# Checkpoint-Restart for a Network of Virtual Machines

Rohan Garg, Komal Sodha, Zhengping Jin, Gene Cooperman

College of Computer and Information Science
Northeastern University, Boston
Boston, Massachusetts 02115

{rohgarg, komal, jinzp, gene}@ccs.neu.edu

September 24, 2013

# Outline

# Outline

# Motivation

- Parallel Computations on the Cloud
- Not everybody uses MPI: IaaS (Infrastructure as a Service)
- Flexibility and maintainability

# Motivation

- Parallel Computations on the Cloud
- Not everybody uses MPI: IaaS (Infrastructure as a Service)
- Flexibility and maintainability

Imagine if you could...

- **deploy complex software configuration** in a secure environment
- **gain high reliability** by running within a virtual machine that is set to take snapshots every minute
- **checkpoint a network** of virtual machines including the state of a parallel computation

# Outline

# Related Work

- Virtual Machine checkpointing
  - QEMU, KVM, Xen, VMware: Snapshotting
  - Remus: High Availability on Xen-based servers
  - VM-$\mu$Checkpoint: High frequency checkpointing on Xen
  - Emulab: Distributed checkpointing with Xen; record-replay of network packets
  - BlobSeer

# Related Work

- Virtual Machine checkpointing
  - QEMU, KVM, Xen, VMware: Snapshotting
  - Remus: High Availability on Xen-based servers
  - VM-$\mu$Checkpoint: High frequency checkpointing on Xen
  - Emulab: Distributed checkpointing with Xen; record-replay of network packets
  - BlobSeer
- Checkpoint-restart
  - BLCR: Kernel-space
  - CryoPid2: Process Pods; 32-bit only
  - CRIU: User-space; Linux containers
  - DMTCP: User-space; distributed

# Outline

# DMTCP and Plugins

DMTCP:

- Distributed MultiThreaded Checkpointing
- User-space
- Transparent checkpointing
- Distributed processes
- Wide range of supported applications: MPI, Perl/Python, GDB, X-windows , Matlab, R

# DMTCP and Plugins

DMTCP:

- Distributed MultiThreaded Checkpointing
- User-space
- Transparent checkpointing
- Distributed processes
- Wide range of supported applications: MPI, Perl/Python, GDB, X-windows , Matlab, R

DMTCP Plugins:

- DMTCP extensions; shared libraries
- Short, well-defined API
- Add support to handle the checkpoint-restart of specific resources

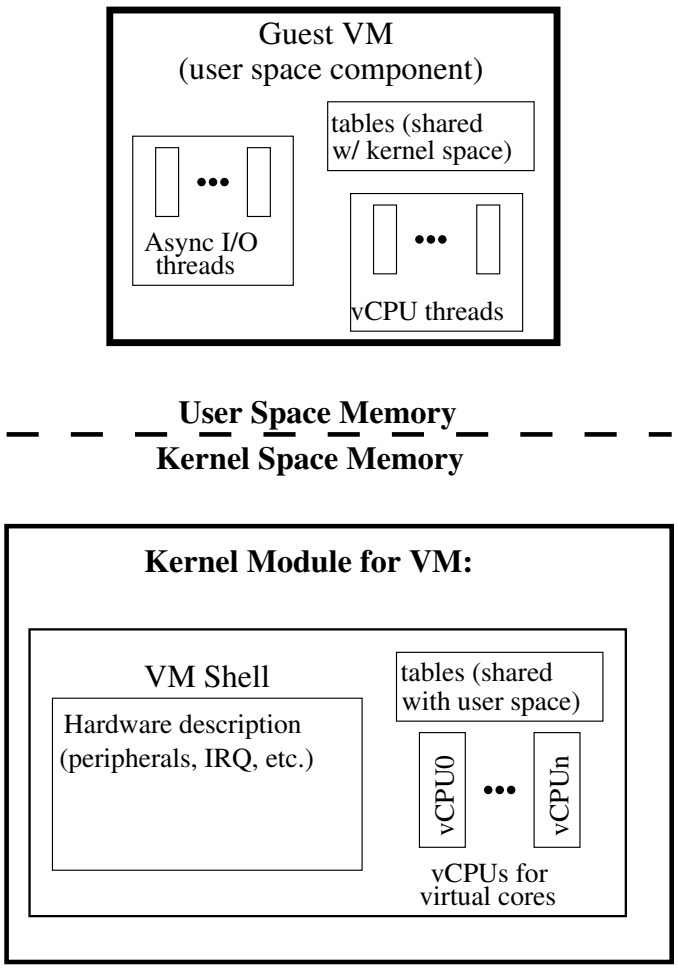
# DMTCP Plugins: Features

Two essential features:

- Wrapper Fuctions:
  - Interpose on library and system function calls
  - Process the arguments; call the interposed function; and return back (possibly modified) return value
- DMTCP Events:
  - Notify plugin of several events: Pre-checkpoint, Post-restart, etc.

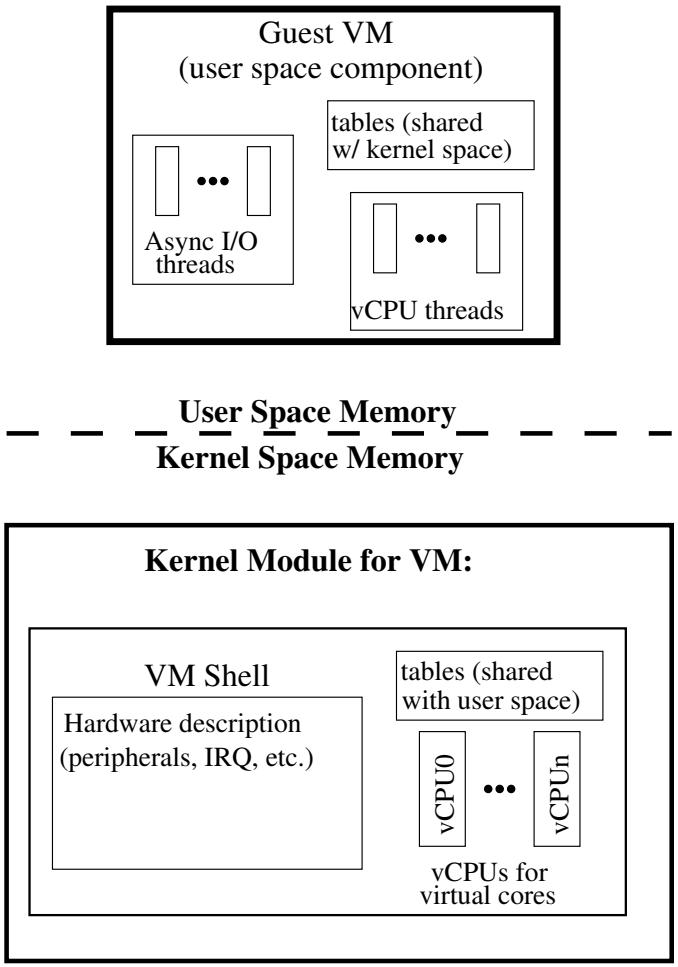
# Generic Checkpoint-Restart for VMs: Background

## Generic VM Architecture



# Generic Checkpoint-Restart for VMs: Background

## Generic VM Architecture




Special Cases:

- Xen, VMware ESXi Server: very thin hypervisor; bare-metal; no host OS
- QEMU: Software emulation; user-space

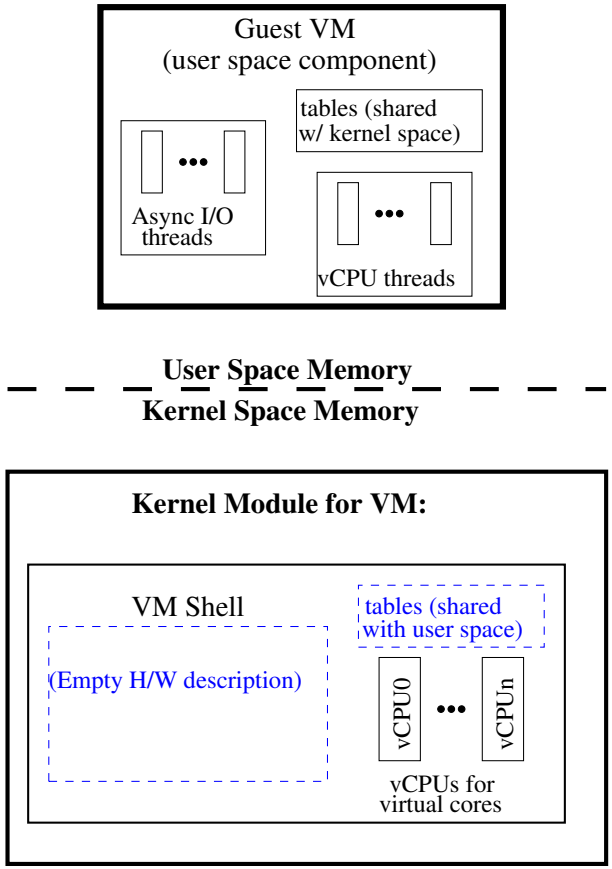
# Generic Checkpoint-Restart for VMs: Background

- DMTCP:
  - Handle user-space memory, file descriptors, sockets, etc.

```
% dmtcp_checkpoint qemu <args-for-qemu>
% dmtcp_command --checkpoint
% dmtcp_restart ckpt-qemu-img.dmtcp
```

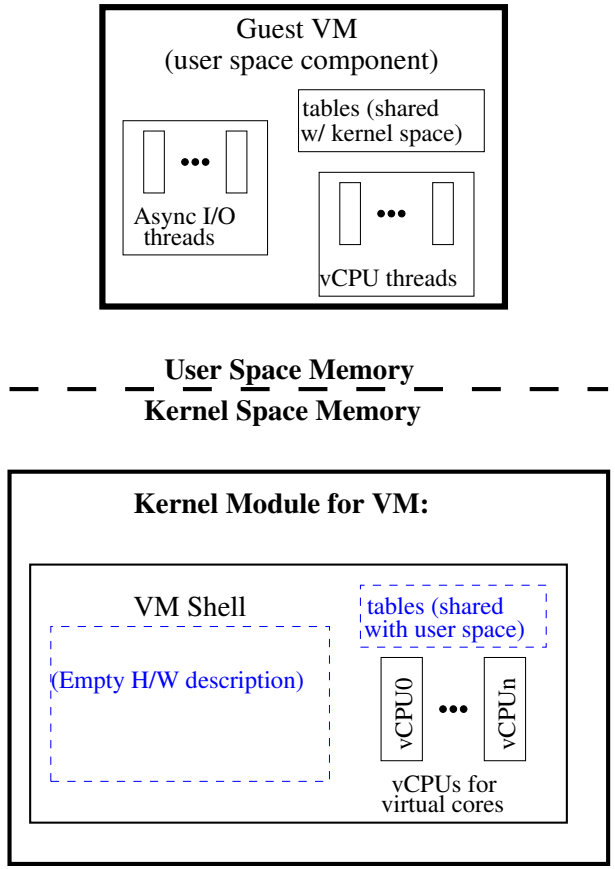
# Checkpoint-Restart for KVM: Key Ideas

- DMTCP KVM Plugin:
  - Launch empty VM *shell*
  - Copy the checkpoint image (they're just bits) from the old checkpointed VM
  - Restore kernel VM driver parameters
  - Patch kernel VM driver parameters



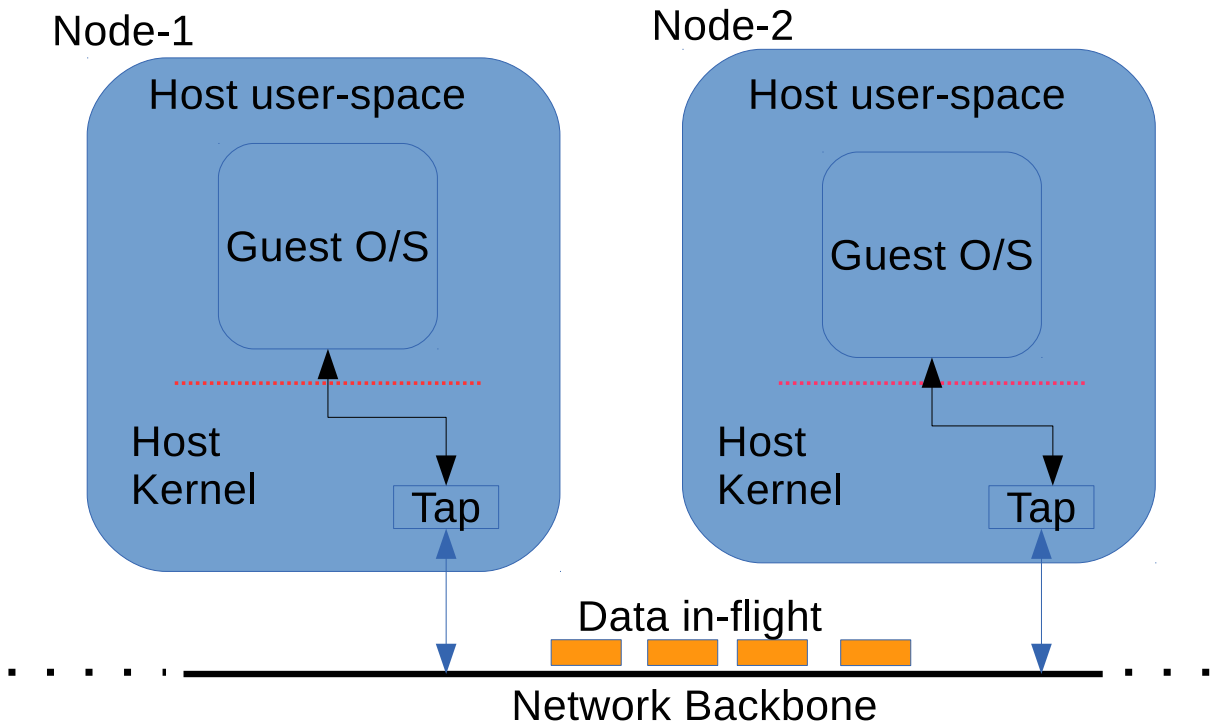
# Checkpoint-Restart for KVM: Key Ideas

- DMTCP KVM Plugin:
  - Launch empty VM *shell*
  - Copy the checkpoint image (they're just bits) from the old checkpointed VM
  - Restore kernel VM driver parameters
  - Patch kernel VM driver parameters

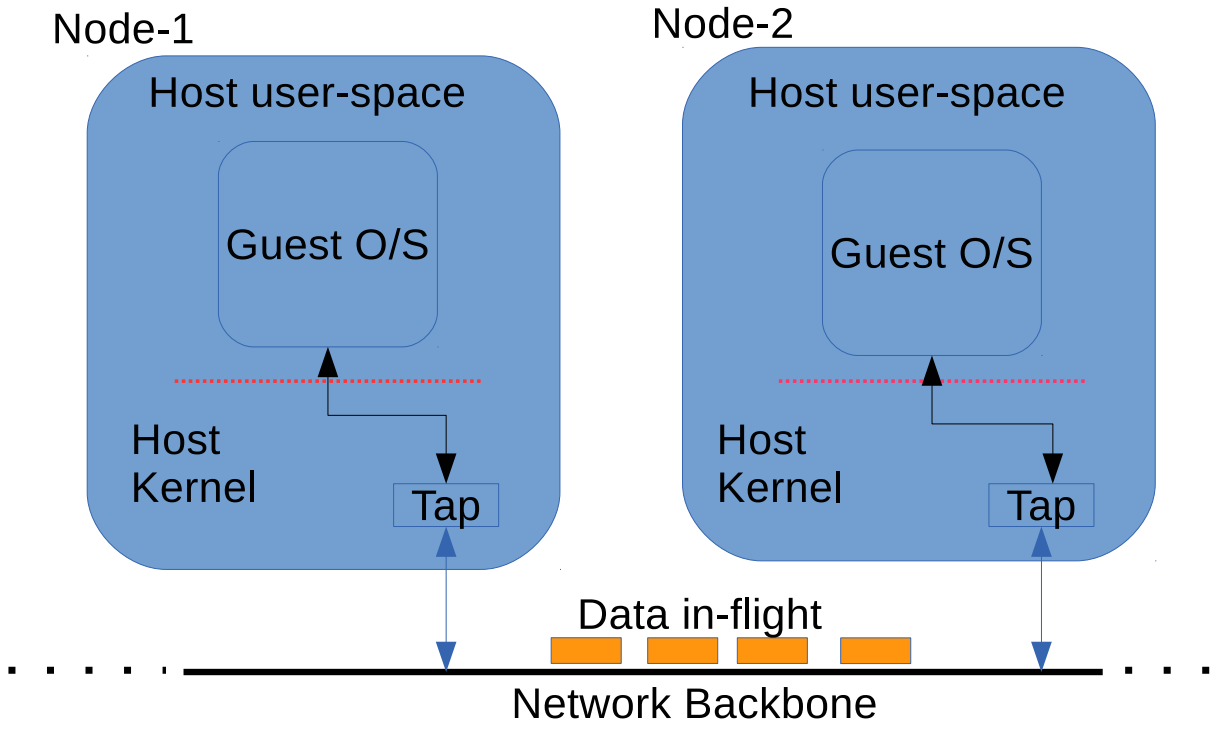



```
% dmtcp_checkpoint \
        --with-plugin dmtcp_kvm_plugin.so \
        qemu -enable-kvm <args-for-qemu>
% dmtcp_command --checkpoint
% dmtcp_restart ckpt-qemu-img.dmtcp
```

# Challenges for checkpointing a network of VMs



# Challenges for checkpointing a network of VMs




Challenges:

- **Synchronization** between VMs
- **Re-generating** the virtual network
- Saving and restoring **in-flight data**

# Challenges for checkpointing a network of VMs: Solutions

- **Synchronization** between VMs

# Challenges for checkpointing a network of VMs: Solutions

- **Synchronization** between VMs
  - DMTCP Co-ordinator

# Challenges for checkpointing a network of VMs: Solutions

- **Synchronization** between VMs
  - DMTCP Co-ordinator
- **Re-generating** the virtual network
- Saving and restoring **in-flight data**

# Challenges for checkpointing a network of VMs: Solutions

- **Synchronization** between VMs
  - DMTCP Co-ordinator
- **Re-generating** the virtual network
- Saving and restoring **in-flight data**
  - DMTCP TUN/TAP Plugin: Heuristic:
    - Quiesce the user-application threads
    - Wait for a fixed time: assume all packets have arrived
    - Write the checkpoint image (if additional packets continue to arrive, try again)
    - Alternative approach: broadcast a *cookie*

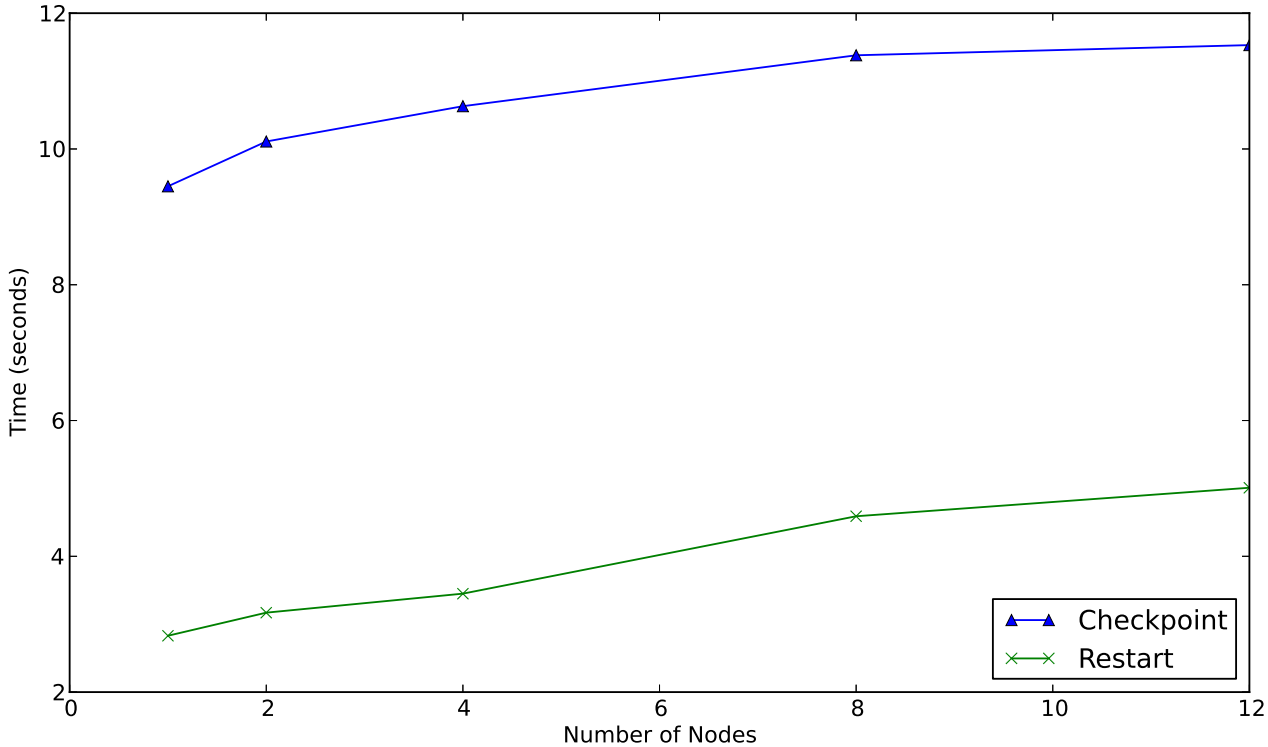
```
% dmtcp_checkpoint \
        --with-plugin dmtcp_kvm_plugin.so \
        --with-plugin dmtcp_tun_plugin.so \
        qemu -enable-kvm <args-for-qemu>
% dmtcp_command --checkpoint
% dmtcp_restart ckpt-qemu-img.dmtcp
```

# Outline

# Experimental Results: Setup

- Network of Virtual Machines
  - 12-node cluster (at University of Alabama, Birmingham)
  - Each node: 12-core Intel Xeon (1.6 GHz) server; 24 GB RAM
  - KVM/QEMU with Tap
  - Host OS: 64-bit CentOS; Linux Kernel 2.6.32
  - Guest OS: Ubuntu 12.04 Server
- Others:
  - Btrfs (nested VMs)
  - DMTCP optimizations
  - Commodity computer

# Experimental Results: Scalability




Checkpoint-restart of HPCC benchmark on a Gigabit Ethernet cluster, (Memory allocated in each case is 1024 MB.)

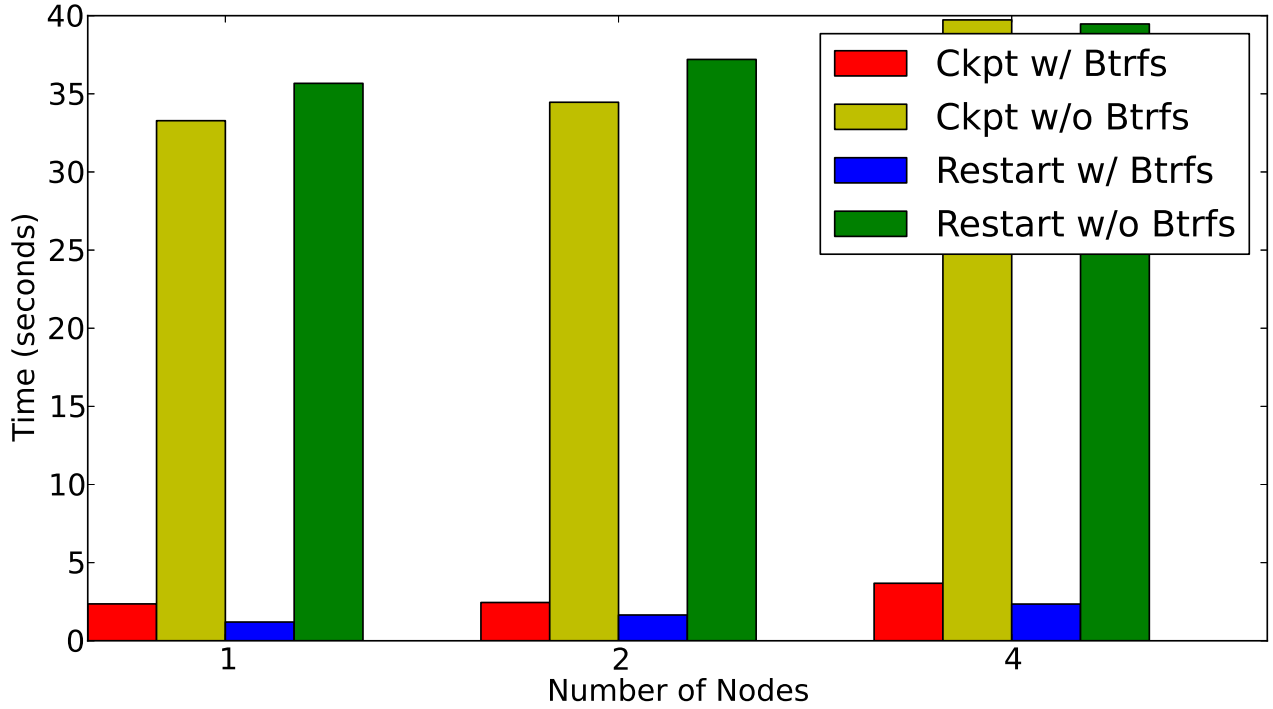
# Experimental Results: Optimizations - I

- Btrfs filesystem
  - Fast, incremental checkpoints
  - Copy-on-write filesystem
  - Going to be the default filesystem (soon?)
  - Nested VMs

# Experimental Results: Optimizations - I

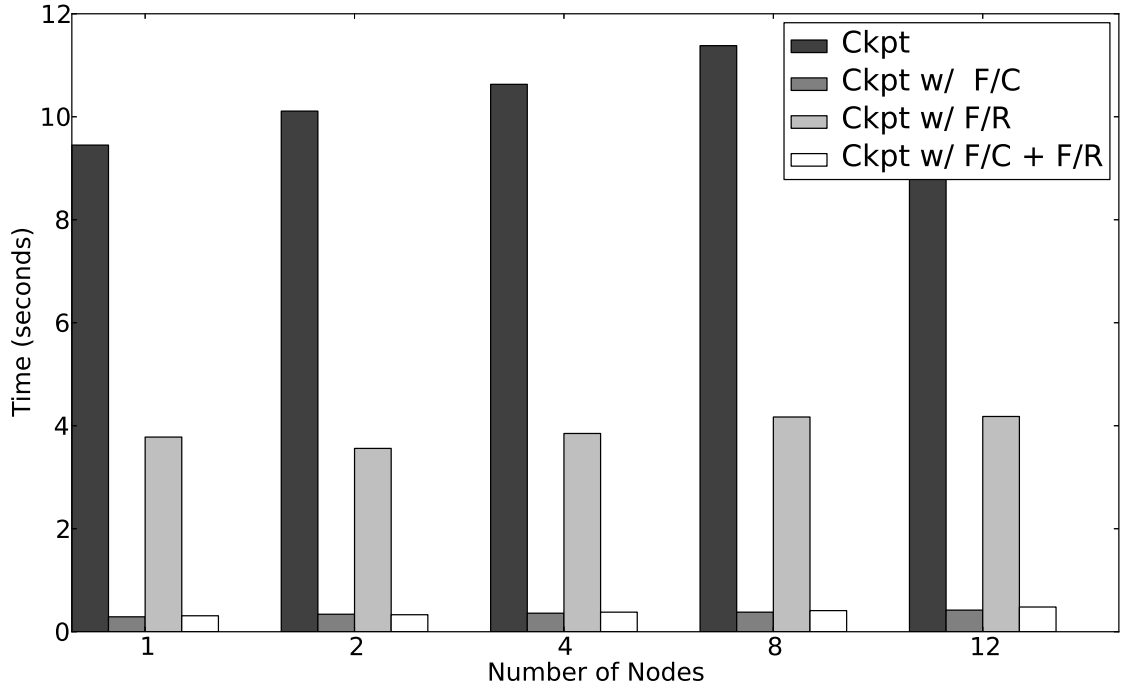
- Btrfs filesystem
  - Fast, incremental checkpoints
  - Copy-on-write filesystem
  - Going to be the default filesystem (soon?)
  - Nested VMs
- DMTCP optimizations
  - *Forked checkpointing*: copy-on-write: fork a child to write checkpoint; parent continues
  - mmap-based *fast restart*: on-demand paging from the checkpoint image

# Experimental Results: Optimizations - II



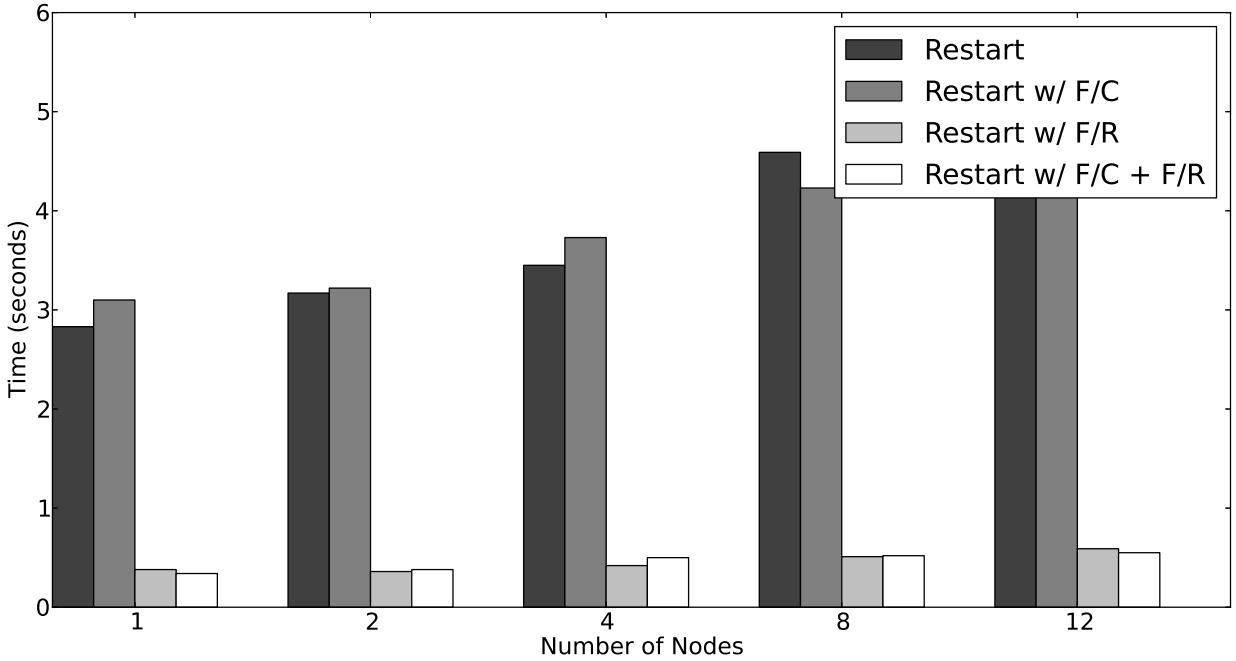

Snapshotting up to four distributed VMs running HPCC under KVM/QEMU. The Btrfs filesystem is used to snapshot the filesystem using nested VMs. (Memory allocated in each case is 384 MB. The size of the guest filesystem is 2 GB.)

# Experimental Results: Optimizations - II




Checkpoint of HPCC benchmark on a Gigabit Ethernet cluster, as influenced by DMTCP's optional optimizations: forked checkpoint (F/C) and fast restart (F/R). DMTCP's default gzip compression of checkpoint images is incompatible with DMTCP F/R, and so is not used in those cases. (Memory allocated in each case is 1024 MB.)

# Experimental Results: Optimizations - II




Restart of HPCC benchmark on a Gigabit Ethernet cluster, as influenced by DMTCP's optional optimizations: forked checkpoint (F/C) and fast restart (F/R). DMTCP's default gzip compression of checkpoint images is incompatible with DMTCP F/R, and so is not used in those cases. (Memory allocated in each case is 1024 MB.)

# Outline

# Conclusion

Summary

- Generic mechanism for checkpoint-restart: QEMU (user-space), Lguest (paravirtualization), QEMU/KVM (hardware-assisted virtualization)
- Btrfs: fast, incremental snapshots
- Low maintainability, high flexibility: plugin with 400 LOC

**Questions?**