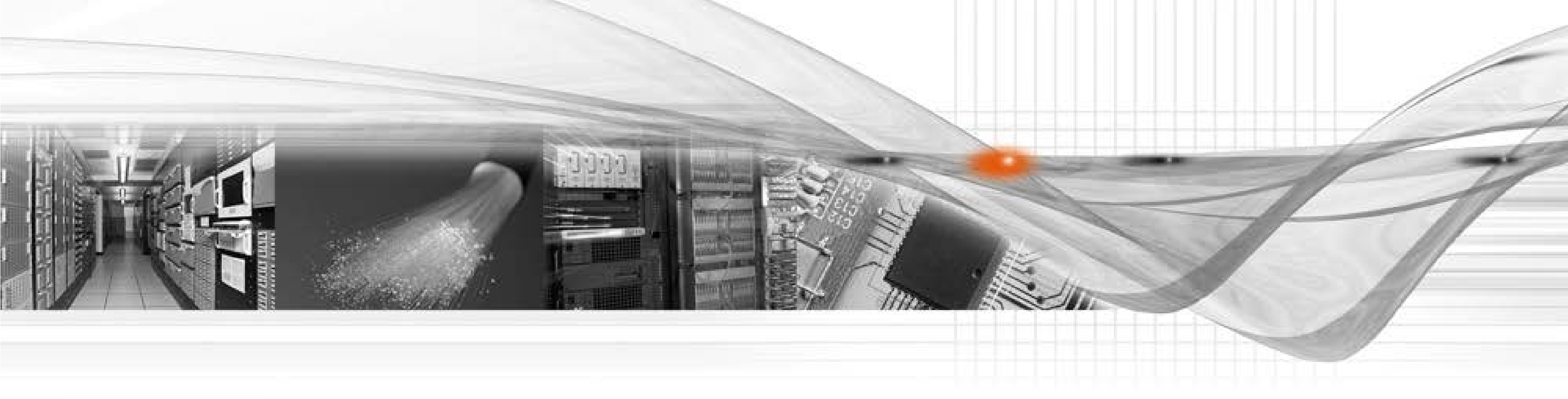
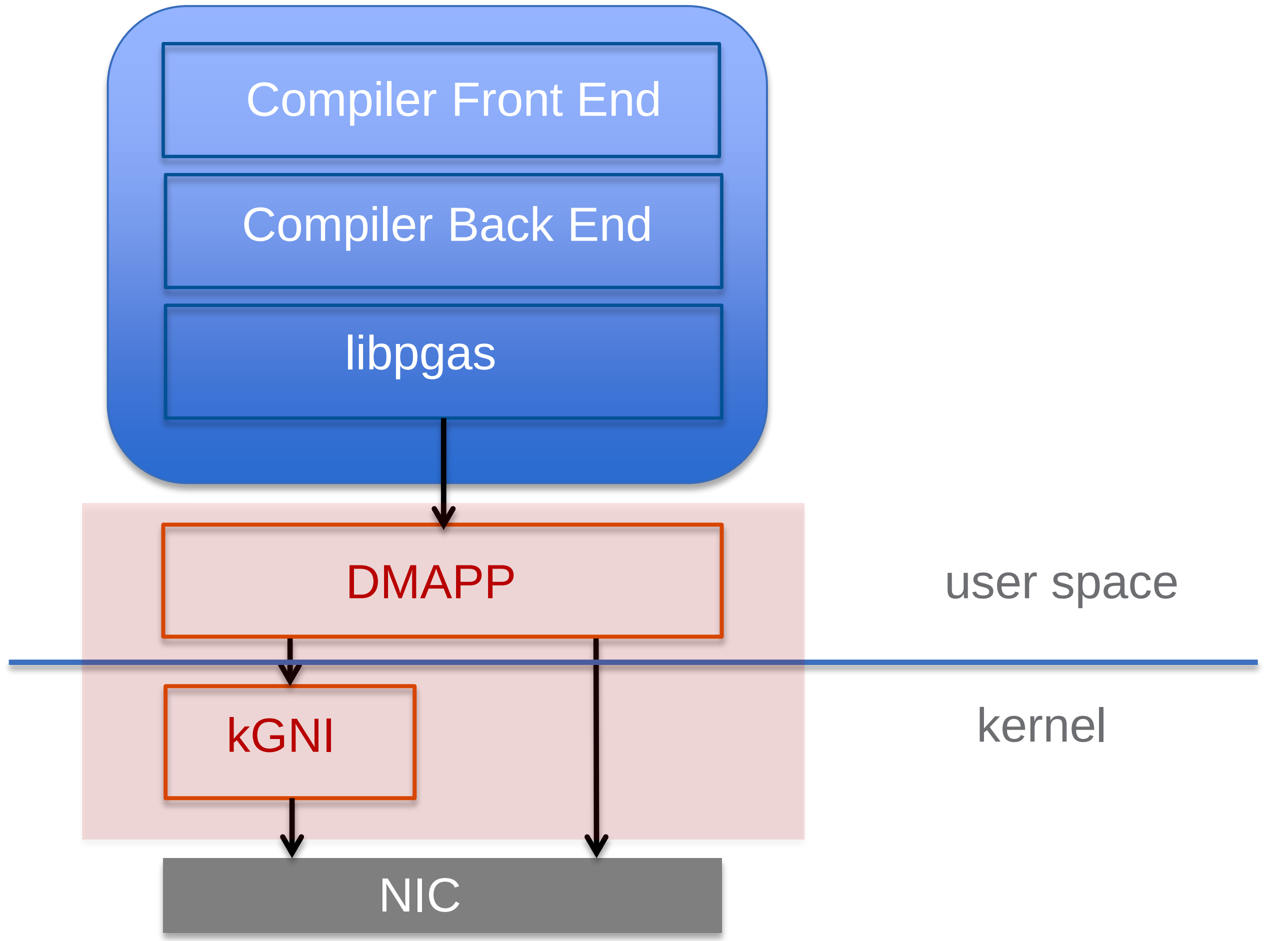


# 2013 OFA Developer Workshop

Scaling with PGAS Languages Panel

Howard Pritchard
CRAY, Inc.

# Cray UPC/CAF Compiler and Network Stack



## DMAPP Summary

- High level API for one-sided program models
  - Optimized for fine grain RMA
  - Hide changes in underlying HW from upper levels as much as possible while not sacrificing performance
- Based loosely on concepts from Cray and Quadrics SHMEM
  - Blocking, nonblocking, nonblocking implicit RMA
- Uses explicit memory "descriptor" arguments
- Supports RMA read/write/atomic memory ops

# DMAPP – Memory Registration (Dealing with the I/O MMU)

- Simple memory registration
- Symmetric memory registration (don’t need to do explicit exchange of memory registration keys)
- Dynamic memory reservation
  - Application “registers” a VM region, but its not really registered at that point
  - Use an “update” registration to fault in pages in region when needed and register with NIC
  - Option to use symmetrically
- DMAPP handles registration of “get” buffers internally

## DMAPP – Other functionality

- Scalable message queue
  - Intended for helping to support PGAS functionality where “active message” support is useful
  - Blocking/polling flavors
- Collectives
  - Uses a “pset” construct similar to MPI groups
  - Uses hw offload if available and operation can be offloaded
- Thread hot (fine grain locking, multiple threads can be using network concurrently)
- Hook functions for library interop, i.e. MPI makes progress even in a DMAPP blocking call

## Thank You

OPENFABRICS ALLIANCE

## Backup Slides

## DMAPP – API example

```
extern dmapp_return_t
dmapp_put_nb(IN  void                  *target_addr,
             IN  dmapp_seg_desc_t      *target_seg,
             IN  dmapp_pe_t            target_pe,
             IN  void                  *source_addr,
             IN  uint64_t              nelems,
             IN  dmapp_type_t          type,
             OUT dmapp_syncid_handle_t *syncid);
```

## DMAPP – RMA functionality

- READ, WRITE
  - Contiguous
  - Strided, gather, scatter, rank/thread strided
- AMOS
  - 32/64 bit integer ops (fadd, cswap, bitwise ops, etc.)
  - 32/64 bit fp ops (add)
- Three synchronization types
  - Blocking (doesn’t return till HW ack back from target)
  - Non blocking explicit (handle returned which is used in subsequent completion call)
  - Non blocking implicit (a series of arbitrary rma ops can be done followed by a dmapp_gsync_wait/test)

# kGNI odds and ends

- Uses pnotify fork functionality to avoid COW problems with registered memory regions
- Provides ummunotify like functionality to keep DMAPP internal memory registration cache from getting in to trouble