Scalable Communication Endpoints for MPI+Threads Applications

Rohit Zambre, * Aparna Chandramowlishwaran, * Pavan Balaji *
*University of California, Irvine | ^Argonne National Laboratory

---

**Introduction**

- **MPI everywhere not scalable on modern systems**
  - Disproportionate increase in number of cores compared to other on-node resources
  - Dwinding share of resources per process
- **MPI+Threads model addresses scalability issue**

**Why this tradeoff?**

- **Endpoint configuration in state-of-the-art MPI libraries:**
  - Why this tradeoff?

**Background**

- OFED stack; QP-depth: 64; Postlist: 32; Unsignaled Completions: 64
- To study effect of feature
- 2 nodes with Intel Haswell (16 cores per socket) @ 2.5 GHz + Mellanox
- Naive solution impacts memory and hardware resource usage

**Resource Sharing Analysis**

- Analytically, four levels of sharing
  - Buffer sharing
  - Context sharing
  - Protection Domain Memory Region sharing
  - Completion Queue sharing

**Scalable Endpoints**

- Based on analysis above, we define six categories of endpoints for N threads:
- Evaluation using 16 threads
- Global array kernel
- DGEMM
-Stencil kernel
- 5-point stencil with 1-D partitioning

**Evaluation Setup**

- 2 nodes with Intel Haswell (16 cores per socket) @ 2.5 GHz + Mellanox
- ConnectX-4 adapter on each node
- To study effect of feature f on multifielded RDMA write message rate: "All w/o P"
- OFED stack; QP-depth: 64; Postlist: 32; Unsignaled Completions: 64

---

*Send a message*

- (1)(i) Write a message descriptor (MD)
- (1)(ii) CPU MMIO-writes to NIC
- (2) NIC DMA-reads MD
- (3) NIC DMA-reads payload
- (4) NIC DMA-writes completion after receiving ACK from target

*Features that help small messages*

- Postlist: Reduces (1)(ii)
- Unsignaled Completions: Reduces (4)
- Inlining: Removes (3)
- Programmed I/O: Removes (2)

*Sending a message*

- Write MD
- Read data
- Write completion
- Acknowledge

---

*Scalable Communication Endpoints*

- A resource sharing model that concretely categorizes the tradeoff space ranging from fully independent paths to fully shared paths

*Naive solution for MPI+Threads: emulate MPI everywhere endpoints*

- Leads to 93.75% wastage of limited hardware resources
- Need a second NIC after using only 6.25% of the resources on the first
- MPI+Threads allows for arbitrary level of sharing: what level of sharing is ideal?
- Depends on performance requirements and availability of resources
- A tradeoff space between performance and sharing resources exists
- Scalable Communication Endpoints

*Why this tradeoff?*

- **Endpoint configuration in state-of-the-art MPI libraries:**
  - Why this tradeoff?

---

*Background*

- Dwindling share of resources per process
- Disproportionate increase in number of cores
- MPI everywhere not scalable on modern systems

---

**Communication Resources**

- Transmit Queue: Queue Pair (QP) in Verbs (consumes memory)
- Completion Queue: Completion Queue (CQ) in Verbs (consumes memory)
- Hardware resource: micro User Access Region (uUAR) within UAR pages on Mellanox InfiniBand (consumes hardware resources)
- Naive solution impacts memory and hardware resource usage

*Memory: Creating 16 naive endpoints will occupy 5.15 MB*

*Hardware resources: much smaller limit than that of memory in general*

*Max of 16K uUARs on ConnectX-4 (1021 naive endpoints); max of 160 HW contexts on Omni-Path*

---

*Scalable Endpoints*

- Based on analysis above, we define six categories of endpoints for N threads:

---

*Evaluation using 16 threads*

- Global array kernel
- DGEMM
- Stencil kernel

*Performance decreases with increasing resource efficiency*

---

*Evaluation Setup*

- 2 nodes with Intel Haswell (16 cores per socket) @ 2.5 GHz + Mellanox
- ConnectX-4 adapter on each node
- To study effect of feature f on multifielded RDMA write message rate: "All w/o P"
- OFED stack; QP-depth: 64; Postlist: 32; Unsignaled Completions: 64