OPENFABRICS ALLIANCE

2024 OFA Virtual Workshop

# OPEN FABRIC INTERFACE 2.0 UPDATE

**Jianxin Xiong**

Intel Corporation

intel.

# OUTLINE

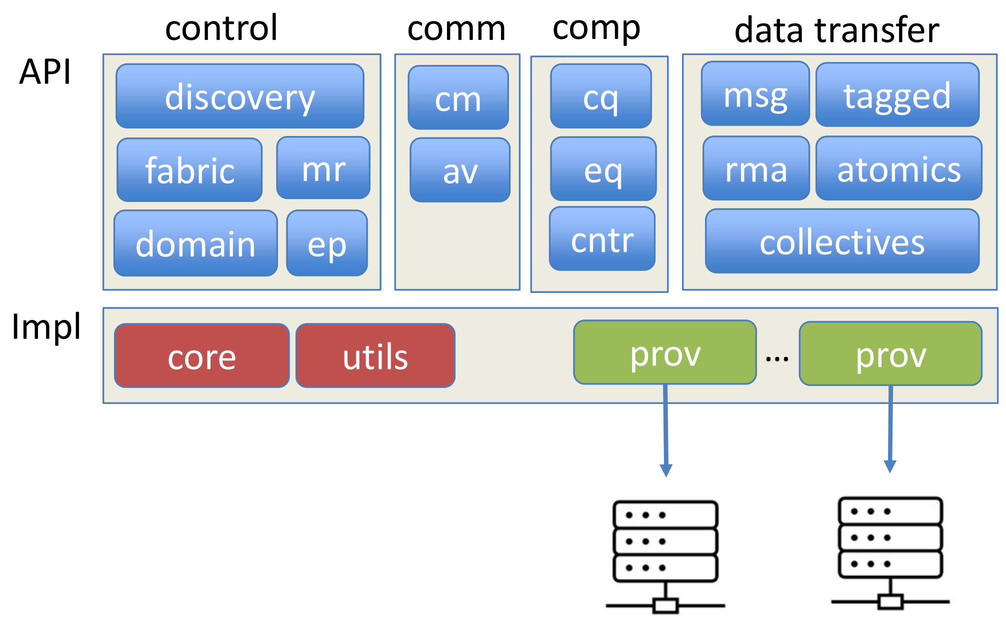
- Introduction
- Proposed OFI 2.0 Changes
- Timeline

# OFI (LIBFABRIC) IN A NUTSHELL

## OFI Features

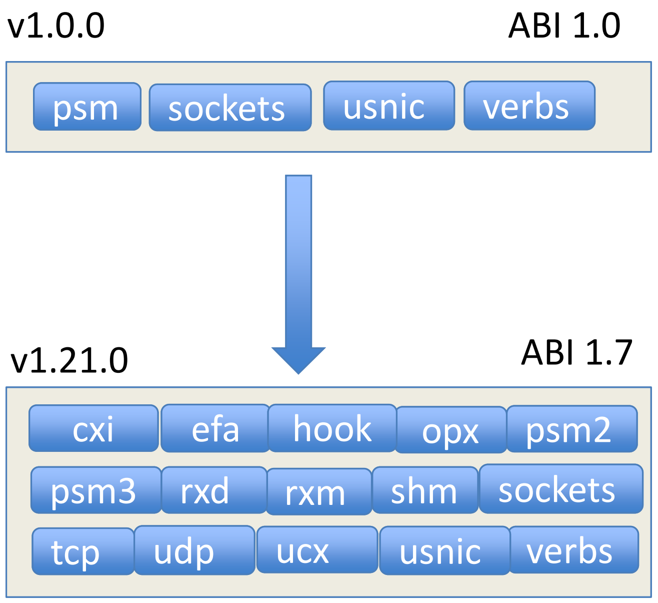
- **Enable advanced fabric features**
  - Optimized software paths
  - OS bypass
  - Zero-copy transfers
  - Minimized memory footprint
- **Fabric portability**
  - Single API, many providers
  - Implementation flexibility for providers
  - Capability discovery at runtime

## OFI Architecture



## THE SUCCESS OF GROWTH

### Providers



### Timeline

- **Initial libfabric commit: Nov 7, 2013**
- **libfabric v1.0.0: Apr 6, 2016**
- **libfabric v1.21.0 (the latest): Mar 29, 2024**
  - 55 releases in total, feature + bug fix
- **major new features since v1.0.0**
  - Authorization keys, multicast, FI_ADDR_STR, FI_LOCAL_COMM, FI_REMOTE_COMM, FI_HMEM, FI_CONTEXT2, new MR mode bits, FI_RMA_PMEM, NIC attributes, collectives
- **middleware**
  - Intel MPI, OpenMPI, MPICH, SHMEM, GASNet, Charm++, oneCCL, NCCL, DAOS, ……

## WHY 2.0?

- **We have been able to maintain API and ABI backward compatibility so far**
  - API: existing application source should be able to compile against newer libfabric headers & libraries and run
  - ABI: existing application binary should be able to run with newer libfabric libraries
  - This is possible because:
    - API changes are always “appending”, never “removing” or “reordering”
    - ABI compatibility stubs are used to do runtime data-structure / parameter conversion

- **Bumping the version to 2.0 allows making changes that breaks API/ABI compatibility**
  - Simplification:
    - remove rare used / hard to use features / options
    - present easier to understand interface to the user
  - Optimization:
    - allow more efficient provider implementation
  - New features:
    - Add new API: doesn’t break API
    - Redefine existing API: may or may not break API

We still want to maintain ABI backward compatibility for features carried over from 1.x!

## PROPOSED 2.0 CHANGES

- **Simplification**
  - Remove asynchronous AV insertion (rarely used)
  - Remove FI_AV_MAP support
  - Remove FI_THREAD_FID and FI_THREAD_ENDPOINT (hard to use)
  - Consolidate control progress and data progress
  - Remove comp_order attributes (rarely supported)
  - Remove total_buffered_recv field (deprecated)
  - Remove fid_wait and fid_poll (reduce complexity)
  - Remove FI_WAIT_MUTEX_COND (unimplemented)
  - Remove FI_MR_BASIC, FI_MR_SCALABLE and FI_LOCAL_MR (deprecated)
  - Remove asynchronous MR registration (unused)
- **Optimization**
  - Restrict an endpoint to a single CQ (more efficient progress)
  - fi_log: new levels, redefine subsys
  - Separate FI_DIRECTED_RECV bits for msg & tagged
  - Refined FI_HMEM capabilities
  - Refined inject size and max size for different ops
- **New features**
  - Add new fi_fabric2 call (consistent fi_info parameter)
  - Add new FI_ATOMIC_DIFF op
  - Add new atomic data types FI_BFLOAT16, FI_FLOAT16
  - Add new peer group feature
  - Define new tag formats

# OFI 2.0 CHANGES (1~2): ADDRESS VECTOR

## Remove asynchronous AV insertion

- **Currently behavior:**
  - when fi_av_open() is called with FI_EVENT flag, insertion on the resulting AV will be asynchronous.
  - The feature is rarely used while makes the implementation more complicated.
- **Proposed change:**
  - Remove the feature

## Remove FI_AV_MAP support

- **Proposed change:**
  - Keep the FI_AV_MAP enum value
  - Make FI_AV_MAP behave the same as FI_AV_TABLE
- **The change is only visible to the provider**
  - Application can continue to use FI_AV_MAP w/o noticing the difference
- **The purpose is to free up some bits in fi_addr_t**
  - See the peer group feature

## OFI 2.0 CHANGES (3~4): THREADING MODEL & PROGRESS

### Simplify threading models

- **Proposed change:**
  - Remove FI_THREAD_FID and FI_THREAD_ENDPOINT
  - Keep FI_THREAD_SAFE, FI_THREAD_DOMAIN, FI_THREAD_COMPLETION
  - Recommend FI_THREAD_DOMAIN for multi-thread app with regular endpoint
  - Recommend FI_THREAD_COMPLETION for multi-thread app with scalable endpoint
- **Reason**
  - The removed threading models are hard to use due to the complexity associated with the completion structure

### Consolidate progress models

- **Proposed change (domain_attr):**

```
enum fi_progress control_progress;
enum fi_progress data_progress;
```

↓

```
enum fi_progress control_progress; // unused
union {
        enum fi_progress data_progress;
        enum fi_progress progress;
};
```

- **Reason**
  - applications usually set them to be the same
  - providers usually use data_progress to determine its behavior

# OFI 2.0 CHANGES (5~6): TX & RX ATTRIBUTES

## Remove comp_order attributes

- **Proposed change:**
  - Remvoe the use of fi_tx_attr->comp_order and fi_tx_attr->comp_order attributes man pages and code
  - Keep the field in the structures for backward compatibility
- **Reason**
  - Most hardware don't support in-order completion (only IB Verbs does)
  - Application don't need this, either.

## Remove total_buffered_recv field

- **Proposed change:**
  - Remove the use of fi_rx_attr->total_buffered_recv from man pages and code
  - Keep the field in the structure for backward compatibility
- **Reason**
  - The field has already been deprecated
  - Even today, it's a hint only. A provider can choose to ignore it.

## OFI 2.0 CHANGES (7~8): WAIT SET & POLL SET

### Remove fid_wait and fid_poll

- **Wait set / poll set allows aggregating multiple wait objects into one**
- **Proposed change:**
  - Remove fid_wait (wait set) and fid_poll (poll set) from the API
- **Reason:**
  - Supporting these adds complexity to the provider implementation
  - Can get the wait object and use native poll / epoll directly instead

### Remove FI_WAIT_MUTEX_COND

- **Proposed change:**
  - Remove the wait object type FI_WAIT_MUTEXT_COND
- **Reason:**
  - It's not implemented by any provider

## OFI 2.0 CHANGES (9~10): MEMORY REGION

### Remove deprecated MR modes

- **Proposed change:**
  - Remove FI_MR_BASIC, FI_MR_SCALABLE and FI_LOCAL_MR
- **Reason:**
  - These MR modes are for compatibility with libfabric versions older than v1.5
  - They have been deprecated for a long time

### Remove asynchronous MR registration

- **Current behavior:**
  - Binding an event queue to a domain with FI_MR_REG flag causes all memory registration on this domain to be asynchronous
- **Proposed change:**
  - Remove this option. Make memory registration to be always synchronous
- **Reason:**
  - No native support
  - Complicate the implementation

# OFI 2.0 CHANGES (11~12): CQ AND FI_LOG

## Restrict an endpoint to one CQ

- **Current Behavior:**
  - An endpoint can bind different CQs for send and recv context
- **Proposed change:**
  - An endpoint can only bind to one CQ
- **Reason:**
  - The change simplifies both application and provider logic for making progress
  - There is no hard reason to use separate CQ

## Refine fi_log

- **Proposed change:**
  - Redefine subsys as a flag
  - Add a new log level (FI_LOG_ERROR), and maybe a level between FI_LOG_INFO and FI_LOG_DEBUG
- **Reason”**
  - subsys is seldom used, changing to flag simplifies the filter logic and allows future extension
  - New log levels are needed for finer control on the verbose level

## OFI 2.0 CHANGES (13~14): CAP BITS

### Separate FI_DIRECTED_RECV for msg &tagged

- **Proposed change:**
  - Add new capability bits for FI_DIRECTED_RECV for msg and tagged ops.
  - Keep the current one to cover both
- **Reason:**
  - Providers may only support the capability for one type of ops

### Refined FI_HMEM capabilities

- **Proposed change:**
  - Add hmem_attr to fi_info.

```
struct fi_hmem_attr {
        char *name;
        enum fi_hmem_iface iface;
        bool dmabuf_reg;
        bool gdr_copy;
        bool async_copy;
};

struct fi_info {
        ……
        struct fi_hmem_attr *hmem_attr;
}
```

## OFI 2.0 CHANGES (15~16): OP SIZES

### Refined inject size for ops

- **Current behavior**
  - The single tx_attr->inject_size covers all ops (msg, tagged, rma)
- **Proposed change**
  - Add query method to fi_tagged_ops, fi_msg_ops, and fi_rma_ops which will return inject size as part of the result
  - The API call will be fi_query_msg, fi_query_tagged, and fi_query_rma.

### Refined max size for ops

- **Current behavior:**
  - ep_attr->max_msg_size set the transport limit
  - atomics and collectives have their own size limits that can be queried by fi_query_atomic and fi_query_collective
  - msg, tagged, and rma may have different limit by there is no way to know
- **Proposed change:**
  - Use the same query method for the inject size to get the max size at the same time.

## OFI 2.0 CHANGES (17~18): FI_FABRIC

### Require fi_info be allocated with API

- **Current behavior:**
  - fi_info can be hand crafted
- **Proposed change:**
  - Require that fi_info should be allocated by fi_alloc_info() or fi_dupinfo() or be returned from fi_getinfo().
- **Reason:**
  - allow the library to allocate hidden fields for internal use

### Add fi_fabric2

- **Current behavior:**

```
int fi_fabric(struct fi_fabric_attr *attr,
              struct fid_fabric **fabric,
              void *context);
```

- **Proposed change:**

```
int fi_fabric2(struct fi_info *info,
              struct fid_fabric **fabric,
              uint64_t flags,
              void *context);
```

- **Reason:**
  - Consistent interface as other open calls
  - Get access to other info not available in fabric_attr

## OFI 2.0 CHANGES (19~20): ATOMICS

### New atomic op FI_ATOMIC_DIFF

- **Proposed change:**
  - Add a new atomic op FI_ATOMIC_DIFF, which performance the operation (target = target – source)
- **Reason:**
  - This is a useful operation that may be supported by some hardware

### New atomic data types FI_BFLOAT16 & FI_FLOAT16

- **Proposed change:**
  - Add new atomic data types FI_BFLOAT16 and FI_FLOAT16
- **Reason:**
  - These are data types used in AI/ML applications

## OFI 2.0 CHANGES (21): PEER GROUP

- **Peer group maps to "communicator" concept of HPC and AI applications**
- **Peer groups are identified as integer "group id", which are then embedded into high bits of "fi_addr_t", with the help of a new function:**

```
fi_addr_t fi_group_addr(fi_addr_t fi_addr, uint32_t group_id);
```

- **The group id is chosen by the user, between 0 and domain_attr->max_group_id.**
- **Peer group support:**
  - Request by setting hints->domain_attr->max_group_id to non-zero
  - Check fi_info->domain_attr->max_group_id for provider support
    - fi_getinfo() may fail if asked for too many
    - May get more than asked for
- **Benefit:**
  - Free up tag bits that might have been used by communicator id
  - Increase the effectiveness of tag hashing for improved tag matching performance

# OFI 2.0 CHANGES (22): NEW TAG FORMAT

- **Current behavior:**
  - ep_attr->mem_tag_format is a bit map with alternating segments of 0's and 1's, representing different semantic fields in the tag.
  - hard to use

0xFFFF0000FFFF0000: four 16-bit fields

- **Proposed change:**
  - Use the lower bits to define a set of "well-known" tag usage models

| Tag format | FI_TAG_BITS | FI_TAG_MPI | FI_TAG_CCL |
|---|---|---|---|
| Tag layout | 64-bit tags | 32-bit tag + 32-bit payload id | 64-bit payload id |
| Matching | Allow wildcard | Allow wildcard | Exact match only |
| Tag setting | Direct set | fi_tag_mpi(tag, payload_id) | Direct set |
| Ignore bits | Direct set | FI_MPI_IGNORE_TAG, FI_MPI_IGNORE_PAYLOAD | 0 |

- **Benefits**
  - Allow providers to optimize tag-matching algorithm

## TIMELINE

- **A longer release cycle for the first 2.0 release**

| 2.0.0 alpha | 2.0.0 beta | 2.0.0 GA |
|---|---|---|
| July 2024 | Sept 2024 | Nov 2024 |

- **What to expect at each stage**
  - 2.0 alpha: mostly feature complete
  - 2.0 beta: feature complete and validated
  - 2.0 GA: issues discovered after beta fixed

- **What about 1.x releases**
  - The libfabric “main” branch is for 2.0 development
  - The 1.x development continues on the “v1.x-main” branch
  - There will be two more feature releases for the 1.x series this year: 1.22 in July and 1.23 in Nov
  - There may be some bug fix releases as well

OPENFABRICS ALLIANCE

2024 OFA Virtual Workshop

## THANK YOU

Jianxin Xiong

Intel Corporation

intel.