Binary Checkpointing with Trick

Meena Arnold
Danny Strauss
Alex Lin
John Penn

Contents

Introduction

- Binary Checkpoint
  - A dump of a Trick simulation's memory state to an executable script
  - The script is used to later restart the Trick simulation from the saved state

- DMTCP (Distributed MultiThreaded Checkpointing)
  - Trick interfaces with an open source application named "DMTCP" to perform binary checkpointing
  - The terms "DMTCP checkpoint" and "binary checkpoint" are used interchangeably in the context of Trick checkpointing
  - Official Site: http://dmtcp.sourceforge.net

Setting up Trick with DMTCP

- Configure Trick with DMTCP

  > cd $TRICK_HOME

  > ./configure --dmtcp=<path_to_dmtcp>

  Example:

  ./configure --dmtcp= /users/trick/dmtcp/trick_dmtcp-1.2.6/dmtcp-1.2.6-linux-x86_64

  - DMTCP releases are kept in /users/trick/dmtcp/
  - It is recommended to use the latest version
  - Although most our systems are 64-bit, both 32-bit and 64-bit versions are available
  - 64-bit versions are labeled with “-linux-x86_64”

Running Trick with DMTCP

- Start the simulation with ‘s_main_dmtcp.py’
  - $TRICK_HOME/bin/s_main_dmtcp.py is a Python script that executes DMTCP with S_main_*.exe as its argument. At this point, DMTCP is a parent process to Trick

    > s_main_dmtcp.py RUN_test/input.py

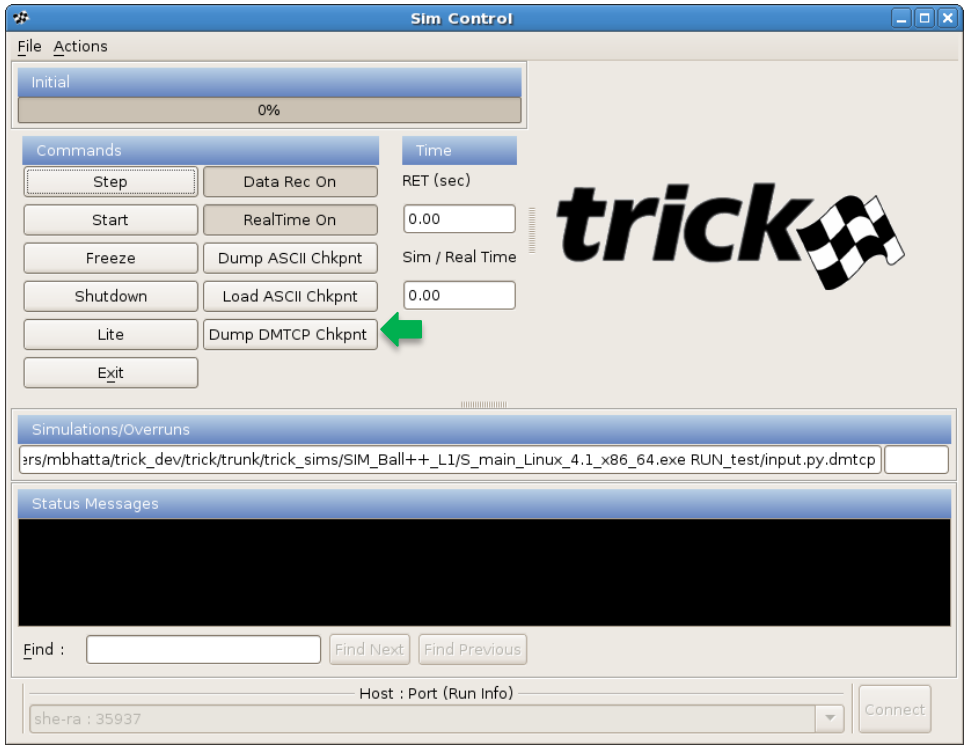
- DMTCP checkpoints are dumped to and loaded from the DMTCP Checkpoint Directory Environment Variable
  - defined in $TRICK_HOME/bin/s_main_dmtcp.py:

    “--ckptdir ${PWD}/dmtcp_checkpoints”

Dumping a Checkpoint

Method 1: Sim Control Panel

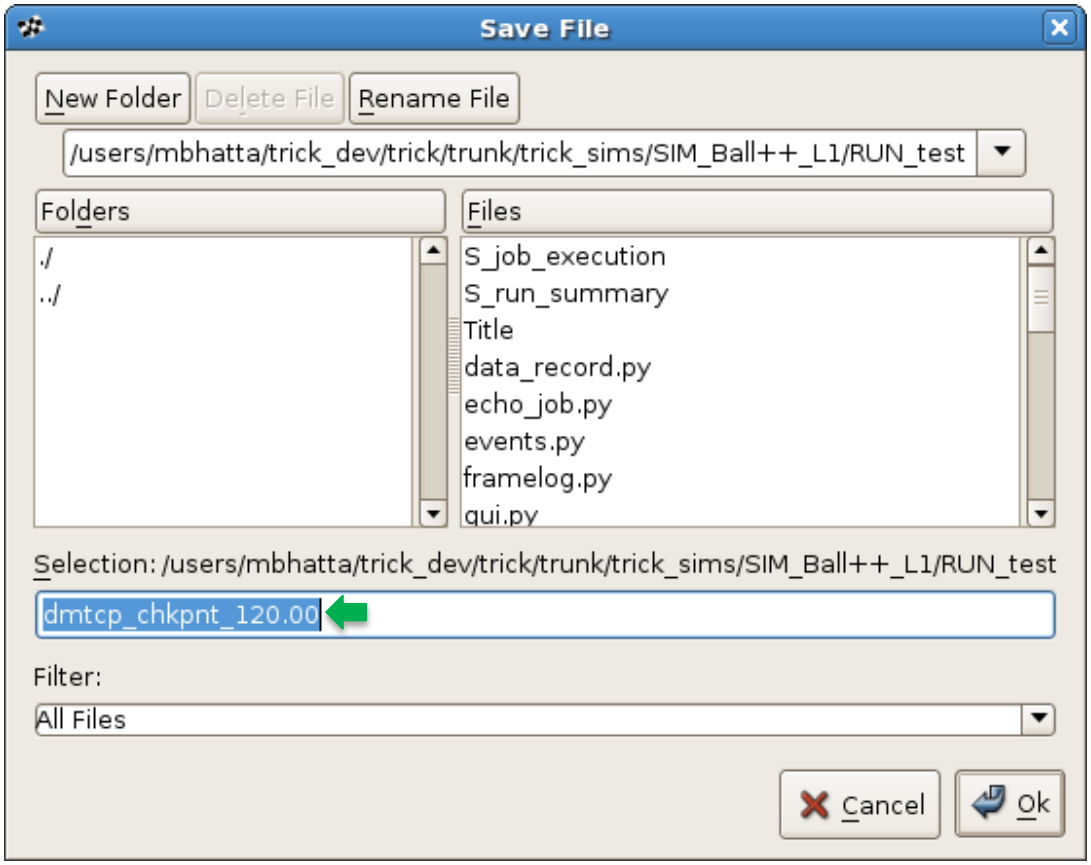
- Click “Dump DMTCP Chkpnt” button during Freeze or Initialization



Dumping a Checkpoint

Method 1: Sim Control Panel, continued

- “dmtcp_chkpnt_120.00” will be saved to the DMTCP Checkpoint Directory (i.e. ‘dmtcp_checkpoints’)



Dumping a Checkpoint

Method 2: Input File

Checkpoint now:

```
trick.dmtcp_checkpoint("filename")
```

```
trick.dmtcp_checkpoint()
```

Binary checkpoints dumped during Initialization will be named "dmtcp_chkpnt_init" unless a new filename is specified.

Checkpoint at <n> seconds:

```
trick.dmtcp_checkpoint(<n>)
```

Periodic checkpoints every <n> seconds:

```
trick.dmtcp_checkpoint_safestore(<n>)
trick.dmtcp_checkpoint_safestore_set_enabled(True|False)
```

Dumping a Checkpoint

Method 2: Input File, continued

Checkpoint at <n> seconds:

```
trick.add_read(<n>, "trick.dmtcp_checkpoint()")
```

Important Notes:

- When dumping binary checkpoints via the “add_read()” command, the Sim Control Panel must be disabled.
- DMTCP is unavailable with the “add_event() “ command

Loading a DMTCP Checkpoint

- For binary checkpoints, Trick will dump an executable restart script. To load (i.e. restart) a DMTCP checkpoint, simply execute the script:

```
> cd dmtcp_checkpoints
> dmtcp_chkpnt_120.00
```

- The simulation will start in the same state it was in at the time it was checkpointed.
- Open files will be restored
- Socket connections between simulations must be restored manually

Loading a DMTCP Checkpoint

Important Notes:

- Binary checkpoints can only be restarted from the command line. The Sim Control Panel is part of the binary executable and will appear when the binary checkpoint restart script is executed.

- DMTCP checkpoints must be loaded from the “DMTCP Checkpoint Directory” (see slide “Running Trick with DMTCP”)

Trick ASCII vs. DMTCP Checkpointing

Capability	Trick ASCII	DMTCP
Primitive types	Y	Y
C structures	Y	Y
C++ class public primitive types	Y	Y
C++ class protected/private primitive types	Y with friend	Y
C++ user defined templates	Some	Y
C++ STLs	Some with macro	Y
C++ references	N	Y
C++ static const variables	N	Y
C++ exotic constructs (class inherits from template)	N	Y

Trick ASCII vs. DMTCP Checkpointing
continued

Capability	Trick ASCII	DMTCP
File pointers and streams	N	Y
Open file contents (partially written files restored)	N	Y
Function pointers	N	Y
System types: sockets, semaphores, mutexes, etc.	N	Y
Global variables	Y if MM knows	Y
Local static variables	N	Y
Python Input Processor State	N	Y
Other languages (e.g. Ada)	Y mapped to C	Y
3rd party libraries (e.g. Matlab model)	N	Y

Trick ASCII vs. DMTCP Checkpointing
continued

Capability	Trick ASCII	DMTCP
Saves original simulation executable	N	Y
Restart using rebuilt (CP) simulation executable	Sometimes	N
Reconnect socket connections	Y with restart jobs	Auto and restart jobs
Creates checkpoint in forked process	Y	Y
Takes simultaneous checkpoints across sims.	Y	Y
Checkpoint non-Trick applications	N	Y
ASCII output (human readable checkpoints)	Y	N
Runs on Linux and Mac OSX	Y	N Linux only
Reloads checkpoint without restarting	Y	N