HPC Challenge Benchmarks in CAF2.0

Department of Computer Science, Rice University
Project URL: http://caf.rice.edu

HPC Challenge Benchmarks

- Measure performance of HPC systems
  - ❖ processor performance
  - ❖ memory subsystem performance
  - ❖ system interconnect performance
- Benchmarks
  - ❖ RandomAccess
  - ❖ EP STREAM Triad
  - ❖ Global FFT
  - ❖ Global HPL
  - ❖ PTRANS
  - ❖ DGEMM
  - ❖ b_eff Latency/BW

EP STREAM Triad

- outlined STREAM triad with explicit shape array declarations for performance

```
double precision, allocatable :: a(:)[*]
double precision, allocatable :: b(:)[*], c(:)[*]
! allocate with the default team
allocate(a(ndim)[], b(ndim)[], c(ndim)[])
...
do round = 1, rounds
  do j = 1, rep
    call triad(a,b,c,n,scalar)
  end do
  call team_barrier()
end do
...
subroutine triad(a, b, c, n ,scalar)
  double precision a(n), b(n), c(n), scalar
  a = b + scalar * c
end subroutine triad
```

Global FFT

- all-to-all collective in transpose

```
complex, allocatable :: c(:,2)[*], spare(:)[*]
...
! permute data to bit-reversed indices using team_alltoall
call bitreverse(c, n_world_size, world_size, spare)
! local FFT for levels that fit in the memory of an image
...
! block to cyclic transpose (implemented with team_alltoall)
call transpose(c, n_world_size, world_size, spare)
! local FFT for remaining levels
...
! cyclic to block transpose (implemented with team_alltoall)
call transpose(c, n_world_size, n_local_size/world_size,spare)
```

Unbalanced Tree Search (UTS)

- load balance via work stealing and sharing

```
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()
  end if
end do
! attempt to steal work from another image
spawn steal_work()[get_random_image()]
! step up lifelines for work sharing
do index = 0, max_neighbor_index - 1
  neighbor = xor(my_rank, 2**index)
  spawn set_lifelines(my_rank, index)[neighbor]
end do
```

RandomAccess

- hypercube-based routing of updates
- overlap computation with synchronization for performance

```
event,allocatable :: delivered(:)[*],received(:)[*]
integer(8),allocatable :: fwd(:,:,:)[*]
do i = world_logsize-1, 0, -1
  ...
  call split(ret(:,last), retsizes(last), &
             ret(:,current), retsizes(current), &
             fwd(1:,out,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), &
               ret(:,current),retsizes(current),&
               fwd(1:,out,i),fwd(0,out,i),bufsize,dist)
    event_notify(received(i+1)[from])
  endif
  count = fwd(0,out,i)
  copy_async(fwd(0:count,in,i)[partner],fwd(0:count,out,i),&
             cr = received(i), dr = delivered(i)[partner])
  ...
end do
```

Global HPL

- block-cyclic data distribution
- row/column team-based communication
- asynchronous broadcast of panels

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

allocate(delivered(1:NUMPANELS)[])
event_init(delivered, NUMPANELS)
...
do j = pp, PROBLEMSIZE - 1, BLKSIZE
  cp = mod(j / (BLKSIZE + 1) - 1, 2) + 1
  ...
  event_wait(delivered(3-cp))
  ...
  if (mycol == cproc) then
    ...
    if (ncol > 0) then
      ! update part of the trailing matrix
      if (NPCOL == 1) call update(m,n,BLKSIZE,ncol,3-cp)
      if (NPCOL /= 1) call update(m,n,0,ncol,3-cp)
    end if
    call fact(m, n, cp)  ! factorize the next panel
    ...
  end if
  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
```

Benchmarks	SLOC
STREAM Triad	63
RandomAccess	409
Global FFT	450
Global HPL	786
UTS	544

# of cores	STREAM‡ (TByte/s)	RA† (GUP/s)	Global FFT† (GFlop/s)	Global HPL† (TFlop/s)	UTS‡ (MNode/s)
64	0.17	0.08	6.69	0.36	163
256	0.67	0.24	22.8	1.36	645
1024	2.66	0.69	67.8	4.99	2371
4096	10.70	2.01	187	18.3	7818

Performance results were collected on: Cray XT4 (†), XT5 (‡)

Development of Coarray Fortran 2.0 is supported by the Department of Energy's Office of Science under cooperative agreements DE-FC02-07ER25800 and DE-FC02- 06ER25754.

Contributors: John Mellor-Crummey (PI), Laksono Adhianto, Guohua Jin, Mark Krentel, Karthik Murthy, Dung Nguyen, William N. Scherer III, Scott K. Warren, Chaoran Yang

SciDAC
Scientific Discovery through Advanced Computing