VCCP: A Transparent, Coordinated Checkpointing System for Virtualization-based Cluster Computing

*N. Saragol **　　*Hong Ong#*　　*K. Chanchio**
Box Leangsuksun+

* Thammasat University, Patumtani, Thailand
\# Oak Ridge National Laboratory, Oak Ridge, TN, USA
\+ Louisiana Tech University, Ruston, LA, USA

Outline

- Introduction
- Goals and Motivations
- Related Works
- VCCP Mechanisms
  - Protocol and Analysis
- Experimental Results
- Progress and Future Works

Introduction

- Fault tolerance is necessary for HPC
  - The more hw/sw components, the higher the chances some of them will fail
  - Bad for long-running parallel applications
- Checkpoint/Restart is a common technique used to provide fault tolerance. Common approaches include:
  - Modifying App. source code
  - Linking App with User-level library
  - Modifying OS kernel

Problems

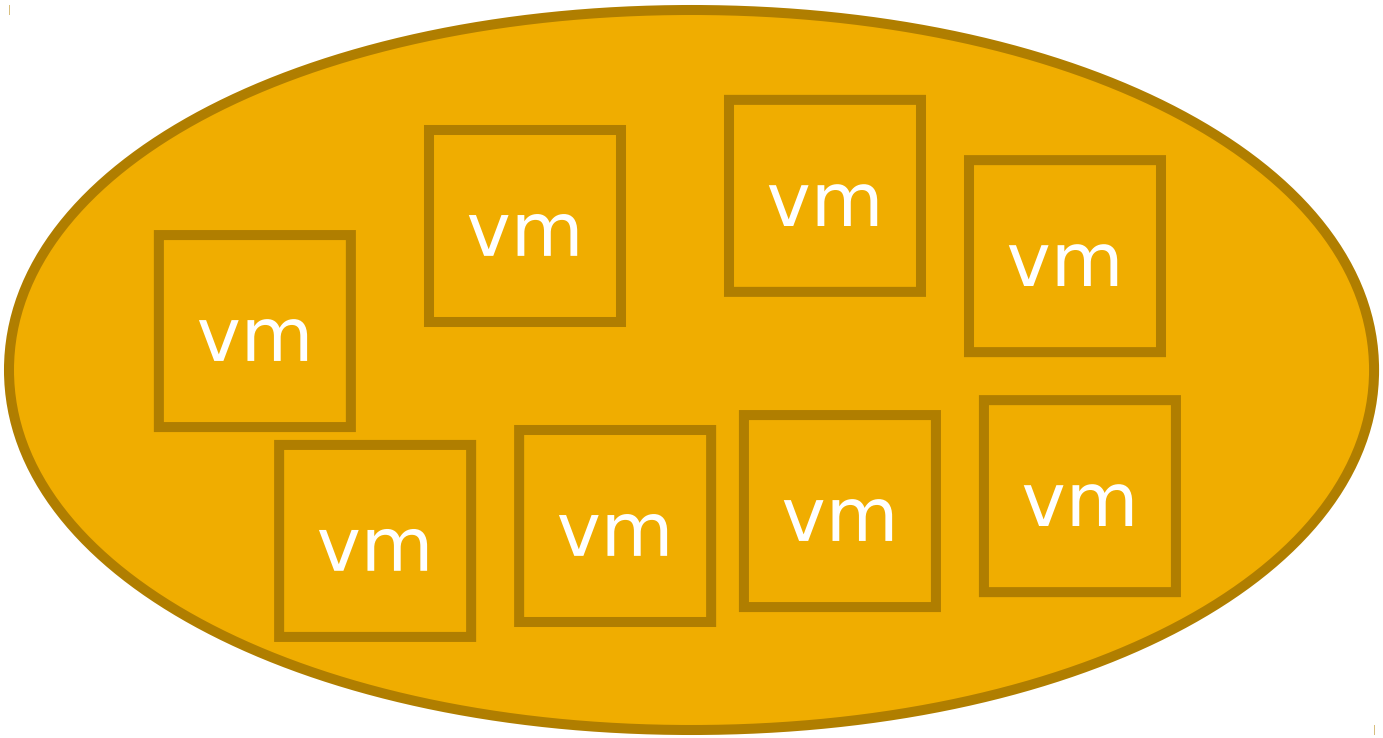
Existing checkpoint/restart tools for parallel processing are:

- **Complex:** require runtime systems and/or compilation tools, thus adding more components system-wide
- **Not user-friendly:** require additional works such as software installation, code modification, kernel modification, recompilation/re-linking.
- **Vendor-specific:** each software tool for parallel processing has a different checkpoint/restart implementation

Goals

- High transparency
  - Checkpoint/restart mechanisms should be ***transparent*** to applications, OS, and runtime environments; no modification required
- Efficiency
  - Checkpoint/restart mechanisms should ***not*** generate unacceptable overheads
    - Normal Execution
    - Communication
    - Checkpointing Delay

Goals (cont.)



etwork of VMs as **an abstract system unit** that all
ckpointing to be conducted **transparently** and
ciently without modifying guest OS or applications.

Motivations

- Leverage **Virtualization Technology** to provide highly transparent checkpointing mechanisms for HPC
  - Hypervisor can *transparently* **save** and **restore** VM state
- **A Cluster of Virtual Machines** can be built to run parallel applications
- Virtual Machine Technology **keeps get better**
  - E.g. HW supports (Intel VT), Virtual I/O

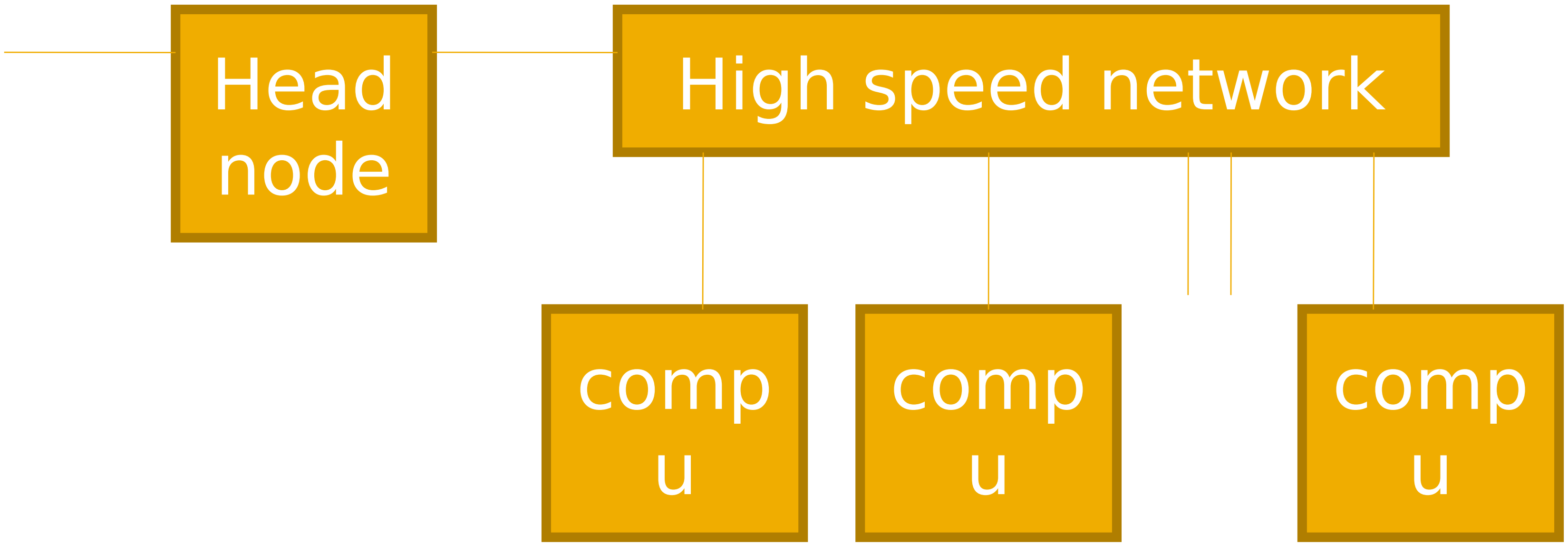
Related Works

- Most coordinated checkpointing approach are implemented at library level
  - E.g. LAM/MPI, MPICHV, CoCheck
- Some is hypervisor-based, but require software component on guest OS
  - Scarpuzza et al.: a Xen-based system on Infiniband
- Highly-transprent, hypervisor-based, but not coordinated checkpointing approach
  - Kangarlou et al. implement a non-blocking snapshot on Xen and VIOLIN virtual network

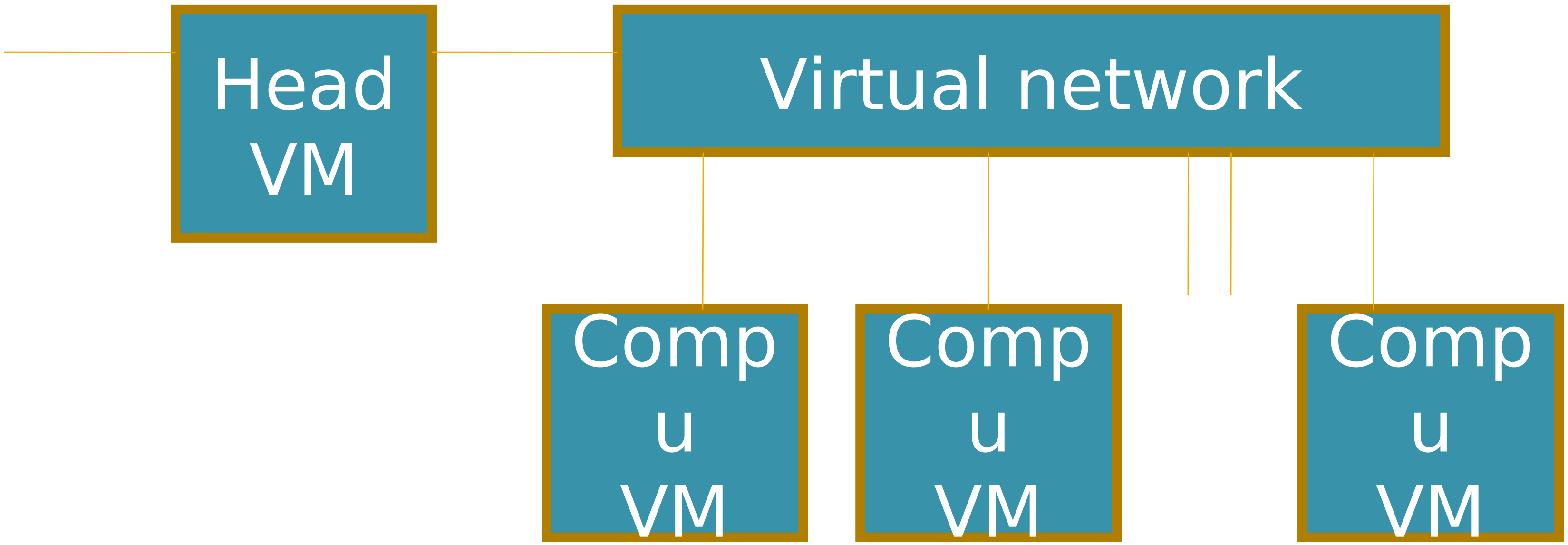
Contributions

- Layered Virtual Cluster Architecture
- Novel Hypervisor-based Coordinated Checkpointing Protocols
  - Transparent C/R mechanisms
- Correctness Analysis
- Preliminary Implementation & Experiments
  - A proof of concepts

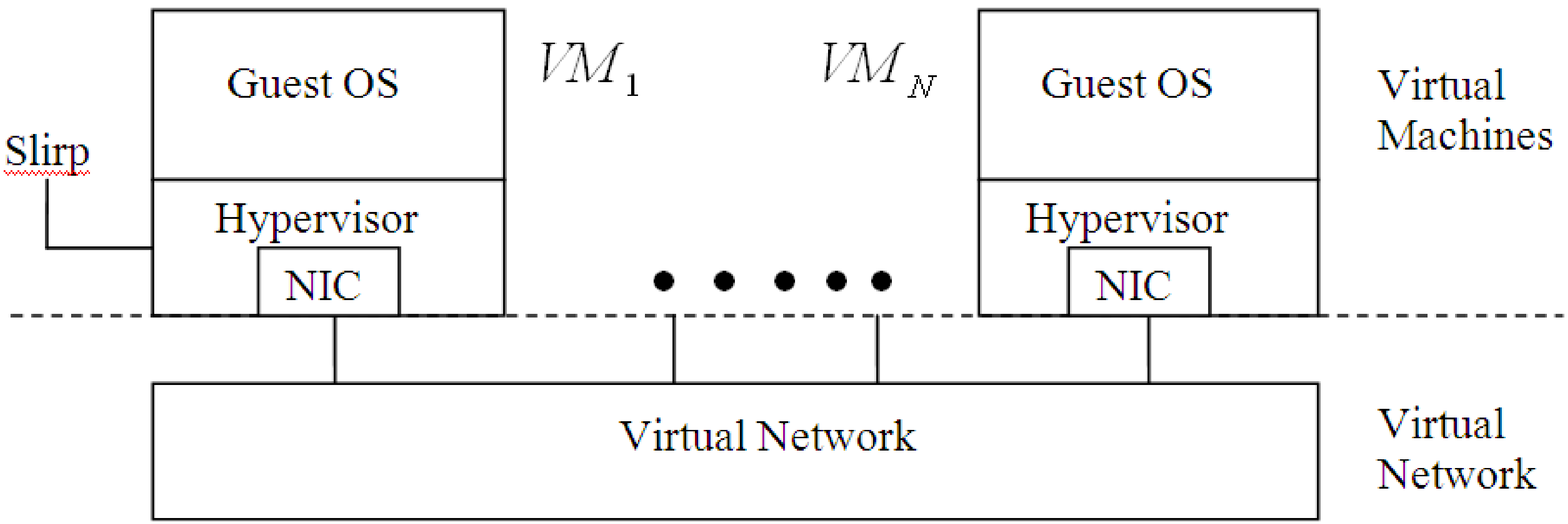
Cluster Architecture



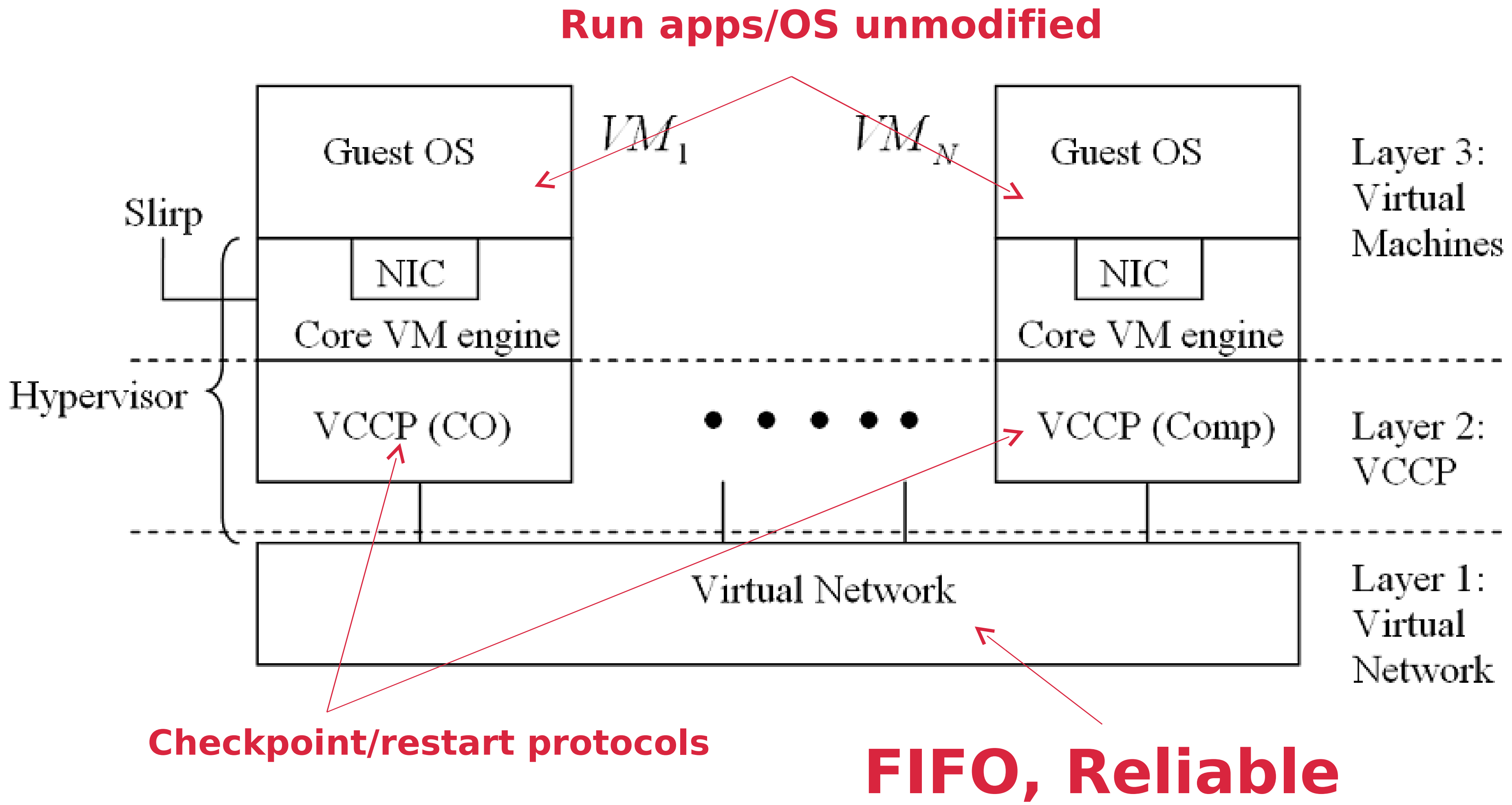
Virtual Cluster Architecture



Typical Virtual Cluster Architecture



Virtual Cluster Architecture

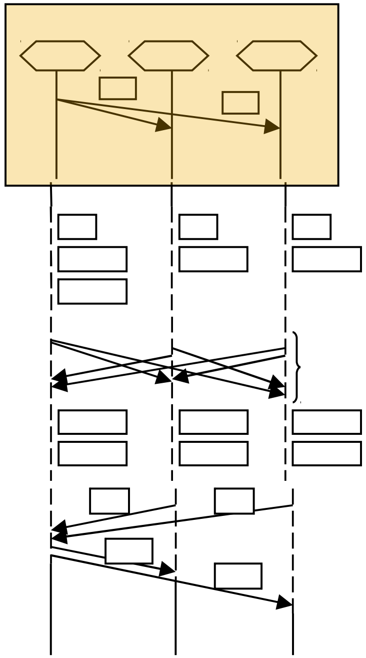


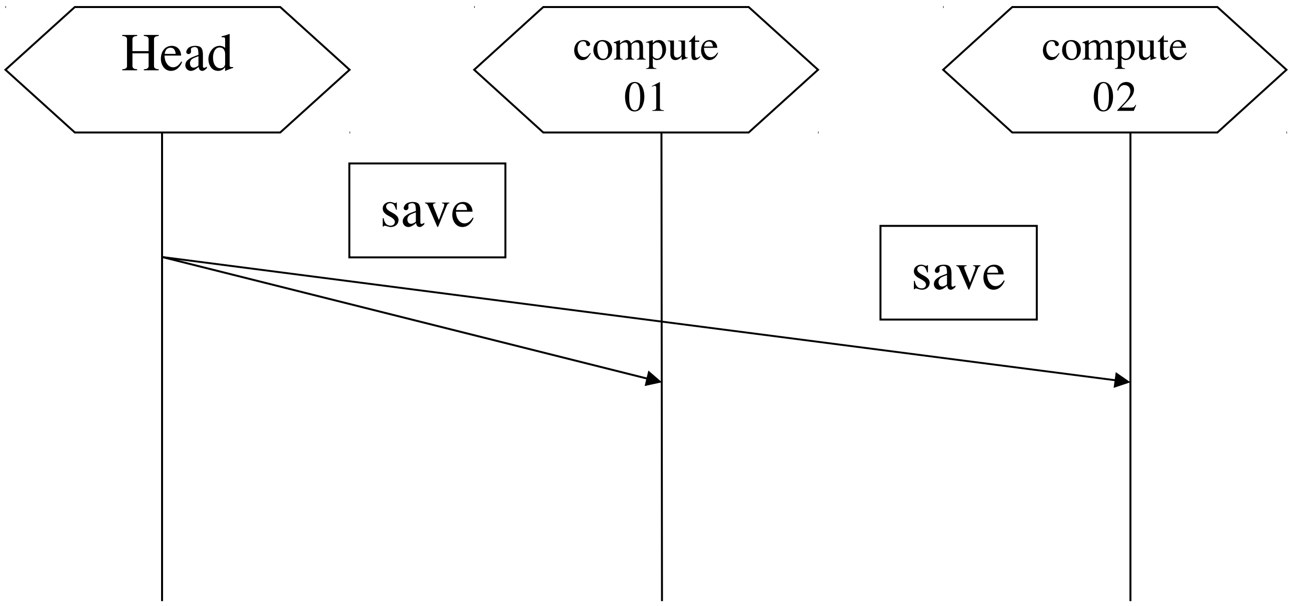
Virtual Cluster CheckPointing (VCCP) Protocol

1. Stop VM computation
2. Flush messages out of the network
3. Locally Save State of every VM
4. Continue computation

VCCP

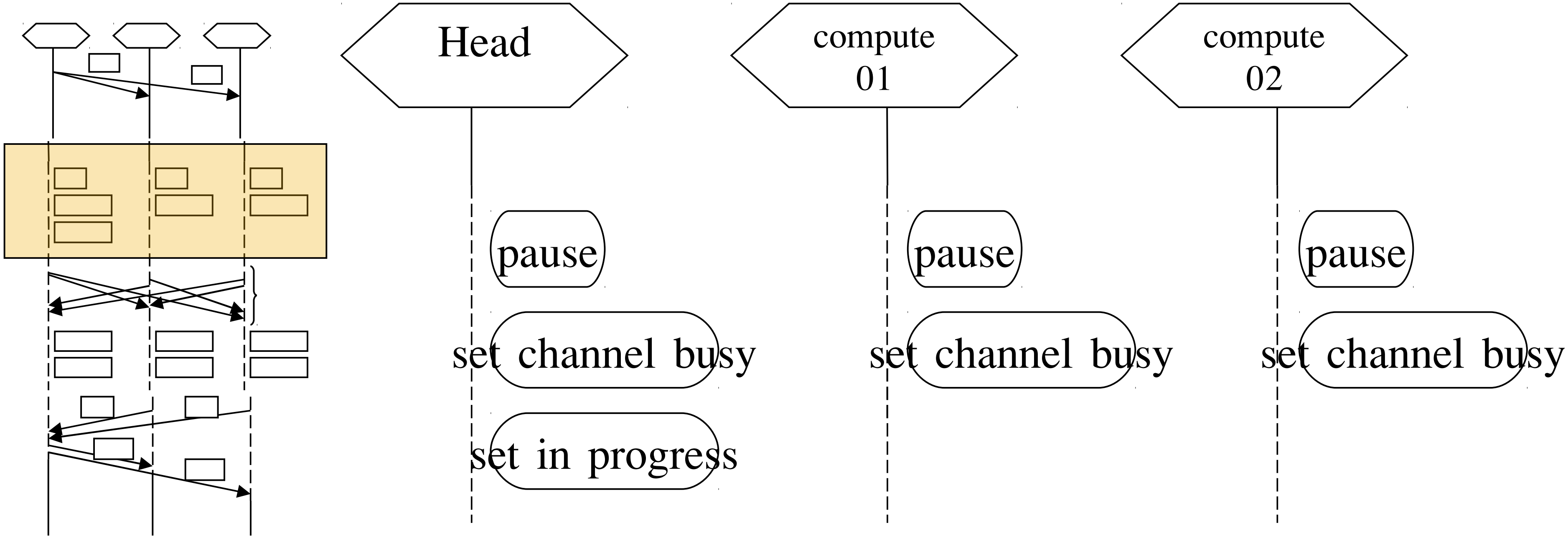
VCCP checkpoint protocol





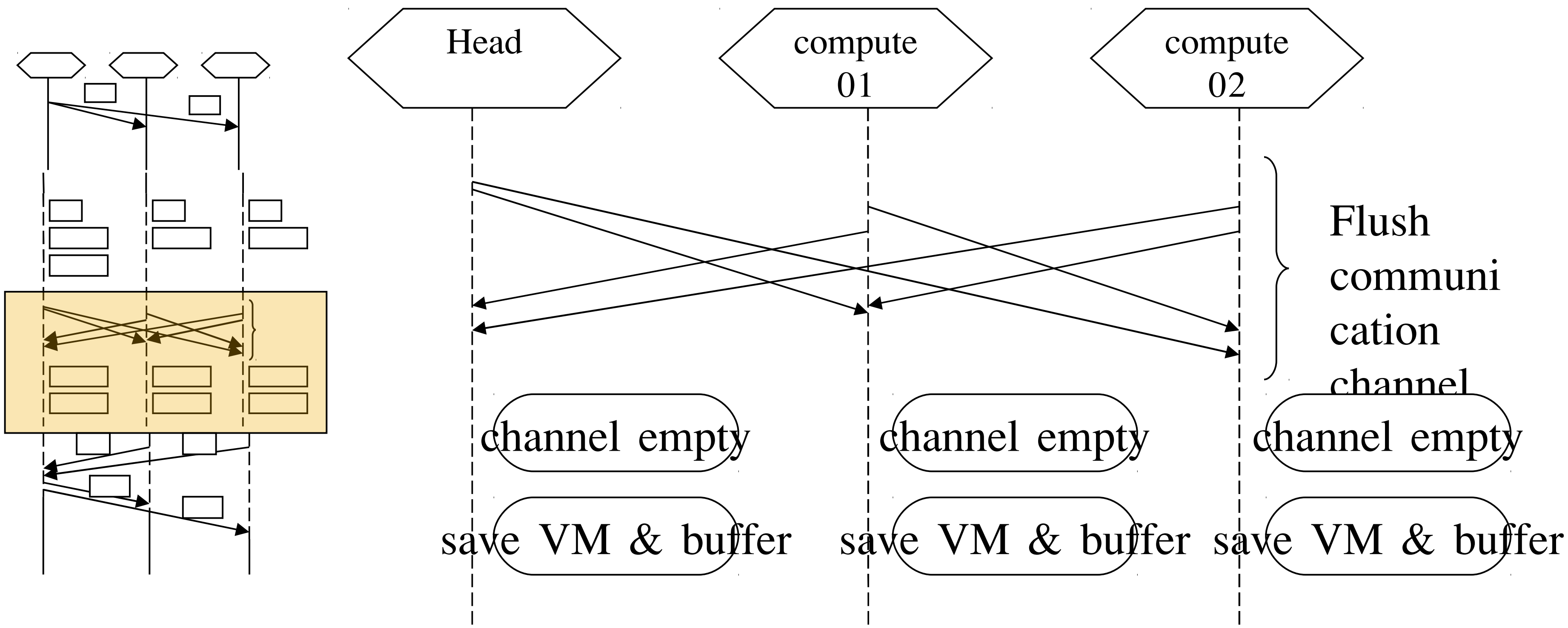
VCCP

VCCP checkpoint protocol



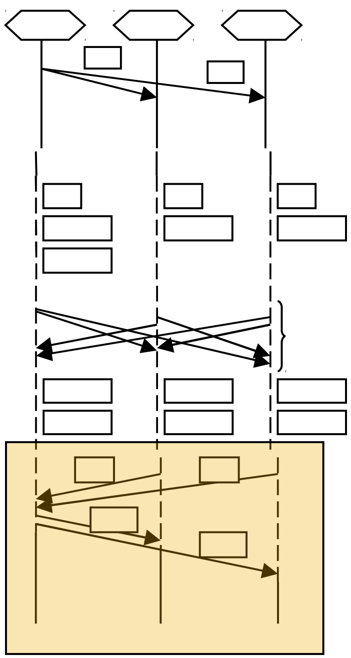
VCCP

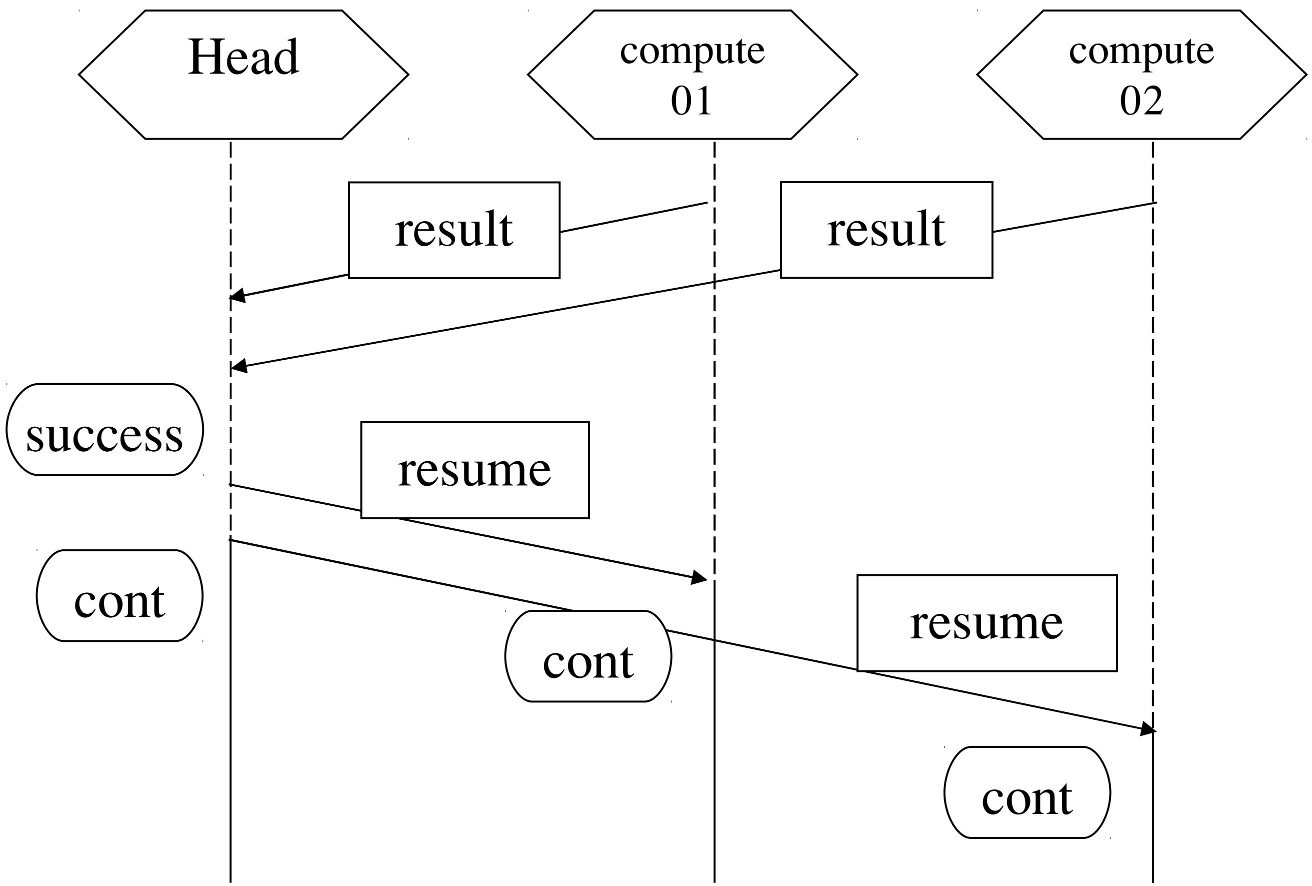
VCCP checkpoint protocol



VCCP

VCCP checkpoint protocol





VCCP recovery protocol
VCCP recovery protocol

Coordinator:

1. Broadcast *loading* requests
2. Load VM state and **Recv Queue**
3. Repeat
4. Receive frames
5. Until (*restore done* frames are received from all other VMs)
6. Broadcast *resume VM* request
7. Resume local VM

Compute:

Upon receiving a *loading* request,

1. Load VM and **Recv Queue**
2. Send *restore done* frame to Coordinator
3. Wait until receiving a *resume VM* frame
4. Resume VM

Correctness Analysis

- **The checkpoint/recovery protocols do not cause problems to normal execution**
  - TAB/VDE/TCP maintain FIFO ordering and reliable
- The protocols can cause clock skew
  - Cannot guarantee pausing and resuming at the exact same time on every VM
- Checkpointing and recovery events can affect RTT (See Next slide)

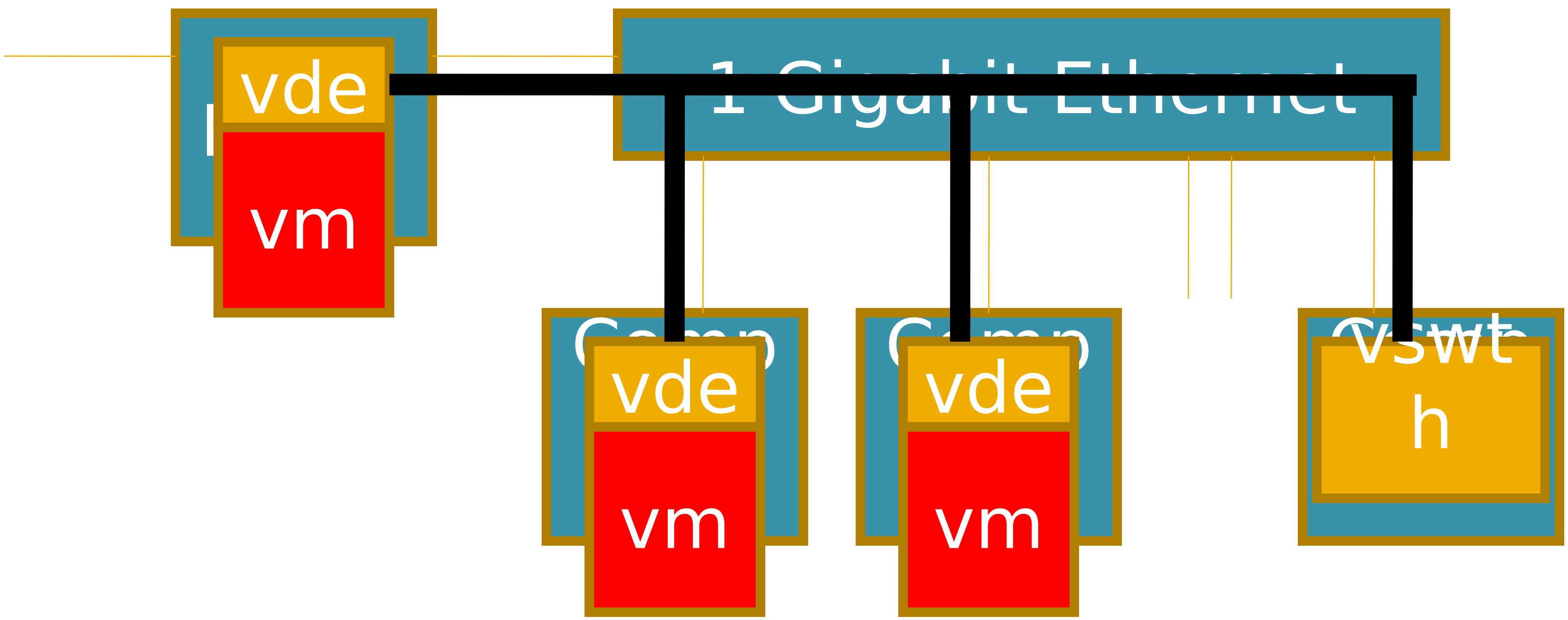
RTT analysis

1. Checkpointing can affect message RTT; depending on the clock skew value
2. This problem also occurs on traditional checkpointing mechanisms
3. Solutions:
   - Make the timeout threshold large
   - Keep the clock skew value small

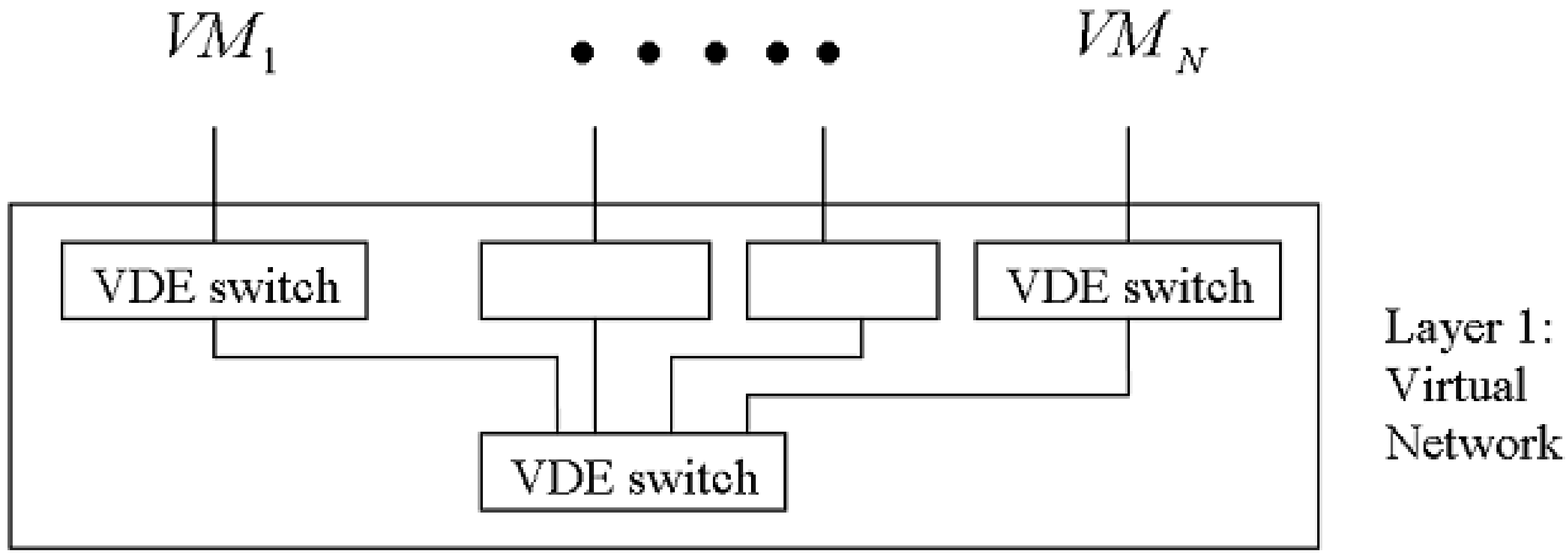
Experiment

- Cluster Node x 9 (8 compute, 1 virtual switch)
  - Processor: Intel Xeon 2.6 GHz x 2
  - Memory: 2 GB
  - HD: 40 GB
  - Host OS: Rock 4.2.1
  - 1 Gigabit Ethernet
- Virtual Cluster x 8
  - Hypervisor: Modified QEMU
  - Guest OS: Damn Small Linux 3.4.1 + OpenMPI 1.2.3
  - Memory: 512 MB
  - HD : 300 MB (4 GB Maximum)

Testbed Configuration



Virtual Distributed Ethernet (VDE)



Benchmarks

NAS Parallel Benchmark

- EP : estimate floating point performance minimal interprocessor communication
- CG : estimate unstructured matrix vector multiplication performance with interprocessor communication
- IS : estimate integer sort performance with interprocessor communication
- MG: estimate data communication performance

Overheads

Kernel EP	*2 nodes*	*4 nodes*	8 nodes
VCCP cluster	57.78	28.71	14.63
QEMU cluster	57.41	28.7	14.59
Overheads	0.37	0.01	0.04
Overheads %	0.64%	0.03%	0.27%

Kernel CG	*2 nodes*	*4 nodes*	8 nodes
VCCP cluster	91.02	128.68	110.45
QEMU cluster	90.37	124.57	109.74
Overheads	0.65	4.11	0.71
Overheads %	0.72%	3.30%	0.65%

Kernel IS	*2 nodes*	*4 nodes*	8 nodes
VCCP cluster	2.48	1.73	1.46
QEMU cluster	2.45	1.69	1.42
Overheads	0.03	0.04	0.04
Overheads %	1.22%	2.37%	2.82%

Kernel MG	*2 nodes*	*4 nodes*	8 nodes
VCCP cluster	7.88	17.05	21.29
QEMU cluster	7.7	16.87	21.17
Overheads	0.18	0.18	0.12
Overheads %	2.34%	1.07%	0.57%

Overheads

Kernel EP	*2 nodes*	*4 nodes*	8 nodes
VCCP cluster	57.78	28.71	14.63
QEMU cluster	57.41	28.7	14.59
Overheads	[illegible]	[illegible]	[illegible]
Overheads %	0.[illegible]	[illegible]	[illegible]

Kernel CG	*2 nodes*	*4 nodes*	8 nodes
VCCP cluster	91.02	128.68	110.45
QEMU cluster	90.37	124.57	109.74
Overheads	[illegible]	4.11	0.71
Overheads %	[illegible]	3.30%	0.65%

Kernel IS	*2 n*[illegible]	[illegible]	[illegible]
VCCP cluster	[illegible]	[illegible]	[illegible]
QEMU cluster	[illegible]	[illegible]	[illegible]
Overheads	[illegible]	[illegible]	[illegible]
Overheads %	1.[illegible]	[illegible]	[illegible]

[illegible]	[illegible]	*4 nodes*	8 nodes
[illegible]	[illegible]	17.05	21.29
[illegible]	[illegible]	16.87	21.17
[illegible]	[illegible]	0.18	0.12
[illegible]	[illegible]	1.07%	0.57%

<u>VCCP v.s. Qemu cluster</u>

Average 1.33 %

Minimum 0.03% (EP 4 Nodes),

Maximum 3.30% (CG 4 Nodes)

Overheads (cont.)

Kernel EP	*2 nodes*	*4 nodes*	8 nodes
VCCP cluster	57.78	28.71	14.63
REAL cluster	51.39	25.73	12.92
Overheads	6.39	2.98	1.71
Overheads %	12.43%	11.58%	13.24%

- VCCP v.s. Qemu cluster
  - Average 1.33 %
  - Minimum 0.03% (EP 4 Nodes), Maximum 3.30% (CG 4 Nodes)
- VCCP v.s. Real cluster
  - Average 12.41 % (EP)
  - Minimum 11.58% ,(EP) Maximum 13.24% (EP)

Overheads (cont.)

Kernel EP	*2 nodes*	*4 nodes*	8 nodes
VCCP cluster	57.78	28.71	14.63
REAL cluster	51.39	25.73	12.92

- VCCP v.s. Q
  - Average 1.3
  - Minimum 0.

  3.30% (CG 4

l cluster

1 % (EP)

58% ,(EP)

24% (EP)

VCCP v.s. Real cluster

Average 12.41 %
Minimum 11.58% ,

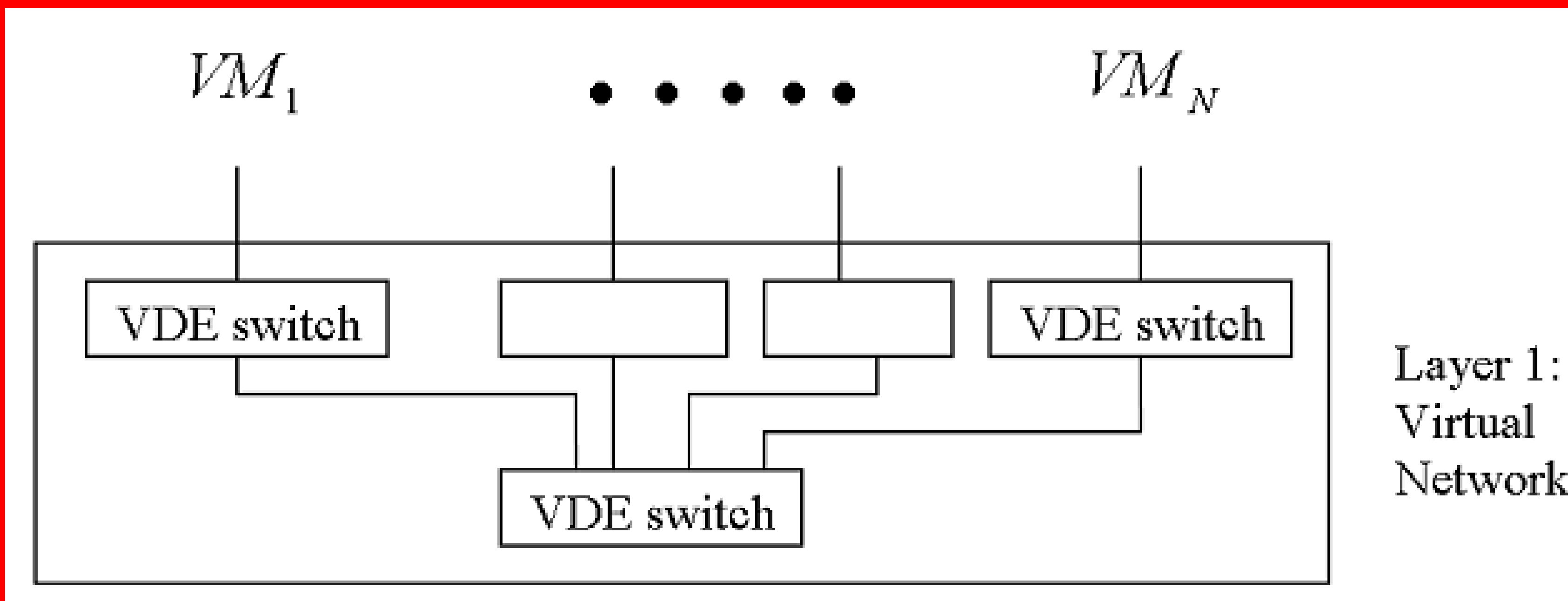
Maximum 13.24%
(No hardware supports)

Overheads

Virtual Network Problems:

1.

2.



Kernel CG	2 nodes	4 nodes	8 nodes
Overheads %	2.34%	1.07%	0.57%

Ke			
Overheads %	1.22%	2.37%	2.82%

Overheads

Virtual Network Problems:

1. VM and VDE compete for resources
2. Hot spot on central VDE switch

Kernel CG	*2 nodes*	*4 nodes*	8 nodes
VCCP cluster	91.02	128.68	110.45
QEMU cluster	90.37	124.57	109.74
Overheads	0.65	4.11	0.71
Overheads %	0.72%	3.30%	0.65%

Kernel IS	*2 nodes*	*4 nodes*	8 nodes
VCCP cluster	2.48	1.73	1.46
QEMU cluster	2.45	1.69	1.42
Overheads	0.03	0.04	0.04
Overheads %	1.22%	2.37%	2.82%

Kernel MG	*2 nodes*	*4 nodes*	8 nodes
VCCP cluster	7.88	17.05	21.29
QEMU cluster	7.7	16.87	21.17
Overheads	0.18	0.18	0.12
Overheads %	2.34%	1.07%	0.57%

Checkpointing Performance

Checkpoint time = Flush time

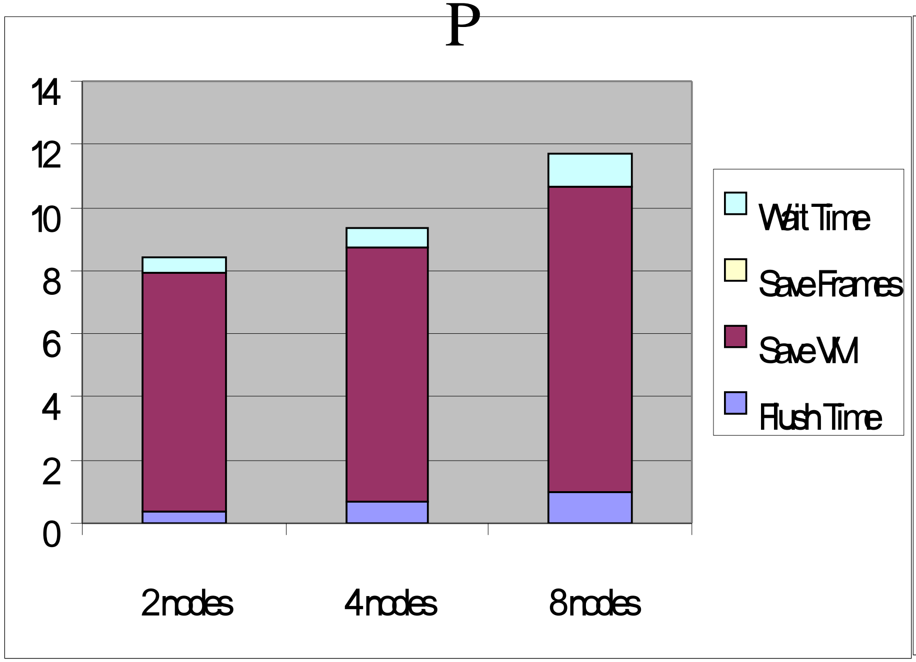
+ Time to save VM State

+ Save messages to disks

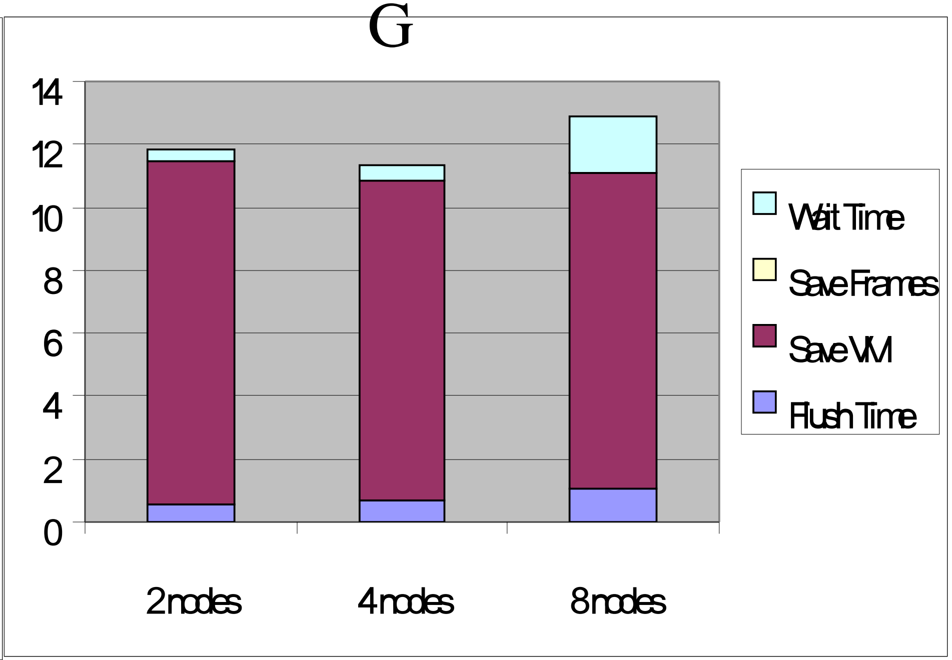
+ Wait time

Checkpointing Performance

EP

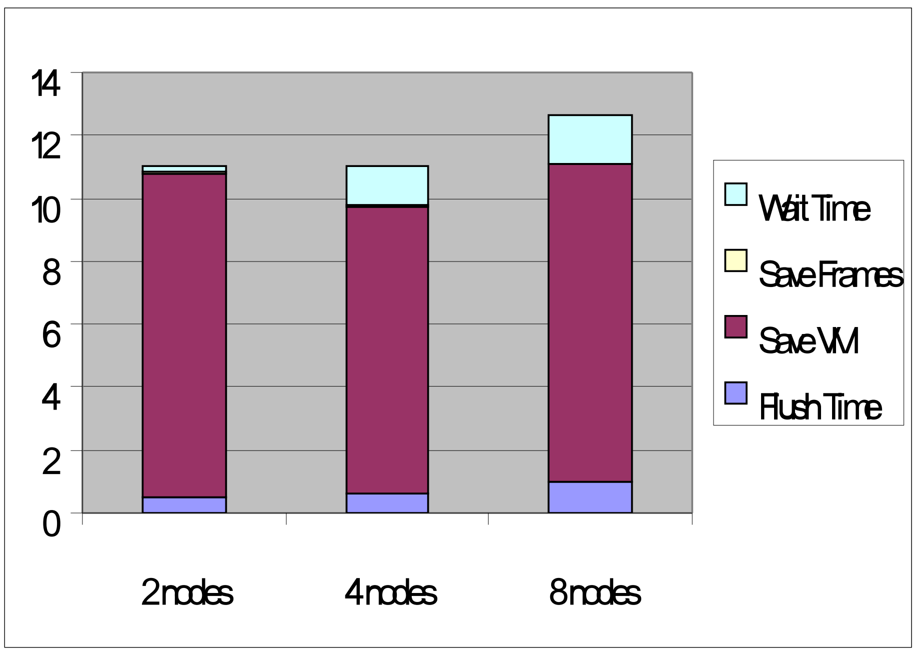


CG

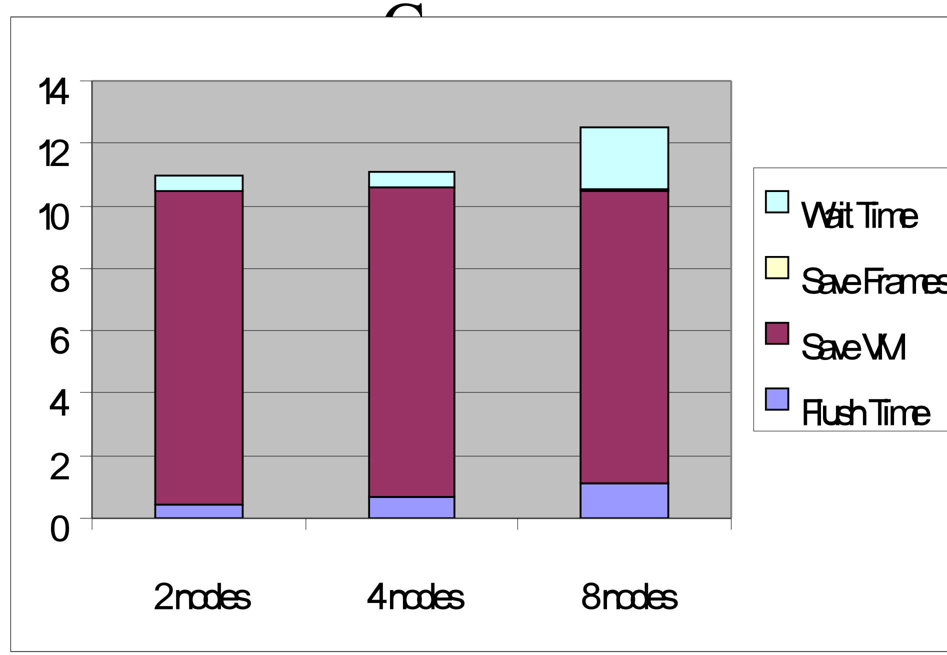


Checkpointing Performance

IS



M



Progress

- We have developed a transparent hypervisor-based coordinated checkpointing system for Virtual Clusters
- **Transparency** is achieved
- **Efficiency** requires further investigations
  - ***Low*** normal execution overhead
  - ***High*** communication overhead
  - ***High*** local checkpointing overhead

Future Works

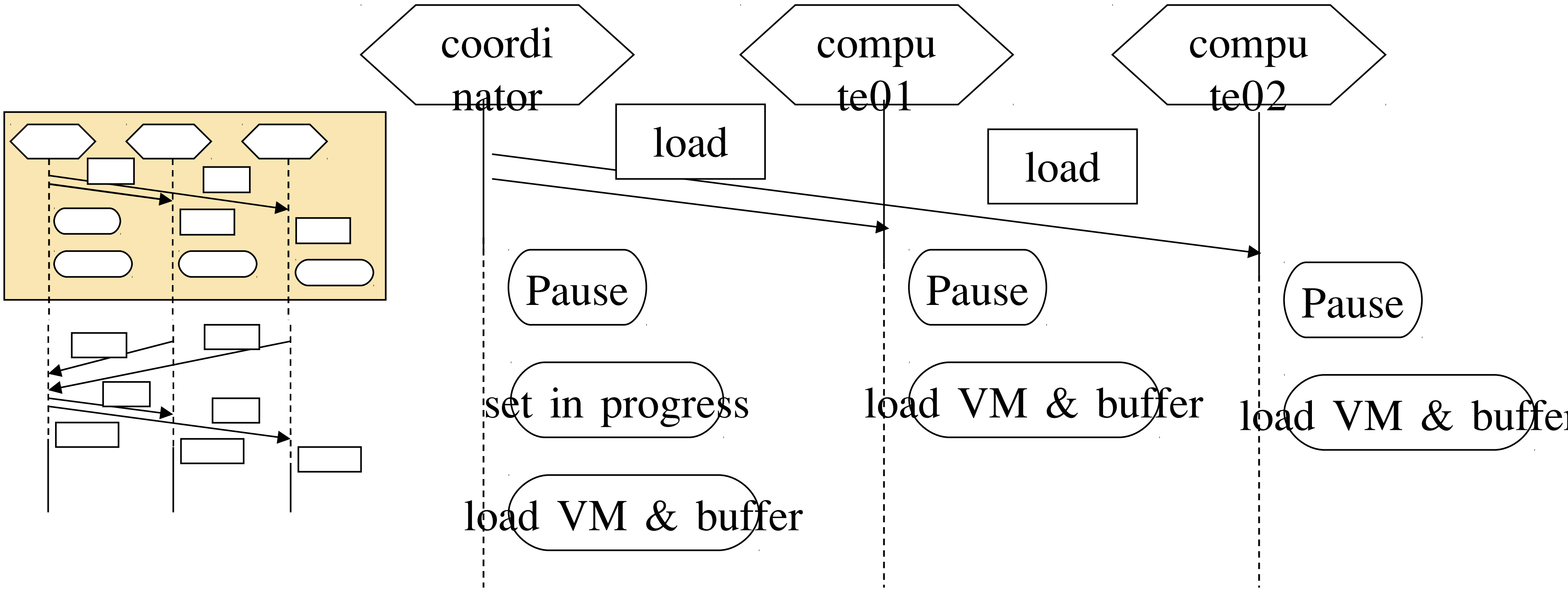
- Study and develop new virtual network
  - Direct TCP connections among VM
  - Virtio
- We are investigating low-latency local VM checkpointing mechanisms
  - Multi-threaded mechanism to save VM state
  - Diskless checkpointing

THANK YOU and QUESTIONS?

Backup

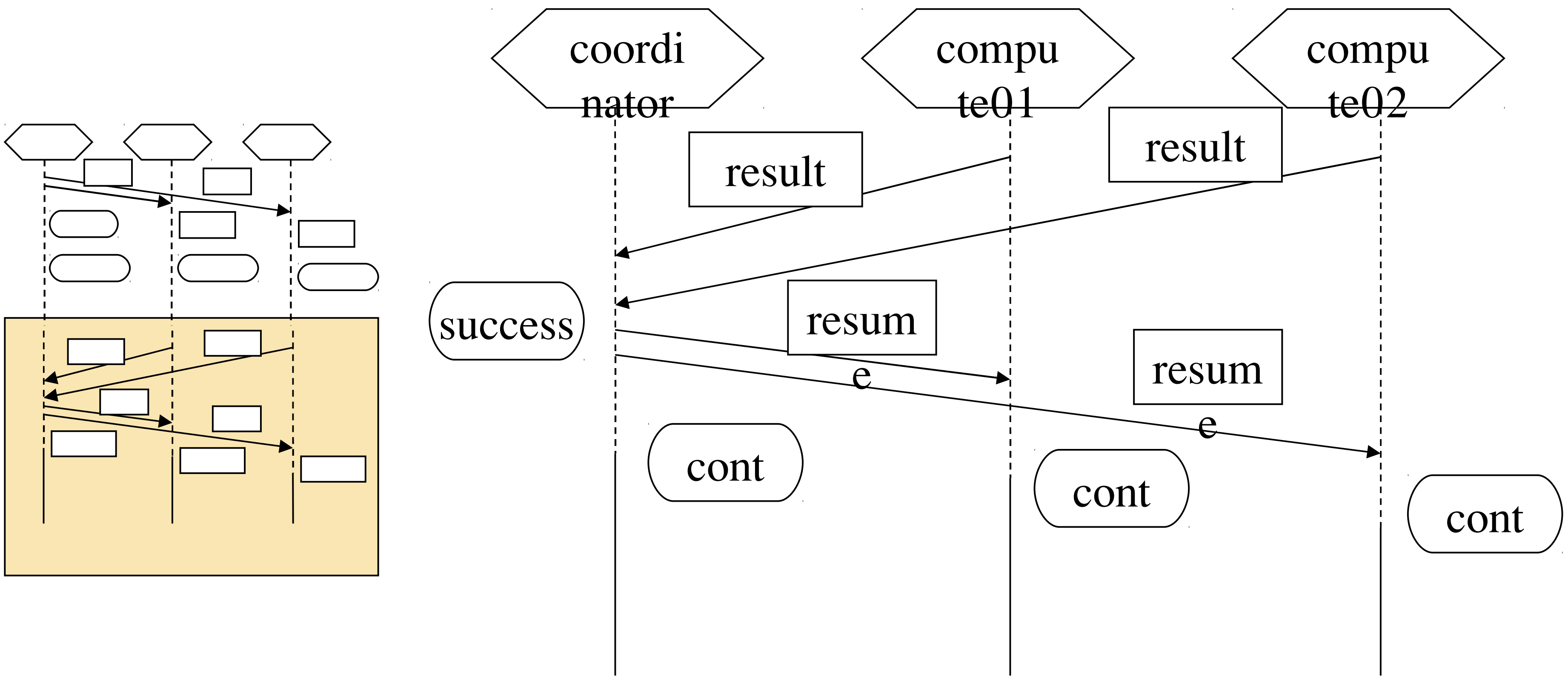
VCCP

VCCP recovery protocol



VCCP

VCCP recovery protocol

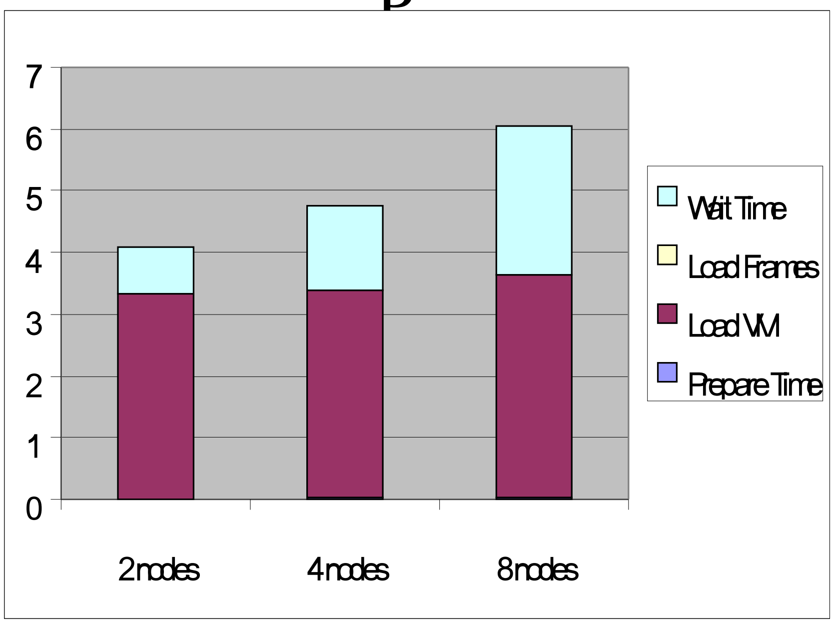


Recovery Performance

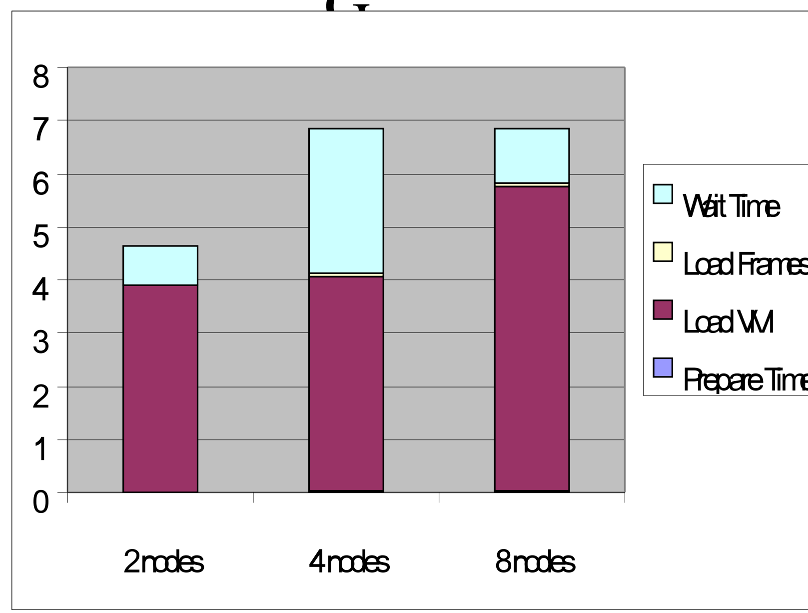
Restoration time = coordinate time + load coordinator's vm time + load coordinator's frame time + wait time

Recovery Performance

E
P

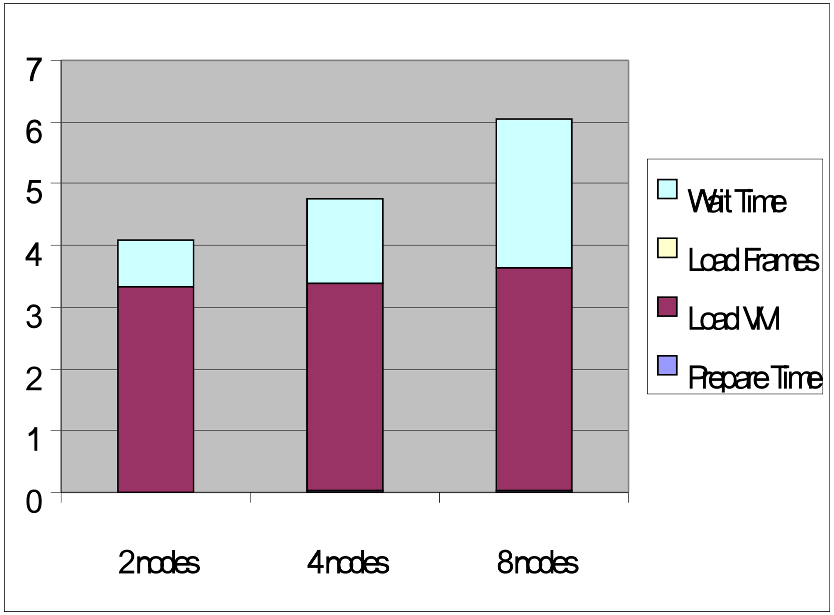


C
G



Recovery Performance

IS



MG