Opportunities and Challenges of Low-Cost HPC in a Bare-Metal Marketplace

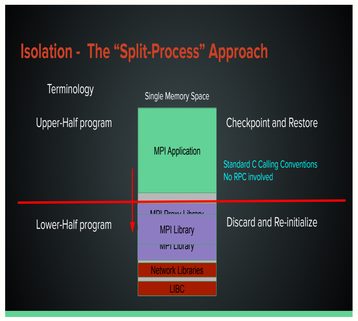
HPC Lab (team leader: Gene Cooperman)
College of Computer and Information Science / Northeastern University

Northeastern University
College *of* Computer and Information Science

Motivation

- Different clusters are co-located in the marketplace with time-varying demands, and opportunities for hardware multiplexing. Leases and reservations need to be preemptible. DMTCP enables low-overhead, low-cost ability to efficiently use preemptible hardware resources in the marketplace.
- DMTCP (Distributed MultiThreaded CheckPointing, `http://dmtcp.sourceforge.net`) is a widely used package for transparent checkpoint-restart. It is sidely used in HPC (e.g., MPI), GPUs, high-performance networks; and applications such as cyber-security, EDA, science, and engineering.

Split Processes with MPI for MANA: Isolation and Flexibility



DMTCP checkpointing: Mature, Real-World Use

- NERSC (National Energy Research Scientific Computing Center): user support for Cray-based Supercomputers
- TACC (Texas Advanced Computing Center): Demonstrated petascale transparent checkpointing
  Checkpointing HPCG: 32,752 CPU cores), and (NAMD: 16,368 CPU cores) for checkpointing MPI applications over InfiniBand at TACC:
  "System-level Scalable Checkpoint-Restart for Petascale Computing", Jiajun Cao et al., *Int. Conf. on Parallel and Dist. Sys.* (ICPADS'16), 2016

Supported by:

(FogRein: Inria Associate Team)

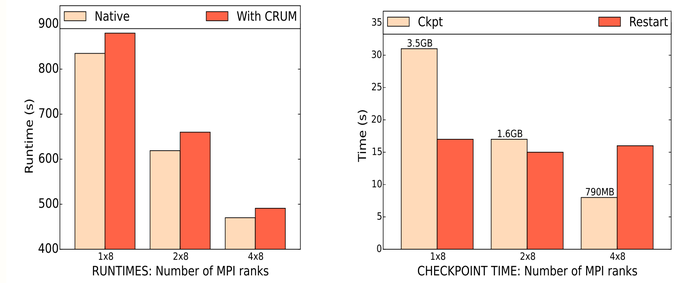
*** CHALLENGES ***

1. **Application transparency:**
   Extending to CUDA, MPI, Edge Computing
   Transparency Issues: Specialized hardware and non-reentrant libraries: many newer RDMA interconnects, GPUs (NVIDIA and AMD), CPU accelerators (e.g., FPGA), proprietary drivers
2. **Low Overhead:**
   DMTCP is the leading current solution for CUDA, MPI
   HPC Issues: Jitter, Stragglers, low-overhead support for newer networks

Performance Results for HPC

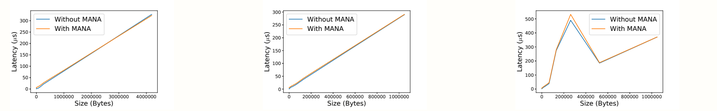
Unique capabilities:
Support for NVIDIA CUDA GPUs; most implementations of MPI

Transparent Checkpointing of CUDA:



(Internal version 2 of this work: less than 1% runtime overhead)

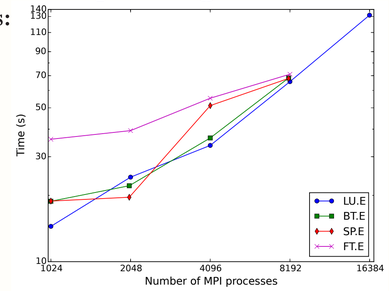
MANA for MPI: $< 1\%$ runtime overhead



(a) Point-to-Point Latency (b) Collective MPI_Gather (c) Collective MPI_Allreduce

Scalable MPI: Checkpointing at Petascale

NAS benchmarks:



Real-world: HPCG (32,752 proc's, 652 s); NAMD (16,368 proc's, 157 s)

*** OPPORTUNITES ***

1. **Load balancing:**
   Dynamically changing capacity; must adapt: A pool of resources exist; Can we remove the problem of fixed silos? Use Marketplace strategies by predicting future capacity changes over time.

2. **Serverless computing:**
   Bootstrapping container-based applications: Fast deployment (checkpoint once, fast restart many times); Easy integration: No host privileges required due to transparency of checkpointing.

3. **Long-running jobs:**
   3.1 **Piece-wise execution of very long jobs:**
   Intermittent availability of resources; No single reservation is sufficient to run the entire job; So, we split a long-running job into multiple shorter jobs.
   3.2 **Fault tolerance:**
   Large jobs crash; Many GPU studies show that the GPU hardware has less reliability than server-class machines. SSDs also have limited reliability — especially due to SSDs "wearing out":
   - "A Large-Scale Study of Flash Memory Failures in the Field", Meza et al., Sigmetrics'2015
   - "A Large Scale Study of Data Center Network Reliability", Meza et al., Internet Measurement Conference 2018. 2018.
   - "Characterizing Disk Failures with Quantified Disk Degradation Signatures: An Early Experience", Huang et al., IEEE Int. Symp. on Workload Characterization, 215

4. **Flexible deadlines (e.g., preemptible and spot instances):**
   GOAL: saving money by running each piece-wise job during cheaper times of day, etc.; Use Marketplace strategies

5. **Scheduling:**
   Can we consider priorities, special classes of jobs?

Future Work

- Split process: checkpoint only the application; not the libraries (e.g., CUDA, MPI, network) and drivers for the underlying hardware
- Checkpoint application only; restart on new hardware/driver for lower layer
- Debugging long-running apps: distributed and parallel applications: Checkpoint the native application, and restart under a model checker (joint work with Inria)
- Checkpointing to Guide Fuzzing in Security (joint work with Raytheon)

Acknowledgments

- A series of National Science grants in support of this effort:
  NSF Award OCI 09-60978, NSF Award OCI 12-29059, NSF Award ACI-1440788, NSF Award OAC-1740218
- A series of 6 grants from Intel Corporation since 2013
- A grant from Raytheon Corporation; and
- National Energy Research Scientific Computing Center (NERSC): Department of Energy.