System-level Scalable Checkpoint-Restart for Petascale Computing

Jiajun Cao [1] Kapil Arya [1] Rohan Garg [1] Shawn Matott [3]
Dhabaleswar K. Panda [2] Hari Subramoni [2] Jérôme Vienne [4]
Gene Cooperman [1]

[1]Northeastern University

[2]The Ohio State University

[3]State University of New York at Buffalo

[4]Texas Advanced Computing Center

December 15, 2016

Overview

Background: Checkpointing in HPC

- Application-level checkpointing
  - Programmer decides when to checkpoint and what data to save
  - Application needs to be rewritten
- Virtual machine snapshotting
  - Virtualization of HPC hardware can be challenging
- System-level checkpointing
  - CRIU (Checkpoint/Restore In Userspace)
    - Lack of support for distributed applications
  - BLCR (Berkeley Lab Checkpoint/Restart)
    - Communication needs to be shut down and restarted at MPI level
    - System V shared memory is not supported
  - DMTCP (Distributed MultiThreaded Checkpointing)
    - No modification to the OS or user program
    - Supports distributed applications
    - Plugin architecture, easy to extend

Background: Hardware/Software

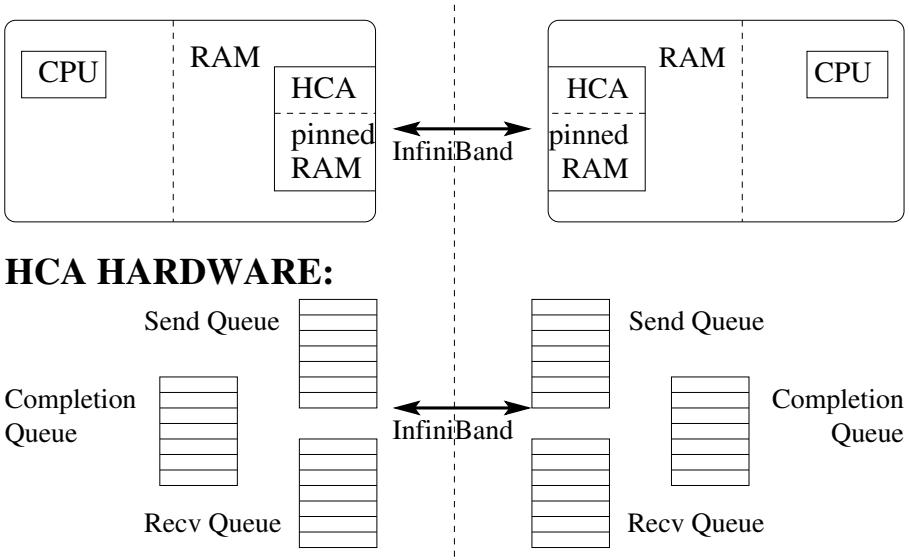

Figure: InfiniBand Concepts

InfiniBand:

- Queue pair-based communication model
- Supports two modes: RC (reliable connection) and UD (unreliable datagram)
- UD is used to bootstrap endpoint connections

Contributions

Performance concern:

- Runtime overhead
- Time for checkpoint

Scalable checkpoint demenstrated: 128 processes *vs.* 32752 processes

Contributions:

- First checkpoint support for a hybrid InfiniBand communication mode that uses both reliable connection (RC) and unreliable datagram (UD).
- Lower the runtime overhead for checkpointing large-scale applications to under 1%.

Implementation: Checkpoint support for UD

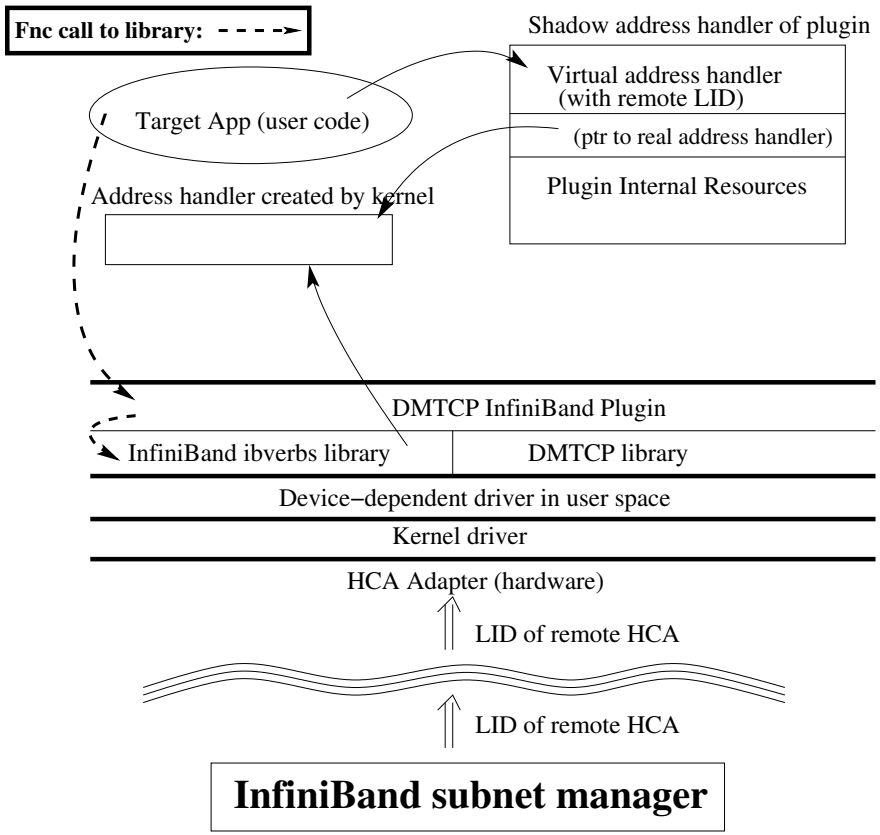

Figure: Virtualization of address handler and LID (local id) of remote HCA (hardware channel adapter).

- LID: part of the AH (Address Handler) used by UD, assigned by hardware
- Wrappers around functions of the InfiniBand library: whenever user code accesses AH, return the virtualized AH
- Build the virtual-to-real global mappings on restart
- When user code accesses AH, translate it into real AH before calling the InfiniBand library

Implementation: Reducing Runtime overhead

Past:

- No way to peek into the current state in the hardware
- Needs to trace send/receive requests
- Runtime overhead: 1.7% at 2K cores, 9% at 4K cores

Current solution:

- Poll the completion queue for a small time window during checkpointing
- If no messages arrive during the window, then no more messages in flight.
- Otherwise, poll during another window

Experimental Results

Num. of processes	Checkpoint time (s)	Restart time (s)	Total ckpt size (TB)	Write (ckpt) bandwidth (GB/s)
8192	136.1	215.3	9.4	69
16368	367.4	706.6	19	52
24000 ([1])	634.8	1183.8	29	46
32752 ([2])	652.8	2539.1	38	60

NOTE: Executed with special permission[1]; and during Stampede maintenance[2] (mostly exclusive access to the cluster).

Table: Checkpoint and restart trends for HPCG (linear algebra); checkpoint image size for each process is 1.2 GB, with 16 images generated on each compute node.

Time to checkpoint NAMD (molecular dynamics) with 16368 processes is 158 s, with a total checkpoint size of 9.8 TB

Conclusion and Future Work

- Driver provides interfaces to peek into the status of the hardware
- Reduce the checkpoint overhead
- New fabrics, e.g., Intel OmniPath, libfabric

Thanks

Questions?

http://dmtcp.sourceforge.net

jiajun@ccs.neu.edu