System-level Transparent Checkpointing for OpenSHMEM

Rohan Garg[1] Gene Cooperman[1] Jerome Vienne[2]

[1]Northeastern University
Boston, MA

[2]Texas Advanced Computing Center
The University of Texas at Austin
Austin, TX

August 2nd, 2016

Extended Collaborative Support Service (ECSS)

ECSS experts, many with advanced degrees in domain areas, are available for collaborations lasting months to a year to help researchers fundamentally advance their use of XSEDE resources.
Expertise is available over a wide range of areas:

- Performance Analysis,
- Petascale Optimization,
- Efficient use of Accelerators,
- I/O Optimization,
- Data Analytics
- Visualization
- use of XSEDE by science gateways

Table of contents

What is Checkpointing ?

Definition

Checkpoint-Restart is the ability to save a set of running processes to a checkpoint image on disk, and to later restart it from disk.

Checkpoint-Restart involves saving and restoring

- all of user-space memory
- state of all threads
- kernel state
- network state
- etc.

Why do we need to use checkpointing?

- Fault tolerance
- Scheduling and process migration
- Faster startup times
- Save/restore workspace (for interactive sessions)
- Managing tails (slower thread tasks) for multi-threaded applications
- Debugging
- Speculative execution (what-if scenarios)
- etc.

DMTCP: Distributed MultiThreaded CheckPointing

- Open source system-level checkpointing
- Transparent to the user
  - Works without modifying the source code or binary
- User-space
  - No kernel modules
- Handles distributed applications
  - Centralized coordinator
- Supports multiple programming language
  - C/C++, Java, Haskell, Lisp, Python, Perl, Matlab, R etc.
- Handles MPI libraries, resource managers, process managers, etc.
  - Open MPI, MVAPICH2, Intel MPI, ...

Available at: `http://dmtcp.sourceforge.net`

Checkpoint-Restart with OpenSHMEM

Ali et al.(2011)

Proposed an application-specific fault tolerance mechanism

Hao et al.

- Presented at OpenSHMEM Workshop 2015
- More generic approach based on User Level Fault Tolerance (ULFM)
- Use shadow memory in which the shared memory regions of peers are backed up by peers.
- The user code is responsible for invoking a chechpoint and for restoring correct operation during a restart
- Copy the shared memory region along with privately mapped memory to a peer process during runtime $\Rightarrow$ This places added pressure on the network fabric and on the RAM.

Design modification of DMTCP to Support OpenSHMEM

The design of DMTCP had to be extended in few areas in order to support both checkpointing of modern MPI implementations and checkpointing of OpenSHEM:

- Unix domain sockets (Earlier MPI implementations generally did not use UNIX domain sockets).
- SysV shared Memory objects: Only BSD-style shared memory regions (using mmap and "MAP_SHARED") was initially supported.
    - OpenSHMEM requires support for large shared memory regions created by the user's application.
    - Absence of virtual memory. (Alternative strategy was created).

Experimental Setup 1/3

Stampede Cluster

- Ranked #12 on latest TOP500
- CentOS 6.4
- 6,400 nodes
  - Linux Kernel 2.6.32-431-el6
  - 16-cores (dual sockets) Sandy Bridge Xeon E5-2680
  - At least 1 Xeon Phi (KNC)
  - 32 GB RAM
- InfiniBand FDR interconnect
- SLURM resource manager
- No swapfile
- Lustre Filesystem 2.5.5

Experimental Setup 2/3

Software used:

- Intel Compiler 13.0.2.146
- MVAPICH2-X 2.0b
- NAS Benchmarks (BT & SP) with OpenSHMEM support

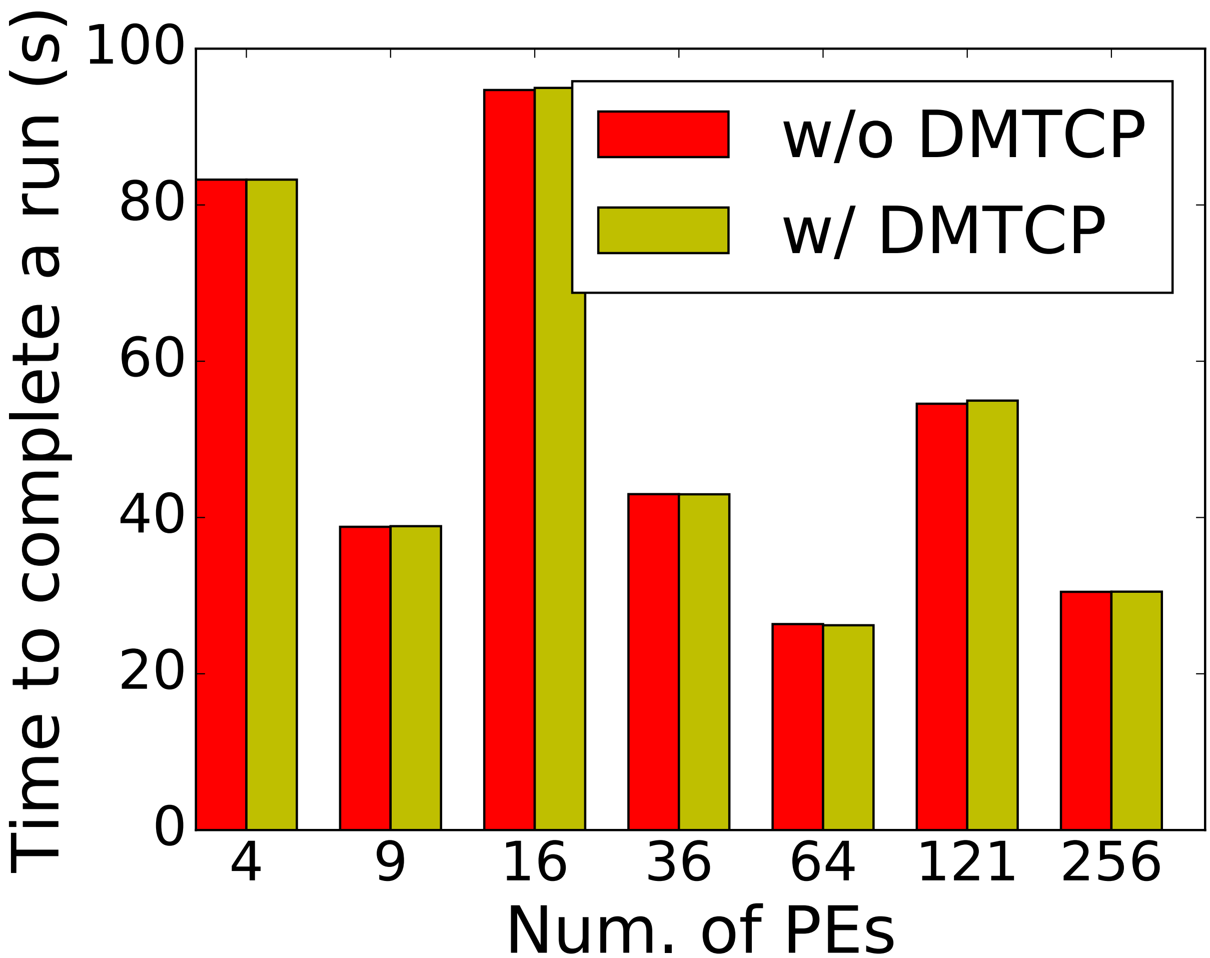
Experimental Setup 3/3

Distribution of Processes

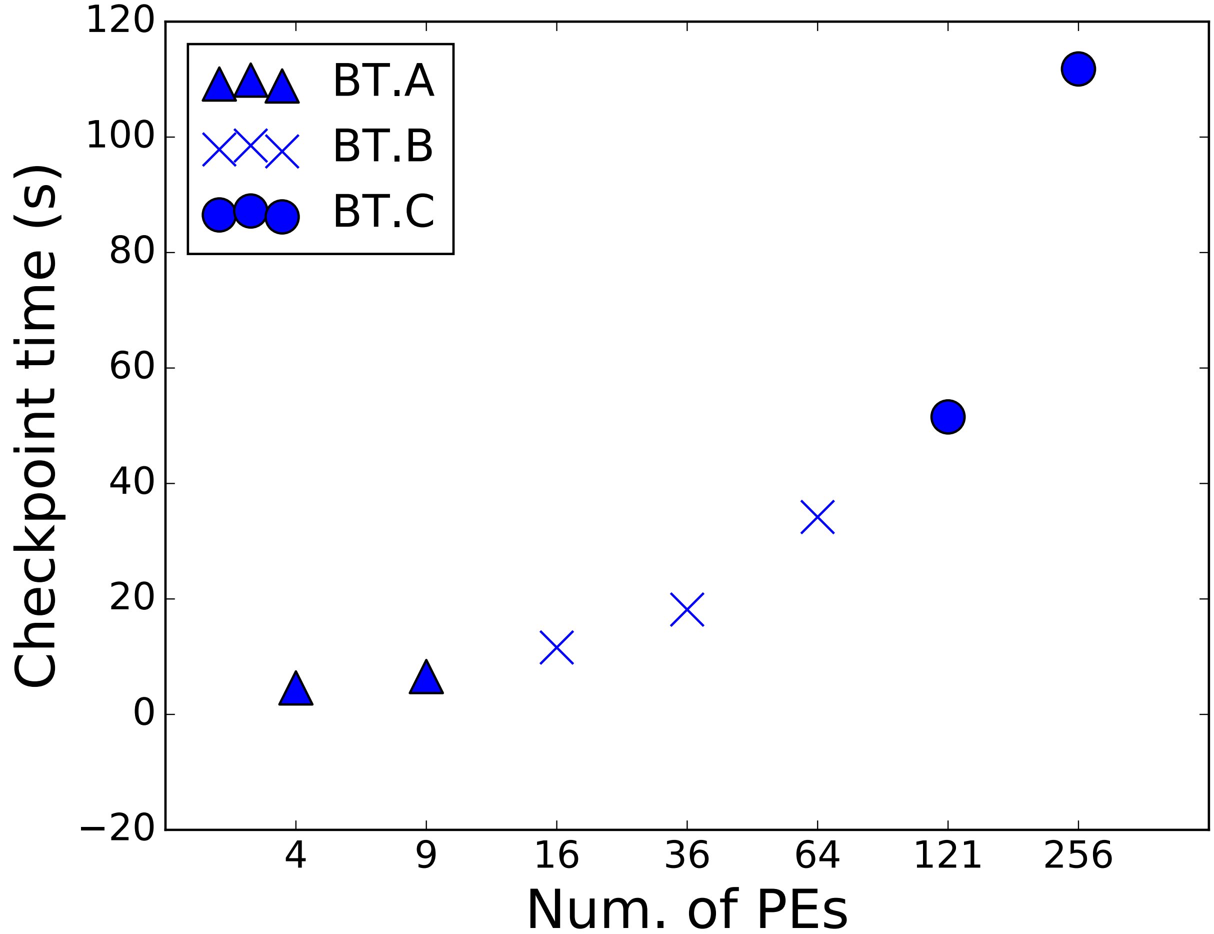
Num of PE's	Num of Nodes	Processes per node	NAS class used
4	2	2	A
9	3	3	A
16	4	4	B
36	6	6	B
64	8	8	B
121	11	11	C
256	16	16	C

For a given number of PE's, all the runs (with and without DMTCP) were conducted on the same set of nodes to reduce the variability due to network topology and traffic.

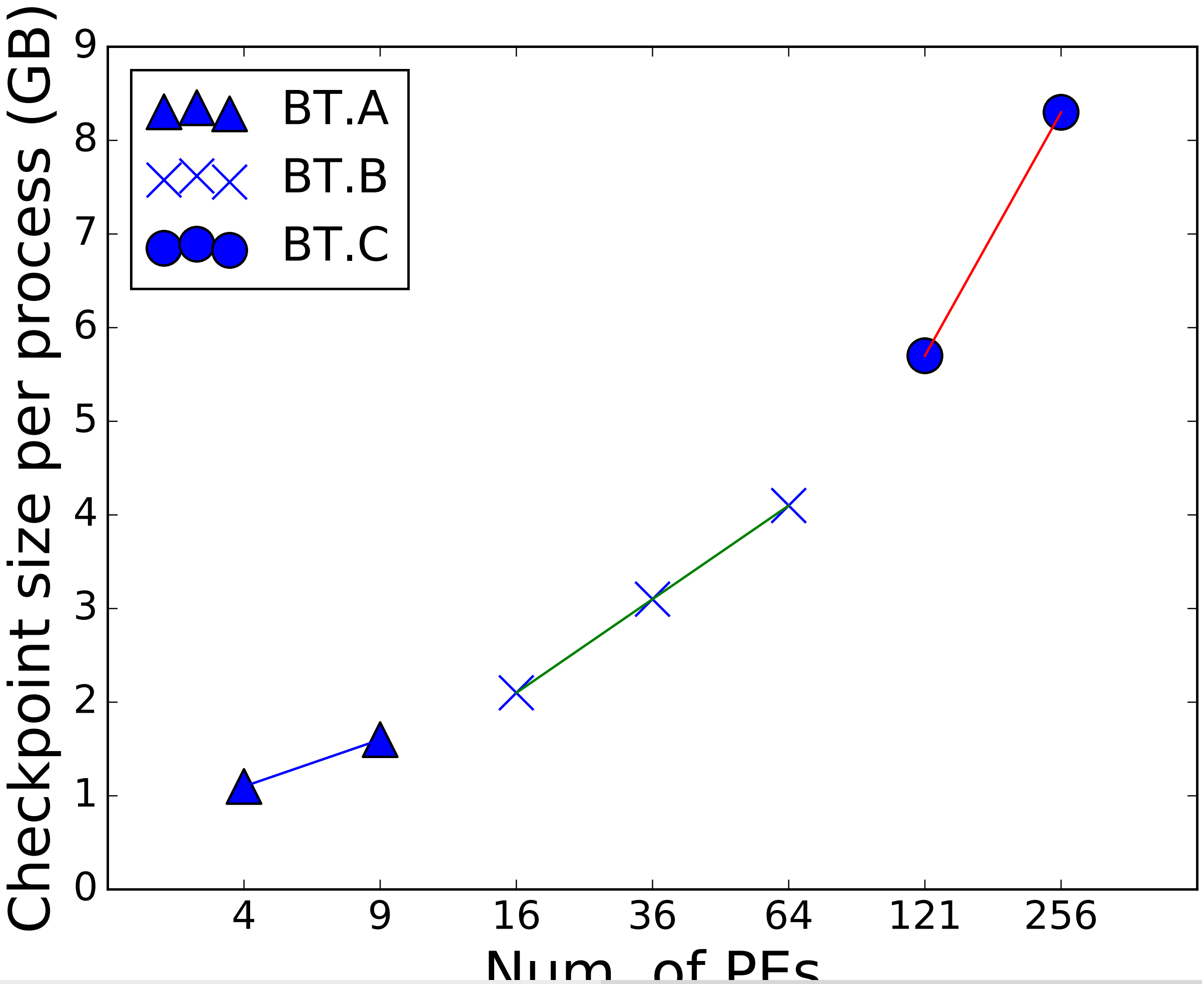
Runtime Overhead (NAS BT)



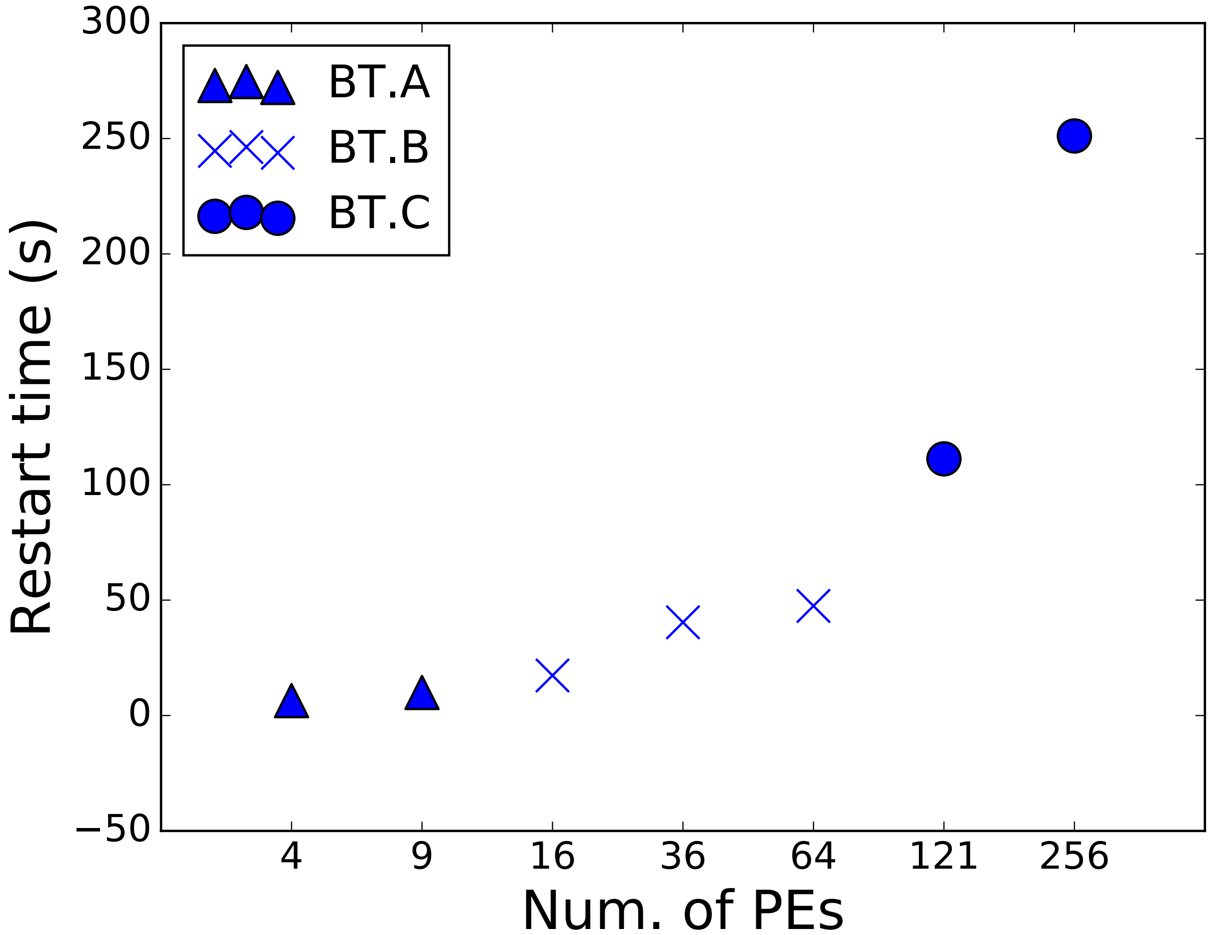
Checkpoint Times for NAS BT



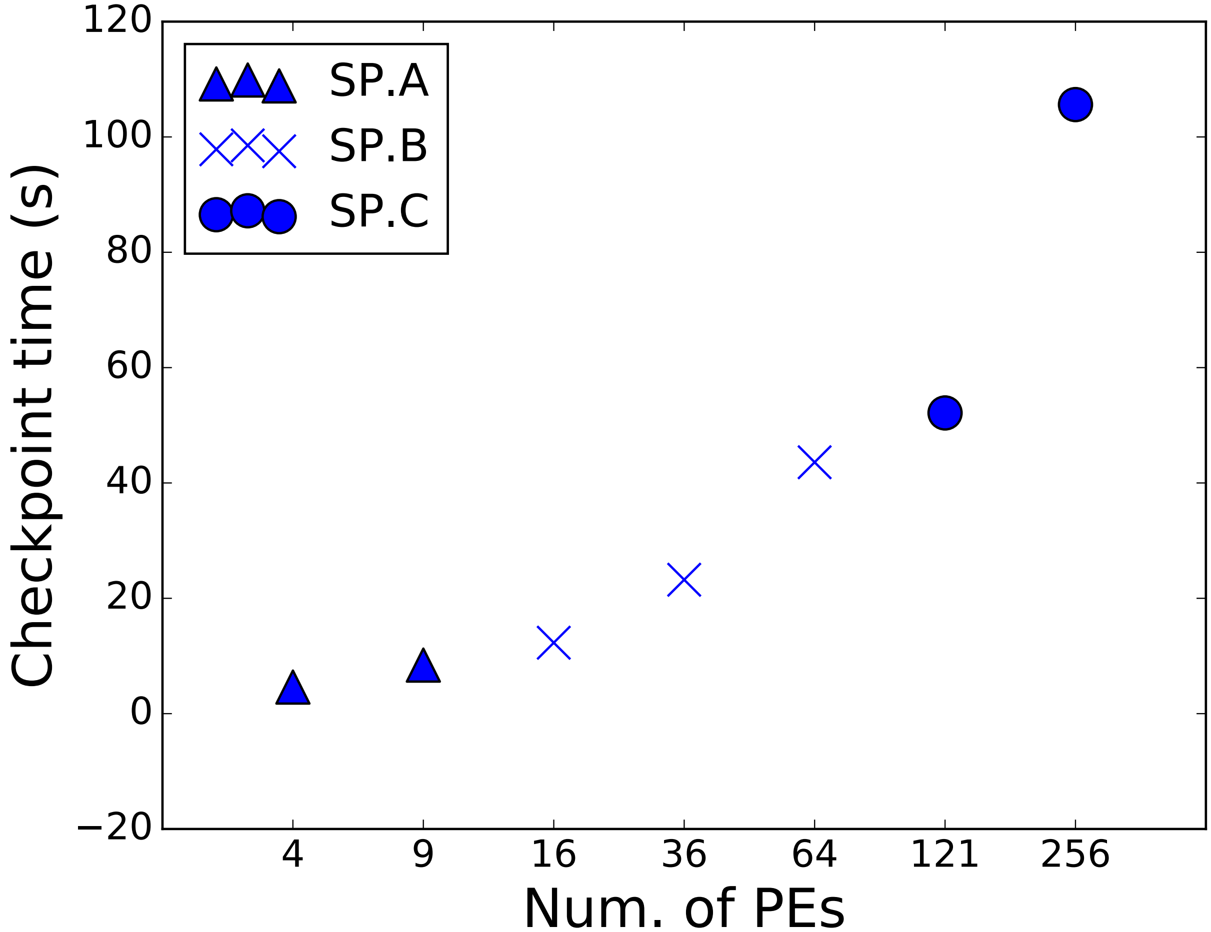
Uncompressed Image Sizes for NAS BT



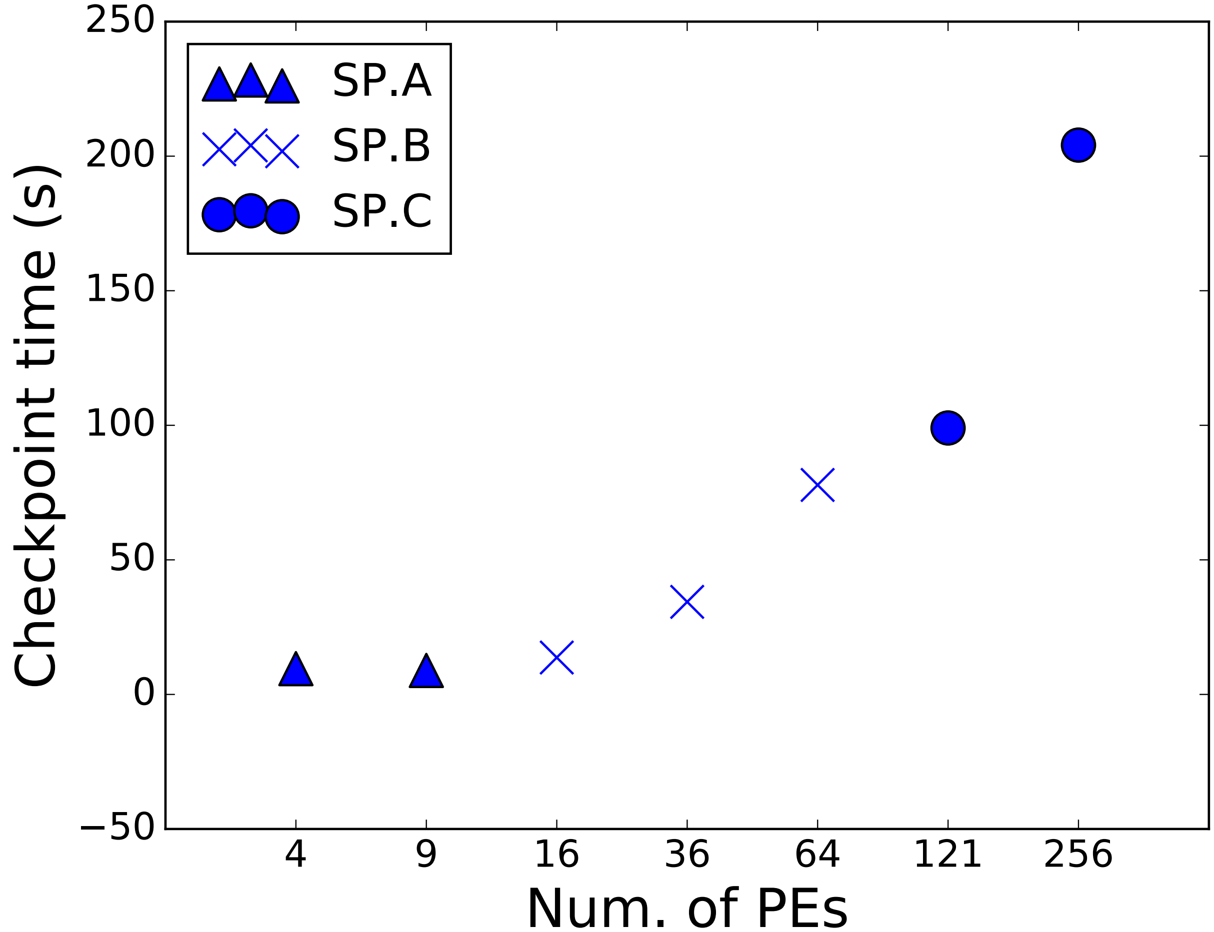
Restart times for NAS BT



Checkpoint Times for NAS SP



Restart times for NAS SP



General comments

- At the largest scale, 256 processes, the total data written to the disk is 2.2 TB, with an effective bandwidth of 20 GB per second.
- In all the cases, the checkpoint times are dominated by the time to write the checkpoint data to stable storage, and the cost for checkpointing the state of the application is negligible.
- We observe that the largest component in a checkpoint image is an OpenSHMEM shared-memory region (90-97% of the total image size in these experiments).

Conclusion and Future Work

- A system-level approach to checkpoint OpenSHMEM was presented
  - Capability of saving the state of an entire computation for restart
- Working on a leader election strategy
  - Only one copy of each shared memory region will be saved on a single node
  - Will reduce the time to write to back-end storage.

Acknowledgment

We would like to thank both Kapil Arya and Jiajun Cao for many useful discussions on the internals of DMTCP, and the design of those internal components. This work was partially supported by the National Science Foundation under Grant ACI-1440788. We also acknowledge the support of the Texas Advanced Computing Center (TACC) and the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1053575.