2021 OFA Virtual Workshop

PROGRESS OF UPSTREAM GPU RDMA SUPPORT

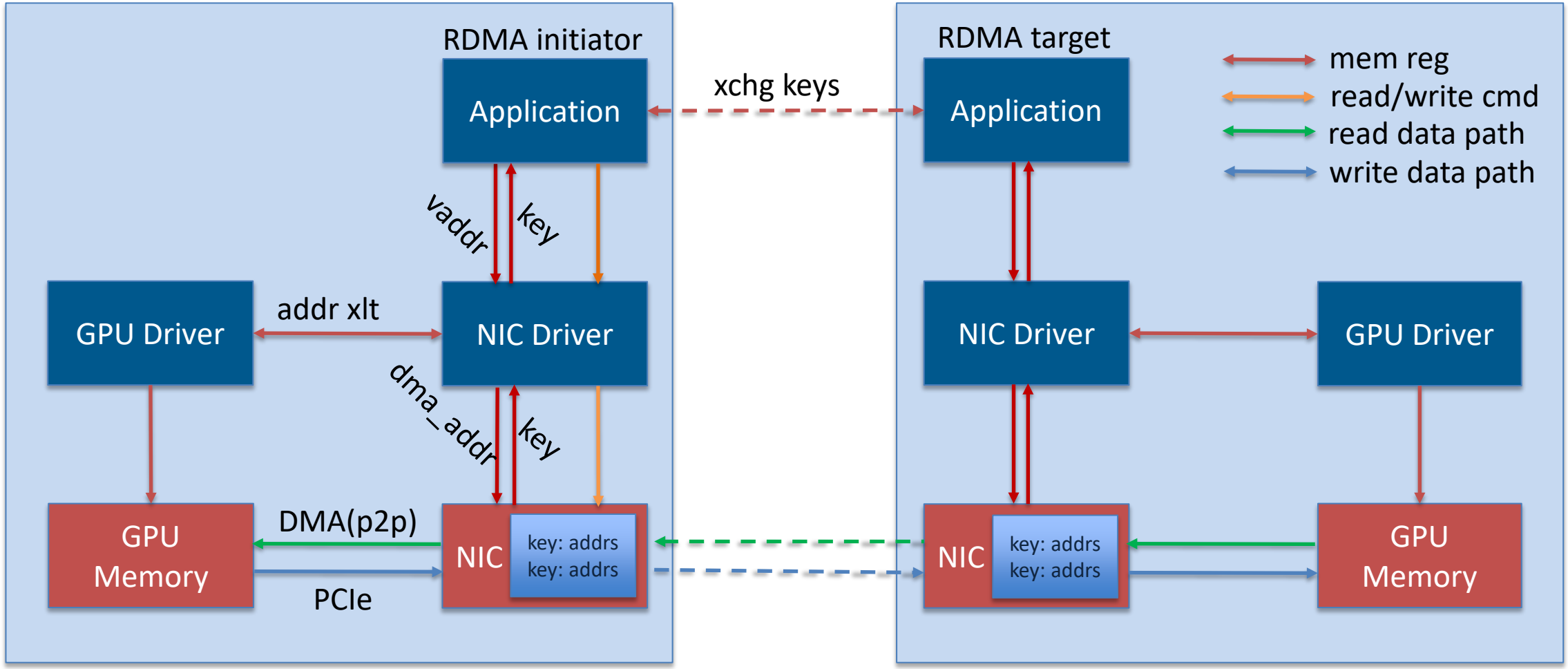
Jianxin Xiong

Intel Corporation

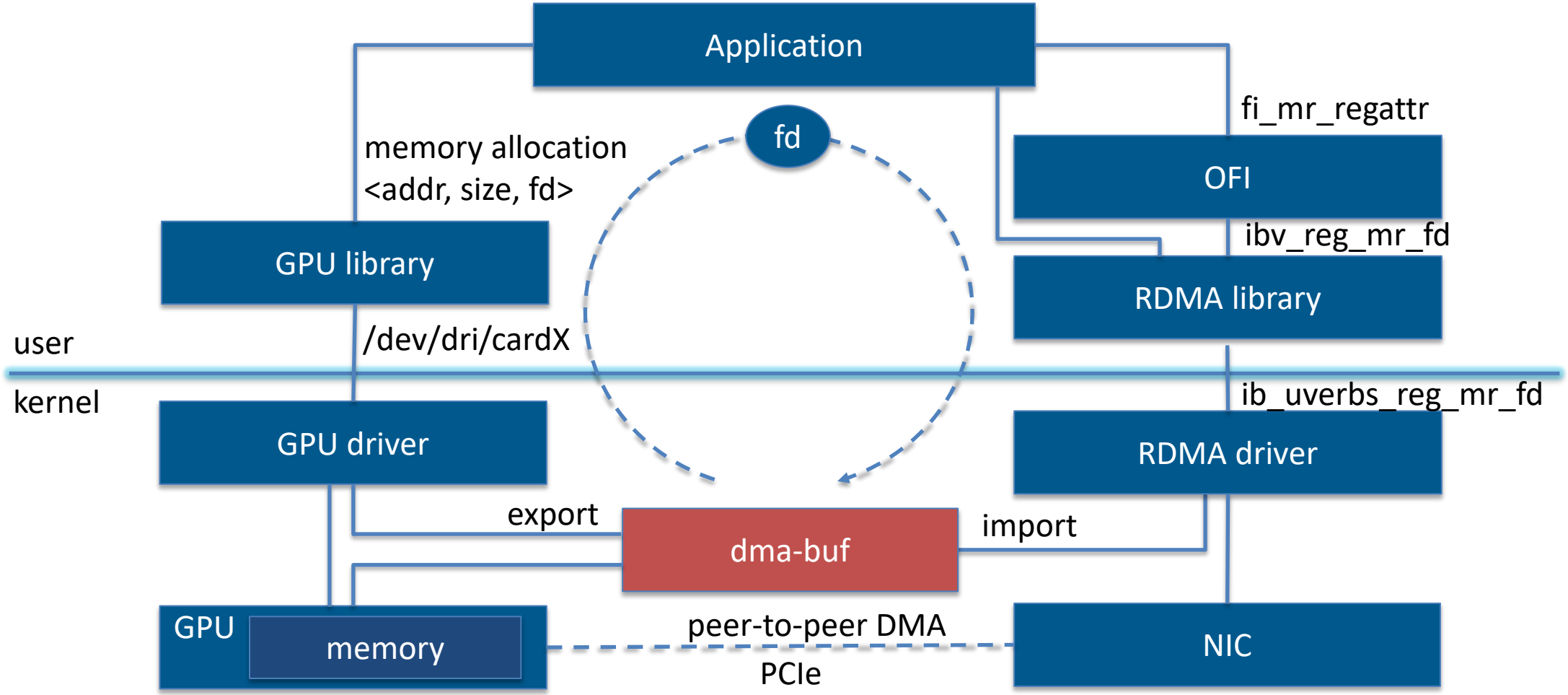
OUTLINE

- **GPU RDMA Overview**
- **Prototype presented on 2020 OFA Workshop**
  - Use dma-buf as the buffer sharing mechanism
  - Requires the buffer to be pinned
  - The goal is to have a solution at upstream
- **Good progress has been made since then**
  - Upstream acceptance achieved
  - Some major design changes
  - Many rounds of code review & refinement
- **Future work**
  - Broader hardware support (GPU & NIC)
  - Software ecosystem enabling

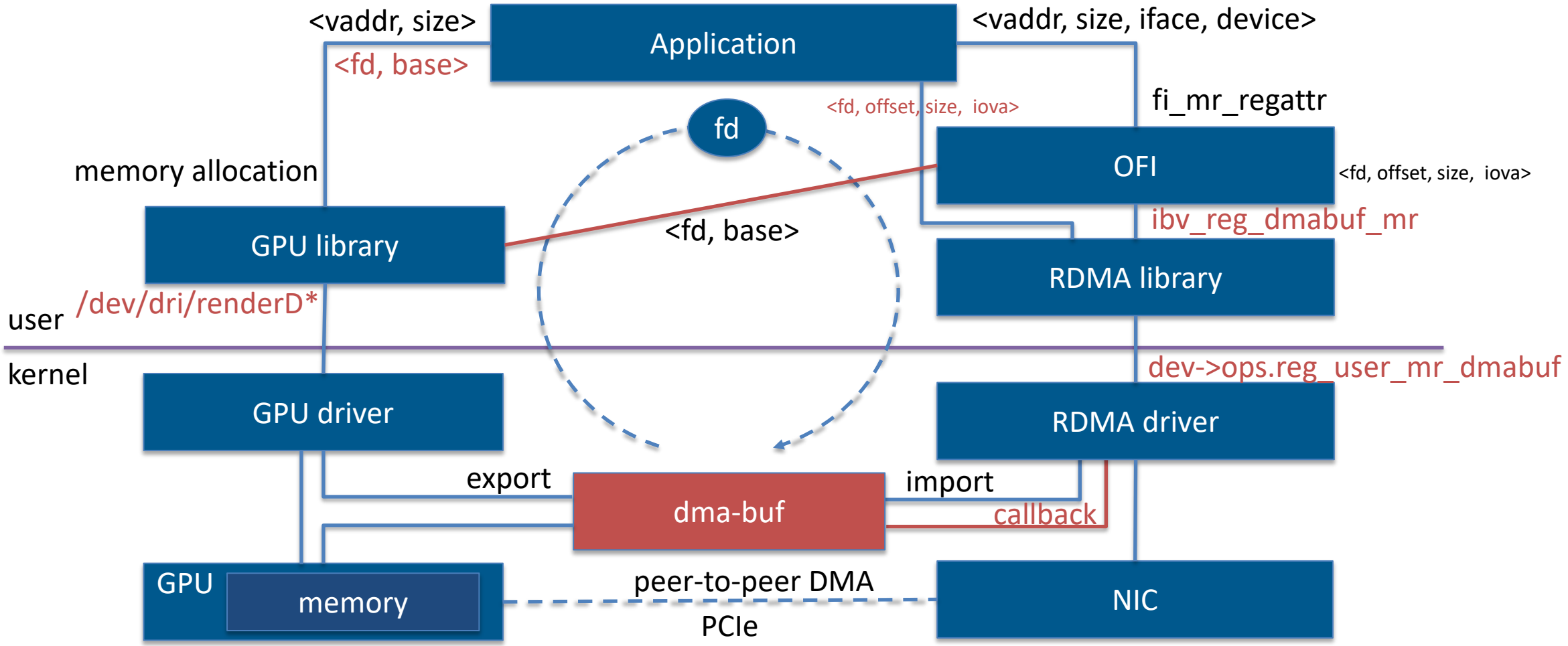
GPU RDMA OVERVIEW



PROTOTYPE PRESENTED LAST YEAR

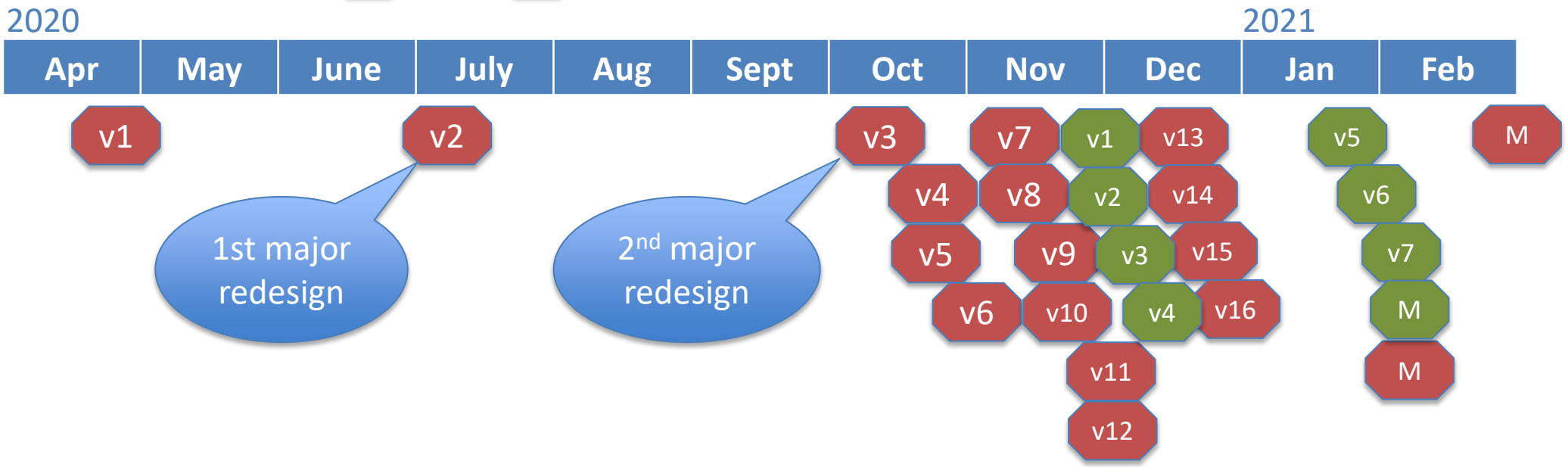


UPDATED SOFTWARE ARCHITECTURE



CURRENT STATUS

- **The latest patch sets have been merged into upstream**
  - Kernel patch set has landed in kernel 5.12
  - User space patch set is included in rdma-core-34.0
- **How we get there**
  - Kernel patch set: v1 …… v16 , merged to linux-rdma on Jan 21, merged by Linus on Feb 22
  - User space patch set: v1 …… v7 , merged to rdma-core on Jan 26

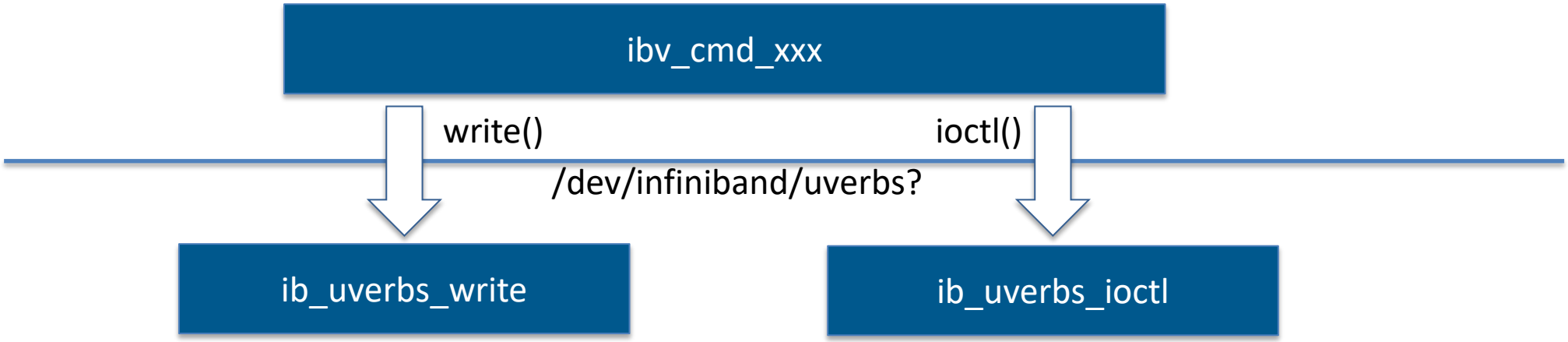


MAJOR CHANGES

	March 2020	March 2021
User verbs command interface	Device write	Device ioctl
User space API	ibv_reg_mr_fd() ibv_rereg_mr_fd() *<addr, len, fd_type, fd>*	ibv_reg_dmabuf_mr() *<offset, len, iova, fd>*
Dma-buf attach method	Static attach	Dynamic attach
Buffer must be pinned	Yes	No
NIC on-demand paging support	Not required	Required
PyVerbs support	No	Yes
Unit tests	No	Yes

Full change history available in the cover letter of the change sets:
[PATCH v16 0/4] RDMA: Add dma-buf support — Linux RDMA and InfiniBand development (spinics.net)
[PATCH rdma-core v7 0/6] Add user space dma-buf support — Linux RDMA and InfiniBand development (spinics.net)

WRITE VS IOCTL



This is the classic interface. Used by commands existed before a certain time.

Commands defined in ib_user_verbs.h

All new commands need to use this interface.

Commands defined in ib_user_ioctl_cmds.h

NEW USER SPACE API

- **A single call to register dma-buf based memory region**
- **Compared with regular memory region:**
  - Use “offset” instead of “vaddr” for the starting address
    - Offset is relative to the beginning of the dma-buf object
  - New “iova” parameter for added flexibility on how to specify the address used in RDMA operations
    - Examples
      - iova == offset: use offset in RDMA commands
      - iova == vaddr: use vaddr in RDMA commands
  - New “fd” parameter for the dma-buf object
- **The library used for device memory allocation should provide ways to:**
  - query the dma-buf fd associated with a virtual address
  - query the offset of a virtual address within the associated dma-buf object

```
struct ibv_mr *
ibv_reg_dmabuf_mr(
    struct ibv_pd *pd,
    uint64_t offset,
    size_t length,
    uint64_t iova,
    int fd,
    int access);
```

DMA-BUF: STATIC VS DYNAMIC ATTACHMENT

- **Static attach**

```
struct dma_buf_attachment *dma_buf_attach(dma_buf, dev);
```

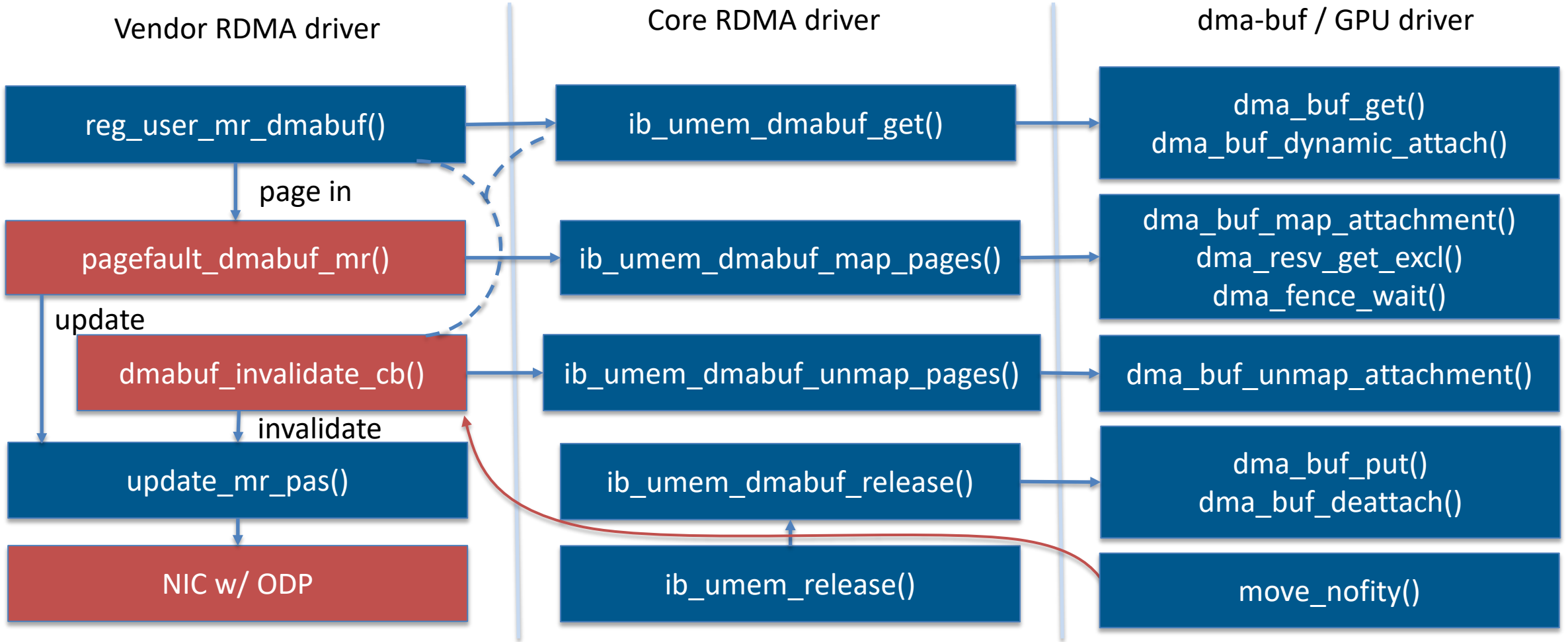
  - Buffer is pinned upon return
  - However, GPU drivers don't allow pinning device memory
    - Move to system memory when static attachment is mapped

- **Dynamic attach**

```
struct dma_buf_attachment *
dma_buf_dynamic_attach(dma_buf, dev, importer_ops, importer_priv);
```

  - Buffer is not pinned
  - Importer provides a callback function which is called whenever the buffer moves
  - Inside the callback, the importer should invalidate address mapping related to the buffer
  - Preferred way for RDMA usage

WORK WITH DYNAMICALLY ATTACHED DMA-BUF



PYVERBS & UNIT TESTS

- **PyVerbs is a Python interface for the Verbs API**
  - Part of the user space rdma-core package
  - Heavily used by the unit tests
- **All verbs API functions are required to have PyVerbs support and unit tests**
- **New additions to PyVerbs**
  - A set of utility functions that allocate dma-buf object via the DRM ioctl interface
  - A new class DmaBuf that wraps around the dma-buf allocation utility functions
  - A new class DmaBufMR as a new type of MR object
  - Updated infrastructure to support modules with mixed Cython and C source
- **New additions to unit tests**
  - A new class DmaBufMRTest to test memory registration functionality
  - A new class DmaBufRC for creating resources for RC traffic test
  - A new class DmaBufTestCase for testing RC traffic using dma-buf based MRs

LIBFABRIC UPDATE

- **Device memory is registered using fi_mr_regattr() which allows passing iface and device attributes**
  - Dma-buf is used when iface is FI_HMEM_ZE
    - Device memory allocation: zeMemAllocDevice()
    - Get dma-buf fd: zeMemGetIpcHandle()
    - For calculating offset: zeMemGetAddressRange()
- **Device memory support is provider dependent**
  - Indicated by the FI_HMEM capability flag
  - Provider can set MR mode FI_MR_HMEM to indicate that registration is always needed for device memory even for local access
  - Support of individual iface is also provider dependent
  - Currently the shm, verbs, and rxm providers support FI_HMEM_ZE
  - The PR that add dma-buf support to the verbs provider:

[prov/verbs: Add dmabuf MR support by j-xiong · Pull Request #6599 · ofiwg/libfabric (github.com)]

```
enum fi_hmem_iface {
        FI_HMEM_SYSTEM  = 0,
        FI_HMEM_CUDA,
        FI_HMEM_ROCR,
        FI_HMEM_ZE,
};

struct fi_mr_attr {
    ……
    enum fi_hmem_iface iface;
        union {
                uint64_t reserved;
                int cuda;
                int ze;
        } device;
};
```

FUTURE WORK

- **Broader hardware support**
  - GPU
    - Allow GPU to optionally pin the buffer?
      - This should work fine with current RDMA stack, just the invalidate callback is never called
    - Some GPU drivers don't support full dma-buf operations
      - e.g. dma_buf_ops.map_dma_buf()
  - NIC
    - Currently on-demand paging capability is required
    - Can this requirement be relaxed?
      - Related to GPU buffer pinning restriction
    - Page-level faulting is somewhat overkill
      - Any NIC supporting MR level faulting?
- **Software ecosystem enabling**
  - From applications to middlewares

2021 OFA Virtual Workshop

THANK YOU

Jianxin Xiong

Intel Corporation

intel