OpenIB Architectural Overview

Roland Dreier

Sean Hefty

Hal Rosenstock

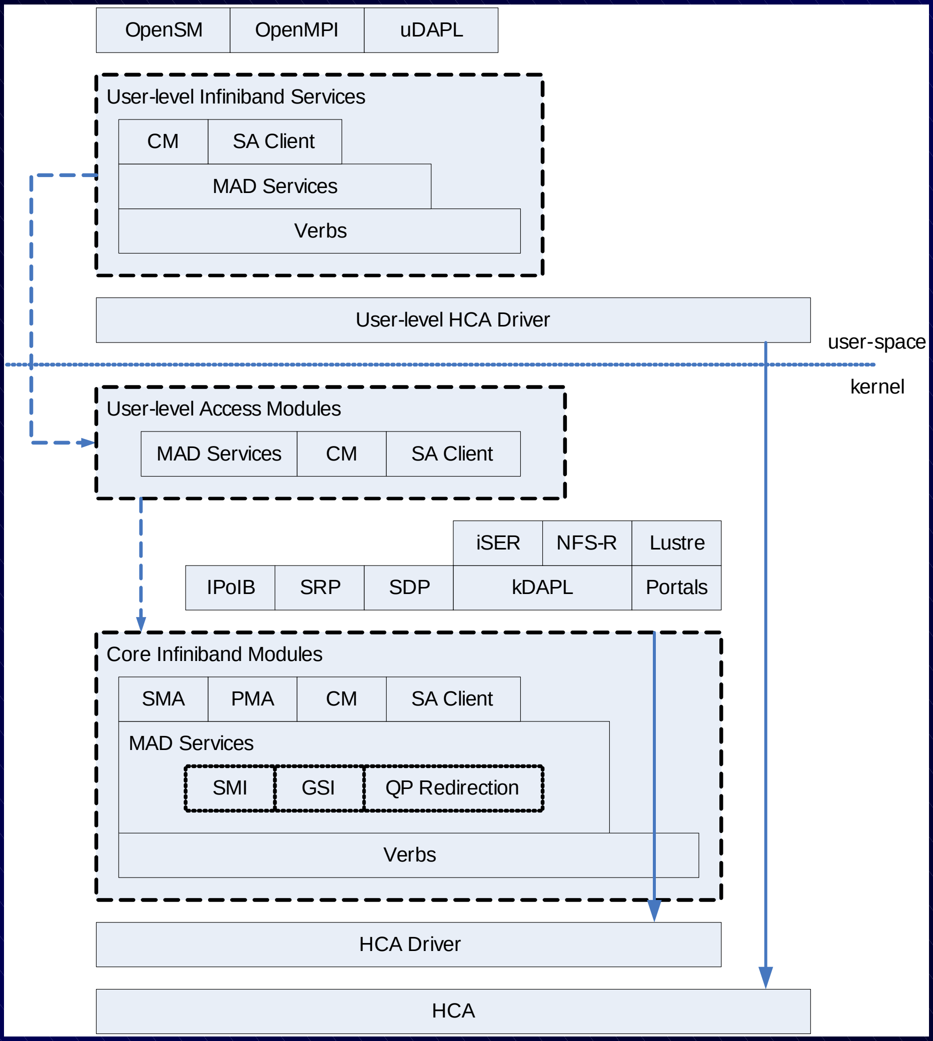
Shahar

12/21/04

Context of Architectural Overview

- Focus on initial components
- What is known now
  - May change/evolve somewhat as implementation develops

Architectural Components



OpenSM | OpenMPI | uDAPL

User-level Infiniband Services

CM | SA Client

MAD Services

Verbs

User-level HCA Driver

user-space

kernel

User-level Access Modules

MAD Services | CM | SA Client

iSER | NFS-R | Lustre

IPoIB | SRP | SDP | kDAPL | Portals

Core Infiniband Modules

SMA | PMA | CM | SA Client

MAD Services

SMI | GSI | QP Redirection

Verbs

HCA Driver

HCA

OpenIB Layers/Components/Modules

- IB driver
- IB core
- IPoIB
- OpenSM
- Diagnostic tools

IB Driver

- Mthca
  - MT23108 and MT25208 HCAs
    - MT25208 in compatibility mode
- Minor features to be completed:
  - RDMA and atomic support
  - APM support
  - These features are well-understood; API already exists; small amount of new code
- Larger features can be implemented as required by applications
  - Memory windows, shared receive queues, Mellanox-style “FMRs”

IB Driver

- Userspace verbs support
  - Requires kernel driver extensions
    - Allocate/map userspace doorbell pages
  - Control path talks to kernel through a file descriptor
    - Use read/write instead of ioctl to avoid “big kernel lock”
    - automatic clean-up when file descriptor is closed

IB Driver

- Userspace verbs support (cont’d)
  - Datapath functionality in a “libmthca”
    - Fast path operations require only a function call from application (through function pointer) to hardware access function
    - No context switches required in fast path
    - Interrupt-driven operation requires kernel to wake up process; performance is limited by kernel interrupt service and scheduler latency
  - Thread safety using pthread mutexes
    - applications must use pthreads -- “raw” threading with clone() won't work

IB Driver

- Optimizations
  - Code designed from the start paying attention to expensive operations (PCI reads, locked operations and cache misses)
  - Some low-hanging fruit in reduction of locking in interrupt service and CQ polling code paths
  - Extend verbs API for multiple CQ event handlers. In conjunction with MSI-X, allows CQs to be bound to a CPU for SMP performance

Core Infiniband Modules

Overview

- Collection of kernel-mode Infiniband modules
- Expose APIs required to access specific Infiniband functionality
  - Verbs
  - MAD services
    - SMI, GSI, QP redirection
  - MAD clients
    - CM, SA client, SMA, PMA

Core Infiniband Modules

Overview

- Entry point for HCA driver registration
  - Notifies clients of device insertion/removal
  - Hardware independent

Core Infiniband Modules

Verbs

- Provide infrastructure for kernel/user communication
- Split between extensions to kernel core IB layer and a “libibverbs” in userspace
- Handle memory pinning (mostly done in userspace with mlock() system call)
- Pass most operations on to device-specific driver (mthca)

Core Infiniband Modules

Verbs

- Provides direct path to HCA driver
  - Shared handles with HCA driver
  - Inline speed path operations for low latency
- Reference counting for proper cleanup
- Direct access to commonly accessed resource attributes
  - QP sizes, CQ sizes

Core Infiniband Modules

MAD Services

- Access to special QPs (QP0 /QP1)
- Request/response matching
  - timeouts
- RMPP support
- Support for QP redirection
  - Is this really needed ?
- Shared CQ to reduce interrupts
- Multi-threaded completion processing
  - One thread per port
- MAD buffer cache
- Zero-copy sending or receiving MADs

Core Infiniband Modules

MAD Services

- Minimal translations between clients and HCA driver
  - Use same work request structure when posting sends
  - Use similar structure when reporting completions
- Queuing of requests to handle QP overrun and for error recovery

Core Infiniband Modules

CM

- Implements CM protocol
  - IBA 1.1 compliance
  - Connection/disconnection requests (RC, UC)
    - REQ, REP, RTU
    - DREQ, DREP
    - REJ
    - MRA
  - Service ID resolution
    - SIDR_REQ, SIDR_REP
  - Path migration
    - LAP, APR
- API and HLD on openib-general list
  - Service ID range

Core Infiniband Modules

SA Client

- Issues and tracks queries to SA
- IPoIB requirements only currently supported
  - PathRecord requests
  - Manages multicast join/leave as well as group creation/deletion (MCMemberRecord)

Core Infiniband Modules

SA Client

- Other queries implemented based on ULP/application request/demand
  - ServiceRecord
    - Applications/ULPs
      - Sandia Portals
      - u/kDAPL
    - Methods
      - Set, Delete, Get
      - GetTable ?

Core Infiniband Modules

SA Client

- Other queries implemented based on ULP/application request/demand (cont'd)
  - MultiPathRecord
    - Multi HCA and port
    - RMPP required (both SA client and SA)
      - Only consumer of dual sided RMPP
    - SA optional feature
      - Not currently planned as part of current OpenSM work

User-level Access Modules

Overview

- Collection of related modules
- Support user-level clients accessing kernel-mode services
  - MAD services
  - MAD clients
    - CM, SA client
    - SMA, PMA are send only clients
      - HCA firmware performs IB agent functions

IPoIB

- Functionality
  - IPv4
    - Unicast
    - Multicast
    - DHCP ?
    - Already implemented; requires more testing
  - IPv6
    - Works; DHCPv6 not tested
  - Open Issues
    - Multicast router
    - Port bonding
  - Connected mode I-D support not currently supported
    - Nor are MIB I-Ds

OpenSM

- Vendor layer
  - Port to gen2 interfaces
    - Solicited send with timeouts
    - Use kernel RMPP ?
- Integrate gen1 changes
  - Primarily Mellanox changes
    - Mellanox Gold 1.6.1 is latest version
- Build environment (autotools)

OpenSM

- CLI (If/when needed)
  - Use standard SA queries
  - If additional functionality needed, special well defined interface for this access to be developed
    - Based on socket or pipes or similar mechanism

Diagnostic Tools

- Proposed Tools and Syntax
  - https://www.openib.org/svn/gen2/trunk/src/userspace/diags/diagtools-proposal.txt
- Host
  - Ibstatus: displays basic information obtained from the local IB driver
  - Ibping: validates connectivity between IB nodes using UD transport (or vendor MAD)
  - Ibroute: displays the unicast or multicast forwarding table for the specified LID
  - Ibtracert: traces the path from a source GID/LID to a destination GID/LID

Diagnostic Tools

- SMA/PMA query tools
  - smpquery: basic subset of standard SMP queries (NodeInfo, PortInfo, etc.)
  - perfquery: obtain the basic performance and error counters from the PMA at the node specified

Diagnostic Tools

Topology File

- Two alternatives
  - Gather topology from live topology and annotate (if necessary)
  - Create an expected topology from configuration and heuristics
- Planning on using first approach (live topology approach)

Topology File Syntax

```
      switchguids=0x8f104003f0313
      Switch  8 "S-0008f104003f0313"          # OpenIB port
0 lid 16
      [5]    "S-0008f104003f0314"[2]
      [6]    "S-0008f104003f0315"[2]
      [8]    "S-0008f104003f0317"[2]
      [7]    "S-0008f104003f0316"[2]

      switchguids=0x8f104003f0314
      Switch  8 "S-0008f104003f0314"          # OpenIB port
0 lid 17
      [2]    "S-0008f104003f0313"[5]
      [3]    "S-0008f104003f0312"[5]
      [4]    "S-0008f104003f0311"[5]
```

Topology File Syntax

```
hcaguids=0x8f10403965028
Hca     2 "H-0008f10403965028"          # OpenIB
[1]     "S-005442ba00001180"[22]                # lid 5

hcaguids=0x8f10403965014
Hca     2 "H-0008f10403965014"          # OpenIB
[1]     "S-005442ba00001180"[12]                # lid 4

hcaguids=0x8f10403965008
Hca     2 "H-0008f10403965008"          # OpenIB
[1]     "S-005442ba00001180"[8]         # lid 2 lmc

hcaguids=0x8f10403965010
Hca     2 "H-0008f10403965010"          # OpenIB
[1]     "S-005442ba00001180"[5]         # lid 3 lmc
```

Diagnostic Tools

- Network
  - ibnetdiscover: performs subnet discovery and outputs a human readable topology file
  - ibswitches: displays switches discovered in subnet from either topology file or live topology
  - Ibhosts: displays HCAs discovered in the subnet from either topology file or live topology
  - ibnetverify: scans the network to validate the connectivity and reports errors (from port counters)

Thank You