Cooperative VM Migration for a virtualized HPC Cluster with VMM-bypass I/O devices

Ryousei Takano, Hidemoto Nakada, Takahiro Hirofuchi,
Yoshio Tanaka, and Tomohiro Kudoh

Information Technology Research Institute,
National Institute of Advanced Industrial Science and Technology (AIST), Japan

IEEE eScience 2012, Oct. 11 2012, Chicago

Background

- HPC cloud is a promising e-Science platform.
  - HPC users begin to take an interest in Cloud computing, e.g., Amazon EC2 Cluster Compute Instances.
- Virtualization is a key technology.
  - Pro: It makes migration of computing elements easy.
    - VM migration is useful for achieving fault tolerance, server consolidation, etc.
  - Con: It introduces a large overhead, spoiling I/O performance.
    - VMM-bypass I/O technologies, e.g., PCI passthrough and SR-IOV, can significantly mitigate the overhead.

VMM-bypass I/O makes it impossible to migrate a VM.

Contribution

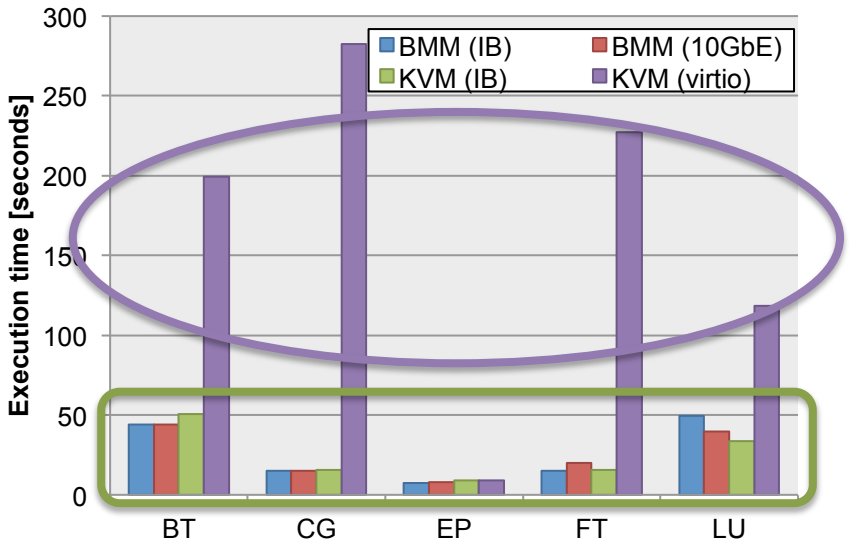
- Goal:
  - To realize VM migration and checkpoint/restart on a virtualized cluster with VMM-bypass I/O devices.
    - E.g., VM migration on an Infiniband cluster
- Contributions:
  - We propose cooperative VM migration based on the Symbiotic Virtualization (SymVirt) mechanism.
  - We demonstrate reactive/proactive fault tolerant (FT) systems.
  - We show postcopy migration helps to reduce the service downtime in the proactive FT system.

Agenda

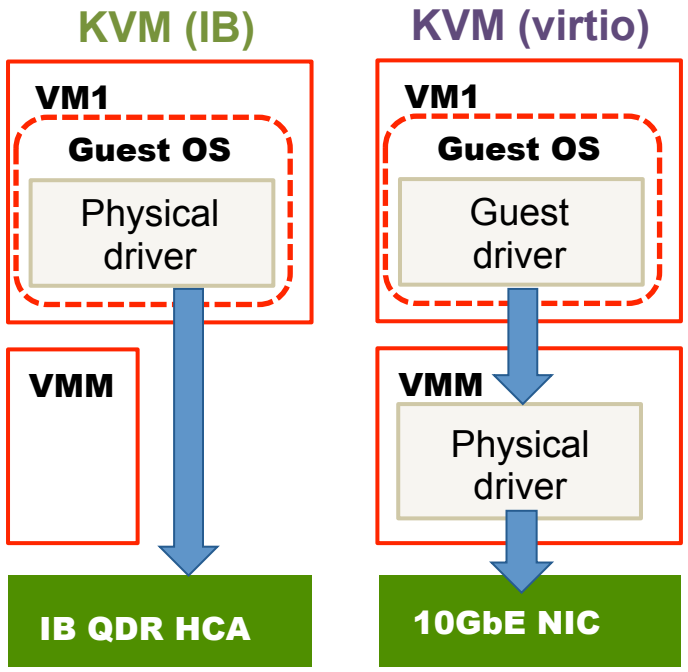
- Background and Motivation
- SymVirt: Symbiotic Virtualization Mechanism
- Experiment
- Related Work
- Conclusion and Future Work

Motivating Observation

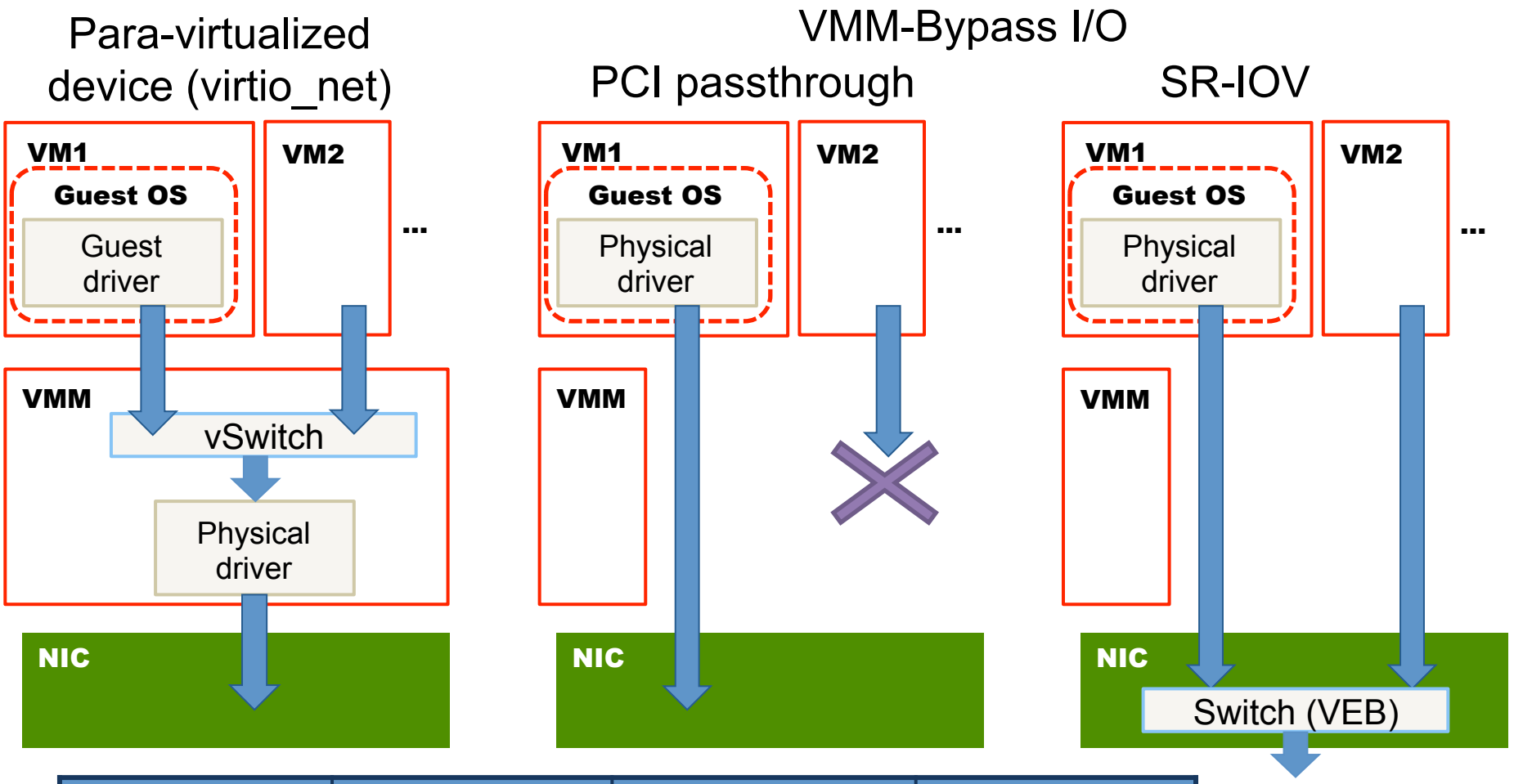
- Performance evaluation of HPC cloud
  - **(Para-)virtualized I/O** incurs a large overhead.
  - **PCI passthrough** significantly mitigate the overhead.



The overhead of I/O virtualization on the NAS Parallel Benchmarks 3.3.1 class C, 64 processes.



BMM: Bare Metal Machine



	Para-virt device	PCI passthrough	SR-IOV
Performance	✖	✔	✔
Device sharing	✔	✖	✔
VM migration	✔	✖	✖

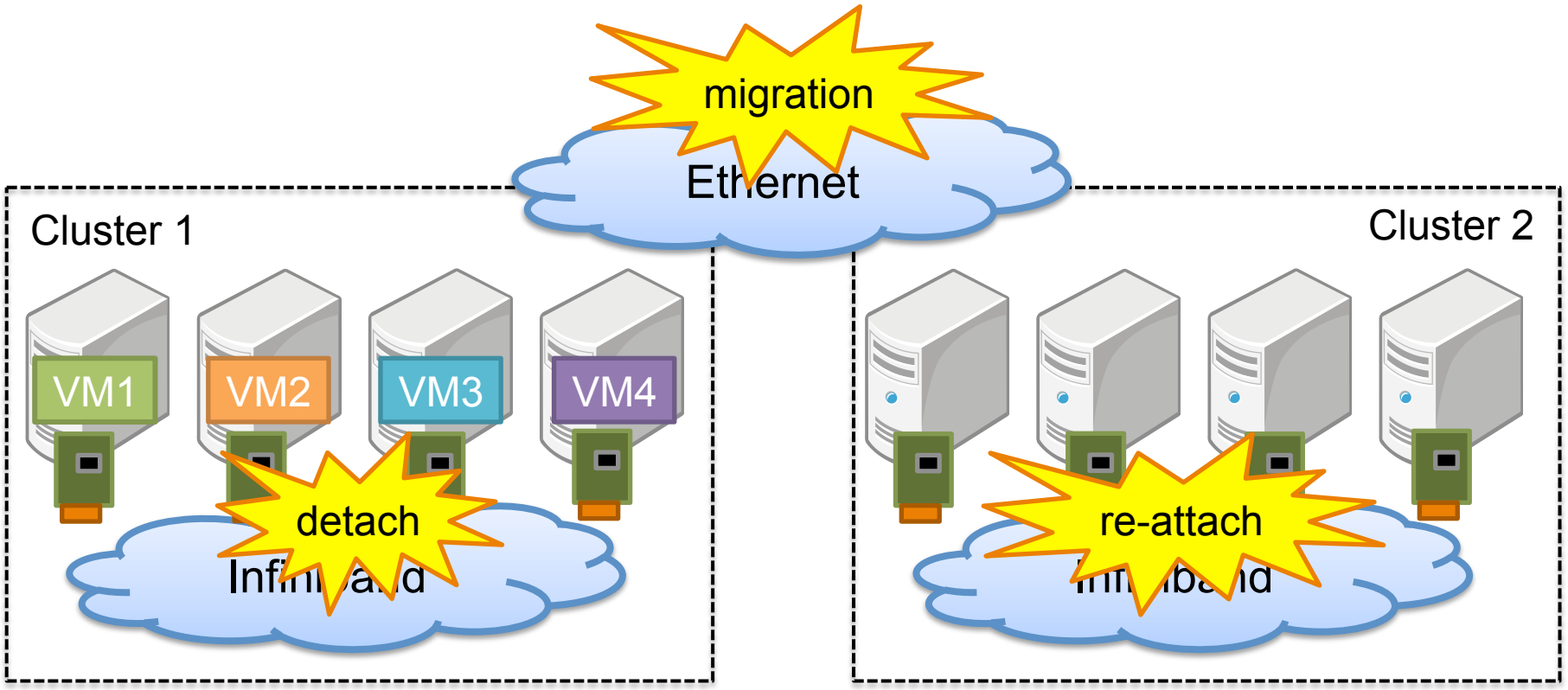
We address this issue!

Problem

VMM-bypass I/O technologies make VM migration and checkpoint/restart impossible.

1. VMM does not know the time when VMM-bypass I/O devices are detached safely.
   - To perform such migration without losing in-flight data, packet transmission to/from the VM should be stopped prior to detaching.
   - With a VMM, it is hard to know the communication status of an application inside the VM, especially if VMM-bypass I/O devices are used.
2. VMM cannot migrate the state of VMM-bypass I/O devices from the source to the destination.
   - With Infiniband, Local ID, QueuePair Numbers, etc.

Goal

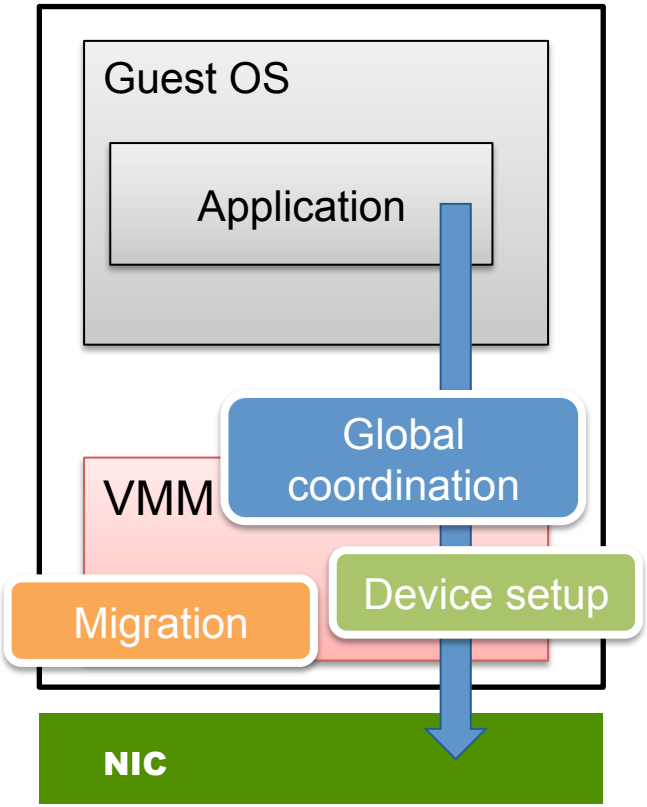


We need a mechanism of combining VM migration and PCI device hot-plugging.

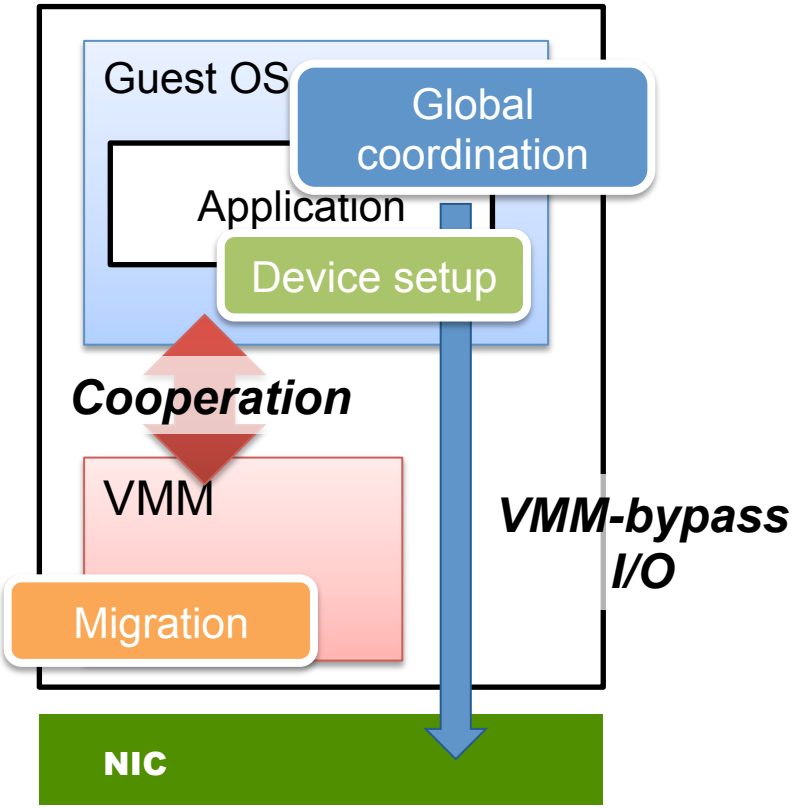
SymVirt: Symbiotic Virtualization

Cooperative VM Migration

Existing VM Migration
(Black-box approach)
Pro: portability

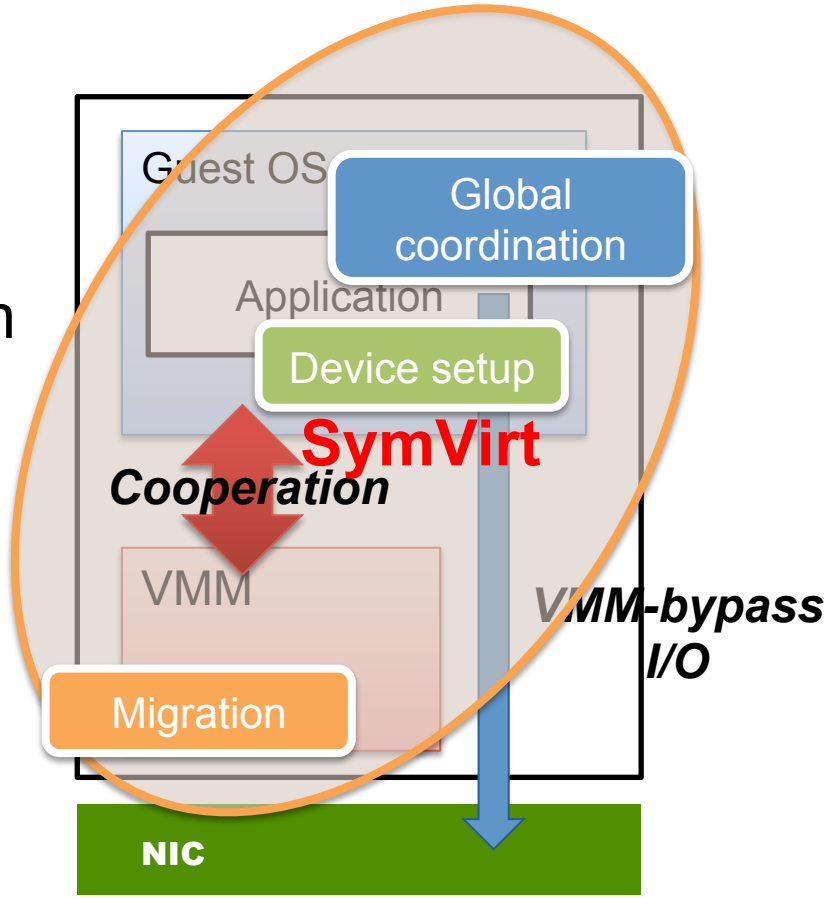


Cooperative VM Migration
(Gray-box approach)
Pro: performance

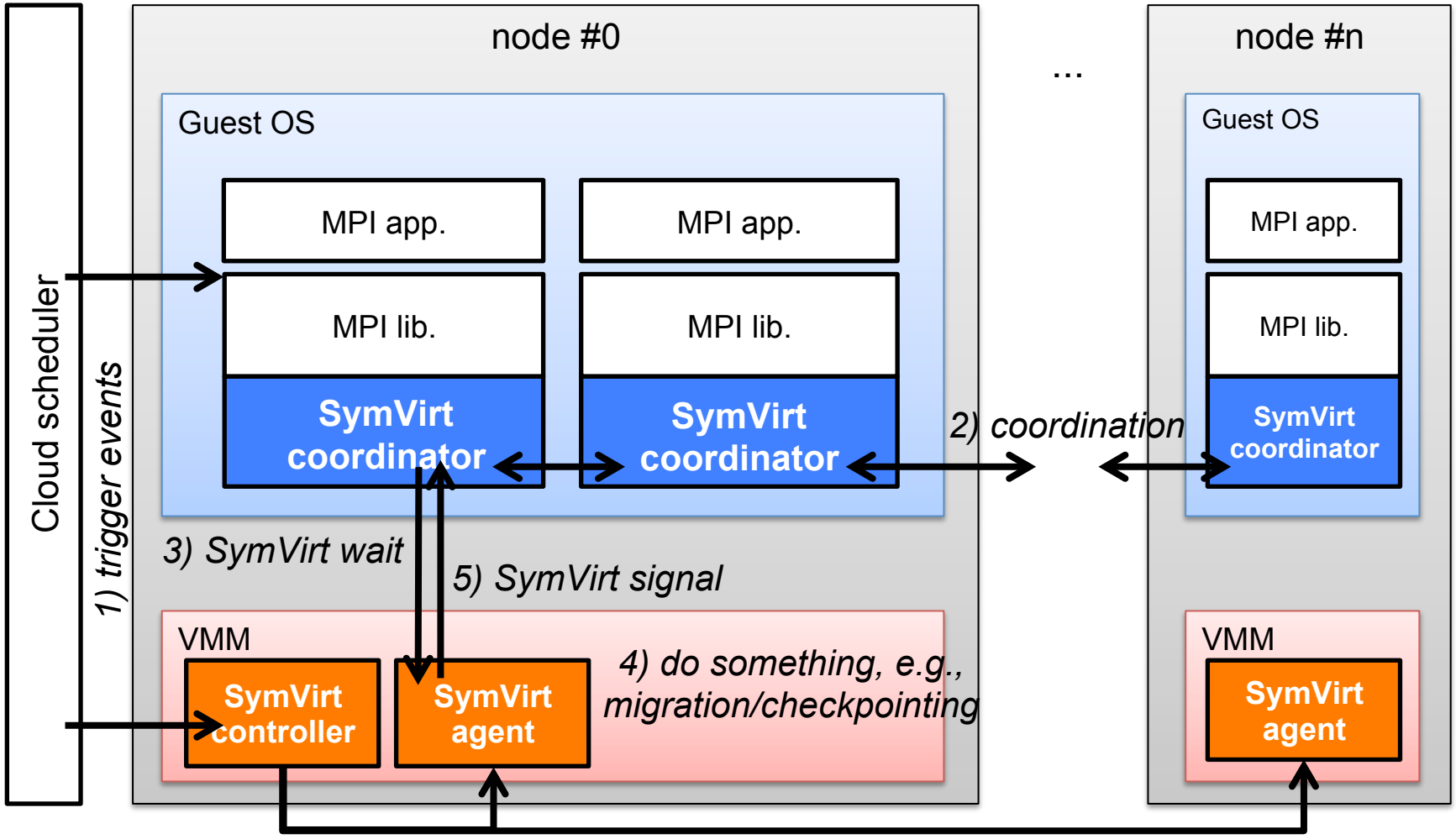


SymVirt: Symbiotic Virtualization

- We focus on MPI programs.
- We design and implement a symbiotic virtualization (SymVirt) mechanism.
  - It is a cross-layer mechanism between a VMM and an MPI runtime system.

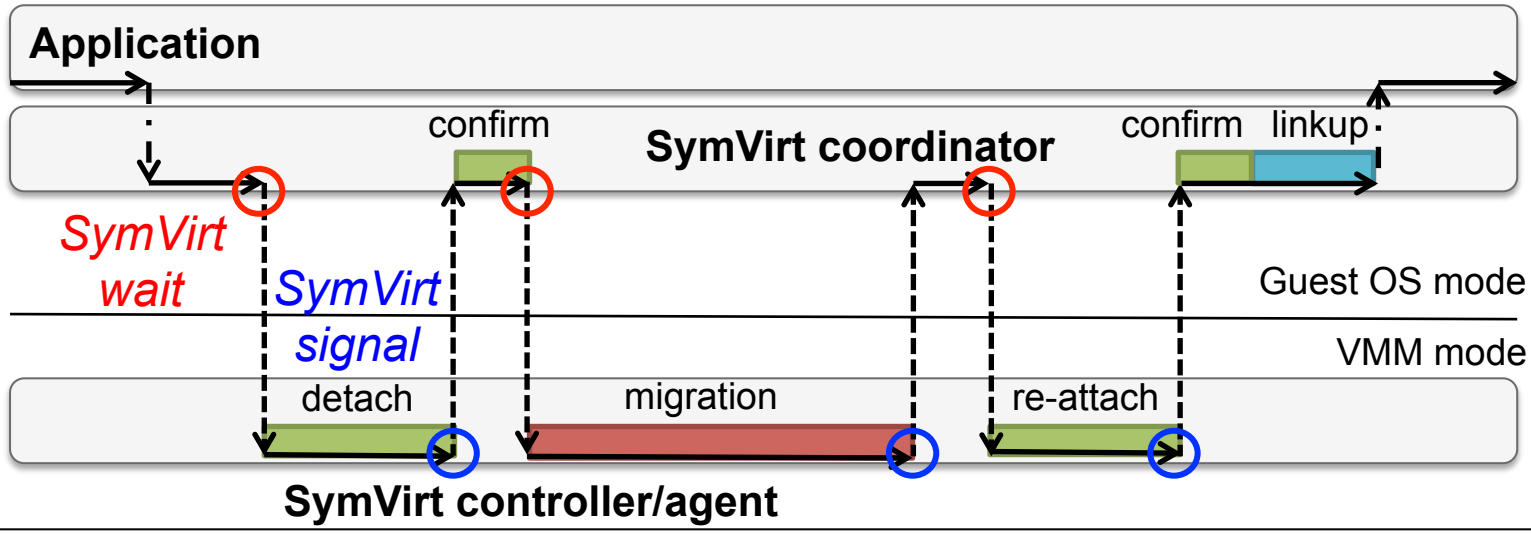


SymVirt: Overview



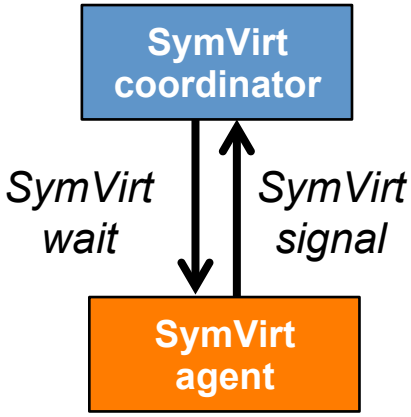
SymVirt wait and signal calls

- SymVirt provides a simple Guest OS-to-VMM communication mechanism.
- SymVirt coordinator issues a SymVirt wait call, the guest OS is blocked until a SymVirt signal call is issued.
- In the meantime, SymVirt agent controls the VM via a VMM monitor interface.



SymVirt: Implementation

- We implemented SymVirt on top of QEMU/KVM and the Open MPI system.
- User application and the MPI runtime system can work without any modifications.
- QEMU/KVM is slightly modified for supporting SymVirt wait and signal calls.
  - A SymVirt wait call is implemented by using a *VMCALL* Intel VT-x instruction.
  - A SymVirt signal call is implemented as a new QEMU/KVM monitor command.



SymVirt: Implementation (cont'd)

- SymVirt coordinator is heavily relied on the Open MPI checkpoint/restart (C/R) framework.
  - Global coordination of SymVirt is the same as a coordination protocol for MPI programs.
  - SymVirt executes VM-level migration or C/R instead of process-level C/R using the BLCR system.
  - SymVirt does not need to take care of changing LIDs and QPNs after a migration, because Open MPI's BTL modules are re-constructed and connections are re-established at continue or restart phases.

BTL: Point-to-Point Byte Transfer Layer

SymVirt: Implementation (cont'd)

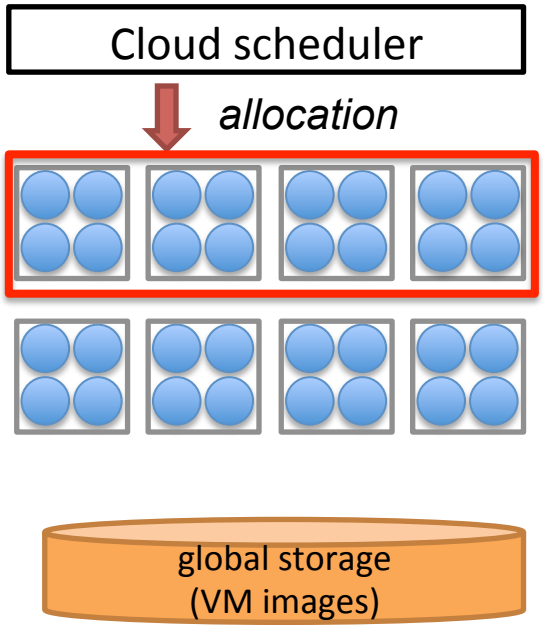
- SymVirt controller and agent are written in Python.

```
import symvirt
agent_list = [migrate_from]
ctl = symvirt.Controller(agent_list)

# device detach
ctl.wait_all()
kwargs = {'tag':'vf0'}
ctl.device_detach(**kwargs)
ctl.signal()

# vm migration
ctl.wait_all()
kwargs = {'postcopy':True, 'uri':'tcp:%s:%d'
          % (migrate_to[0], migrate_port)}
ctl.migrate(**kwargs)
ctl.remove_agent(migrate_from)
```

```
# device attach
ctl.append_agent(migrate_to)
ctl.wait_all()
kwargs = {'pci_id':'04:00.0',
'tag':'vf0'}
ctl.device_attach(**kwargs)
ctl.signal()

ctl.close()
```

SymPFT: Proactive FT system

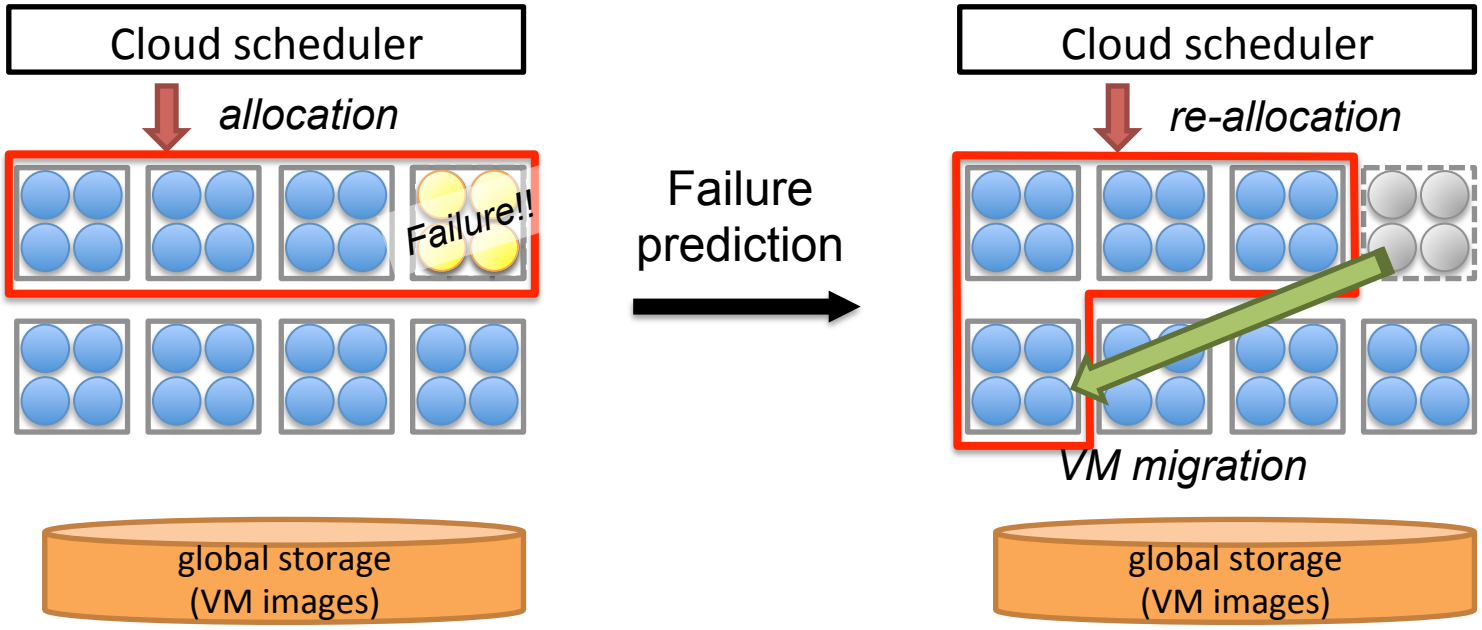
- A VM-level fault tolerant (FT) system is a use case of SymVirt.



User requires a virtualized cluster consists of 4 nodes (16 CPUs).

SymPFT: Proactive FT system

- A VM-level fault tolerant (FT) system is a use case of SymVirt.
- A VM is migrated from a “unhealthy” node to a “healthy” node before the node crashes.



Experiment

Experiment

- The overhead of SymPFT
  - We used 8 VMs on an Infiniband cluster.
  - We migrated a VM once during a benchmark execution.
- Two benchmark programs written in MPI
  - memtest: a simple memory intensive benchmark
  - NAS Parallel Benchmarks (NPB) version 3.3.1
- Overhead reduction using postcopy migration

Experimental setting

We used a 16 node Infiniband cluster, which is a part of the AIST Green Cloud.

Blade server（Dell PowerEdge M610）	
CPU	Intel quad-core Xeon E5540/2.53GHz x2
Chipset	Intel 5520
Memory	48 GB DDR3
InfiniBand	Mellanox ConnectX (MT26428)

Blade switch	
InfiniBand	Mellanox M3601Q (QDR 16 ports)

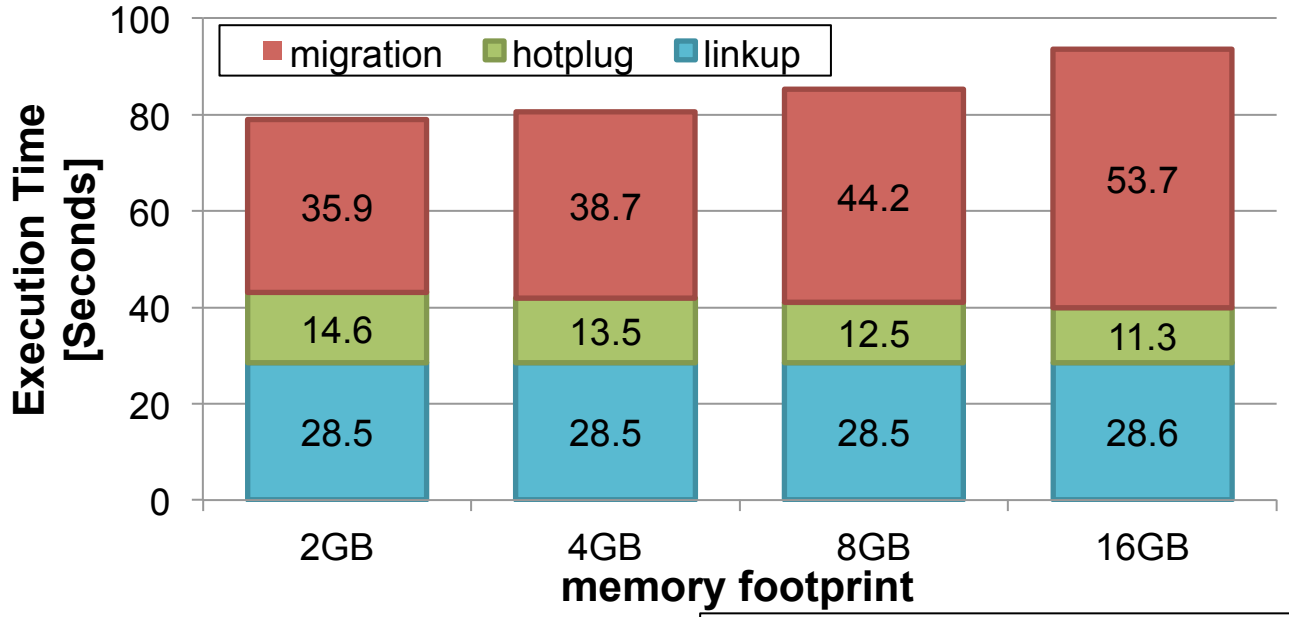
Host machine environment	
OS	Debian 7.0
Linux kernel	3.2.18
QEMU/KVM	1.1-rc3
MPI	Open MPI 1.6
OFED	1.5.4.1
Compiler	gcc/gfortran 4.4.6

VM environment	
VCPU	8
Memory	20 GB

Only 1 VM runs on 1 host, and an IB HCA is assigned to the VM by using PCI passthrough. ➔

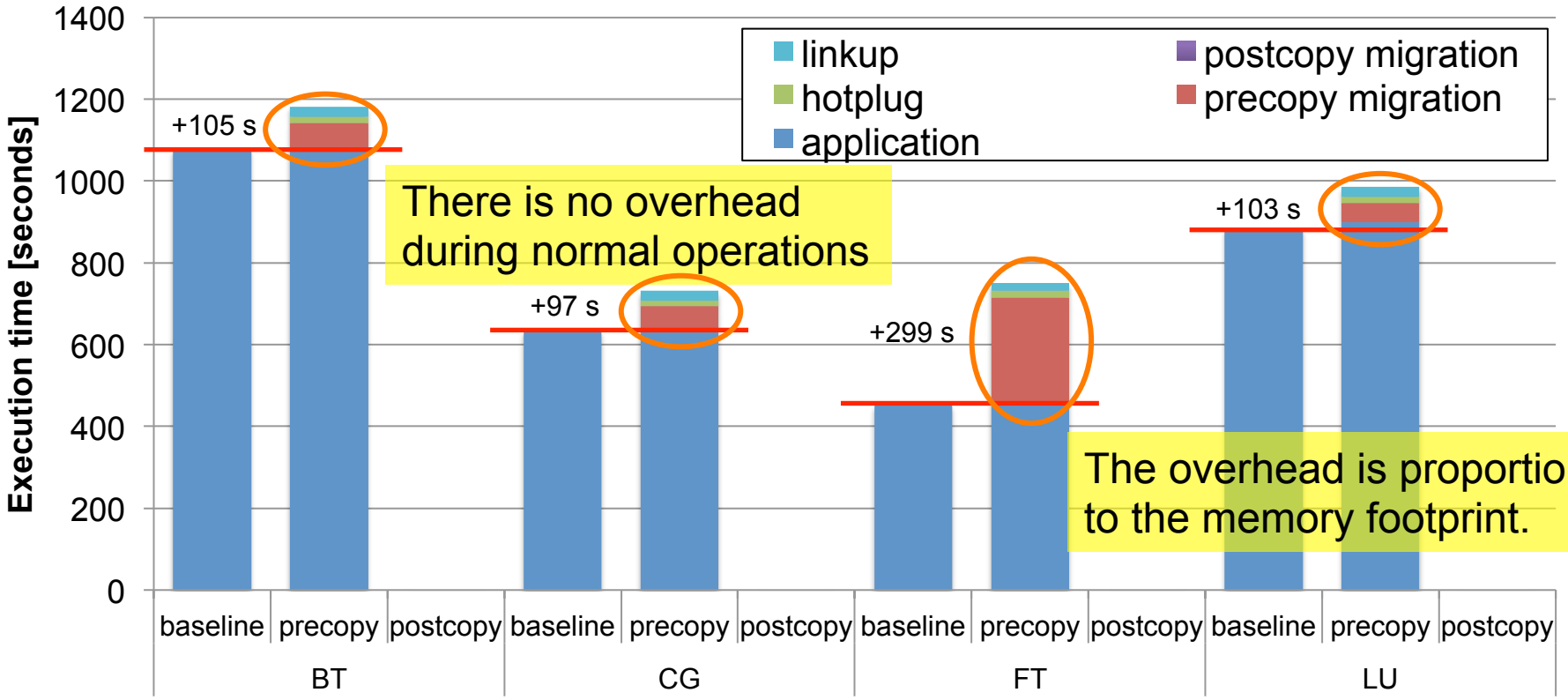
Result: memtest

- The migration time is dependent on the memory footprint.
  - The migration throughput is less than 3 Gbps.
- Both hotplug and link-up times are approximately constant.
  - The link-up time is not a negligible overhead. c.f., Ethernet



This result does not include our proceeding.

Result: NAS Parallel Benchmarks

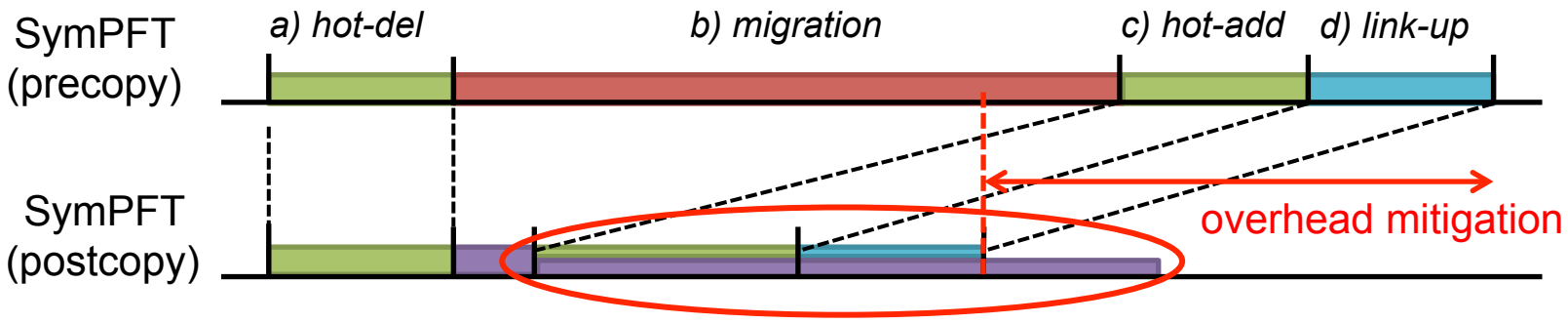


Transferred Memory Size during VM Migration [MB]

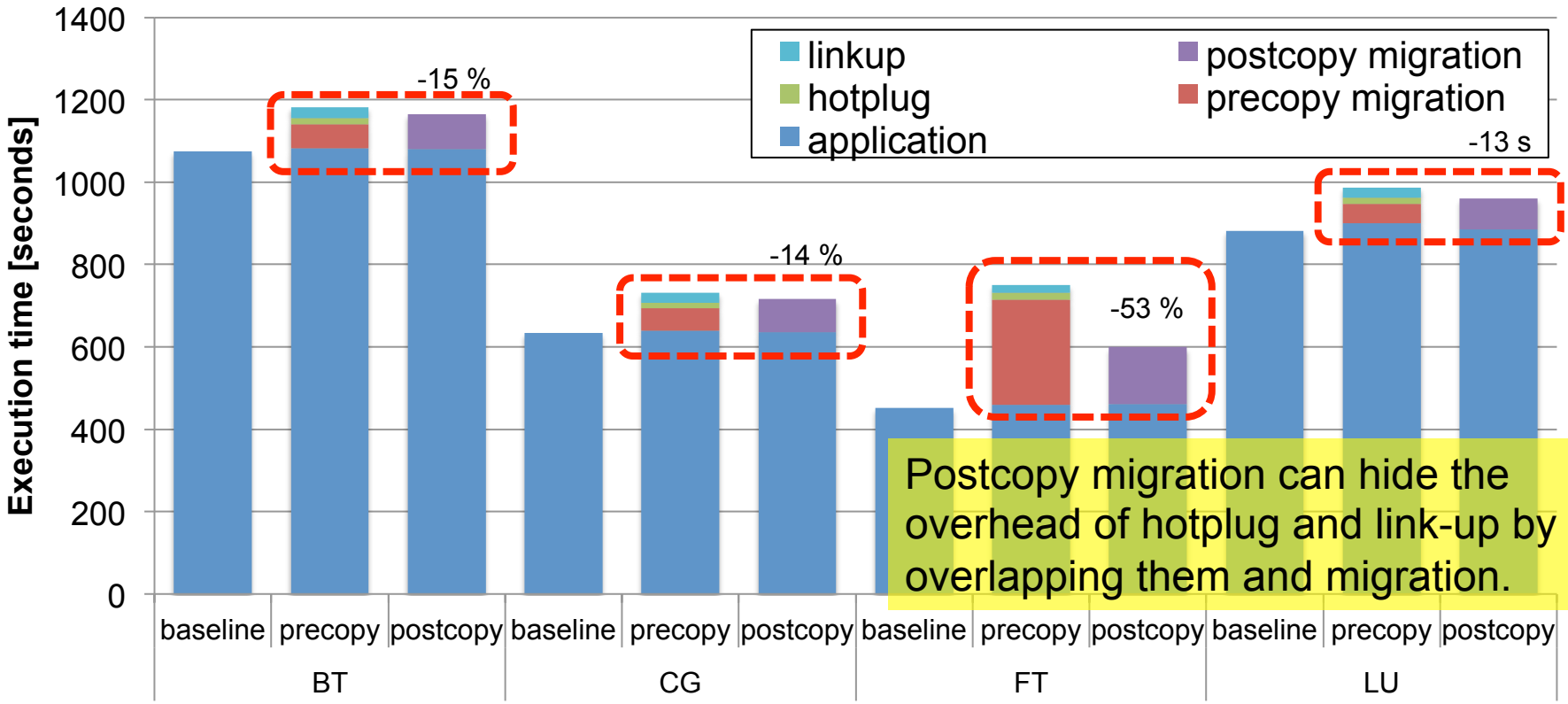
BT	CG	FT	LU
4417	3394	15678	2348

Integration with postcopy migration

- In contrast to precopy migration, postcopy migration transfers memory pages on demand after the execution node is switched to the destination.
- Postcopy migration can hide the overhead of the hot-add and link-up times by overlapping them and migration.
- We used our postcopy migration implementation for QEMU/KVM, *Yabusame*.



Result: Effect of postcopy migration



Postcopy migration can hide the overhead of hotplug and link-up by overlapping them and migration.

Transferred Memory Size during VM Migration [MB]

BT	CG	FT	LU
4417	3394	15678	2348

Related Work

- Some VM-level reactive and proactive FT systems have been proposed for HPC systems.
  - E.g., **VNsnap**: a distributed snapshots of VMs
    - The coordination is executed by snooping the traffic of a software switch outside the VMs.
  - They do not support VMM-bypass I/O devices.
- **Mercury**: a self-virtualization technique
  - An OS can turn virtualization on and off on demand.
  - It lacks a coordination mechanism among distributed VMM.
- **SymCall**: an upcall mechanism from a VMM to the guest OS, using a nested VM Exit call
  - SymVirt is a simple hypercall mechanism from a guest OS to the VMM, assuming it works in cooperation with a cloud scheduler.

Conclusion and Future Work

Conclusion

- We have proposed a **cooperative VM migration** mechanism that enables us to migrate VMs with VMM-bypass I/O devices, using a simple Guest OS-to-VMM communication mechanism, called **SymVirt**.
- Using the proposed mechanism, we demonstrated a **proactive FT system** in a virtualized Infiniband cluster.
- We also confirmed that **postcopy migration** helps to reduce the downtime in the proactive FT system.

- SymVirt can be useful for not only fault tolerant but also load balancing and server consolidation.

Future Work

- Interconnect-transparent migration, called *"Ninja migration"*
  - We have submitted another conference paper.
- Overhead mitigation of SymVirt
  - Very long link-up time problem
  - Better integration with postcopy migration
- A generic communication layer supporting cooperative VM migration
  - It is independent on an MPI runtime system.

Thanks for your attention!

科研費 KAKENHI

This work was partly supported by JSPS KAKENHI 24700040 and ARGO GRAPHICS, Inc.