Progress Report on Transparent Checkpointing for Supercomputing

Jiajun Cao, Rohan Garg

College of Computer and Information Science, Northeastern University

{jiajun,rohgarg}@ccs.neu.edu

August 21, 2015

Overview

Checkpointing: current status

- **Application-specific** – Needs to modify user code; difficult to port to new applications
- **BLCR** – Single-process kernel-level solution; MPI implementations and resource managers need to add extra checkpoint-restart modules; requires tuning for different kernel versions
- **CRIU for Containers** – Linux kernel configured to expose certain of its internals in user-space; difficult to use with MPI
- **DMTCP** – No need to modify the OS or the user program; transparent support for distributed applications (TCP/InfiniBand)

The DMTCP Approach

- Direct support for distributed applications "out-of-the-box"
- Entirely in user-space, no root privileges
- Transparently inter-operates with:
  1. Major MPI implementations: (MVAPICH2, Open MPI, Intel MPI, MPICH2)
  2. Resource managers (e.g., SLURM, Torque, planned for LSF)
     - Resource Manager plugin (batch-queue)
  3. High-performance networks
     - InfiniBand plugin
  4. MPI process managers: (e.g., Hydra, PMI, `mpispawn`, `ibrun`)
  5. All recent versions of Linux kernel
- Extensible: supports application-specific plugins
  - A "cut-out" plugin — if memory is zero, don't save
  - Checkpoint only when specified by the application

DMTCP and the Software Engineering Burden

Lines of code: a proxy for the required software effort

- DMTCP
  - C/C++: 22,276 (+ 1657 for comments)
  - Headers: 6,693 (+ 2173 for comments)
- InfiniBand plugin
  - C/C++: 8,500 (+ 705 for comments)
  - Headers: 342 (+ 229 for comments)
- RM plugin
  - C/C++: 3,859 (+ 451 for comments)
  - Headers: 591 (+ 753 for comments)

* All measurements using `cloc`

Why are we doing this?

DMTCP-style transparent checkpointing is used with full memory dumps.

1990s (Beowulf cluster):
one disk per node; full memory dump is efficient

2000s (Back-end storage):
more RAM per compute node; many compute nodes writing to a small number of storage nodes, full memory dumps become less efficient

Future (Local SSD storage):
a few compute nodes writing to one SSD; full memory dump is efficient

Conclusion: DMTCP-style checkpointing on supercomputers will be practical in the future. The time to plan for this is now!

Experiments on MGHPCC: Runtime overhead

MGHPCC: Massachusetts Green High-Performance Computing Center, `http://www.mghpcc.org/`

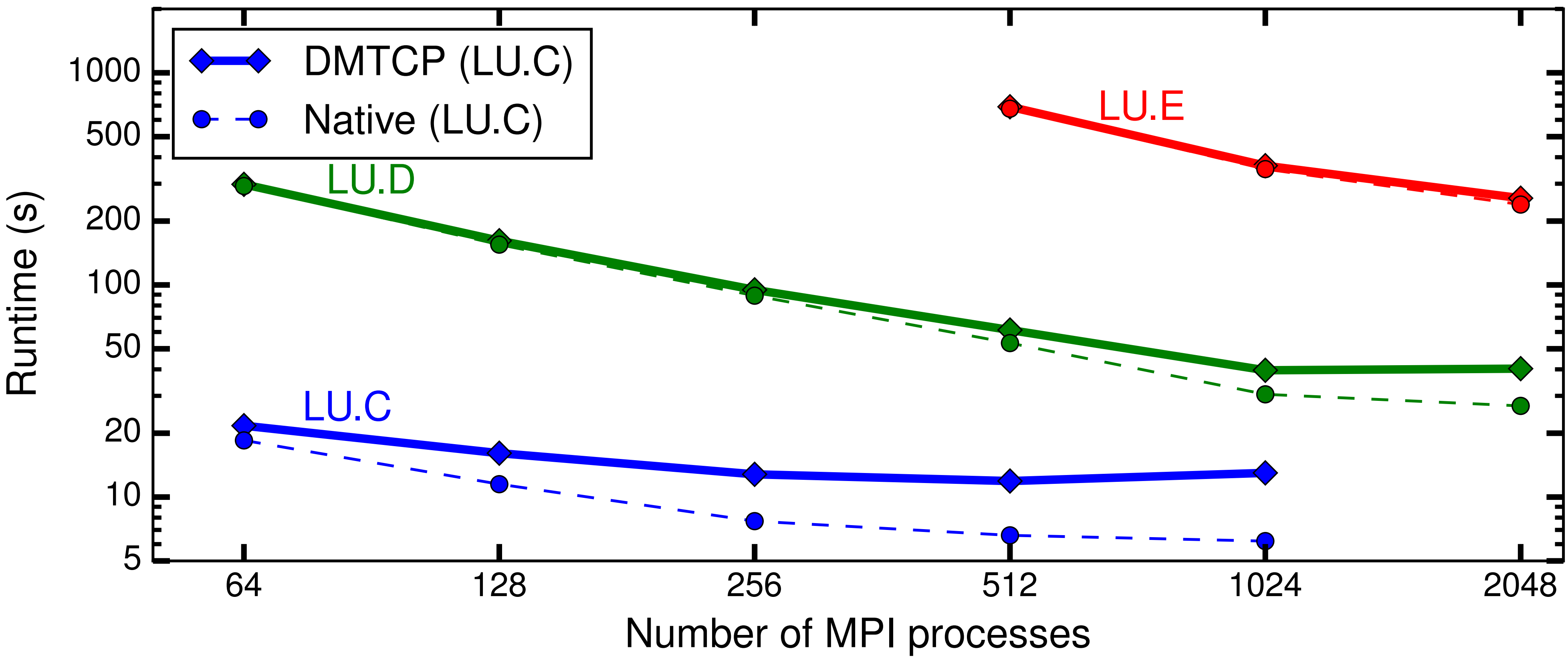


Figure: Comparison of runtime overhead (Open MPI); for jobs requiring at least 50 seconds the runtime overhead was small ($< 2\%$); for jobs requiring less than 50 seconds, the DMTCP startup time was a significant fraction of the total.

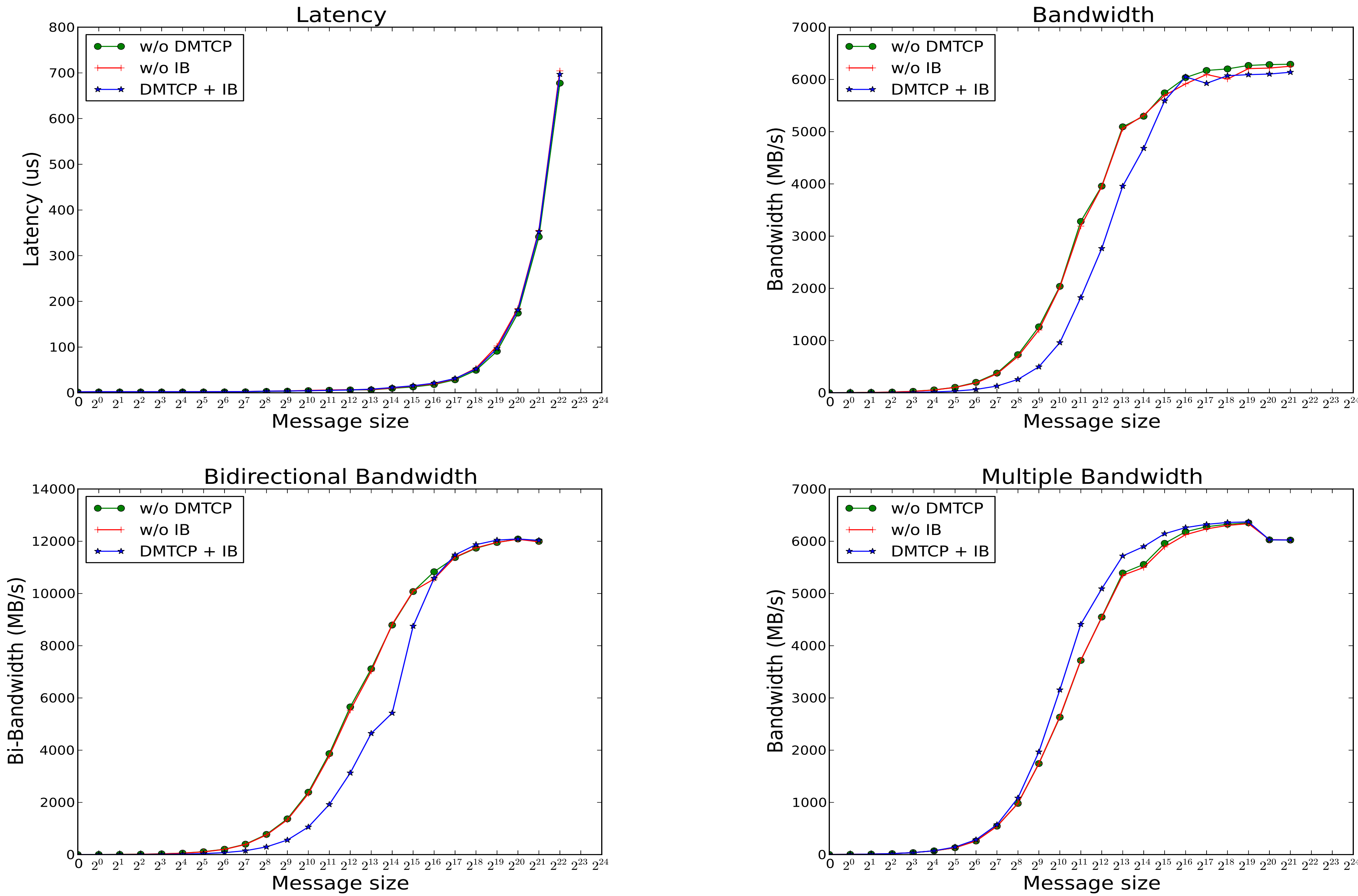
Experiments on MGHPCC: Checkpoint overhead

NAS benchmark	Number of processes	Ckpt time (s)	Ckpt size (MB)
LU.E	128×4	70.8	350
LU.E	64×8	136.6	356
LU.E	32×16	222.6	355
LU.E	128×16	70.2	117

Table: Checkpoint times and image sizes for the same NAS benchmark, under different configurations. The checkpoint image size is for a single MPI process. The checkpoint time is roughly proportional to the total size of the checkpoint images on a single node.

NOTE: These were preliminary results with Open MPI ver. 1.6 with a NFS shared filesystem.

Experiments at Stampede (TACC): Runtime overhead



Experiments at Stampede (TACC): Runtime overhead

NAS benchmark	Number of processes	Native (s)	with DMTCP (s)	Overhead (%)
LU.C	2	354.7	356.7	0.6
LU.C	4	179.9	180.5	0.3
LU.C	8	95.3	95.8	0.5
LU.D	16	1104.0	1104.9	0.0
LU.D	32	497.0	508.4	2.2
LU.D	64	293.5	296.5	1.0
LU.D	128	156.9	159.0	1.3
LU.E	256	1604.8	1635.2	1.9
LU.E	512	619.3	628.9	1.6
LU.E	1024	535.3	544.6	1.7
LU.E	2048	168.7 (162.0)	184.7 (165.4)	9.5 (2.0)
LU.E	4096	94.4 (86.7)	167.5 (91.3)	77.4 (5.0)

Table: Runtime overhead for the NAS LU benchmark. MVAPICH-1.9 with the MV2_ON_DEMAND_THRESHOLD environment variable was used upto 1024 cores. For 2048 cores and for 4096 cores MVAPICH-2.1 was used. The numbers in the parentheses are the actual computation times as reported by the benchmark.

Experiments at Stampede (TACC): Checkpoint overhead

Checkpoint times are approximately proportional to the image size. For a given problem class the image size decreases with the number of processes.

NAS benchmark	Number of processes	Ckpt time (s)	Ckpt size (MB)	Restart time (s)
LU.C	2	16.0	330	7.5
LU.C	4	8.6	174	4.3
LU.C	8	5.0	98	3.2
LU.D	16	32.0	650	12.9
LU.D	32	16.9	350	8.1
LU.D	64	9.9	190	6.5
LU.D	128	6.1	115	4.9
LU.E	256	33.8	700	18.9
LU.E	512	19.6	370	19.1
LU.E	1024	12.4	210	11.3
LU.E	2048	(in progress)	(in progress)	(in progress)
LU.E	4096	(in progress)	(in progress)	(in progress)

Table: Checkpoint times and image sizes for the NAS LU benchmark.

Micro-benchmarks: improving performance

- Performance bug in DMTCP found through micro-benchmarks (valloc)
- Performance bug in the IB plugin found through micro-benchmarks (malloc): a fix is planned

Experiences while scaling up

4 cores:

- Lustre FS bug (`https://jira.hpdd.intel.com/browse/LU-6528`): workaround: verify `mkdir` by calling `stat`

32 cores:

- Multiple processes on a compute node: fixes to DMTCP's code that handles shared memory segments

64 cores:

- Raised value of `MV2_ON_DEMAND_THRESHOLD`

128 cores:

- PMI external socket: added it to the list of "external" sockets

2048 cores:

- Initialization too slow with MVAPICH-1.9: switch to MVAPICH-2.1

16,384 cores:

- ???

Thanks

Questions?