OPENFABRICS ALLIANCE

Scalable Fabric Interfaces

Sean Hefty

Intel Corporation

10TH ANNUAL INTERNATIONAL DEVELOPER WORKSHOP

OFI software will be backward compatible

OFI WG Charter

Develop an extensible, open source framework and interfaces aligned with ULP and application needs for high-performance fabric services

Enable..

High performance

Minimal footprint

Reduced cache and memory footprint

Optimized software path to hardware

- Independent of hardware interface, version, features

Extensible

App-centric

Analyze application needs

- Implement them in a coherent, concise, high-performance manner

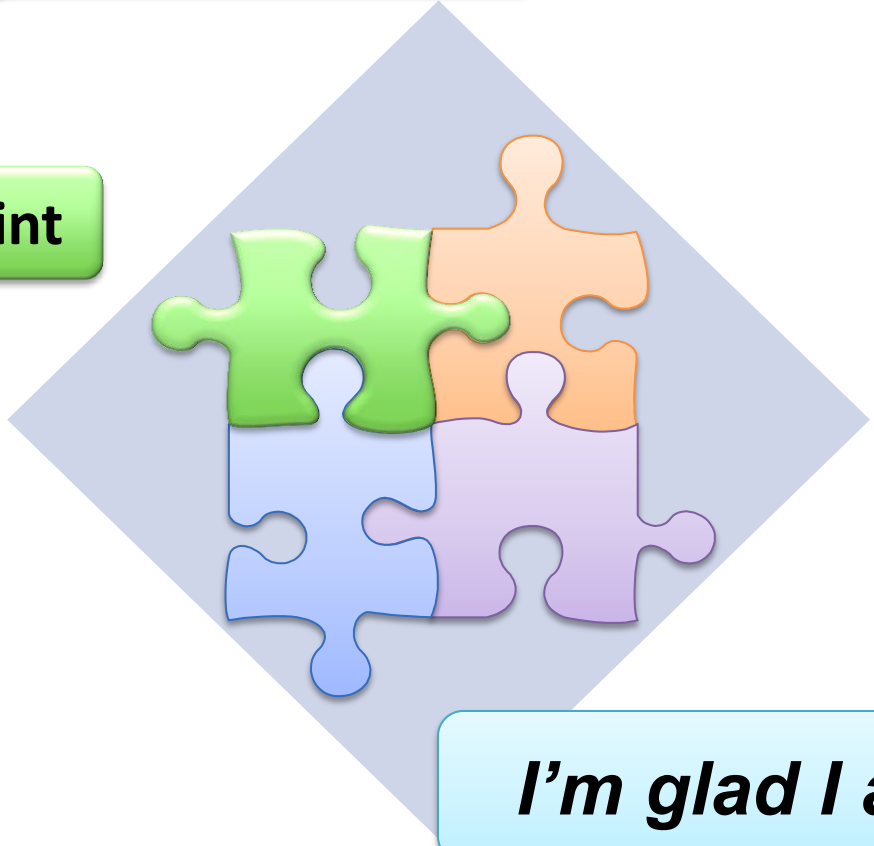
More agile development

- Time-boxed, iterative development
- Application focused APIs
- Adaptable

OPENFABRICS ALLIANCE

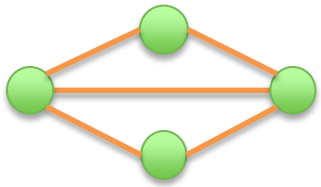
How can an API affect application scalability?

Minimal footprint



I'm glad I asked.

Communication



OPENFABRICS ALLIANCE

Reliable data transfers, zero copies to thousands of processes

```
struct rdma_route {
    struct rdma_addr          addr;
    struct ibv_sa_path_rec *path_rec;
    ...
};

struct rdma_cm_id {...};

rdma_create_id()
rdma_resolve_addr()
rdma_resolve_route()
rdma_connect()
```

Src/dst addresses stored per endpoint

456 bytes per endpoint

Path record per endpoint

Resolve single address and path at a time

All to all connected model for best performance

Scalable Communication

- *Application* driven communication models
- Reliable unconnected transfers
  - Abstract hardware features
    - SRQ, XRC, dynamic connections
- Optimize addressing
  - Resolve multiple resolution requests at once
  - Compact address data storage
    - Compressed address ranges, path data
- Support multiple resolution mechanisms
  - Optimized for different topologies and fabric sizes

SFI - Address Vectors

Store addresses/host names
- Insert range of addresses with single call

Share between processes

Reference entries by handle or index
- Handle may be encoded fabric address

Reference vector for group communication

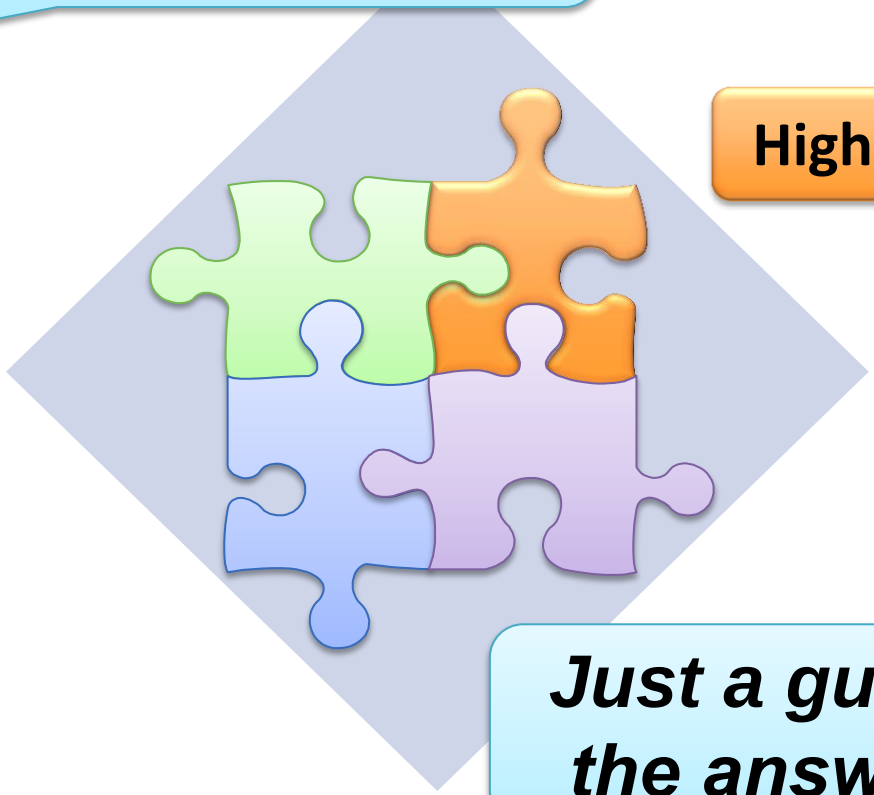
Example only

Start Range	End Range	Base LID	SL
host10	host1000	50	1
host1001	host4999	2000	2

Enable provider optimization techniques
- Greatly reduce storage requirements

OPENFABRICS ALLIANCE

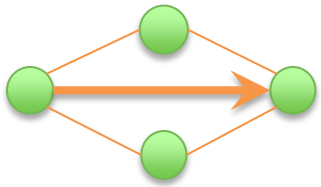
Can API changes unlock higher performance?



High performance

Just a guess, but is the answer "yes"?

Application Send



OPENFABRICS ALLIANCE

Significant SW overhead

```
struct ibv_sge {
    uint64_t        addr;
    uint32_t        length;
    uint32_t        lkey;
};

struct ibv_send_wr {
    uint64_t            wr_id;
    struct ibv_send_wr *next;
    struct ibv_sge     *sg_list;
    int                 num_sge;
    enum ibv_wr_opcode opcode;
    int                 send_flags;
    uint32_t            imm_data;
    ...
};
```

Application request

<buffer, length, context>

3 x 8 = 24 bytes of data needed

SGE + WR = 88 bytes allocated

Requests may be linked - next must be set to NULL

Must link to separate SGL and initialize count

App must set and provider must switch on opcode

Must clear flags

28 additional bytes initialized

Provider Send

```
For each work request
    Check for available queue space
    Check SGL size
    Check valid opcode
    Check flags x 2
    Check specific opcode
    Switch on QP type
        Switch on opcode
    Check flags
        For each SGE
            Check size
            Loop over length
    Check flags
    Check
    Check for last request
Other checks x 3
```

Most often 1
(overlap operations)

Often 1 or 2
(fixed in source)

Artifact of API

QP type usually fixed in source

Flags may be fixed or app may have taken branches

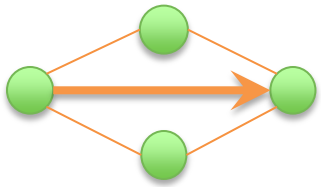
19+ branches including loops

100+ lines of C code
50-60 lines of code to HW

Scalable Transfer Interfaces

- *Application* optimized code paths based on usage model
- Optimize call(s) for single work request
  - Single data buffer or 2-entry SGL
  - Still support more complex WR lists/SGL
- Per endpoint send/receive operations
  - Separate RMA function calls
- Pre-configure data transfer flags
  - Known before post request
  - Select software path through provider

SFI – Send Message



OPENFABRICS ALLIANCE

50-60 lines of C-code

```
Allocate WR
Allocate SGE
Format SGE – 3 writes
Format WR – 6 writes
```

generic send call

```
Loop 1
    Checks – 9 branches
    Loop 2
            Check
            Loop 3
    Checks – 3 branches
Checks – 3 branches
```

Reduce setup cost
- Tighter data

25-30 lines of C-code

```
Direct call – 3 writes
```

optimized send call

```
Checks – 2 branches
```

Eliminate loops and branches
- Remaining branches predictable

Selective optimization paths to HW
- Manual function expansion

Completions

OPENFABRICS ALLIANCE

Application accessed fields

```
struct ibv_wc {
    uint64_t        wr_id;
    enum ibv_wc_status status;
    enum ibv_wc_opcode opcode;
    uint32_t        vendor_err;
    uint32_t        byte_len;
    uint32_t        imm_data;
    uint32_t        qp_num;
    uint32_t        src_qp;
    int             wc_flags;
    uint16_t        pkey_index;
    uint16_t        slid;
    uint8_t         sl;
    uint8_t         dlid_path_bits;
};
```

App must check both return code and status to determine if a request completed successfully

Provider must fill out all fields, even those ignored by the app

Provider must handle all types of completions from any QP

Developer must determine if fields apply to their QP

Single structure is 48 bytes likely to cross cacheline boundary

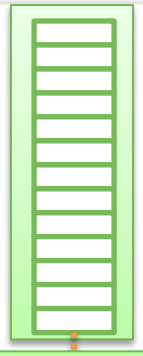
Scalable Completion Interfaces

- *Application* optimized code paths based on usage model
- Use compact data structures
  - Only needed data exchanged across interface
  - Limited to fields required by application
  - Separate addressing from completion data
  - Report errors ‘out of band’
- Per CQ operations
  - Support multiple wait objects
  - Allow provider to optimize event signaling

SFI – Events

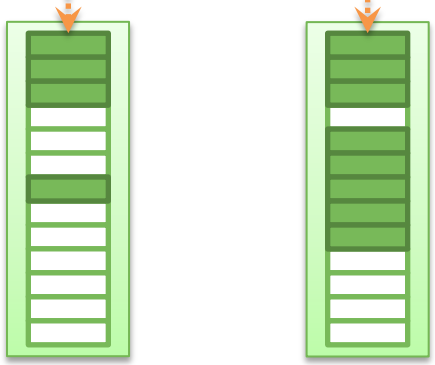
OPENFABRICS ALLIANCE

Generic completion



App selects completion structure

read CQ



Send: +4-6 writes, +2 branches
Recv: +10-13 writes, +4 branches

Op context

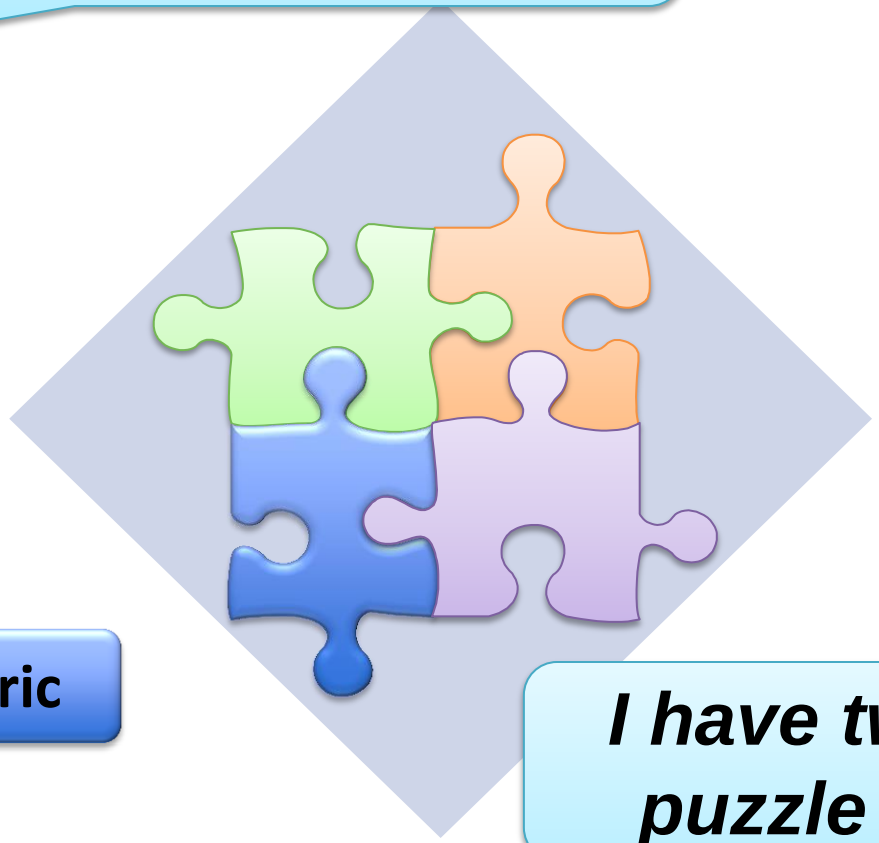
optimized CQ

+1 write, +0 branches

Support provider updating counters

OPENFABRICS ALLIANCE

Is there anything else behind this proposal?



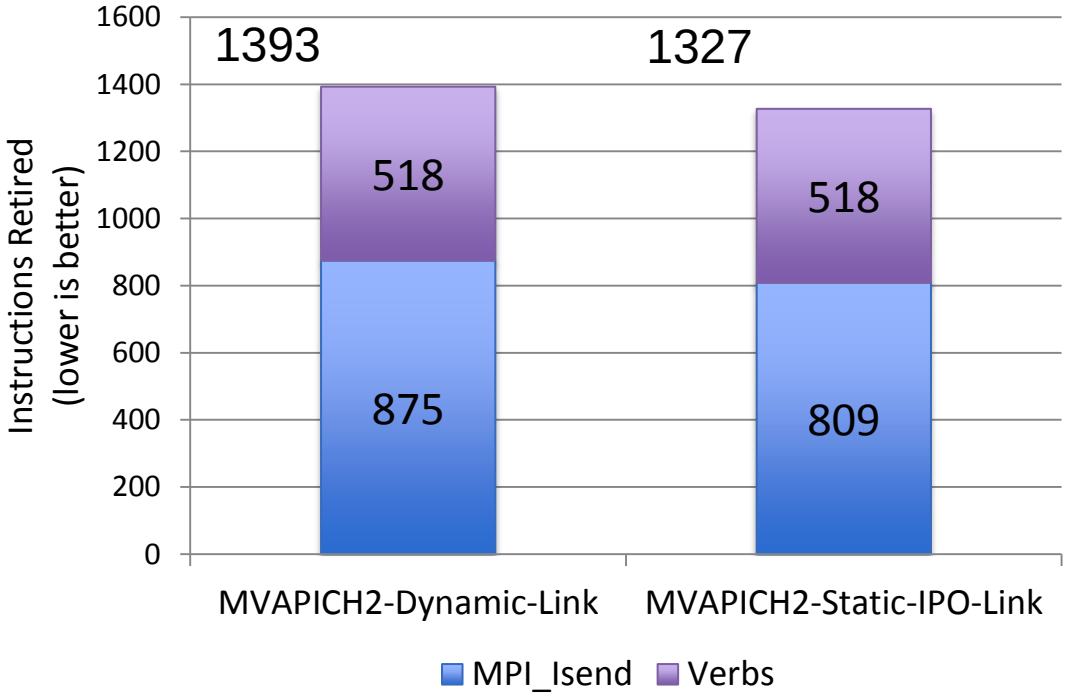
App-centric

I have two more puzzle pieces.

Application Interface Mismatch

Instructions retired in MPI_Isend

Lookup connection, check memory registration, formatting requests, etc.



MVAPICH2-2.0rc1 (latest) code is used with default configuration options (CH3:mrail)
All userspace instructions are counted for full execution of MPI_Isend
Memory copies and locks are also included in the component that uses them

MVAPICH2 lib compile flags: '-O3 –DNDEBUG –ipo'
App compile flags: '-O3 –DNDEBUG –ipo -finline-limit=2097152 -no-inline-factor -inline-max-per-routine=10000000 -inline-max-per-compile=10000000 -Bstatic -lmpich -Bdynamic -lopa -lmpl -libverbs -libumad -libmad -lrdmacm -lrt -lpthread

Application-Centric Interfaces

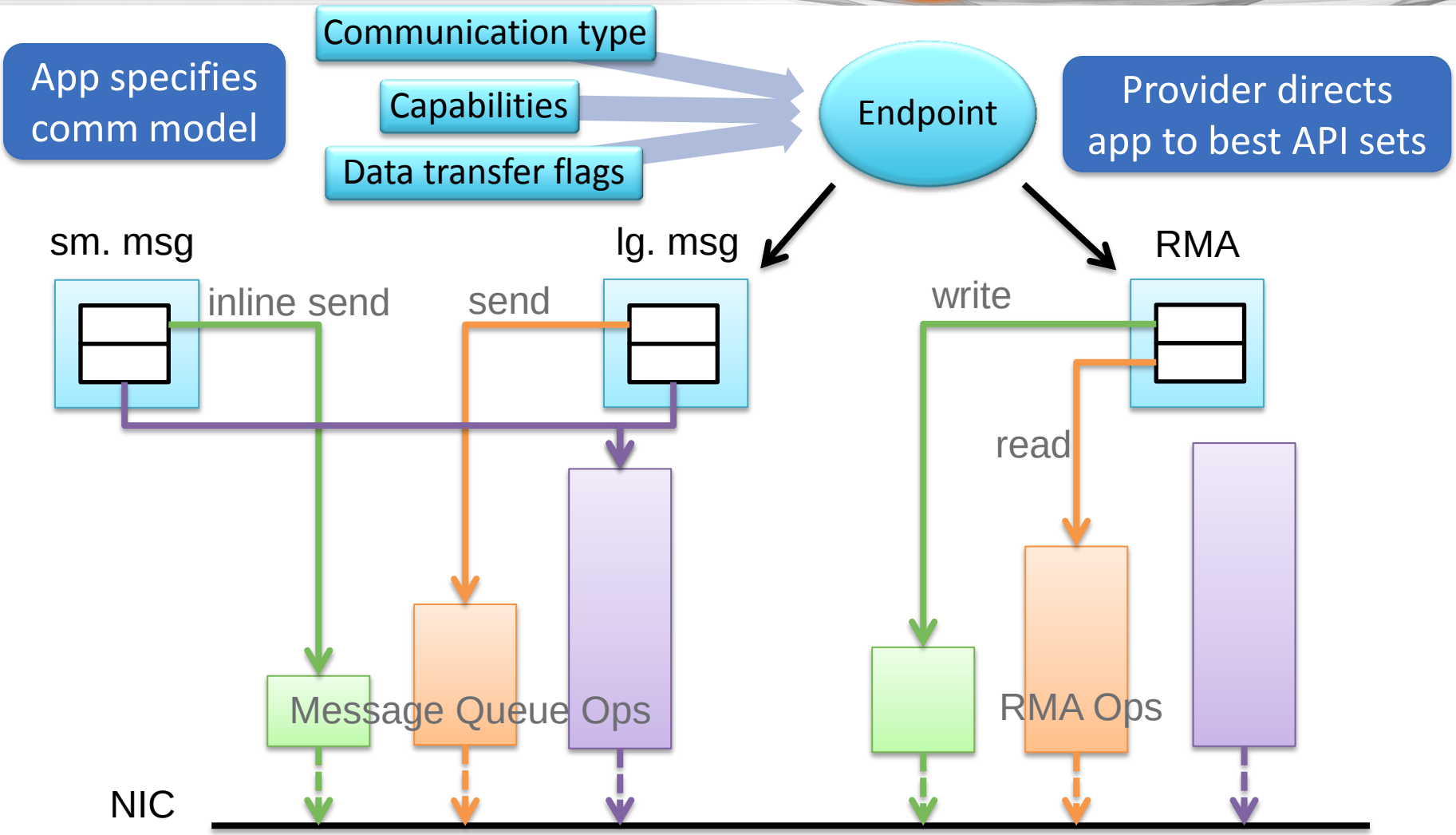
Reducing instruction count *requires* a better application impedance match

- Collect application requirements
- Identify common, fast path usage models
  - Too many use cases to optimize them all
- Build primitives around *fabric services*
  - Not device specific interface

Application-Centric Interfaces

- **Myth**: app-centric interfaces imply more overhead
  - Poor implementations result in poor performance
  - Difficult to use APIs are likely to result in poor implementations
  - Provider knows best method for accessing their HW
  - These are still low-level interfaces (C), just not device interfaces (assembly)

Application Configured Interfaces



OPENFABRICS ALLIANCE

What's the purple piece representing again?

Extensible

Extensible Framework

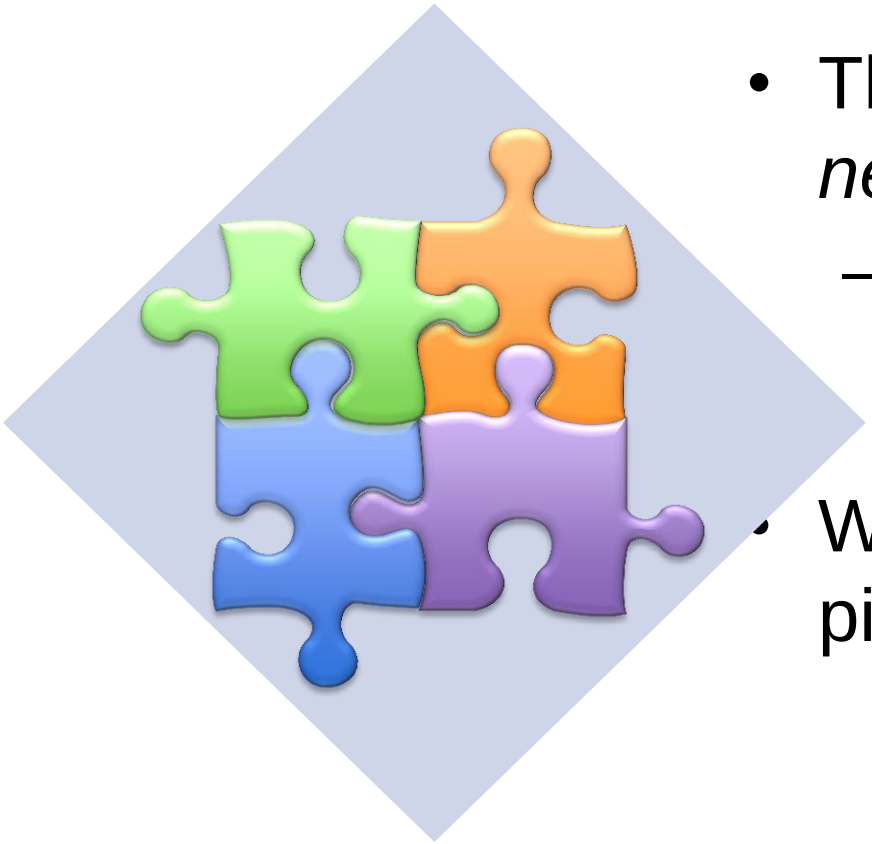
Focus on longer-lived interfaces – software leading hardware

- Take growth into consideration
- Reduce effort to incorporate new *application* features
  - Addition of new interfaces, structures, or fields
  - Modification of existing functions
- Allow time to design new interfaces correctly
  - Support prototyping interfaces prior to integration

Future Extensions

- Design framework and APIs with anticipated capabilities
  - Stage delivering features
- Documentation defines supported usage models
- Use static inline calls to simplify application interactions with objects
  - Convert object-oriented model to procedural model

Claim



- These concepts are *necessary*, not revolutionary
  - Communication addressing, optimized data transfers, app-centric interfaces, future looking
- Want a solution where the pieces fit tightly together

OPENFABRICS ALLIANCE

Thank you!