Screencast: Tuning the Openib BTL (v1.2 series)

Jeff Squyres
May 2008
openib BTL Parameters

```ompi_info --param btl openib```

- Shows all openib BTL MCA parameters
  - …there are a lot!
- Also try:
  ```ompi_info --param btl openib \ --parsable```

- What do they all mean?
openib BTL Parameter Prefix

- All parameter names are prefixed
  - Guarantees uniqueness between components
    - “btl_openib_”
- Prefix will not be shown here for brevity
  - “foo” → “btl_openib_foo”
Simple Parameters

- **max_btls**: integer
  - -1 (use all, default) or >0
  - Max number of IB ports to use (start: port 0)
- **mtu**: integer (default per hardware)
  - 1=256 bytes, 2=512 bytes, 3=1024 bytes, 4=2048 bytes, 5=4096 bytes
- **ib_service_level**: integer (default 0)
  - Direct mapping to virtual lane
Receive Queues

Per-peer receive queues
- For peer 1:
- For peer 2:
- For peer 3:
- For peer 4:

Shared receive queue

Less than N x M buffers
Receive Parameters (v1.2.x)

- **rd_num**: integer
  - Number per-peer receive buffers

- **use_srq**: 0 or 1
  - **srq_rd_max**: integer
    - Max number of posted receives in the SRQ
    - Set absolute limits
  - **srq_rd_max_per_peer**: integer
    - Max number of posted receives per peer
    - Uses “stats game” -- log2(num_MPI_procs)
  - **srq_sd_max**: integer
    - Max number of posted sends to peer SRQ
Short Eager RDMA Params

Num peers: 2
Num buffers: 12
Threshold: 5
Short Eager RDMA Params

• `use_eager_rdma`: 0 or 1
• `eager_rdma_threshold`: integer
  ▪ Number of receives before setup eager RDMA
• `max_eager_rdma`: integer
  ▪ Max number of peers to use eager RDMA
• `eager_rdma_num`: integer
  ▪ Number of posted receive buffers per peer
Long Message Protocol

1 2 3

RDMA fragments
Send / receive fragments
Match fragment
Long Message Parameters

- `eager_limit`: integer
  - Max size of “eager” (short) messages
- `max_send_size`: integer
  - Max size of “prime the pipeline” fragments
- `min_rdma_size`: integer
  - Offset where to start RDMA
- `max_rdma_size`: integer
  - Max size of long message RDMA fragments
v1.2 Long Message Params

- **eager_limit**: Fragments of size up to `max_send_size`
- **min_rdma_size**: Fragments of size up to `max_rdma_size`

Categories:
- 1
- 2
- 3
Disabling “Eager” Completion

- pml_ob1_use_early_completion
  - “Early completion” latency optimization
  - Enabled (set to 1) by default

- Behavior can be disabled by setting this MCA parameter to 0
  - Can cause problems (hangs) in some applications that do not enter the MPI library frequently
Timeout Parameters

- All are directly given to verbs API
- `btl_openib_ib_min_rnr_timer`: 0-31
  - Receiver not ready timer (seconds)
- `btl_openib_ib_timeout`: 0-31
  - InfiniBand transmit timeout, plugged into:
    \[ 4.096\mu s \times 2^{\text{btl_openib_ib_timeout}} \]
- `btl_openib_ib_retry_count`: 0-7
- `btl_openib_ib_rnr_retry`: 0-7
Freelist Parameters

- “Freelists” maintained of registered memory buffers
  - Indexed by count of buffers (not size)
- `free_list_max`: integer
  - Max number of buffers in freelist (-1 = infinite)
- `free_list_num`: integer
  - Initial number of buffers
- `free_list_inc`: integer
  - Number of buffers to add when empty
Memory Pool Parameter

- mpool__rdma__cache__size__limit: integer
  - In “rdma” mpool component; not openib BTL
  - Memory pool
  - Max limit on user-registered memory

- Used in conjunction with openib BTL parameters, can establish a maximum limit of all registered memory
Registered Memory Footprint

• Still quite complicated!
  ▪ Sum of combinations of many MCA parameters
  ▪ FAQ web page gives good description

• Total registered memory can be limited
  ▪ May need to use an Excel spreadsheet…
MPI Layer Parameters

• `mpi_leave_pinned`: 0 (default) or 1
  ▪ Leave user buffers registered ("pinned")
  ▪ **Extremely important for benchmarks that re-use buffers!**

• `mpi_paffinity_alone`: 0 or 1
  ▪ Must be manually set
  ▪ Assume MPI job is "alone" on the node
  ▪ Pin MPI processes \( \rightarrow \) processors starting with 0

• `mpi_yield_when_idle`: 0 or 1
  ▪ When busy-polling, call `yield()`
Sidenote: Portable Linux Processor Affinity (PLPA)

- Sub-project of Open MPI
- Small library to do processor affinity
  - Pin process A to processor X
  - API for processor affinity has changed 3 times
  - Depends on glibc, kernel, and distro versions
- PLPA provides stable API
- New version can map (socket, core) tuples to Linux virtual processor ID
  - `plpa_taskset(1)` command
More Information

• Open MPI FAQ
  ▪ General tuning
    http://www.open-mpi.org/faq/?category=tuning
  ▪ InfiniBand / OpenFabrics tuning
    http://www.open-mpi.org/faq/?category=openfabrics