CAFe: A Unified PGAS Programming Model for Heterogeneous Computing

Craig Rasmussen, Soren Rasmussen, William Dumas
University of Oregon
Matt Sottile, Galois Inc
Dan Nagle, NCAR

The Los Alamos Roadrunner Challenge — a forerunner to tomorrow's architectures?

- Roadrunner 2008
  - hybrid design
  - 6480 AMD Opteron dual-core processors
  - 12,960 IBM PowerXCell accelerators
- Roadrunner presented a programming challenge
- Several teams were started to port important LANL apps to the IBM Cell
  - essentially wrote applications entirely from scratch
- We wrote a Fortran source-to-source translator for algorithms using dense arrays
  - 29 X speedup

CAFe: Coarray Fortran Extensions for Heterogeneous Computing

- Fortran is a parallel language. Fortran added coarrays for parallel computing in 2008 (with additional features added in the 2015 standard).
  - However, the coarray programming model *does not* support
    - attached accelerator devices
    - communication between distributed memory hierarchies
    - remote execution of tasks

CAFe provides a unified parallel model — not so when adding OpenMP/OpenACC directives

- Coarray Fortran has several parallel constructs
  - process teams, synchronization, collectives, critical regions
  - parallel loops (DO CONCURRENT)
  - put and get of memory regions to/from remote processes, [ ] syntax
- Coarrays (or MPI) plus OpenMP/OpenACC have similar constructs
  - However!
    - a programmer must learn and use two separate parallel languages
    - the two languages have different constructs to do the same thing
    - the competing constructs are not compatible with each other
    - num_gangs( ), acc_malloc( ), acc_memcpy_from_device_async( )
    - wait, reduction

CAFe adds three important concepts to parallel Fortran

- Subimages
  - A Fortran image (similar to an MPI process) may create one or more subimages. A subimage could represent an attached accelerator device or a cohort of threads running on a set of homogeneous cores.
- Explicit memory placement
  - Coarray memory may be explicitly allocated and deallocated on a subimage by its parent image.
  - Provides memory affinity for NUMA shared memory multi-cores
- Remote execution and synchronization of tasks
  - Tasks (functions or code blocks) may be executed on a subimage by its parent image. Execution of these tasks may be synchronized with Fortran 2015 events.

CAFe syntax editions (shown in light blue)

- Obtain access to an accelerator device

```
dev1 = get_subimage(dev_id, device_type=CUDA, err=)
```

- Memory allocation (also affinity) and deallocation on a device

```
allocate(A(N)[*], subimage=dev1)
deallocate(A)
```

- Transfer memory (after initialization)

```
A(:)[dev1] = A(:);    B(:)[dev2] = B(:)
```

- Remote execution and synchronization of tasks on two subimages using memory previously allocated on the subimages

```
call task1(A[dev1])  [[dev1, WITH_EVENT=evt]]
call task2(B[dev2])  [[dev2, WITH_EVENT=evt]]
event wait (evt, until_count=2)
```

Single-Source Shortest Path Algorithm: Coding example

```
!! get the Fortran image selector(s) for the accelerator device
!
  dev = get_subimage(acc_id)

!! allocate space on the accelerator
!
  if (dev /= THIS_IMAGE()) then
     allocate(      TT(NX,NY,NZ)[*]) [[dev]]
     allocate(Changed(NX,NY,NZ)[*]) [[dev]]
  end if

!! initialize and copy values to the device
!
  TT      = INFINITY
  TT[dev] = TT

!! loop until converged
!
  do while (.NOT. done)
     call sweep(NX,NY,NZ, NFS, U[dev], TT[dev], Offset[dev], Changed[dev]) [[dev]]

     !! see if any travel times have changed
     !
     Changed = Changed[dev]
     if (sum(Changed) == 0) done = .TRUE.

  end do
```

OpenCL code automatically created by OFP compiler from original *CAFe* source

```
!! WARNING - this code is not readable, portable nor maintainable

TYPE(CLBuffer) :: cl_TTBuf_
TYPE(CLBuffer) :: cl_Changed_
TYPE(CLKernel) :: cl_sweep_

cl_sweep_ = createKernel(cl_dev_,"sweep")

cl_size__ = 4*newNX*newNY*newNZ
cl_TT_ = createBuffer(cl_dev_,CL_MEM_READ_WRITE,cl_size__,C_NULL_PTR)
cl_Changed_ = createBuffer(cl_dev_,CL_MEM_READ_WRITE,cl_size__,C_NULL_PTR)

cl_status__ = writeBuffer(cl_TT_,C_LOC(TT),cl_size__)
cl_status__ = writeBuffer(cl_Changed_,C_LOC(Changed),cl_size__)

cl_status__ = setKernelArg(cl_sweep_,0,NX)
cl_status__ = setKernelArg(cl_sweep_,5,clMemObject(cl_TT_))
cl_status__ = setKernelArg(cl_sweep_,7,clMemObject(cl_Changed_))

DO WHILE(.not. done)
   cl_status__ = run(cl_sweep_,3,cl_gwo__,cl_gws__,cl_lws__)
   cl_status__ = clFinish(cl_sweep_%commands)

   cl_status__ = readBuffer(cl_Changed_,C_LOC(Changed),cl_size__)
   IF (sum(Changed) .le. 0) done = .TRUE.
   cl_status__ = setKernelArg(cl_sweep_, 9,stepsTaken)
END DO
```

CAFe publications

- C. Rasmussen, M. Sottile, S. Rasmussen, D. Nagle, and W. Dumars. CAFe: Coarray Fortran extensions for heterogeneous computing. Paper to be presented at *High-Level Parallel Programming Models and Supportive Environments, 21st International Workshop, IPDPS 2016*, Chicago, IL, USA, May 23, 2016.

- A. D. Malony, S. McCumsey, J. Byrnes, C. Rasmussen, S. Rasmussen, E. Keever, and D. Toomey. A Data Parallel Algorithm for Seismic Raytracing. Paper to be presented at *The International Meeting on High-Performance Computing for Computational Science, VECPAR 2016*, Porto, Portugal, June 28th-30th, 2016.