2011 HPC Challenge Class II Submission: Coarray Fortran 2.0

John Mellor-Crummey, Laksono Adhianto
Mark Krentel, Guohua Jin, Karthik Murthy,
William Scherer III, Chaoran Yang
Department of Computer Science
Rice University
Coarray Fortran (CAF)

- Global address space SPMD parallel programming model
  — one-sided communication
- Simple, two-level memory model for locality management
  — local vs. remote memory
- Programmer has control over performance critical decisions
  — data partitioning
  — data movement
  — synchronization
- Adopted in Fortran 2008 standard
Coarray Fortran 2.0 (CAF 2.0)

• Teams: process subsets, like MPI communicators
  —formation using team_split (like MPI_Comm_split)
  —collective communication (two-sided)
  —barrier synchronization

• Coarrays: shared data allocated across processor subsets
  —declaration: double precision :: a(:,:)[*]
  —dynamic allocation: allocate(a(n,m)[@row_team])
  —access: x(:,n+1) = x(:,0)[mod(team_rank()+1, team_size())]

• Latency tolerance
  —hide: asynchronous copy, asynchronous collectives
  —avoid: function shipping

• Synchronization
  —event variables: point-to-point sync; async completion
  —finish: SPMD construct inspired by X10

• Copointers: pointers to remote data
Our HPC Challenge Goal: Productivity

- Priorities, in order
  - performance, performance, performance
  - source code volume

- Productivity = performance / (lines of code)

- Implications for our implementations
  - FFT (revised implementation for this year)
    - use global transposes to keep computation local
  - EP STREAM Triad
    - outline a loop for best compiler optimization
  - Randomaccess
    - batch updates and use software routing for higher performance
  - HPL
    - operate on blocks to leverage a high performance DGEMM
  - Unbalanced Tree Search (UTS)
    - evaluate how CAF 2.0 supports dynamic load balancing
    - use function shipping to implement work stealing and work sharing
Productivity = Performance / SLOC

Performance (on Cray XT4 and XT5)

<table>
<thead>
<tr>
<th># of cores</th>
<th>STREAM Triad* (TByte/s)</th>
<th>RandomAccess‡ (GUP/s)</th>
<th>Global HPL † (TFlop/s)</th>
<th>Global FFT † (GFlop/s)</th>
<th>UTS* (MNode/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>0.17</td>
<td>0.08</td>
<td>0.36</td>
<td>6.69</td>
<td>163.1</td>
</tr>
<tr>
<td>256</td>
<td>0.67</td>
<td>0.24</td>
<td>1.36</td>
<td>22.82</td>
<td>645.0</td>
</tr>
<tr>
<td>1024</td>
<td>2.66</td>
<td>0.69</td>
<td>4.99</td>
<td>67.80</td>
<td>2371</td>
</tr>
<tr>
<td>4096</td>
<td>10.70</td>
<td>2.01</td>
<td>18.3</td>
<td>187.04</td>
<td>7818</td>
</tr>
<tr>
<td>8192</td>
<td>21.69</td>
<td></td>
<td></td>
<td>357.80</td>
<td>12286</td>
</tr>
</tbody>
</table>

*Jaguar - XT5 ‡Jaguar - XT4 †Franklin - XT4

Source lines of code

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Source Lines</th>
<th>Reference SLOC</th>
<th>Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Randomaccess</td>
<td>409</td>
<td>787</td>
<td>48%</td>
</tr>
<tr>
<td>EP STREAM</td>
<td>63</td>
<td>329</td>
<td>81%</td>
</tr>
<tr>
<td>Global HPL</td>
<td>786</td>
<td>8800</td>
<td>91%</td>
</tr>
<tr>
<td>Global FFT</td>
<td>450</td>
<td>1130</td>
<td>60%</td>
</tr>
<tr>
<td>UTS</td>
<td>544</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Notes
- STREAM: 82% of peak memory bandwidth
- HPL: 49% of FP peak at 4096 cores (uses dgemm)
- Issues with GASNet 1.17 runtime for 4K and more processors on XE6
Unbalanced Tree Search (UTS)

- Exploration of an unbalanced implicit tree
- Fixed geometric distribution, depth 18, 270 billion nodes

```fortran
! while there is work to do
do while(queue_count .gt. 0)
   call dequeue_back(descriptor)
   call process_work(descriptor)
   ! check if someone needs work
   if ((incoming_lifelines .ne. 0) .and. &
       (queue_count .ge. lifeline_threshold)) then
      call push_work()
   endif
endo

! attempt to steal work from another image
victim = get_random_image()

spawn steal_work()[victim]

! set up lifelines on hypercube neighbors
do index = 0, max_neighbor_index-1
   neighbor = xor(my_rank, 2**index)
   spawn set_lifelines(my_rank, index)[neighbor]
enddo
```

- Slope shows all PE working
- Tight grouping of lines shows good load balance

Cray XT5, 12 cores/node

Total nodes processed per PE

Separate line for each of 128 PEs
FFT

- Radix 2 FFT implementation
- Block distribution of coarray “c” across all processors
- Sketch in CAF 2.0:

```fortran
complex, allocatable :: c(:,2)[*], spare(:)[*]
...
!
permute data to bit-reversed indices (uses team_alltoall)
call bitreverse(c, n_world_size, world_size, spare)
!
local FFT computation for levels that fit in the memory of an image
do l = 1, loc_comm-1 ...
!
transpose from block to cyclic data distribution (uses team_alltoall)
call transpose(c, n_world_size, world_size, spare)
!
local FFT computation for remaining levels
do l = loc_comm, levels ...
!
transpose back from cyclic to block data distribution (uses team_alltoall)
call transpose(c, n_world_size, n_local_size/world_size, spare)
```
double precision, allocatable :: a(:)[*], b(:)[*], c(:)[*]

... 

! each processor in the default team allocates their own array parts
allocate(a(local_n)[], b(local_n)[], c(local_n)[])

...

! perform the calculation repeatedly to get reliable timings
do round = 1, rounds
  do j = 1, rep
    call triad(a,b,c,local_n,scalar)
  end do
  call team_barrier() ! synchronous barrier across images in the default team
end do
...

! perform the calculation with top performance
! assembly code is identical to that for sequential Fortran

subroutine triad(a, b, c, n ,scalar)
  double precision :: a(n), b(n), c(n), scalar
  a = b + scalar * c  ! EP triad as a Fortran 90 vector operation
end subroutine triad
event, allocatable :: delivered(:)[*], received(:)[*] ! (stage)
integer(i8), allocatable :: fwd(:, :, :, :)[*] ! (#, in/out, stage)

! hypercube-based routing: each processor has 1024 updates
do i = world_logsize-1, 0, -1 ! log P stages in a route
  call split(retain(:, last), ret_sizes(last), &
             retain(:, current), ret_sizes(current), &
             fwd(1:, in, i), fwd(0, out, i), bufsize, dist)
  if (i < world_logsize-1) then
    event_wait(delivered(i+1))
    call split(fwd(1:, in, i+1), fwd(0, in, i+1), &
               retain(:, current), ret_sizes(current), &
               fwd(1:, out, i), fwd(0, out, i), bufsize, dist)
    event_notify(received(i+1)[from]) ! signal buffer is empty
  endif
  count = fwd(0, out, i)
  event_wait(received(i)) ! ensure buffer is empty from last route
  fwd(0:count, in, i)[partner] = fwd(0:count, out, i) ! send to partner
  event_notify(delivered(i)[partner]) ! notify partner data is there
end do
HPL

• Block-cyclic data distribution
• Team based collective operations along rows and columns
  — synchronous max reduction down columns of processors
  — asynchronous broadcast of panels to all processors

```fortran
! type declaration
type(paneltype) :: panels(1:NUMPANELS)
event, allocatable :: delivered(:)[*]
...

! loop over columns
do j = pp, PROBLEMSIZE - 1, BLKSIZE
  cp = mod(j / BLKSIZE, 2) + 1
  ...
  event_wait(delivered(3-cp))
  ...
  if (mycol == cproc) then
    ...
    if (ncol > 0) ! update part of the trailing matrix
      call fact(m, n, cp) ! factor the next panel
    ...
  endif
  call team_broadcast_async(panels(cp)%buff(1:ub), panels(cp)%info(8), &
    delivered(cp))
  ! update rest of the trailing matrix
  if (nn-ncol>0) call update(m, n, col, nn-ncol, 3 - cp)
  ...
end do
```
Experimental Setup

• Rice Coarray Fortran 2.0
  —source to source translation from CAF 2.0 to Fortran 90
    – generated code compiled with Portland Group’s pgf90
  —CAF 2.0 runtime system built upon GASNet (versions 1.14 .. 1.17)
    —scalable implementation of teams, using $O(\log P)$ storage

• Experimental platforms: Cray XT4, XT5, and XE6
  —systems
    – Franklin - XT4 at NERSC
      2.3 GHz AMD “Budapest” quad-core Opteron, 2GB DDR2-800/core
    – Jaguar - XT4 at ORNL
      2.1 GHz AMD quad-core Opteron, 2GB DDR2-800/core
    – Jaguar - XT5 at ORNL
      2.6 GHz AMD “Istanbul” hex-core Opteron, 1.3GB DDR2-800/core
    – Hopper - XE6 at NERSC
      2.1 GHz AMD dual-twelve cores Magnycours, 1.3GB DDR3-1333/core

  —network topologies
    – XT4, XT5: 3D Torus based on Seastar2 routers; XE6: Gemini
Productivity = Performance / SLOC

Performance (on Cray XT4 and XT5)

<table>
<thead>
<tr>
<th># of cores</th>
<th>STREAM Triad* (TByte/s)</th>
<th>RandomAccess‡ (GUP/s)</th>
<th>Global HPL † (TFlop/s)</th>
<th>Global FFT † (GFlop/s)</th>
<th>UTS* (MNode/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>0.17</td>
<td>0.08</td>
<td>0.36</td>
<td>6.69</td>
<td>163.1</td>
</tr>
<tr>
<td>256</td>
<td>0.67</td>
<td>0.24</td>
<td>1.36</td>
<td>22.82</td>
<td>645.0</td>
</tr>
<tr>
<td>1024</td>
<td>2.66</td>
<td>0.69</td>
<td>4.99</td>
<td>67.80</td>
<td>2371</td>
</tr>
<tr>
<td>4096</td>
<td>10.70</td>
<td>2.01</td>
<td>18.3</td>
<td>187.04</td>
<td>7818</td>
</tr>
<tr>
<td>8192</td>
<td>21.69</td>
<td></td>
<td></td>
<td>357.80</td>
<td>12286</td>
</tr>
</tbody>
</table>

*Jaguar - XT5 ‡Jaguar - XT4 † Franklin - XT4

Source lines of code

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Source Lines</th>
<th>Reference SLOC</th>
<th>Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Randomaccess</td>
<td>409</td>
<td>787</td>
<td>48%</td>
</tr>
<tr>
<td>EP STREAM</td>
<td>63</td>
<td>329</td>
<td>81%</td>
</tr>
<tr>
<td>Global HPL</td>
<td>786</td>
<td>8800</td>
<td>91%</td>
</tr>
<tr>
<td>Global FFT</td>
<td>450</td>
<td>1130</td>
<td>60%</td>
</tr>
<tr>
<td>UTS</td>
<td>544</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Notes
- STREAM: 82% of peak memory bandwidth
- HPL: 49% of FP peak at @ 4096 cores (uses dgemm)
- Issues with GASNet 1.17 runtime for 4K and more processors on XE6
Source Code Breakdown

<table>
<thead>
<tr>
<th></th>
<th>STREAM</th>
<th>RandomAccess</th>
<th>FFT</th>
<th>HPL</th>
<th>UTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation</td>
<td>32</td>
<td>188</td>
<td>180</td>
<td>536</td>
<td>267</td>
</tr>
<tr>
<td>Communication &amp; synchronization</td>
<td>1</td>
<td>12</td>
<td>4</td>
<td>46</td>
<td>17</td>
</tr>
<tr>
<td>Declaration</td>
<td>17</td>
<td>118</td>
<td>103</td>
<td>109</td>
<td>151</td>
</tr>
<tr>
<td>Comments &amp; spaces</td>
<td>13</td>
<td>91</td>
<td>163</td>
<td>95</td>
<td>109</td>
</tr>
<tr>
<td>Total</td>
<td>63</td>
<td>409</td>
<td>450</td>
<td>786</td>
<td>544</td>
</tr>
</tbody>
</table>