OPENFABRICS ALLIANCE

Management Scalability

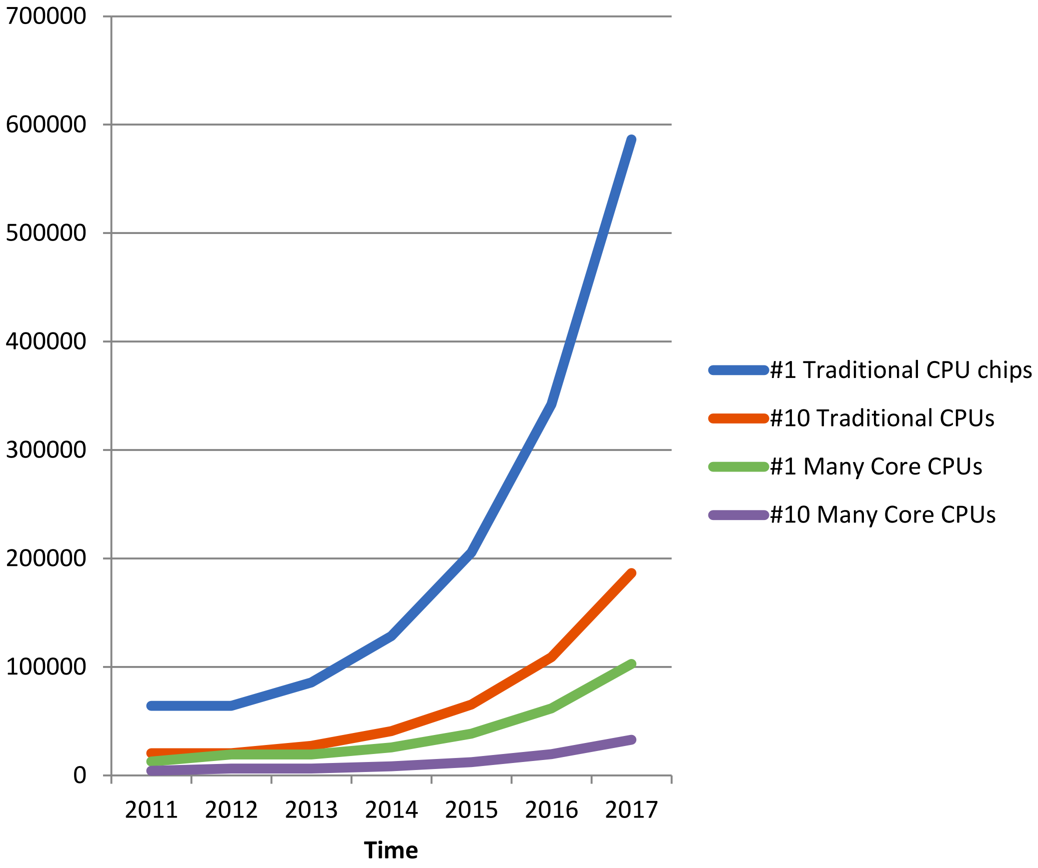
10TH ANNUAL INTERNATIONAL DEVELOPER WORKSHOP

Author: Todd Rimmer
Date: April 2014

Agenda

- Projected HPC Scalability Requirements
- Key Challenges
  - Path Record
  - IPoIB
  - Mgmt Security
  - Partitioning
  - Multicast
  - Notices
  - SA interaction
- Call to Action

Projected HPC Scalability Requirements



- Perf increase 2x/year
- Rapidly increasing node counts
  - HPC and Cloud
- Due to slower pace of interconnect speed growth
  - need multi-rail clusters
  - HCA counts will grow even faster

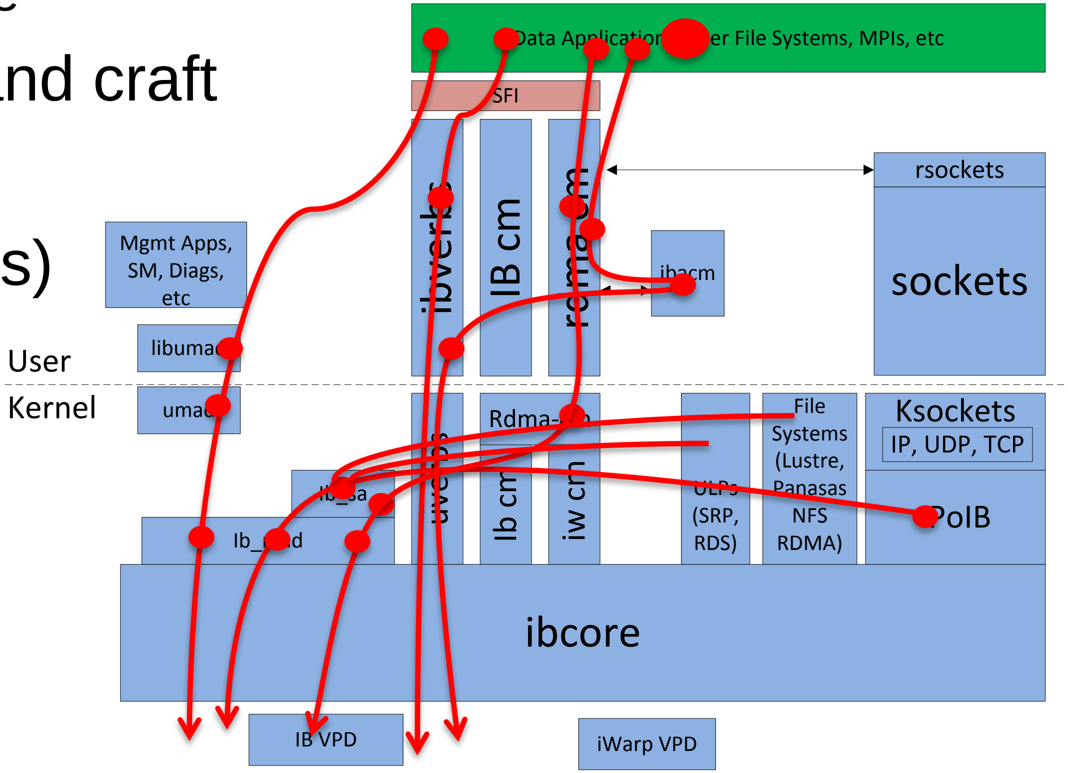
Key Mgmt Scalability Bottlenecks

- PathRecord Query
- IPoIB ARP

PathRecord Query Today

- User apps
  - Use rdma CM
    - ibacm optional cache
  - Use libumad and hand craft
  - Use UD QPs
  - Hand build PR (MPIs)
  - Other (via IPoIB …)
- Kernel ULPs
  - Call ib_sa

No single place to put PR optimizations



PathRecord Query Scalability

- Need to 1st standardize a user space API
  - Libfabrics (OFI WG) and RDMA CM are logical choices
- Use API in all ULPs, benchmarks, demos, tools, diagnostics, etc.
  - Both kernel and user space
  - So everyone benefits from scalability improvements
- Decouple API from IPoIB
  - Multi rail clusters may not want IPoIB on all rails

PathRecord Query

- Need a plugin architecture behind the API
- Need a variety of plugins
  - Small clusters can do direct PathRecord query
  - Modest clusters can do PathRecord caching
  - Large clusters need PathRecord replicas or ibssa
  - Huge clusters need algorithmic approaches
    - Topology dependent optimizations
  - Permit research and experimentation
- Start with direct, ibssa and cached plugin

One size does not fit everyone

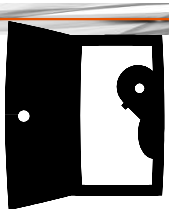
IPoIB ARP Scalability

- Need a multi-tiered approach in IPoIB
  - Modest clusters can do standard ARP/broadcast
    - Perhaps with long ARP timeouts (hours, days)
  - Large clusters need pre-loaded ARP tables
  - Huge clusters need algorithmic approaches
    - Topology dependent
- Need to 1st standardize a plug-in API
- API needs to tie into PathRecord Plug-In
- Implement std ARP and pre-loaded plugins 1st

Other Mgmt Issues

- Umad security
- Partitioning
- Multicast
- Notices
- SA interaction pacing

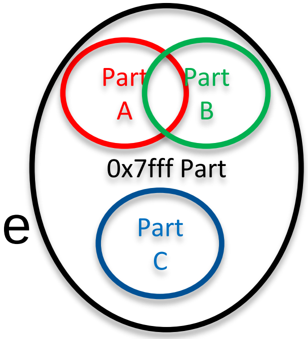
Mgmt Security



- Umad security issues
  - Requires root access by default
  - Use of umad by applications forces opening security
  - Umad is too easy a vehicle to attack server or cluster
- First steps
  - Rapidly move applications away from using umad
  - Simplify API, remove apps hand building packets
    - Multicast membership, Notices, etc
  - Remove need for SM and diagnostics to be root
    - Need ability for secured umad use

Partitioning

- Proper Operation will be necessary for HPC Cloud
- Don’t assume full membership in default partition
  - Carefully reading of IBTA 1.2.1 reveals:
  - Default partition is just a power on default, not a guarantee
  - If it was a guarantee, IBTA partitioning would be useless
    - everyone could use 0xffff to talk to anyone
  - Only guarantee is membership in 0x7fff to permit SA query
- Fix P_Key assumptions in SA queries, ibacm, tools, etc
  - Proper use of PathRecord query will solve most of this
  - Search local P_Key table to decide if 0x7fff or 0xffff present
- IPoIB react to P_Key table changes during Port Initialize
  - especially entry 0
- PKey indexes can change between boot and port Active



Multicast

- Multicast in IBTA
  - Each node can join/leave a group only once
  - Multicast join/leave are for whole node
- Multicast use goes beyond just IPoIB
  - ibacm, MPI collectives, kernel bypass for FSI
  - RDMA CM has some APIs, needs to coordinate w/kernel
- Need API w/kernel muxing of multicast membership
  - IBTA compliant node level interactions with SM/SA
  - Allow multiple processes, kernel and user to join a group
  - Automated cleanup when processes die
  - Also removes another need for umad access by apps

Notices

- Use of Notices by applications is scalability issue
  - Can force O(N) messages from SM on each event
  - Example: turn off 100 nodes in 10K fabric -> 1M notices
  - Example: turn off 50K nodes in 100K fabric -> 2.5B notices
- At host need Notice muxing
  - Each node register/receive/deregister only once
  - Need kernel muxing of notice registration
  - Need kernel muxing of notice delivery/ack
  - Need cleanup when processes die
  - Also removes another need to umad access by applications
- Should we restrict or disable use of notices?

SA Interaction Scalability

- Centralization of PR, Multicast and Notices is 1st step
- This then permits tuning of SA interactions based on scale

- SA Response Timeout/Retry Handling
  - Clients today use fixed timeouts
  - Timeouts chosen a priori without knowledge of SA nor fabric load
- Need centralized config of timeouts and retry settings
  - As opposed to per application constants
- Retries should perform non-linear backoff

- SA Busy Response Handling
  - Present OFA code does immediate retry
  - Prevents SA from using BUSY to pace its workload
  - SA forced to discard
- BUSY should cause client backoff before attempting retry
  - Non-linear backoff also recommended

Next Steps

- Lets all collaborate to solve these challenges
- Your participation in discussion is encouraged

Lets be committed to solving these long standing issues

Summary

- Cluster sizes will grow year over year
- OFA has some long standing scalability issues
- Solutions are possible
- Lets all commit to making it happen

Thank You

OPENFABRICS ALLIANCE

10TH ANNUAL INTERNATIONAL DEVELOPER WORKSHOP