DMTCP: Bringing Checkpoint-Restart to Python

Kapil Arya Gene Cooperman

Northeastern University

June 27, 2013

What is Checkpoint-Restart?

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.

What is Checkpoint-Restart?

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.

Traditional Motivation:

- Long running computation on a cluster
- Computation takes 30 days
- On day 29...

What is Checkpoint-Restart?

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.

Traditional Motivation:

- Long running computation on a cluster
- Computation takes 30 days
- On day 29. . . a node crashes. Disaster!!
- Must restart from beginning

What is Checkpoint-Restart?

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.

Traditional Motivation:

- Long running computation on a cluster
- Computation takes 30 days
- On day 29. . . a node crashes. Disaster!!
- ~~Must restart from beginning~~
- Restart from last checkpoint

DMTCP: Distributed MultiThreaded CheckPointing

- Userspace; works with unmodified binaries.
- Project with roots going back 8 years.
- LGPL, freely available from http://dmtcp.sourceforge.net
- Binary packages in Debian/Ubuntu, Fedora, and OpenSUSE.
- Using DMTCP:

```
$ dmtcp_checkpoint python ./a.py
$ dmtcp_command --checkpoint
$ ./dmtcp_restart_script.sh
```

- Demo!

DMTCP: Distributed MultiThreaded CheckPointing

- Userspace; works with unmodified binaries.
- Project with roots going back 8 years.
- LGPL, freely available from http://dmtcp.sourceforge.net
- Binary packages in Debian/Ubuntu, Fedora, and OpenSUSE.
- Using DMTCP:

```
$ dmtcp_checkpoint python ./a.py
$ dmtcp_command --checkpoint
$ ./dmtcp_restart_script.sh
```

- Checkpoint-Restart involves saving and restoring:
  - all of user space memory.
  - the state of all user threads (for a multithreaded process).
  - the kernel state (the state of open file descriptors, signal handlers, etc.).
  - parent-child relationships (for process trees).
  - network state such as sockets (for distributed processes).

Checkpoint-Restart: Use Cases

- Fault tolerance (restart after a crash)
- Skip past long initialization times
  - Monte Carlo simulations
- Save/restore workspace in scientific programming environments
  - Very long-running interactive sessions
- Applications with CPU-intensive front-end and interactive analysis of results at the back-end
- Debugging
  - Checkpoint image as the "ultimate bug report"
  - Add reversibility to an existing debugger
- Process migration
- Simulations
- Speculative execution

DMTCP-Python Integration

Everything discussed so far was available years ago.

DMTCP-Python Integration

Everything discussed so far was available years ago.

This talk is about making Python aware of DMTCP!

DMTCP-Python Integration Through a Python Module

- Checkpointing an interactive Python session.

```
$ dmtcp_checkpoint python
>>> import dmtcp
>>> x = 1
>>> dmtcp.checkpoint() # Request a checkpoint
>>> # Checkpoint image has been created
...
>>> x = 2
>>> # Whoops, environment messed up, let's restore
>>> dmtcp.restore()
>>> print x
1
```

DMTCP-Python Integration Through a Python Module

- Checkpointing an interactive Python session.

```
$ dmtcp_checkpoint python
>>> import dmtcp
>>> x = 1
>>> dmtcp.checkpoint() # Request a checkpoint
>>> # Checkpoint image has been created
...
>>> x = 2
>>> # Whoops, environment messed up, let's restore
>>> dmtcp.restore()
>>> print x
1
```

- Restoring the session from the shell.

```
$ ./dmtcp_restart_script.sh
>>> print x
1
```

DMTCP Python Module: Checkpoint Hooks

```
import dmtcp
def my_ckpt():
    print "About to checkpoint." # Pre checkpoint hook
    dmtcp.checkpoint() # Create checkpoint
    if dmtcp.isResume():
        print "Resuming after a checkpoint." # Resume hook
    else:
        print "Restarting from a previous checkpoint." # Restart hook
```

DMTCP Python Module: Checkpoint Hooks

```
import dmtcp
def my_ckpt():
    print "About to checkpoint." # Pre checkpoint hook
    dmtcp.checkpoint() # Create checkpoint
    if dmtcp.isResume():
        print "Resuming after a checkpoint." # Resume hook
    else:
        print "Restarting from a previous checkpoint." # Restart hook
```

- Checkpoint-Resume.

```
$ dmtcp_checkpoint python ./a.py
"About to checkpoint."
"Resuming after a checkpoint."
...
```

DMTCP Python Module: Checkpoint Hooks

```
import dmtcp
def my_ckpt():
    print "About to checkpoint." # Pre checkpoint hook
    dmtcp.checkpoint() # Create checkpoint
    if dmtcp.isResume():
        print "Resuming after a checkpoint." # Resume hook
    else:
        print "Restarting from a previous checkpoint." # Restart hook
```

- Checkpoint-Resume.

```
$ dmtcp_checkpoint python ./a.py
"About to checkpoint."
"Resuming after a checkpoint."
...
```

- Restarting from a previous checkpoint.

```
$ ./dmtcp_restart_script.sh
"Restarting from a previous checkpoint."
...
```

DMTCP Python Module: Managing Sessions

- Features:
  - Switching between sessions.
  - Removing unwanted sessions.

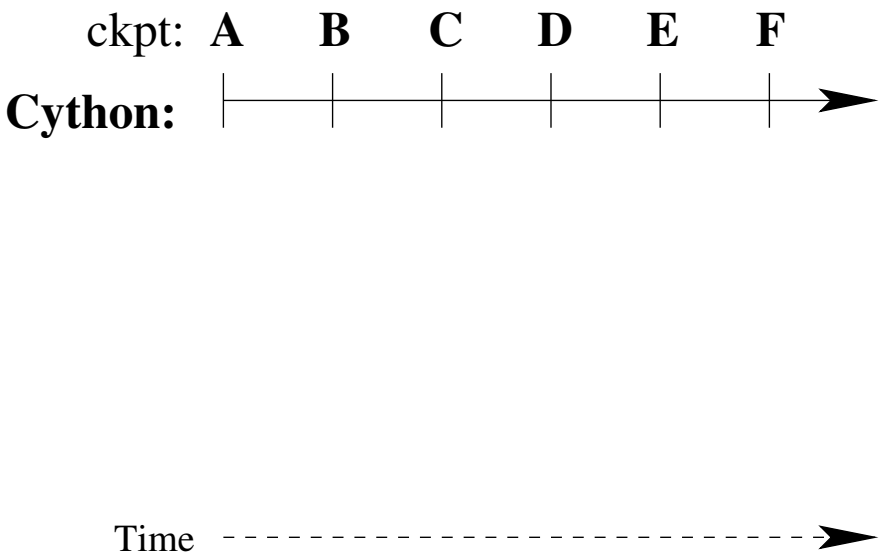
```
$ python
>>> import dmtcp
>>> dmtcp.listSessions()
[1] ('Mon Jun 24 15:35:58 2013', 'dmtcp_restart_script_4...e.sh')
[2] ('Mon Jun 24 15:36:57 2013', 'dmtcp_restart_script_4...4.sh')
>>> # Let's restore session 2
>>> dmtcp.restore(2)
```

DMTCP and IPython.parallel

- Parallel computations with IPython.parallel:
  - Checkpoint controller and multiple engines as a single unit.
  - Restart engines on a different set of nodes.
  - Restart all engines on a single node for debugging.

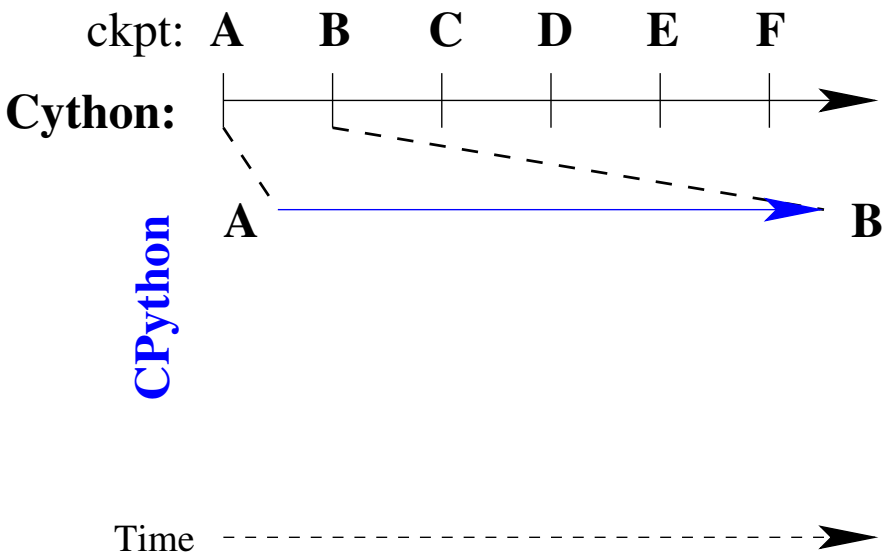
Checking Cython with Multiple CPython Instances

- **Cython** (Compiled): Fast; mostly correct.
- **CPython** (Interpreted): Slow; always correct.
- Checkpoint at regular intervals when executing the compiled version.
- Restart from checkpoint; execute in interpreted mode for verification.
- Run multiple instances of interpreted mode in parallel.
  - Can verify Cython in parallel on a cluster.



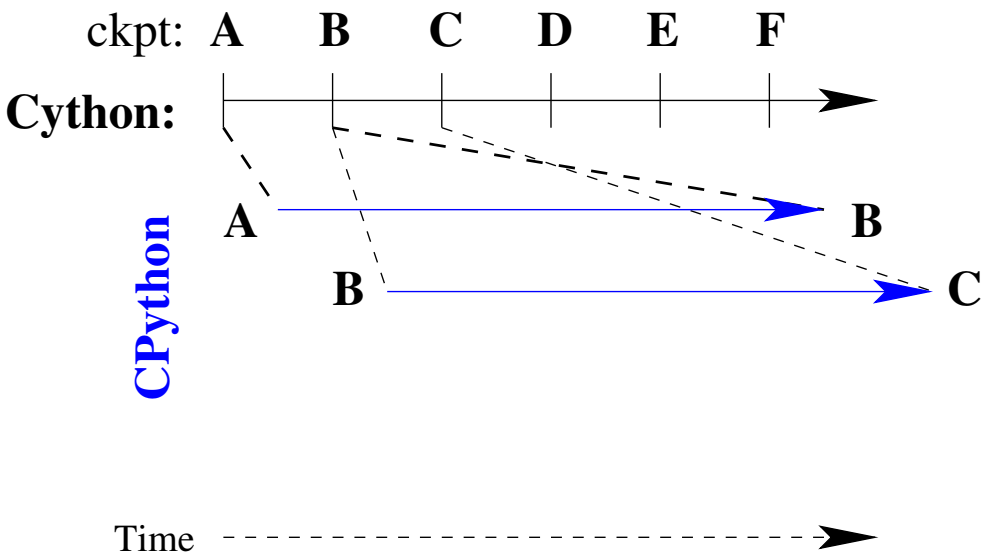
Checking Cython with Multiple CPython Instances

- **Cython** (Compiled): Fast; mostly correct.
- **CPython** (Interpreted): Slow; always correct.
- Checkpoint at regular intervals when executing the compiled version.
- Restart from checkpoint; execute in interpreted mode for verification.
- Run multiple instances of interpreted mode in parallel.
  - Can verify Cython in parallel on a cluster.



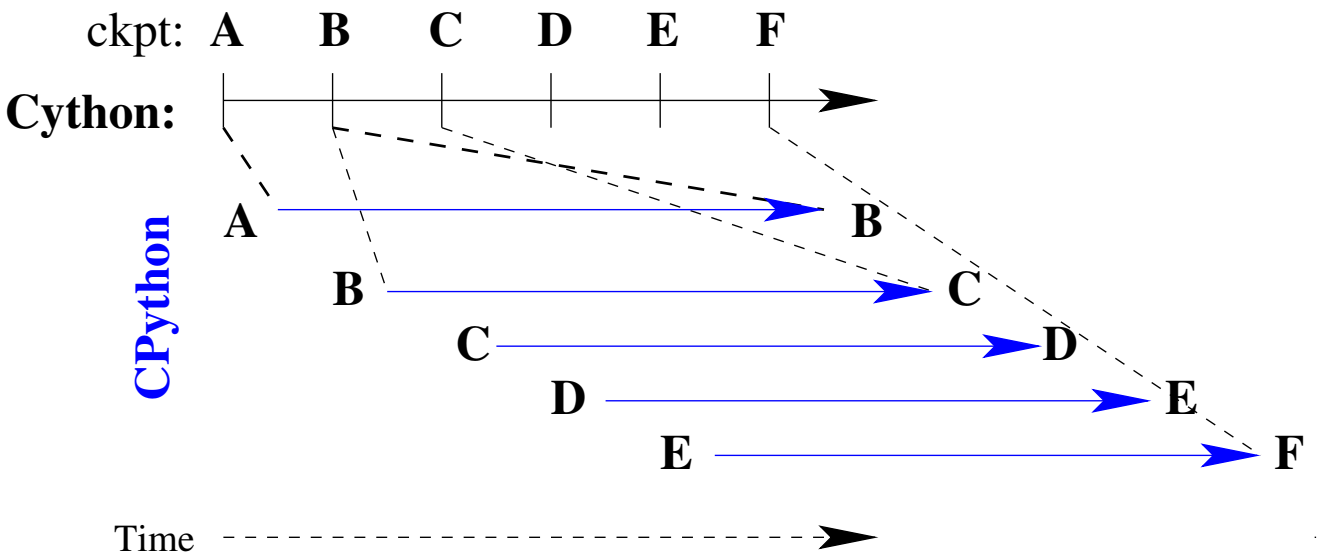
Checking Cython with Multiple CPython Instances

- **Cython** (Compiled): Fast; mostly correct.
- **CPython** (Interpreted): Slow; always correct.
- Checkpoint at regular intervals when executing the compiled version.
- Restart from checkpoint; execute in interpreted mode for verification.
- Run multiple instances of interpreted mode in parallel.
  - Can verify Cython in parallel on a cluster.



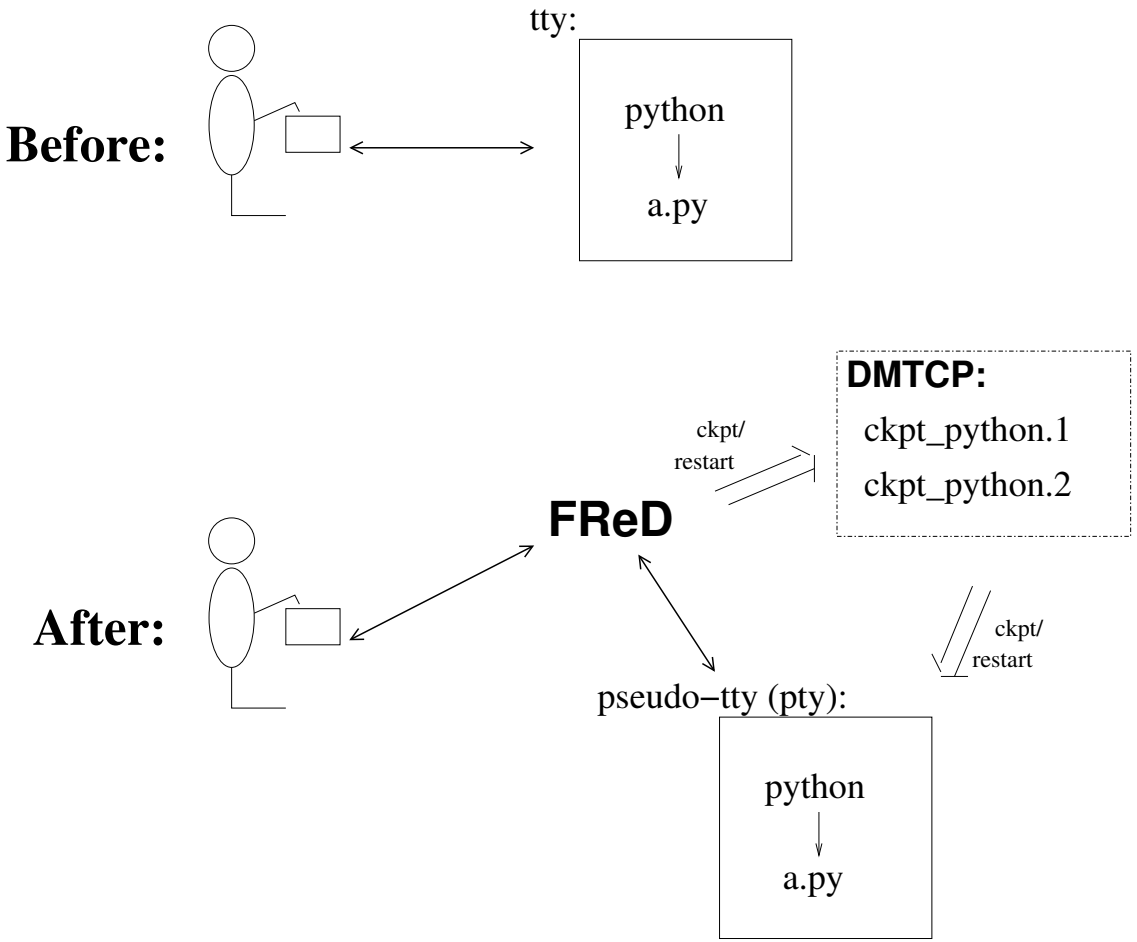
Checking Cython with Multiple CPython Instances

- **Cython** (Compiled): Fast; mostly correct.
- **CPython** (Interpreted): Slow; always correct.
- Checkpoint at regular intervals when executing the compiled version.
- Restart from checkpoint; execute in interpreted mode for verification.
- Run multiple instances of interpreted mode in parallel.
  - Can verify Cython in parallel on a cluster.



Reversible Debugging with FReD

- Fast Reversible Debugger.
- Implemented as a set of Python scripts.



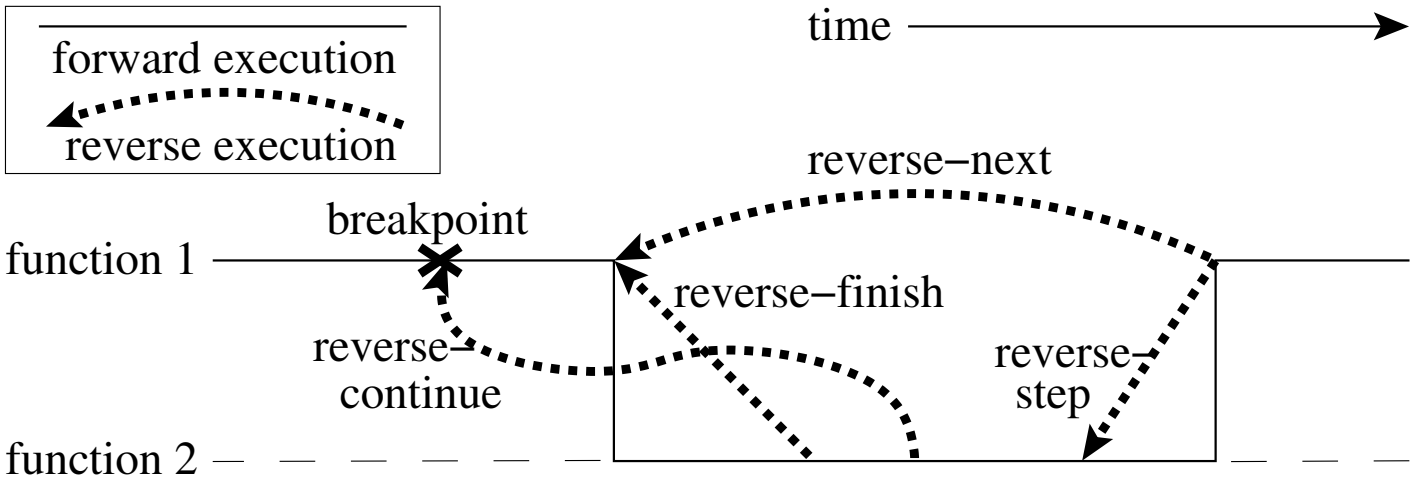
FReD: reverse-XXX commands

step : enter a function

next : step over a function

continue : execute until next breakpoint

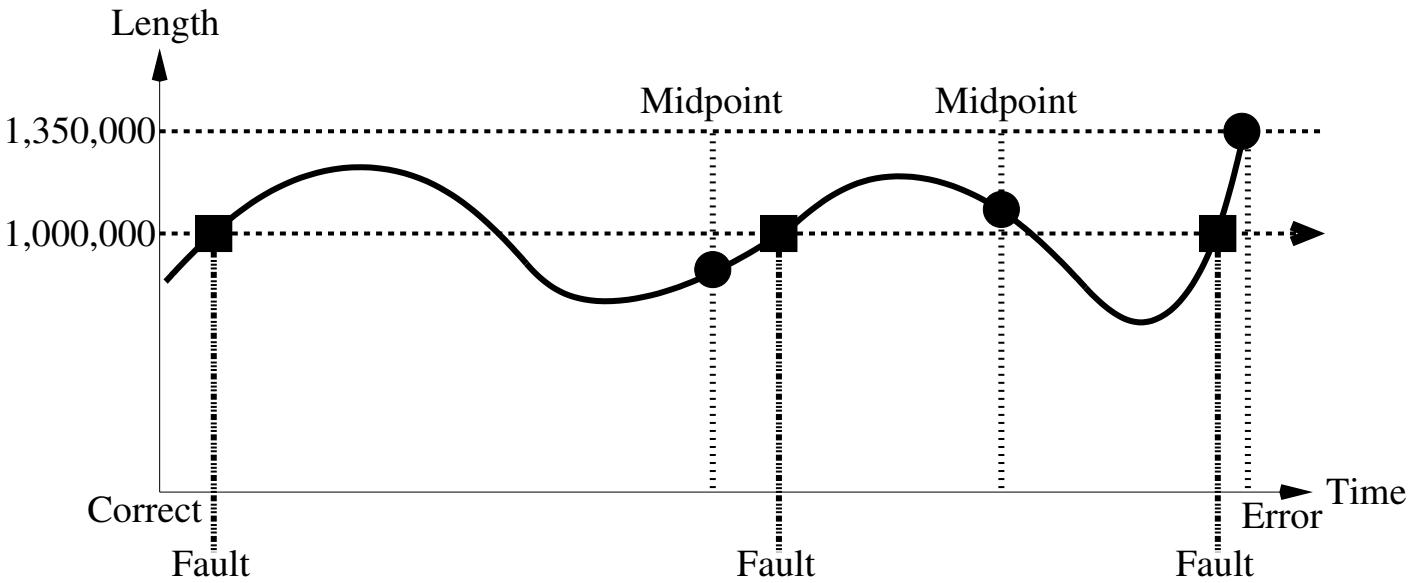
finish : execute until end of function



Undo: *if n commands beyond the last checkpoint, then restart and re-execute first n-1 commands*

Extend to: reverse-step, reverse-next, reverse-finish, reverse-watch, etc.

FReD: Reverse Expression Watchpoint



Example: repeated insertions and deletions on a bounded linked list.

Reverse Expression Watchpoint: Demo

Resources

- DMTCP: `http://dmtcp.sourceforge.net`
- FReD: `https://github.com/fred-dbg`