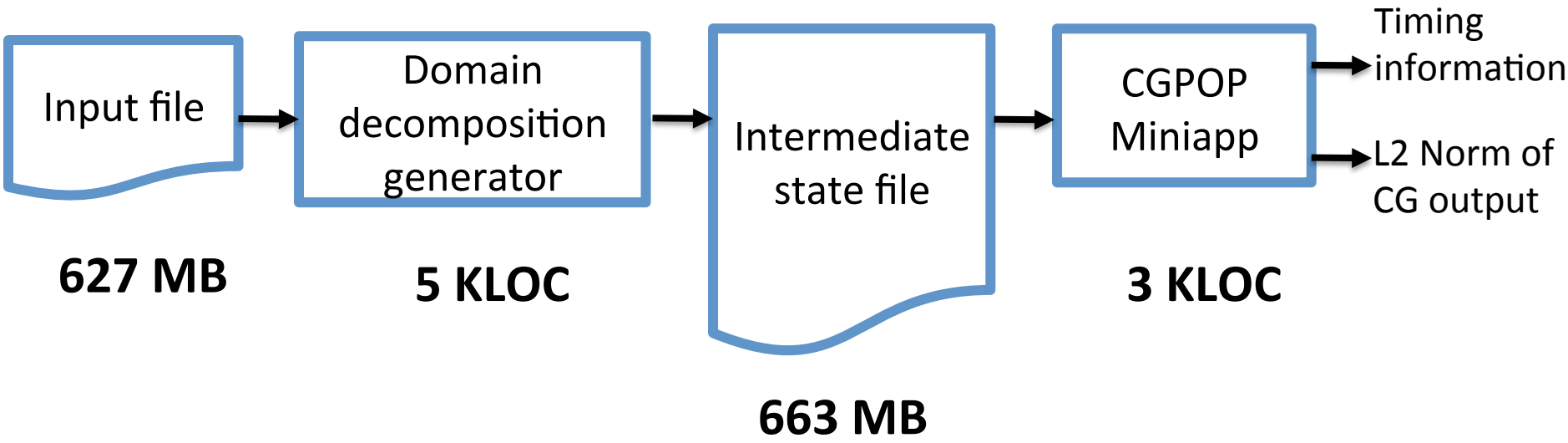
- 3600 x 2400 grid (strong scaling)
- Relative cost of the baroclinic component dominates at fewer than 10000 cores
- Barotropic and 3D-update dominate at greater than 10000 cores
- We extracted CGPOP from the barotropic component, and the conjugate gradient algorithm within it specifically.

CGPOP as a Performance Proxy



CGPOP Architecture

CGPOP is 3KLOC, how did we get it down to that size:



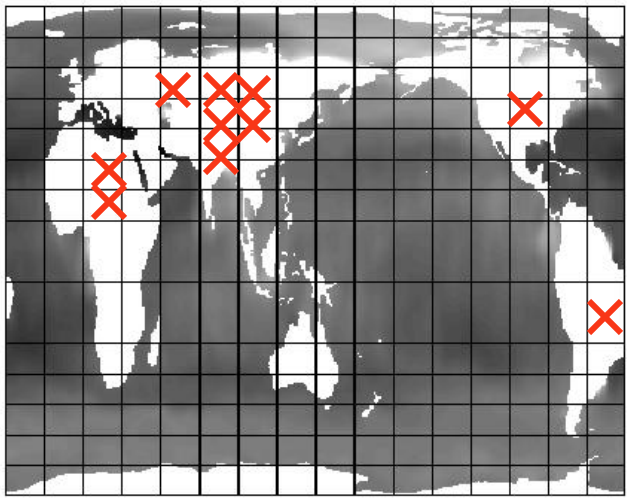
Outline

Evaluating Programming Models

The CGPOP Miniapp

Evaluating CAF

Two primary versions of CGPOP



Two versions:

- World is conceptually big 2D array
- World split into blocks
- Processes take ownership of blocks
- Land blocks are eliminated

In 2d version:

- 2D array
- Neighboring points stored in ghost cells

In 1d version:

- 1D array with indirect access
- Only ocean points are stored
- Neighboring points stored in halo

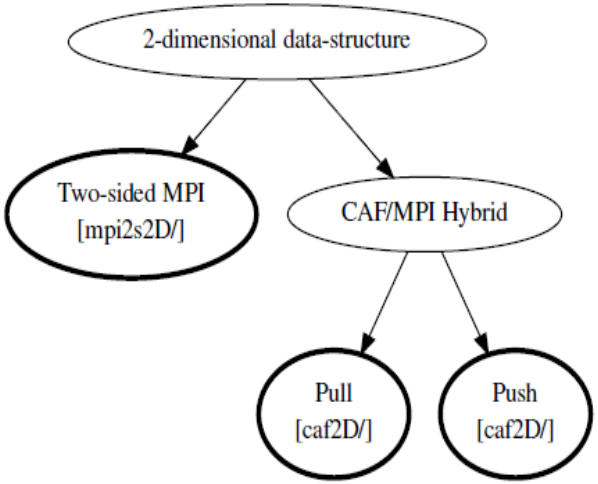
Neither variant is best on all platforms

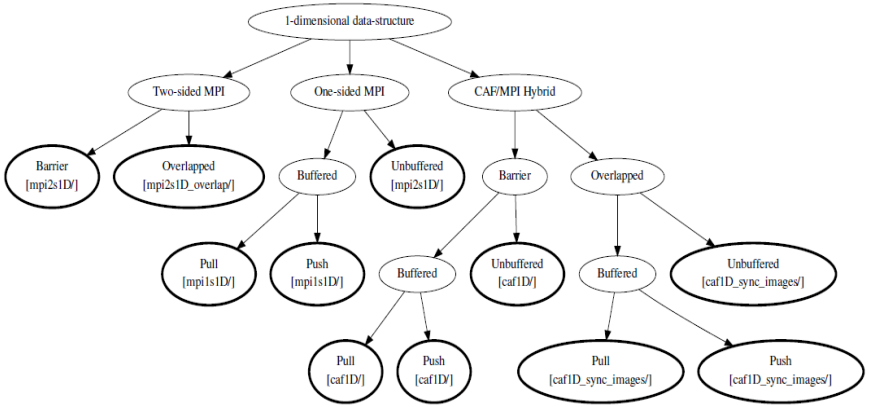
Many possible CAF implementations…

Differ on how they conduct the communication routine

- Buffer data or don't?
- Push data or pull?
- Overlap computation/communication or don't?

It's easier to explore this space with miniapp than fullapp





An issue with all CAF versions

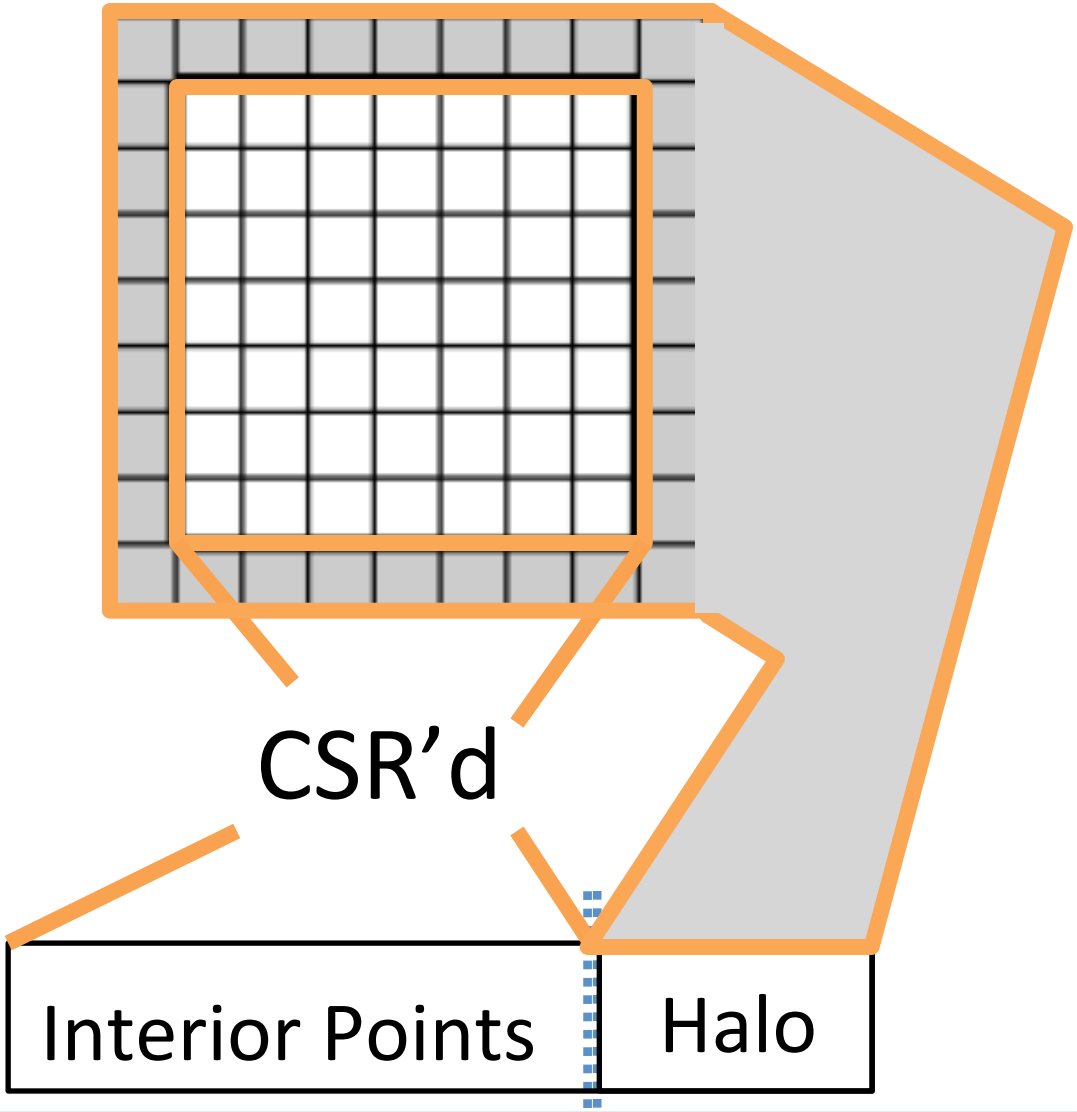
With 2-sided communication:

Sender	Receiver
Receiving proc ID	Sending proc ID
Address to send	Address to receive

With 1-sided pull communication:

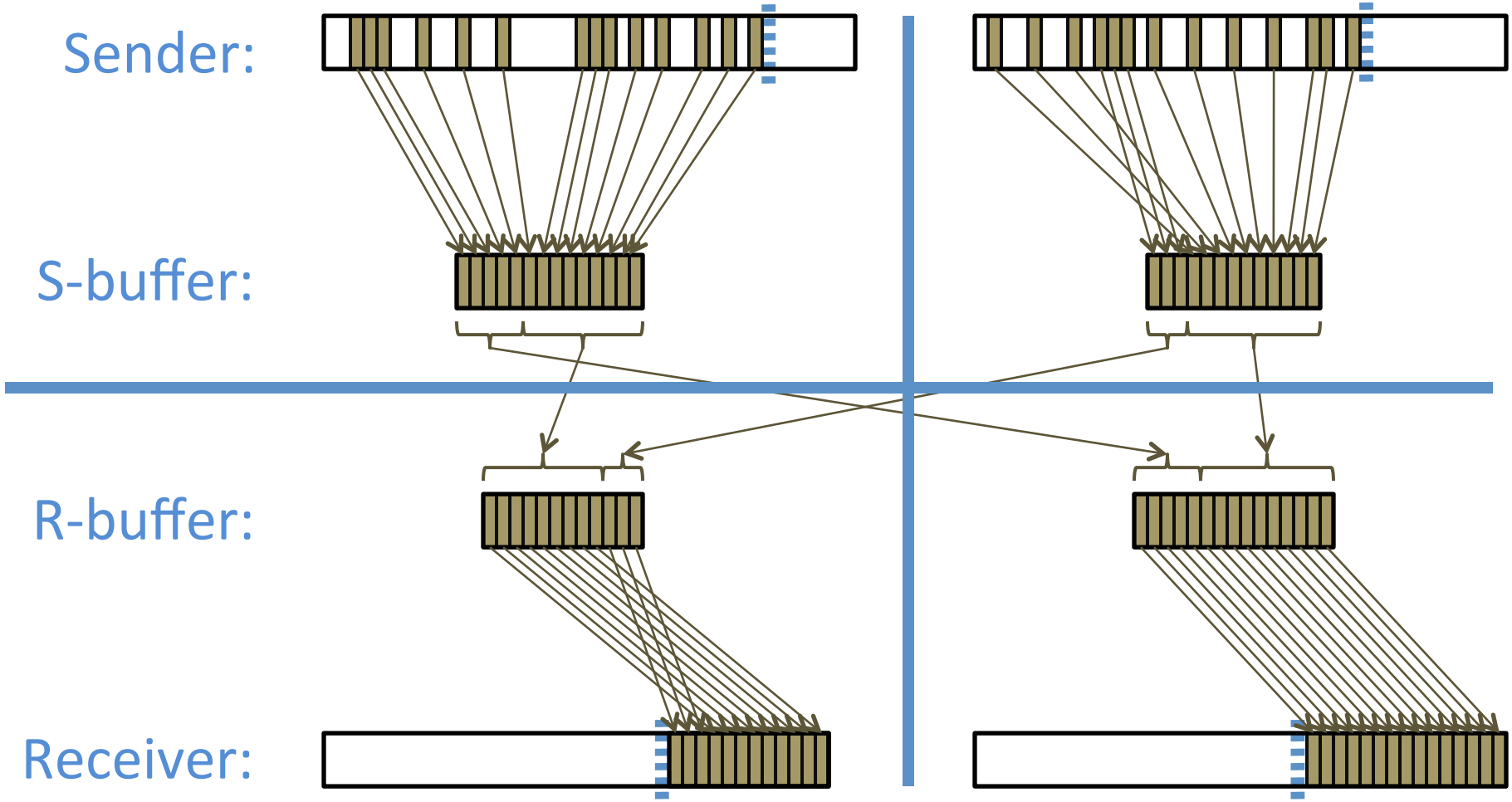
Sender	Receiver
	Sending proc ID
	Address to pull from
	Address to place into

2D and 1D Data-Structures



Initial MPI implementation

Communication to populate halo



Simple CAF Code for UpdateHalo

```
subroutine UpdateHalo ( array )
    ! ** Input parameters and local variables:   **
    real , intent(inout) :: array (:)[:]
    integer :: I      ! dummy counter

    sync all

    !***
    ! iterate through halo elements , grabbing fresh
    ! values from remote images.
    !***
    do i=startOfHaloIdx , endOfHaloIdx
        array(i) = array(halo2grab(i))[haloOnProc(i)]
    enddo
    sync all
end subroutine UpdateHalo
```

Need for Data Aggregation

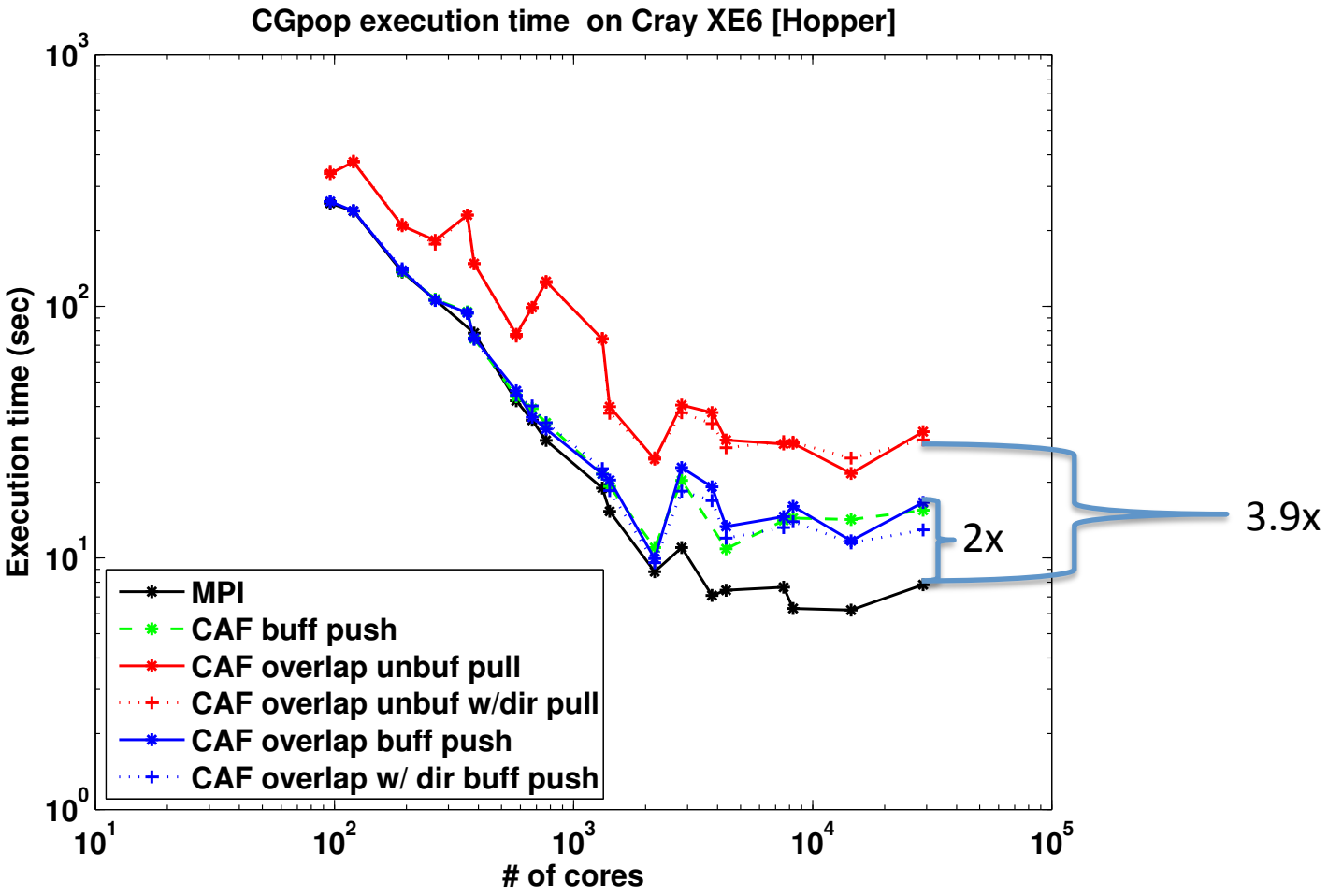
```
subroutine UpdateHalo(array)
  ! ** Input parameters and local variables: **
real(r8), intent(inout) :: array(:)
integer(i4) :: src,dest,len,iptr,tag
integer(i4) :: ierr,i, placeval,j

  ! ** Gather data to be sent into a buffer **
do i=1,lenSendBuffer
    sendBuffer(i) = array(halo2send(i))
enddo

  sync all

  ! ** Pull data from buffer to neighbors **
do i=1,nRecv
               iptr  = ptrRecv(i)
         len   = RecvCnt(i)
         dest  = rNeigh(i) + 1
         pullval = pull(i)

        BufferRecvDbl(iptr:iptr+len-1) = &
              BufferSendDbl(pullval:pullval+len-1)[dest]
    enddo


       sync all

  ! ***
  !  Indirect address from the recvBuffer to the
  !  receiving side's array
  ! ***
do i=1,lenRecvBuffer
    array(recv2halo(i)) = recvBuffer(i)
enddo
end subroutine UpdateHalo
```

```
subroutine UpdateHalo(array)
    ! ** Input parameters and local variables: **
    real(r8), intent(inout) :: array(:)
    integer(i4) :: src,dest,len,iptr,tag
    integer(i4) :: ierr,i, placeval,j

    ! ** Gather data to be sent into a buffer **
    do i=1,lenSendBuffer
        sendBuffer(i) = array(halo2send(i))
    enddo

    sync all

    ! ** Pull data from buffer to neighbors **
    do i=1,nRecv
                iptr  = ptrRecv(i)
            len   = RecvCnt(i)
            dest  = rNeigh(i) + 1
            pullval = pull(i)

           BufferRecvDbl(iptr:iptr+len-1) = &
                BufferSendDbl(pullval:pullval+len-1)[dest]
        enddo

    sync all

    ! ***
    !  Indirect address from the recvBuffer to the
    !  receiving side's array
    ! ***
    do i=1,lenRecvBuffer
        array(recv2halo(i)) = recvBuffer(i)
    enddo

end subroutine UpdateHalo
```

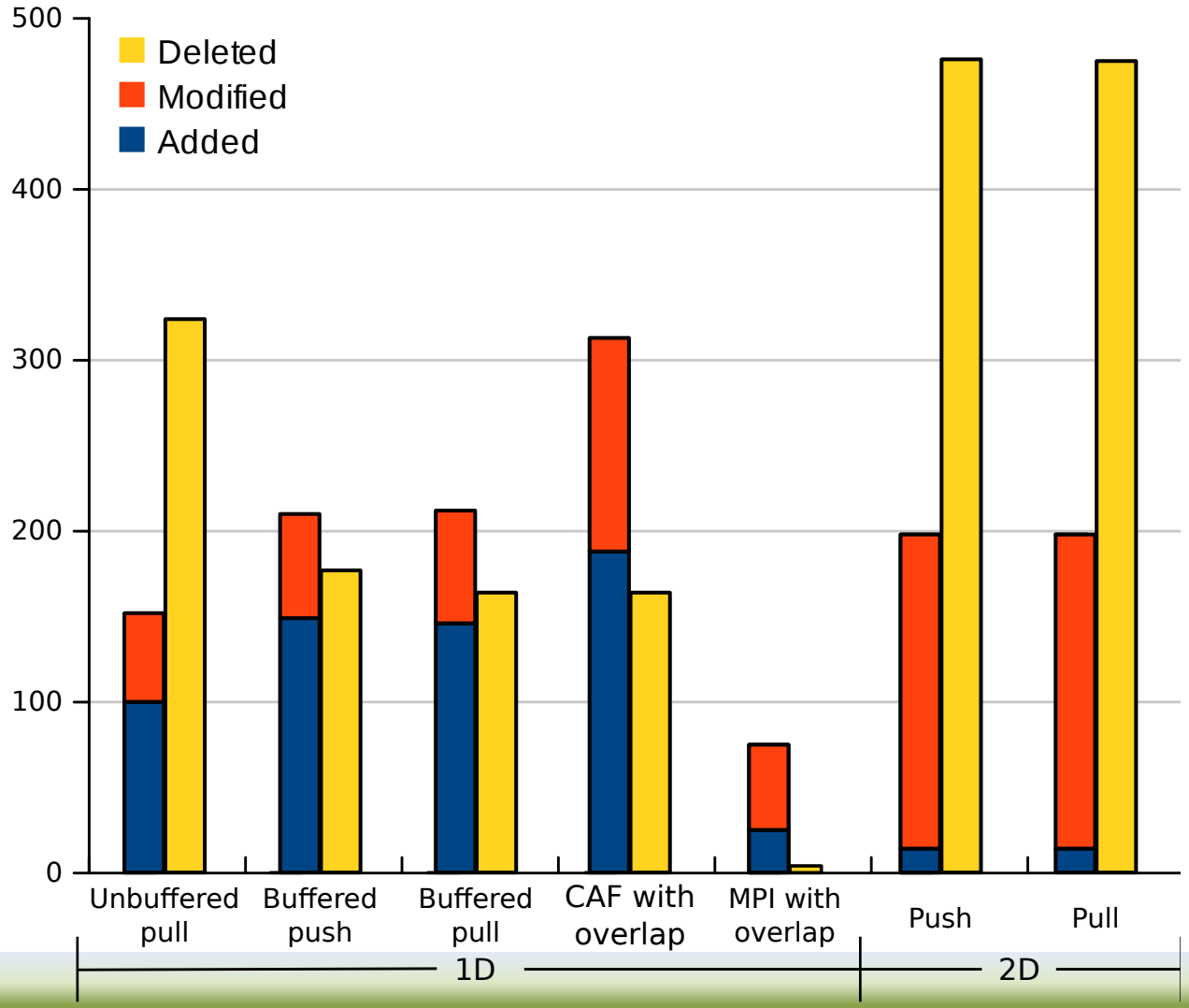
```
subroutine UpdateHalo(array)
      ! ** Input parameters and local variables: **
real(r8), intent(inout) :: array(:)
integer(i4) :: src,dest,len,iptr,tag
integer(i4) :: ierr,i, placeval,j

! ** Gather data to be sent into a buffer **
do i=1,lenSendBuffer
    sendBuffer(i) = array(halo2send(i))
enddo

sync all

    ! ** Pull data from buffer to neighbors **
        do i=1,nRecv
              iptr  = ptrRecv(i)
           len   = RecvCnt(i)
           dest  = rNeigh(i) + 1
           pullval = pull(i)

          BufferRecvDbl(iptr:iptr+len-1) = &
                BufferSendDbl(pullval:pullval+len-1)[dest]
      enddo

sync all

! ***
!  Indirect address from the recvBuffer to the
!  receiving side's array
! ***
do i=1,lenRecvBuffer
    array(recv2halo(i)) = recvBuffer(i)
enddo

end subroutine UpdateHalo
```

```
subroutine UpdateHalo(array)
      ! ** Input parameters and local variables: **
real(r8), intent(inout) :: array(:)
integer(i4) :: src,dest,len,iptr,tag
integer(i4) :: ierr,i, placeval,j

! ** Gather data to be sent into a buffer **
do i=1,lenSendBuffer
    sendBuffer(i) = array(halo2send(i))
enddo

sync all

! ** Pull data from buffer to neighbors **
do i=1,nRecv
              iptr  = ptrRecv(i)
         len   = RecvCnt(i)
         dest  = rNeigh(i) + 1
         pullval = pull(i)

        BufferRecvDbl(iptr:iptr+len-1) = &
             BufferSendDbl(pullval:pullval+len-1)[dest]
    enddo
      sync all


       ! ***
       !  Indirect address from the recvBuffer to the
       !  receiving side's array
       ! ***
      do i=1,lenRecvBuffer
            array(recv2halo(i)) = recvBuffer(i)
      enddo
end subroutine UpdateHalo
```

Resulting Performance



Code Volume Comparison



Suggestions for CAF

- Analyze Code and Automatically Aggregate multiple sends conducted in a loop
- Add gather/scatter operation to language:

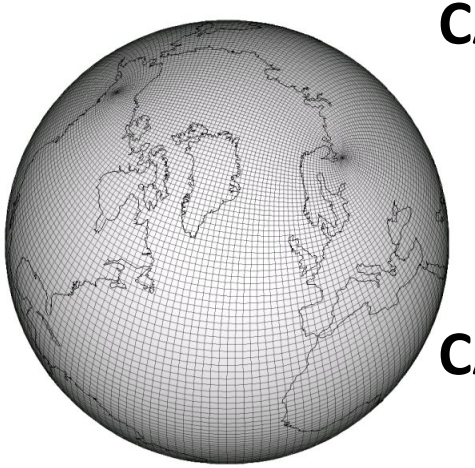
```
array(recv2halo(:)) =
   array(halo2send(:))[source(i)]

   recv2halo – Lists elements to receive
   halo2send – Lists elements to send
   [source(i)] – Indicates to access an
array on the sending proc
```

Suggestions for CAF

- Missing Reductions in standard
  - Cray has reductions that work on certain machines
- Explicit Synchronization Management

  ! dir$ pgas defer_sync

Conclusions



CAF's strong points:

- Decreased line count with non-buffered version
- Can compile with existing Fortran compiler
- Easy learning curve

CAF's weak points:

- Had to explicitly buffer data

Adoption: a chicken and egg scenario

- Developers want to see studies examining applications similar to theirs.
- Compiler researchers need developers to help port applications (due size and domain specific nature).
- Miniapps can help break this cycle

Download Source code and Tech Report at:

http://www.cs.colostate.edu/hpc/cgpop

Importance of Interconnect