ALICE O2 Presentation

Efficient Live Checkpointing Mechanisms for computation and memory-intensive VMs in a data center

Kasidit Chanchio
Vasabilab
Dept of Computer Science,
Faculty of Science and Technology,
Thammasat University
http://vasabilab.cs.tu.ac.th

Outline

- Introduction and problems
- Checkpointing mechanisms
- Our Proposal
  - Time-bound Live Checkpointing (TLC)
  - A Scalable Checkpointing Technique
- Conclusion and Future Works

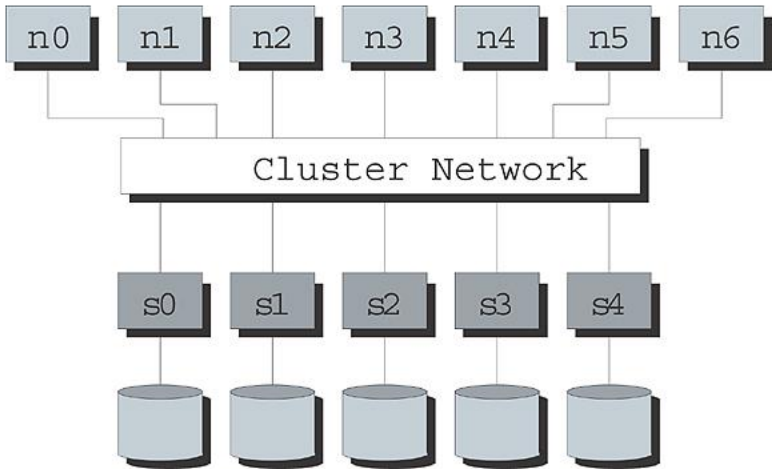
Introduction

vasabiLab
Virtualization Architecture and ScalABle Infrastructure Laboratory

- Today, applications require more CPUs and RAM
  - Big Data Analysis
  - Large Scale simulation
  - Scientific Computation
  - Legacy Applications, etc.
- Cloud computing has become a common platform for large-scale computations
  - Amazon offers VM with 8 vcpus and 68.4GiB Ram
  - Google offers VM with 8 vcpus and 52GB Ram
- Large-scale applications can have long exe time
  - In case of failures, users must restart apps from beginning

How do we handle server crashes?

- **Checkpointing:** The state of long running apps should be saved regularly so that the computation can be recovered from the last saved state if failures occur
- It usually take a **long time to save state** of CPU and memory-intensive apps
  - **Downtime** could also be high



- Parallel File System (PFS) can be a **bottleneck** and slowdown the entire system when saving state of multiple nodes simultaneously

From linux.sys-con.com/node/38285

What is Checkpointing?

vasabiLab
Virtualization Architecture and ScalABle Infrastructure Laboratory

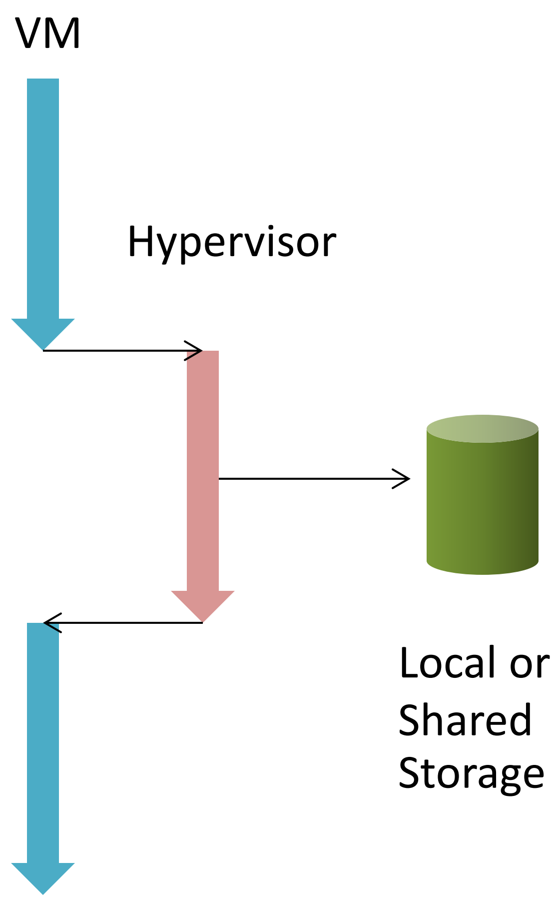
- Periodically Save Computation State to Persistent Storage for recovery if failures occur

Application-Level	Modify App	More works on development	Know exactly what to save
User Level	Link with Chkpt library	Depend on exe environments	Don't have to recompile app
OS-Level	Modify Kernel	Depend on Kernel version	Can reuse executable
VM-Level	Modify Hypervisor	Must handle all VM state	Transparent to Guest OS/App
Linux/Hardware			

VM Checkpointing

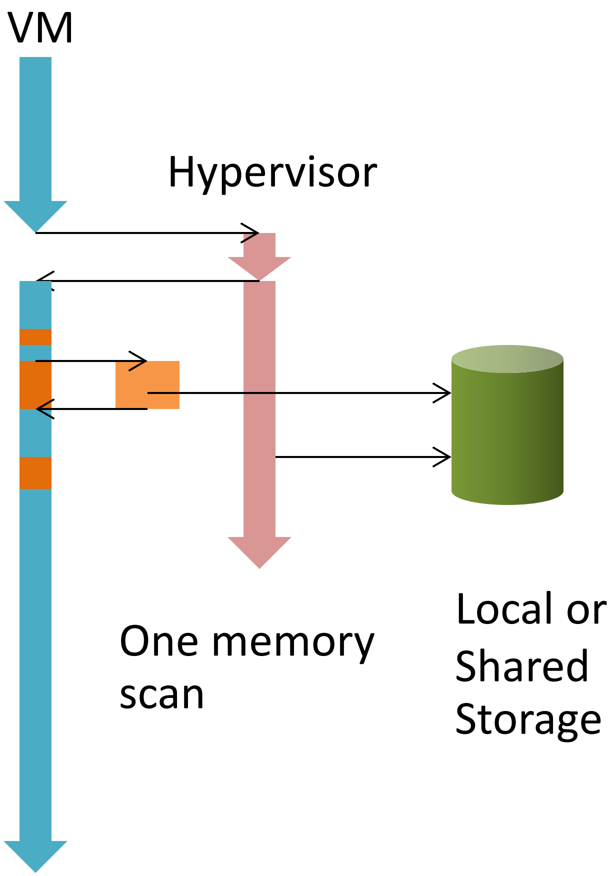
- Highly Transparent to Guest OS & Applications
- Save all apps and execution environments
- Techniques:
  – Stop & Save [kvm]
  – Copy on Write & Chkpt Thread [vmware ESXi]
  – Copy to Memory Buffer [TLC 2009]
  – Live replication to a backup host [Remus]
  – **Time-bound Live Checkpointing** [TLC]

1. Stop and Save



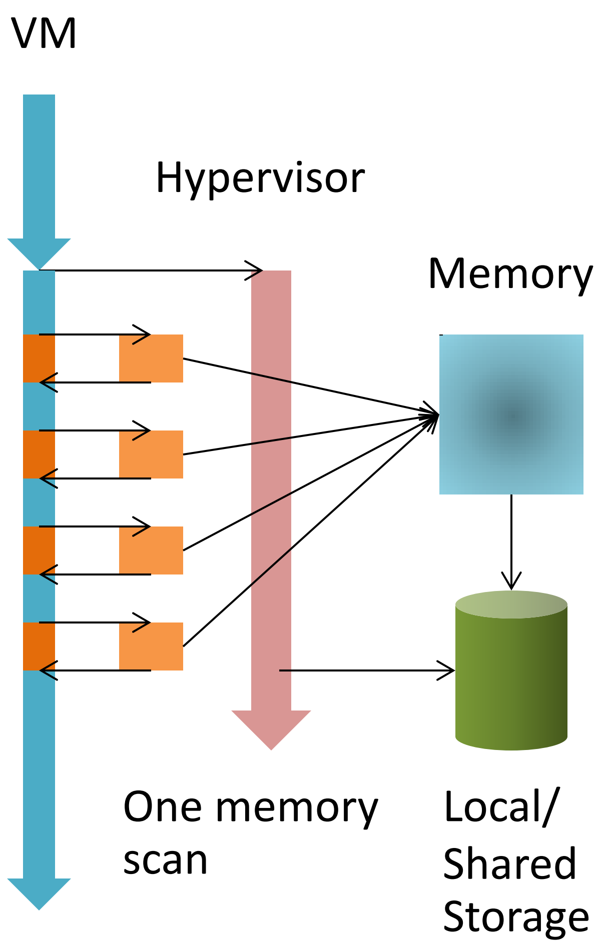
- Stop the VM to save state to disk
- Long Downtime and Checkpoint time
- Saving to shared storage is necessary if want to restore on a new host
- Saving to shared storage cause higher checkpoint time

2. Copy on Write



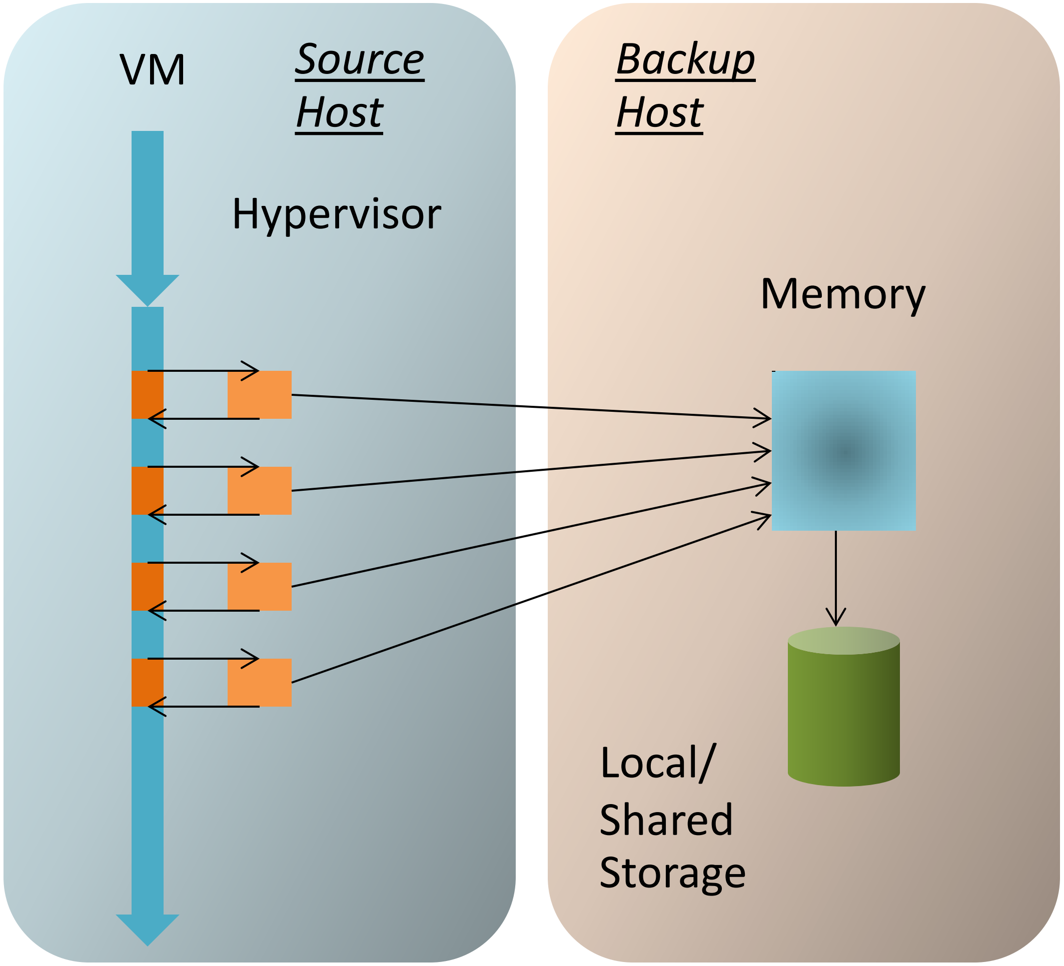
- Hypervisor create a thread to scan memory and save unmodified pages
- If VM modifies a page, hypervisor copy the original contents of that page to directly to disk
- Can cause high downtime if large number of pages are modified in a short period of time

3. Memory Buffer



- Hypervisor create a thread to scan memory and save unmodified pages
- Hypervisor stop VM to copy dirty pages to a memory buffer and write the buffer to disk later when checkpointing done
- Need large amount of memory

4. Replication



- Hypervisor stop VM periodically to copy and sync state information with a backup host
- Great for High Availability
- Need to reserve resource on a backup host for the VM throughout its lifetime

Time-bound Live Migration

- TLC is based on the Time-bound, Thread-based Live Migration (TLM) [CCgrid 2014]
- Basic Principles of TLM:
  - TLM finishes within a **bounded period of time**,i.e., one round of memory scan
  - Performs with **best efforts** to minimize downtime
  - Dynamically adjust VM computation speed to **reduce downtime** by balancing **dirty page generation rate** and **available data transfer bandwidth**

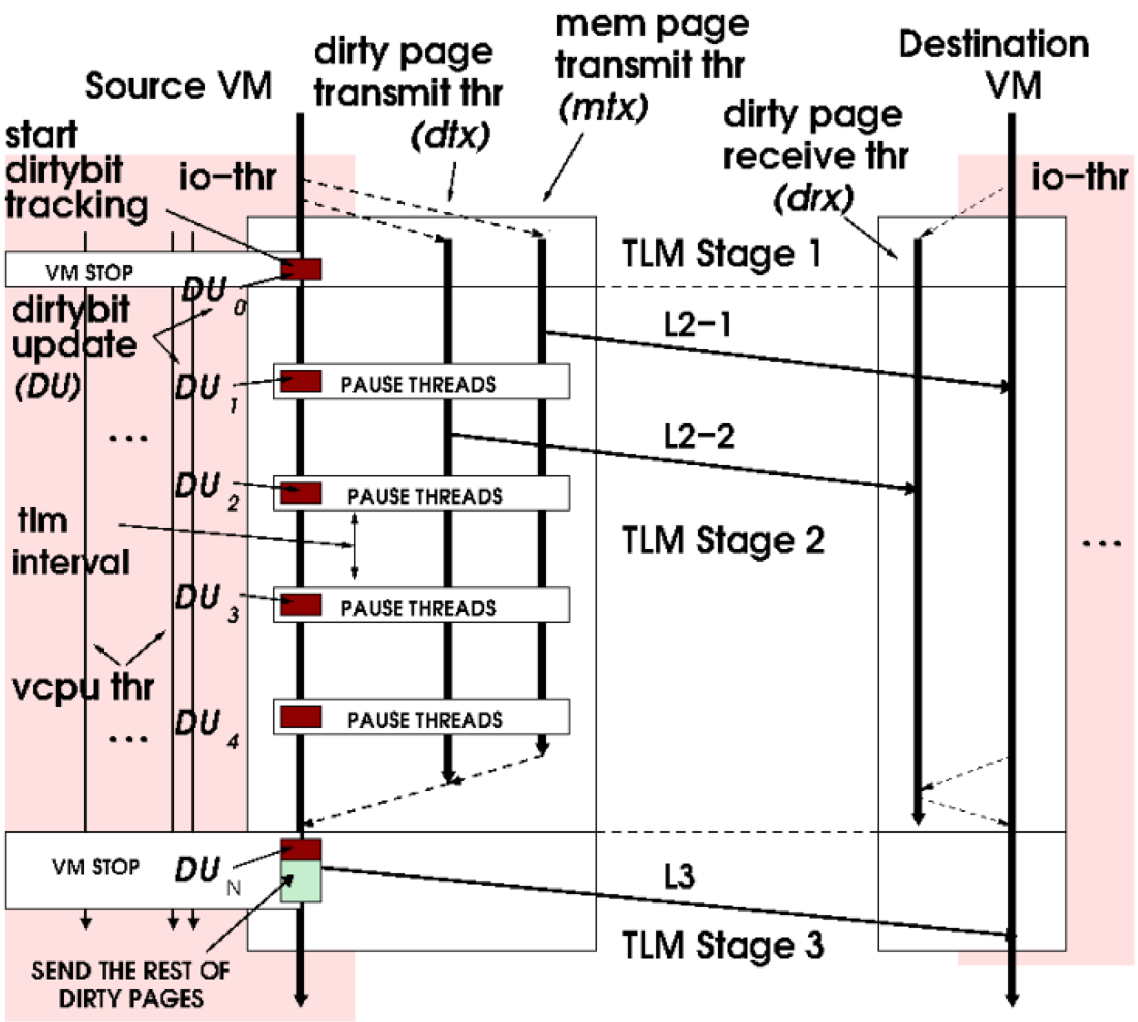
TLM Design

VM State Transfer

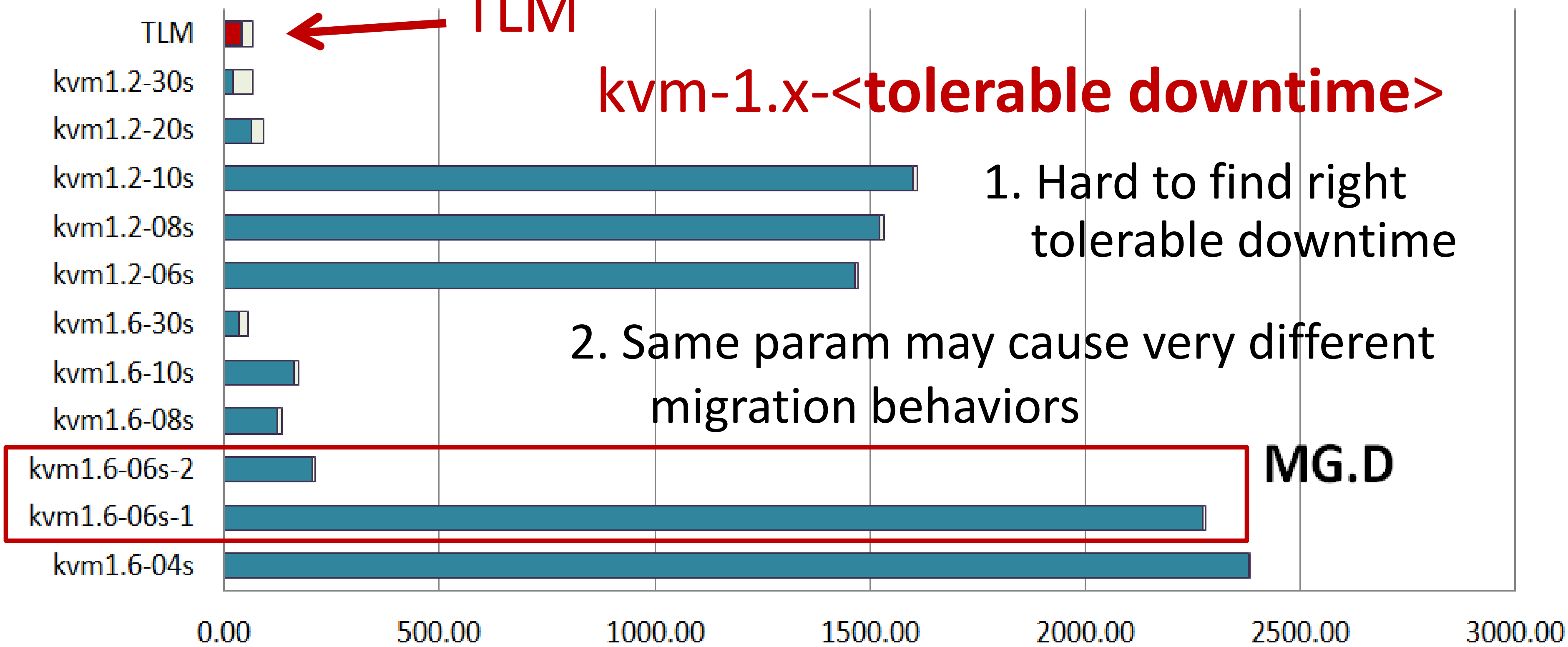
- Add two threads to source hypervisor
  - Mtx: scan entire ram
  - Dtx: new dirty pages
- Use two receiver threads to dest

Downtime reduction

- Manage Resource Allocation and handle downtime minimization

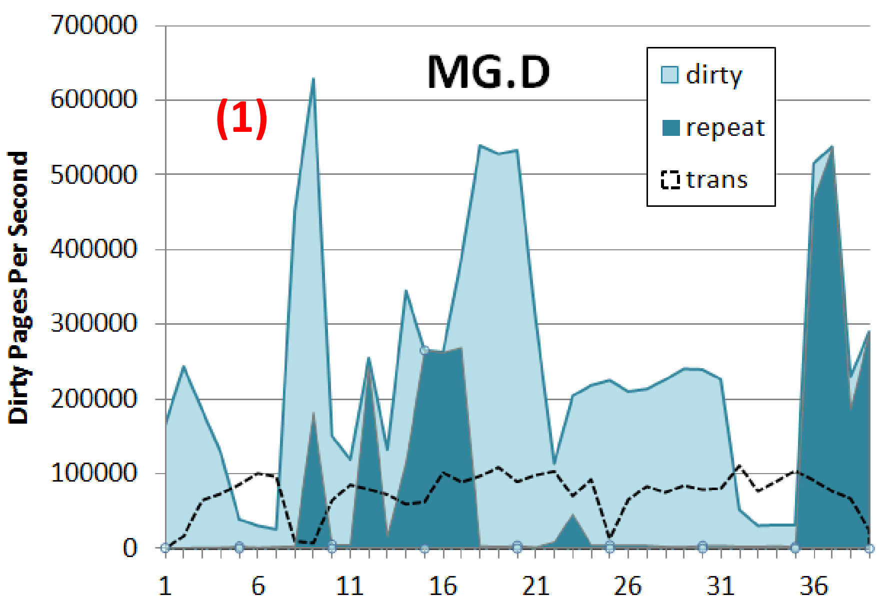


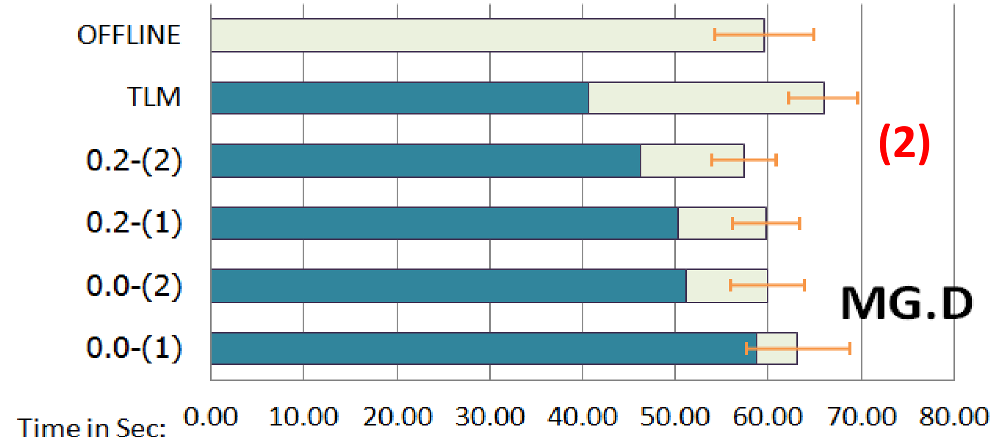
Kvm Migration and Downtime (over a 10 Gbps network)



	kvm1.6-04s	kvm1.6-06s-1	kvm1.6-06s-2	kvm1.6-08s	kvm1.6-10s	kvm1.6-30s	kvm1.2-06s	kvm1.2-08s	kvm1.2-10s	kvm1.2-20s	kvm1.2-30s	TLM
Live Migrate	2377.96	2271.91	204.71	124.88	161.87	33.42	1465.00	1521.46	1600.38	64.24	20.62	40.72
Downtime	3.92	6.38	6.91	7.89	10.40	22.77	7.06	9.37	10.69	28.24	45.65	25.27

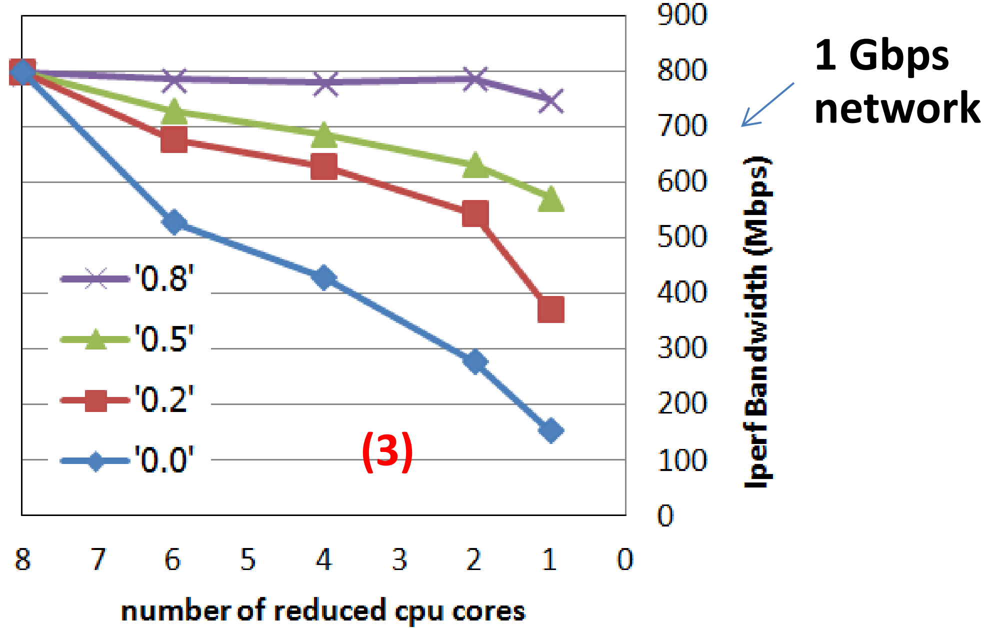
TLM:Kernel MG Class D





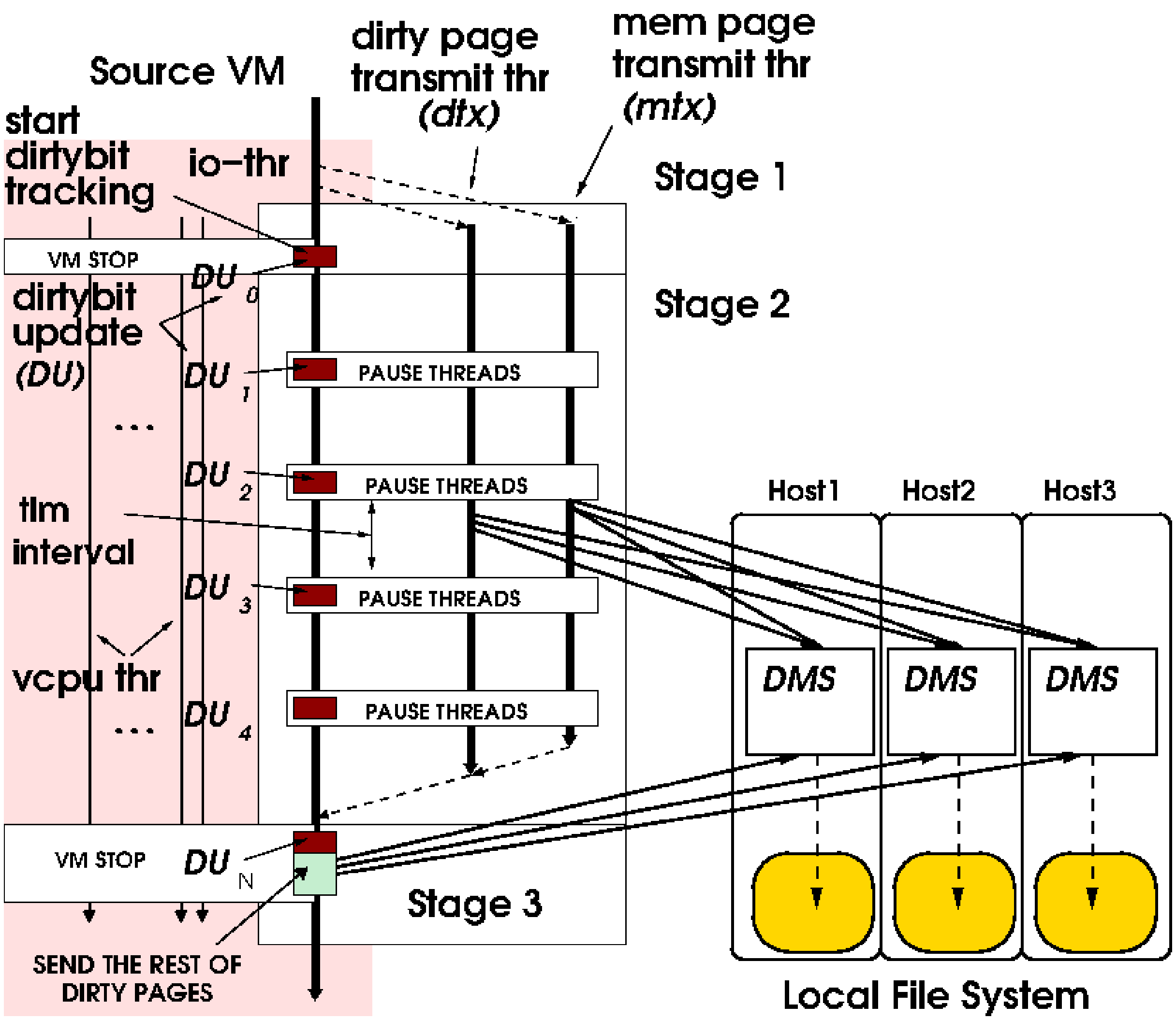
	0.0-(1)	0.0-(2)	0.2-(1)	0.2-(2)	TLM	OFFLINE
Live Migrate	58.70	51.12	50.34	46.21	40.72	0.00
Downtime	4.52	8.85	9.41	11.23	25.27	59.60

- 36GB VM Ram, 27.3GB WSS
- Low locality, 600,000 pages can be updated in one secon[d] but pages are transfer no more than 100,000 page/sec
- Reasonable Bandwidth



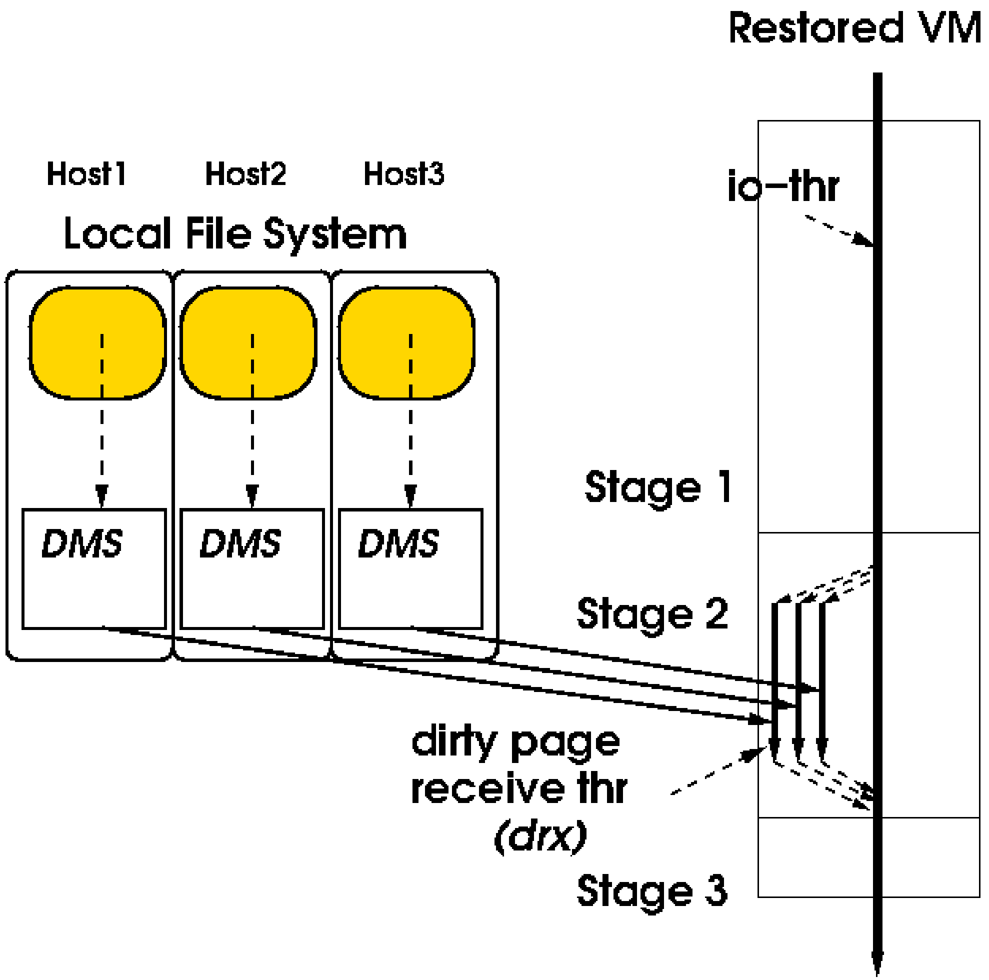
Time-bound Live Checkpointing (TLC)

- Based on TLM
- Send state evenly to set of Distributed Memory Servers
- Let each DMS saves the state to local disk when finish Stage 3
- Each DMS can write state to PFS later
- Perf: migtime + 1/3 of saving the entire VM state to local disk



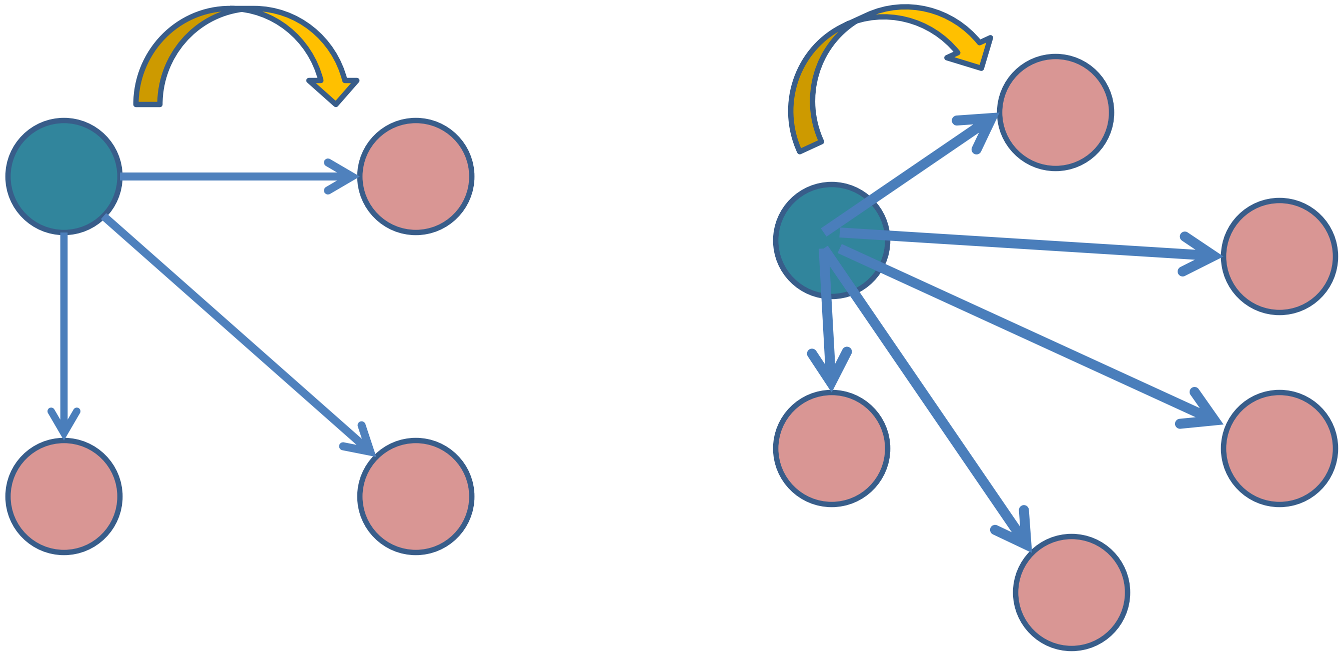
Time-bound Live Checkpointing (TLC)

- Based on TLM
- Each DMS load state info from local disk
- When the loading is done, send data simultaneously to the restored VM
- The restored VM put the transmitted state info at the right place and resume computation
- Perf: 1/3 of traditional VM restoration time



How do we make TLC checkpointing scale?

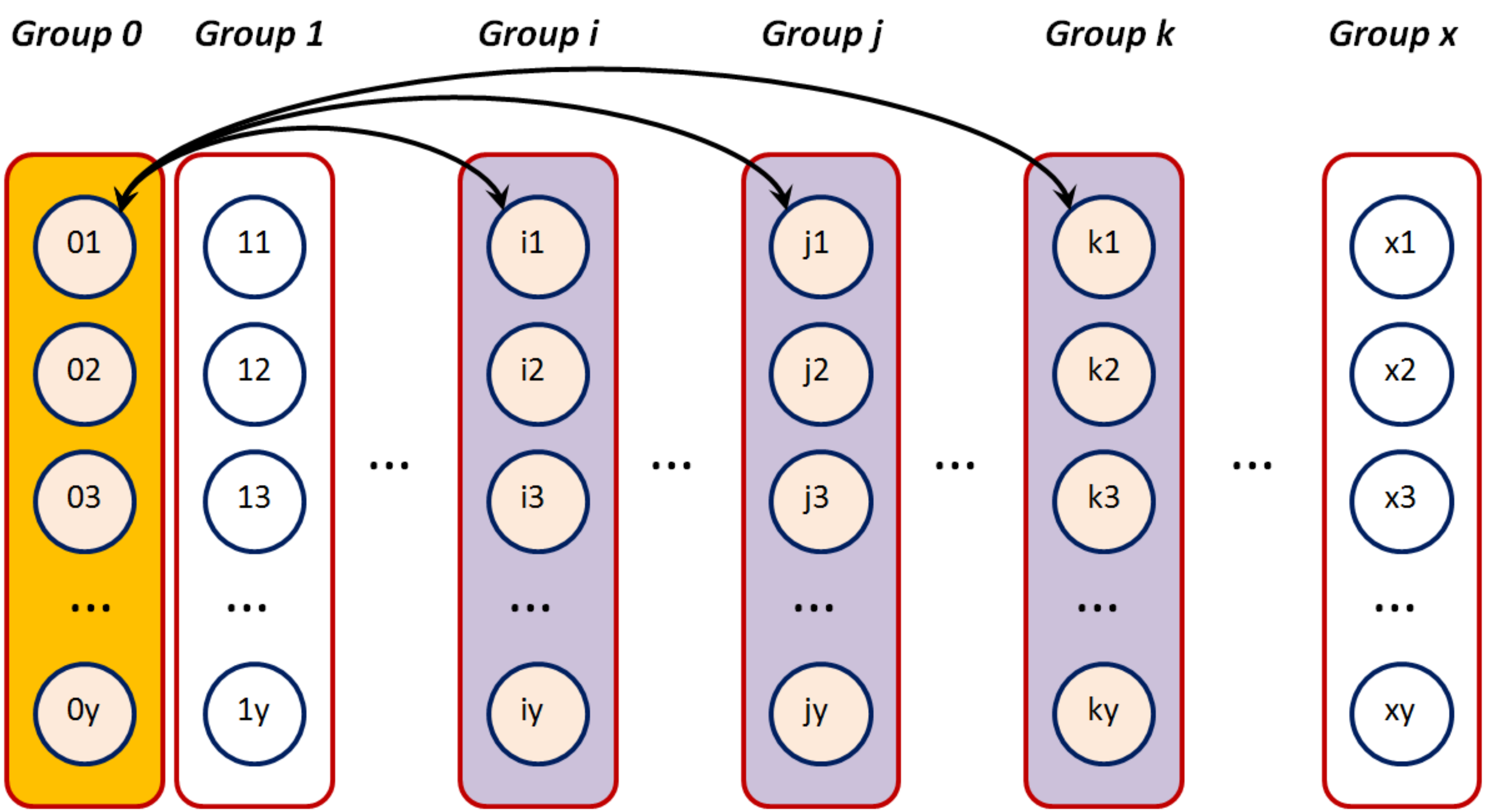
- Define a set of host, namely a ***circle***
- Let each host in the same circle takes turn to checkpoint while the rests help saving its state



Scalable Checkpointing

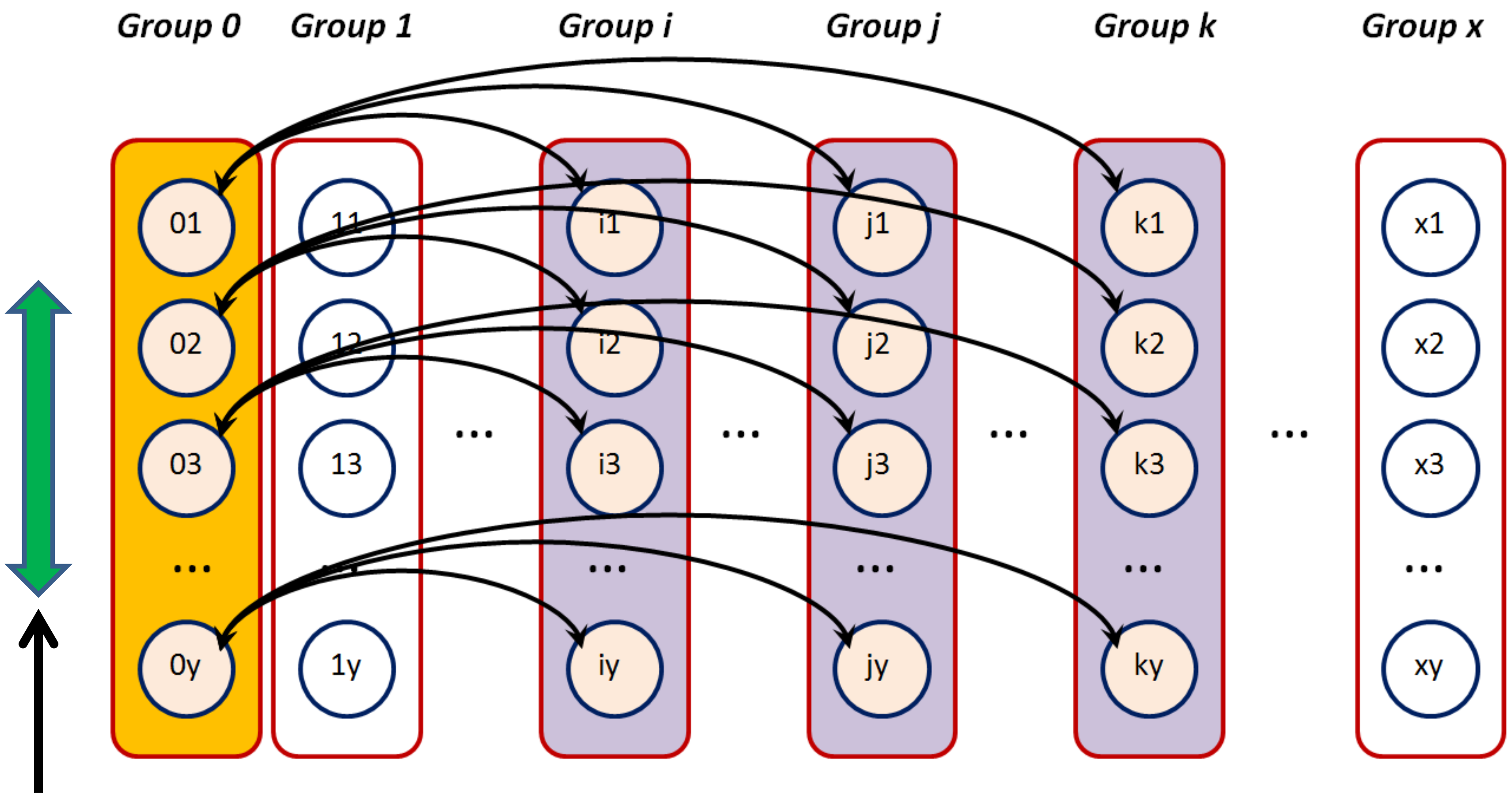
vasabiLab
Virtualization Architecture and ScalABle Infrastructure Laboratory

- Put each host in a circle into a separate ***group***



Scalable Checkpointing

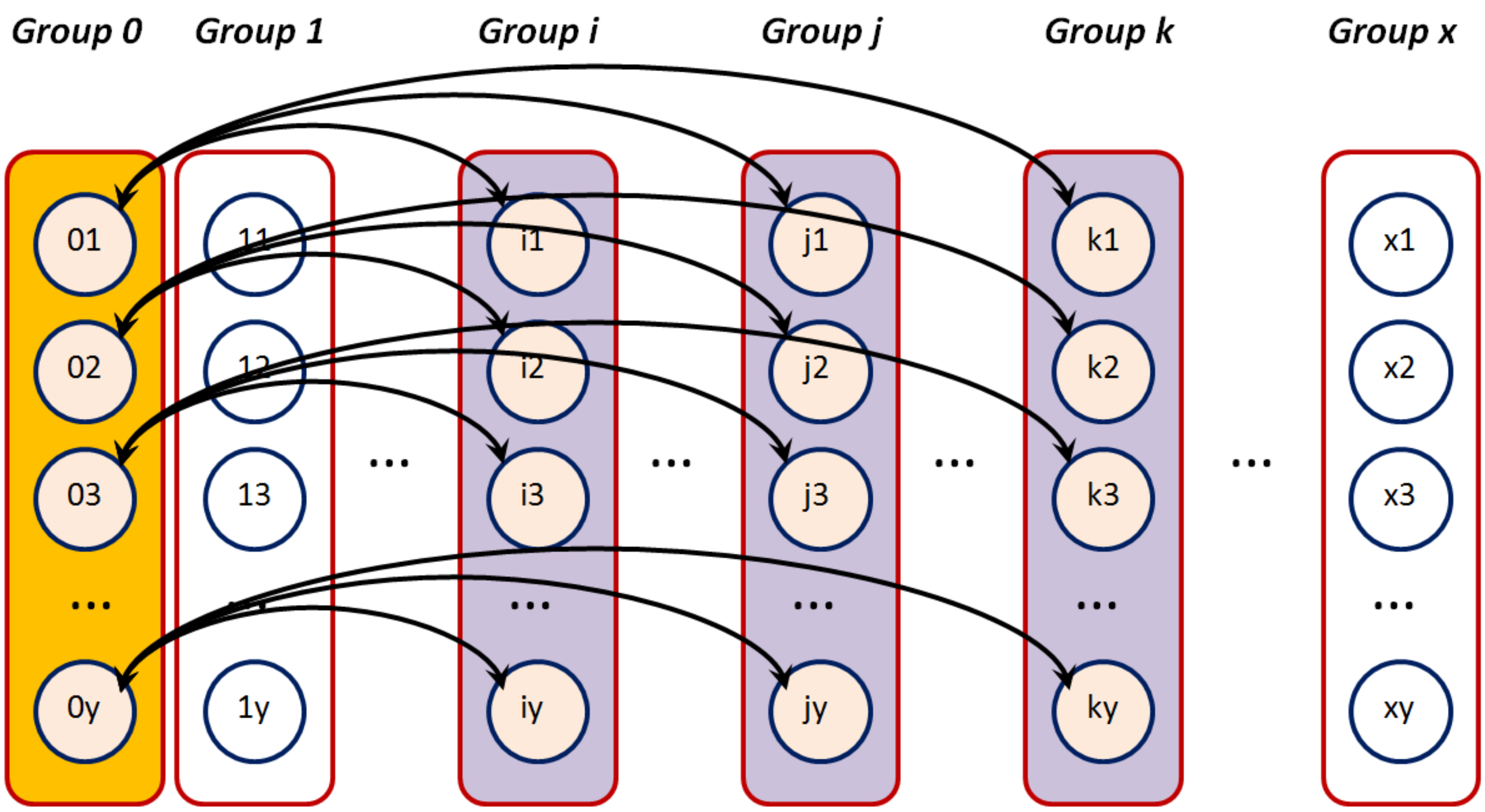
- VM on host in the same ***group*** chkpt at the same time



VMs in the same group could be communicating with one another

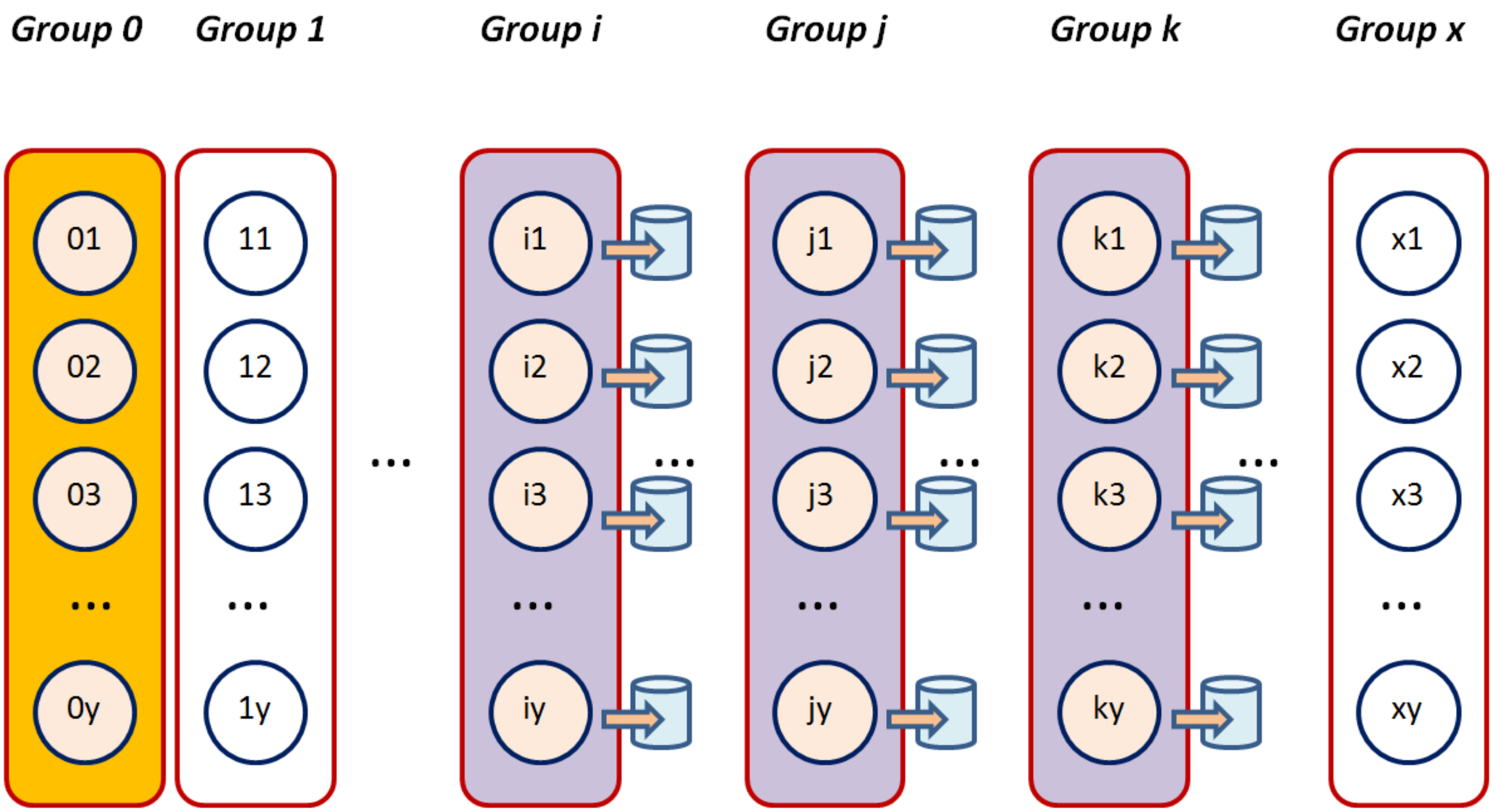
Scalable Checkpointing

- VM on host in the same ***group*** chkpt at the same time



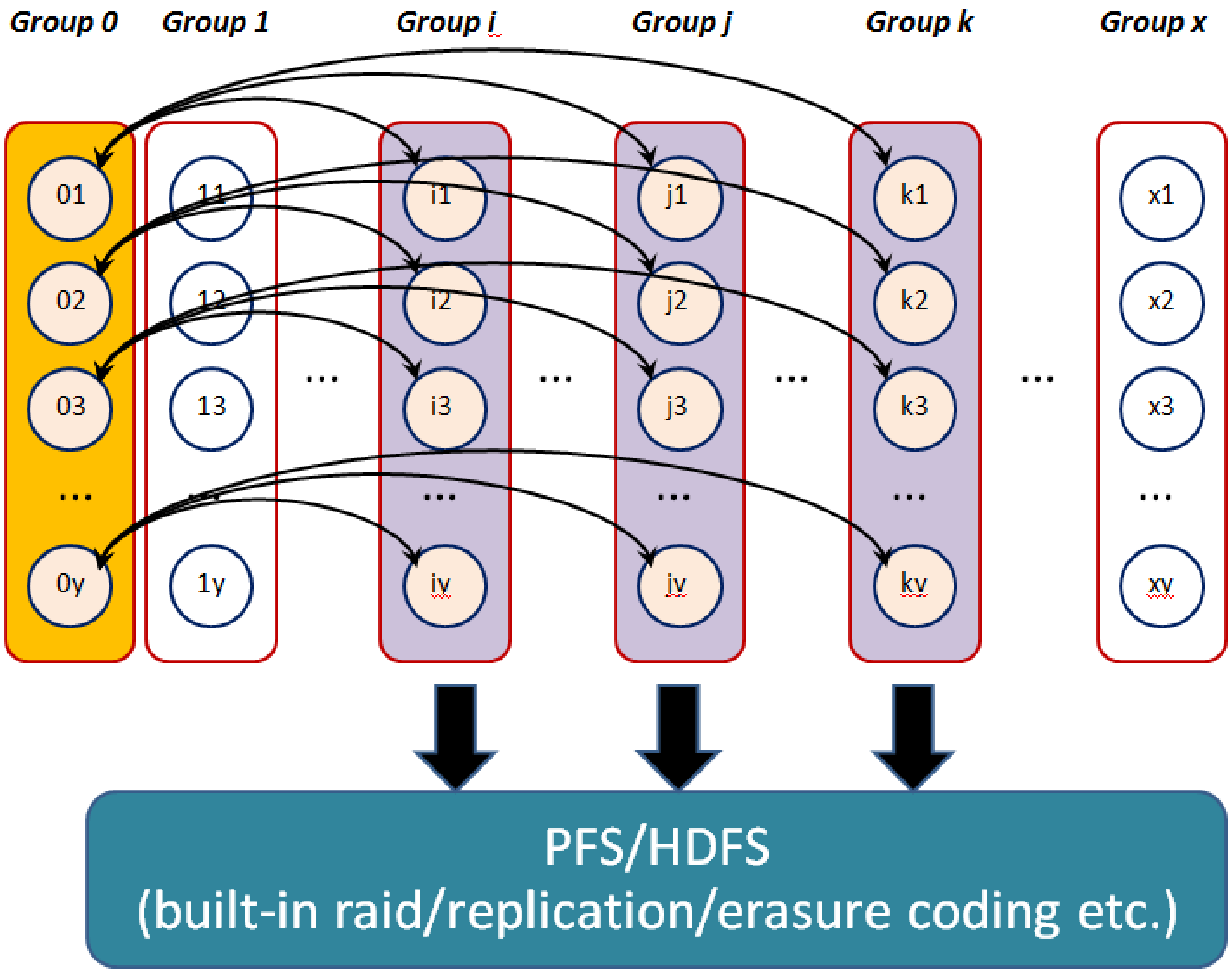
Scalable Checkpointing

- Every DMS on a helping host save state to local disk



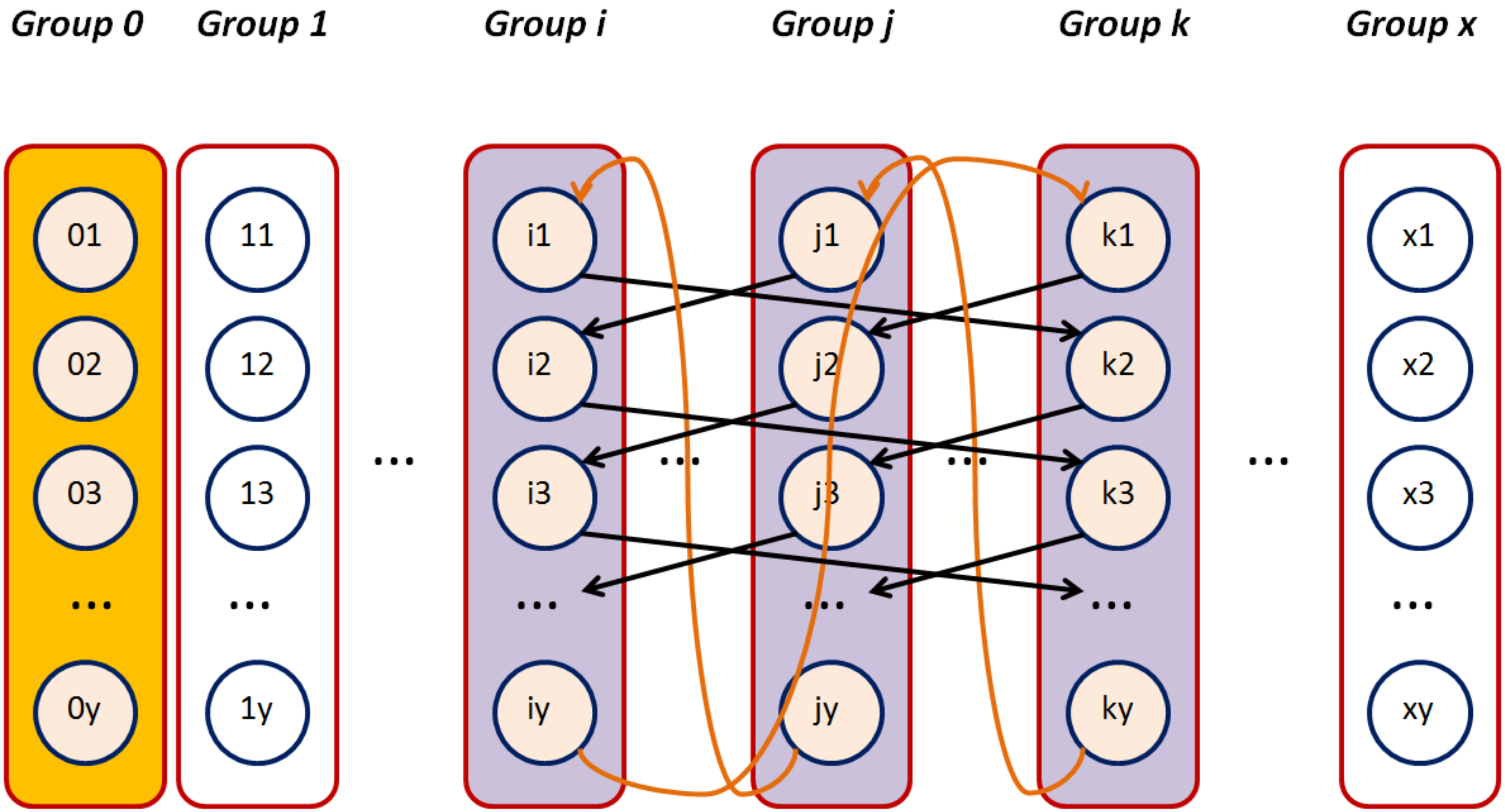
Scalable Checkpointing

- DMS can later saves state to PFS



Scalable Checkpointing

- Or, DMS can collaborate to replicate state information



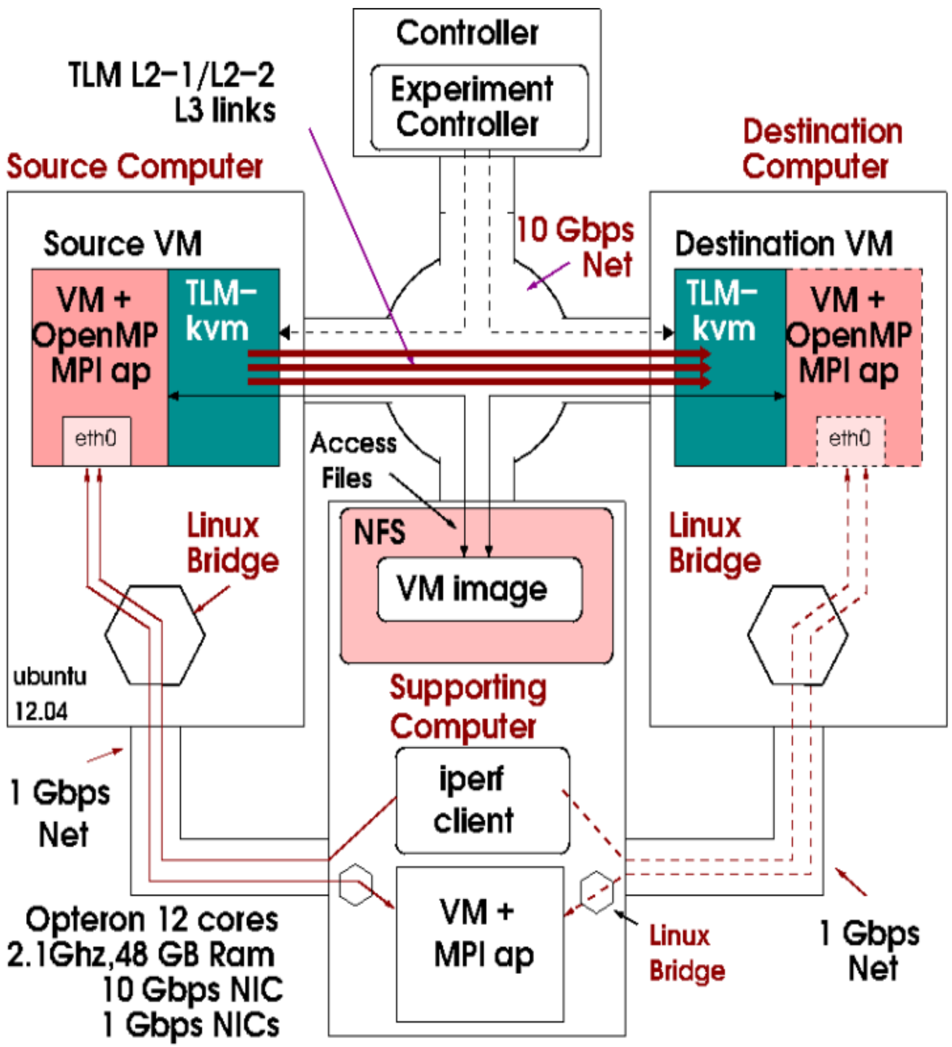
Conclusion and Future Works

vasabiLab
Virtualization Architecture and ScalABle Infrastructure Laboratory

- We propose a Time-bound Live Checkpointing (TLC) mechanism
  - Finish in a bound time period (proportional to Ram size)
  - Provide best effort downtime minimization
  - Reduce dirty page generation rate to minimize downtime
- We propose using a set of the Distributed Memory Server to speed up checkpointing time
- We propose a method to perform checkpointing at a large scale
- We have implemented TLC and DMS and conducted preliminary experiments
- Next, we will evaluate the scalable checkpointing ideas
- Thank you. Questions?

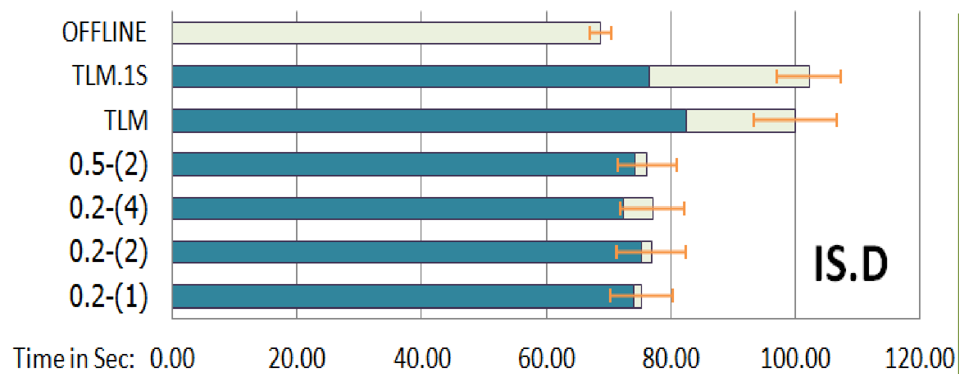
BACKUP

Experimental Setup

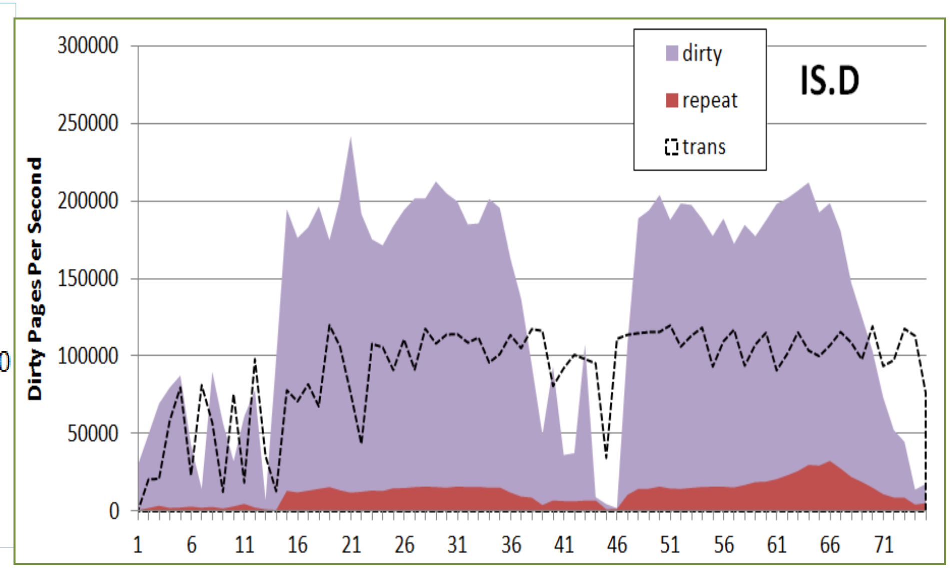


- Each VM uses 8 vcpu
- NAS Parallel Benchmark v3.3
  - OpenMP Class D (and MPI Class D in paper)
- VM migrate from source to dest computer
- Two separate networks:
  - 10 Gbps for migration
  - 1 Gbps for iperf
- Iperf fires from supporting computer
- VM disk image of migrating VM is on NFS

TLM Performance: Kernel IS Class D



	0.2-(1)	0.2-(2)	0.2-(4)	0.5-(2)	TLM	TLM.1S	OFFLINE
Live Migrate	74.14	75.28	72.39	74.22	82.51	76.67	0.00
Downtime	1.26	1.68	4.72	2.00	17.51	25.60	68.80



- 36GB VM Ram, 34.1GB WSS
- Update large amount of pages continuously
- VM page transfer rate is about half of dirty page generation
- The migration tome of TLM and TLM.1S are close
- TLM downtime is about 0.68 of that of TLM.1S