Virtualization and Cloud Computing Research at Vasabilab

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

vasabiLab
VIRTUALIZATION ARCHITECTURE AND SCALABLE INFRASTRUCTURE LABORATORY

DEPARTMENT OF COMPUTER SCIENCE THAMMASAT UNIVERSITY

TU science

Outline

- **Introduction to vasabilab**
- Research Projects
  - Virtual Machine Live Migration and Checkpointing
  - Cloud Computing

VasabiLab

- Virtualization Architecture and ScalABle Infrastructure Laboratory
  - Kasidit Chanchio, 1 sys admin, 2 Phd, 3 MS
  - Virtualization, HPC, systems
- Virtualization:
  - Thread-based Live Migration and Checkpointing of Virtual Machines
  - Coordinated Checkpointing Protocol for a Cluster of Virtual Machines
- Cloud Computing:
  - Science Cloud: The OpenStack-based Cloud implementation for Faculty of Science

Time-Bounded, Thread-Based Live Migration of Virtual Machines

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

Outline

- **Introduction**
- Virtual Machine Migration
- Thread-based Live Migration Overview
- Experimental Results
- Conclusion

Introduction

- Cloud computing has become a common platform for large-scale computations
  - Amazon AWS offers 8 vcpus with 68.4GiB Ram
  - Google offers 8 vcpus with 52GB Ram

Introduction

- Cloud computing has become a common platform for large-scale computations
  - Amazon AWS offers 8 vcpus with 68.4GiB Ram
  - Google offers 8 vcpus with 52GB Ram
- Applications require more CPUs and RAM
  - Big Data Analysis needs big VMs
  - Web Apps need huge memory for caching
  - Scientists always welcomes computing powers

Introduction

- Data Center has hundreds or thousands of VMs running. It is desirable to be able to live migrate VMs **efficiently**
  - Short migration time: flexible resource utilization
  - Low downtime: low impacts on application
- Users should be able to keep track of the progress of live migration
- We assume scientific workloads are computation intensive and can tolerate some downtime

Contributions

- Define a Time-Bound principle for VM live migration
- Our solution takes less total migration time than that of existing mechanisms.
  - 0.25 to 0.5 time that of qemu-1.6.0, the most recent (best) pre-copy migration mechanism
- Our solution can achieve low downtime comparable to that of pre-copy migration
- Create a basic building block for Time-Bound, Thread-based Live Checkpointing

Outline

- Introduction
- **Virtual Machine Migration**
- Thread-based Live Migration Overview
- Experimental Results
- Conclusion

VM Migration

VM Migration *is the ability to relocate a VM between two computers while the VM is running with minimal downtime*

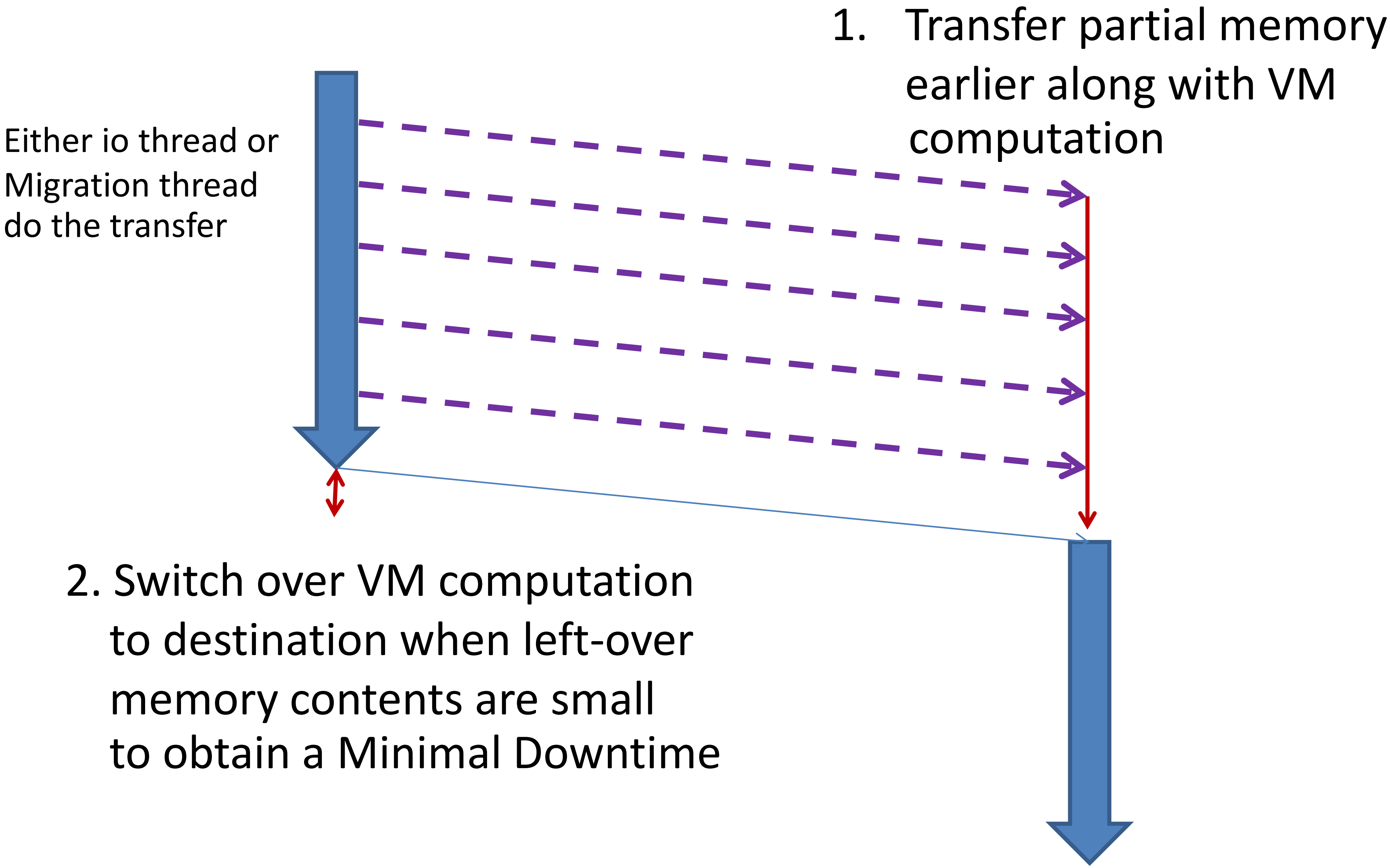
VM Migration

- VM Migration has several advantages:
  - Load Balancing, Fault-Resiliency, Data Locality
- Base on Solid Theoretical Foundation [M. Harchol Balter and A. Downey, Sigmetric96]

VM Migration

- VM Migration has several advantages:
  - Load Balancing, Fault-Resiliency, Data Locality
- Base on Solid Theoretical Foundation
- Existing Solutions
  - Traditional **Pre-copy** Migration: qemu-1.2.0, vmotion, hyper-v
  - **Pre-copy** with delta compression: qemu-xbrle
  - **Pre-copy** with migration thread: qemu-1.4.0, 1.5.0
  - **Pre-copy** with migration thread, auto converge: 1.6.0
  - Post-copy, etc.

Original Pre-copy Migration



Problems

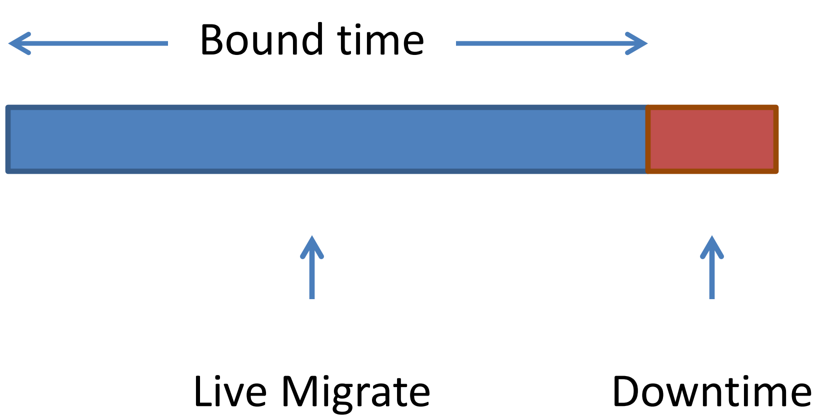
- Existing solutions cannot handle VMs with large-scale computation and memory intensive workloads well
  - Takes a long time to migrate
  - Have to migrate offline
- E.g. Migrate a VM running NPB MG Class D
  - 8 vcpus, 36 GB Ram
  - 27.3 GB Working Set Size
  - Can generate over 600,000 dirt pages in a sec.

Outline

- Introduction
- Virtual Machine Migration
- **Thread-based Live Migration Overview**
- Experimental Results
- Conclusion

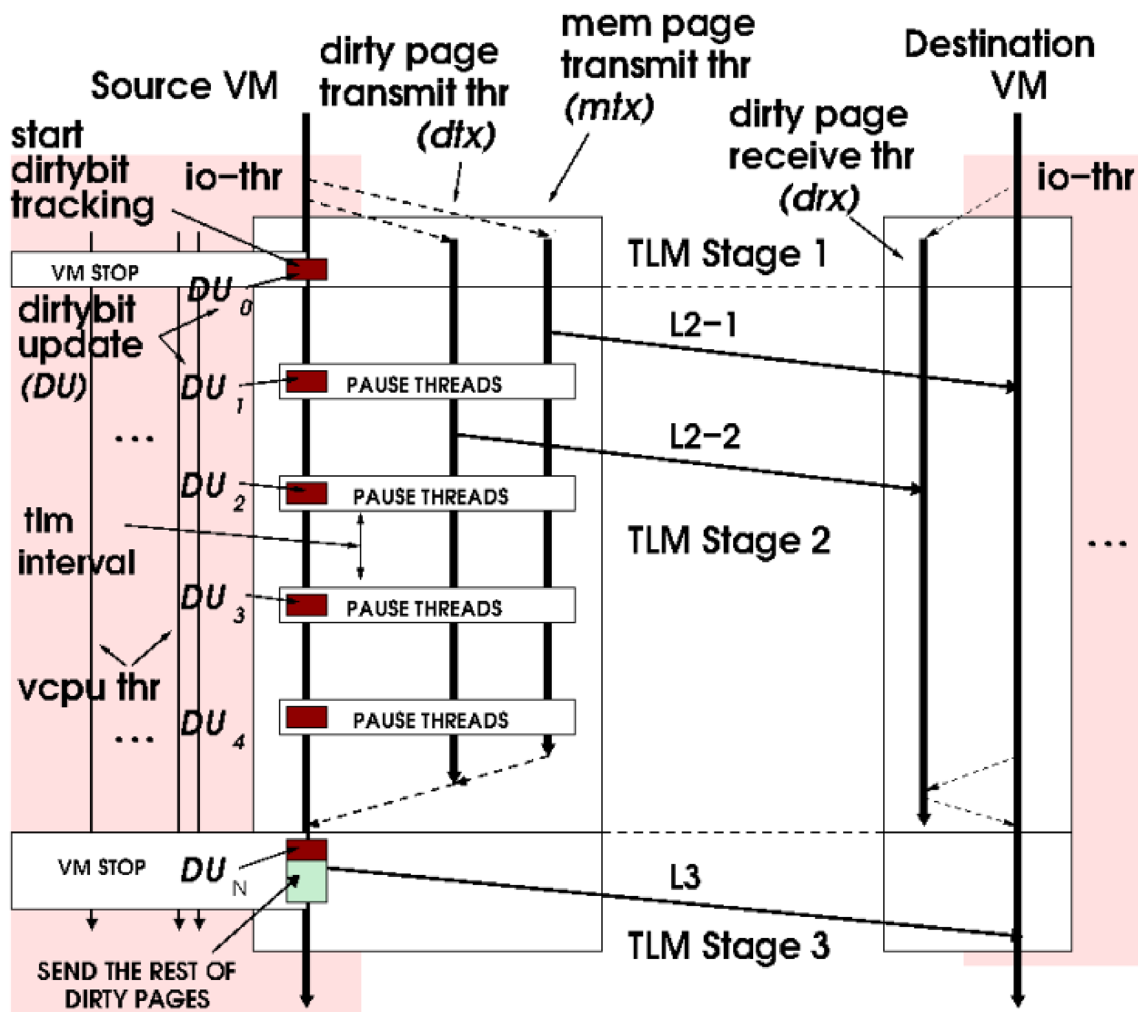
Time-Bound Scheme

- New perspective on VM Migration: Assign additional threads to handle migration
- Time: finish within a **bounded period of time**
- Resource: **best efforts** to minimize downtime while maintaining acceptable IO-bandwidth



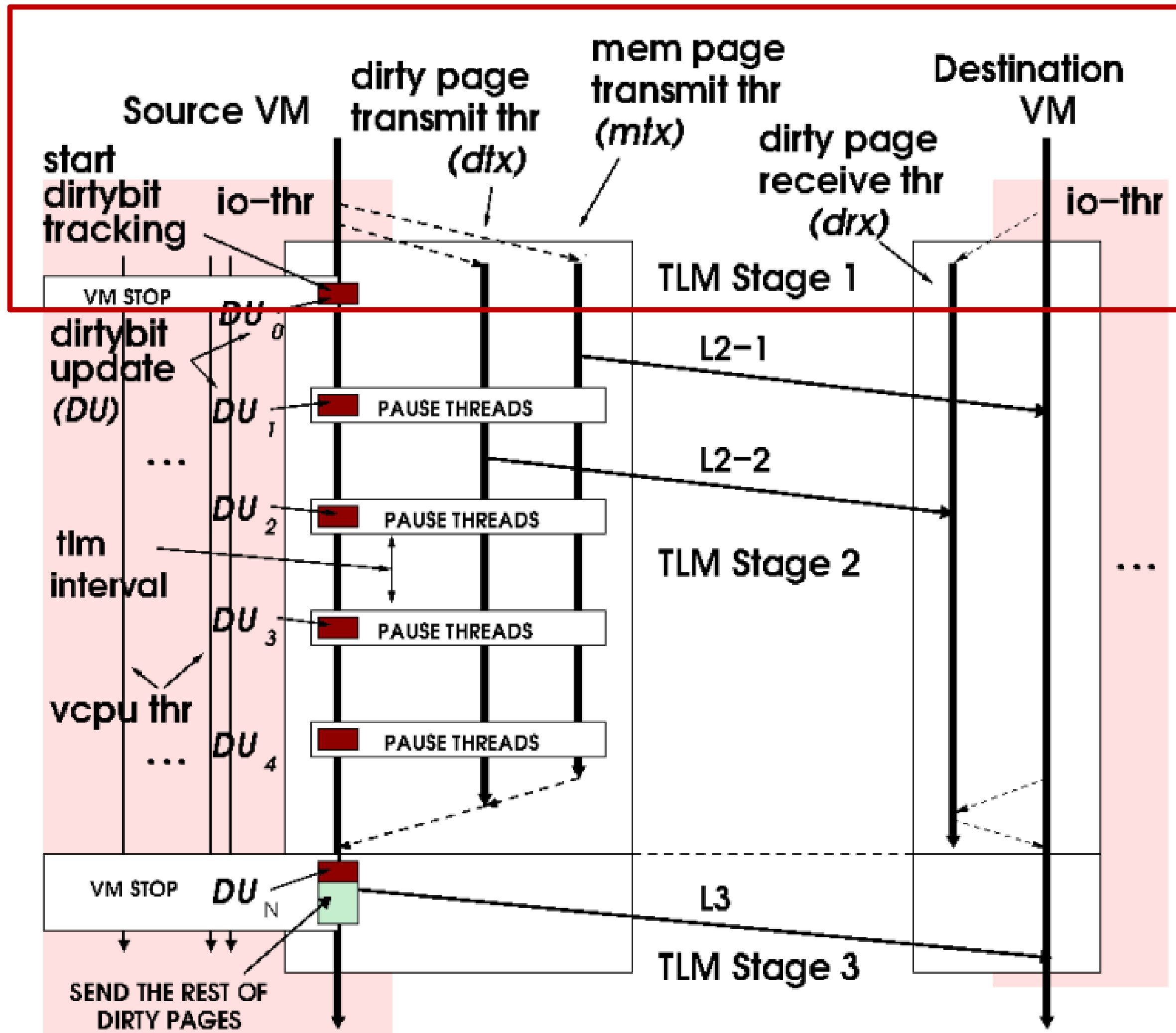
Thread-based Live Migration

- Add two threads
  - Mtx: save entire ram
  - Dtx: new dirty pages
- Operate in 3 Stages



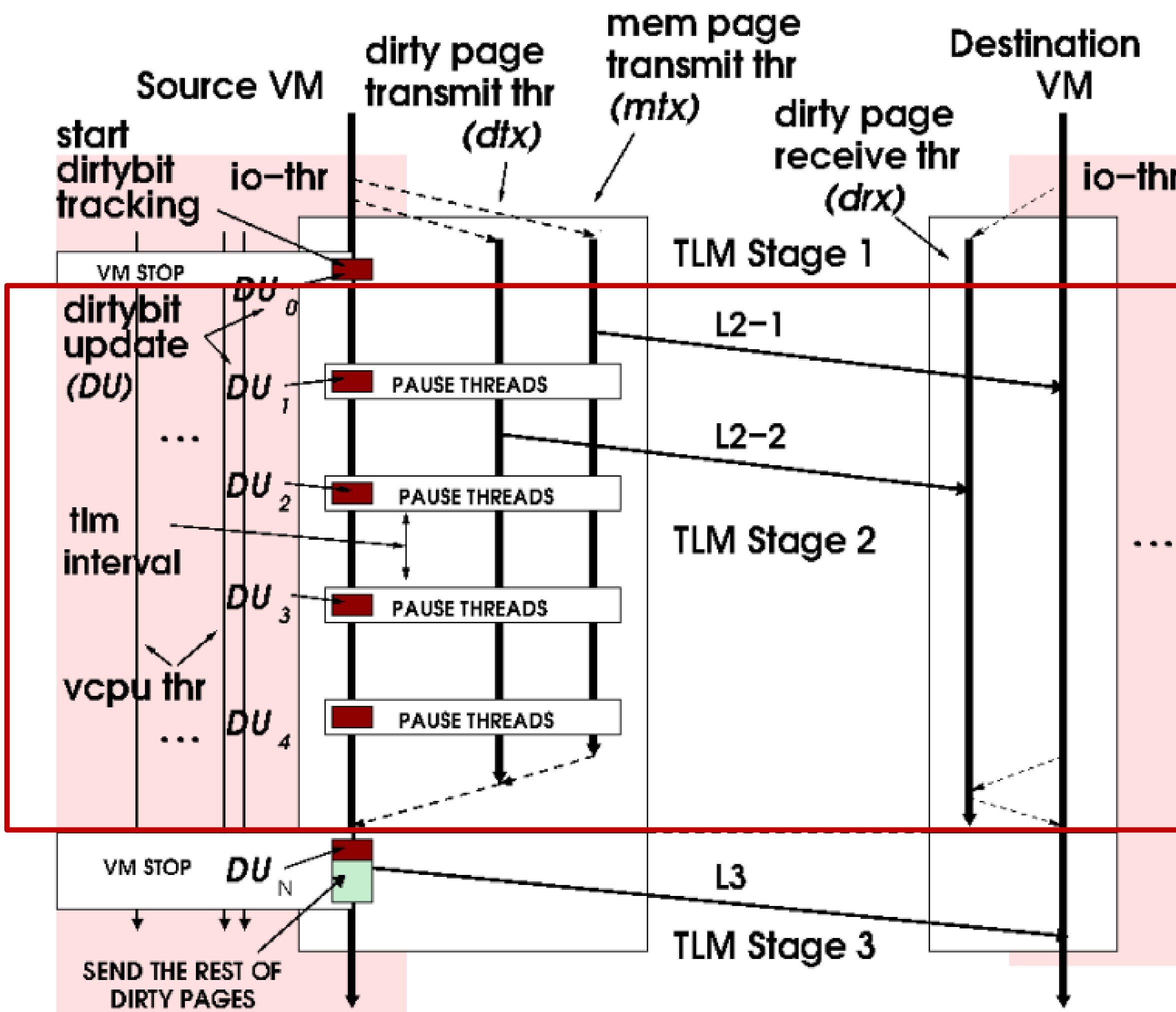
Thread-based Live Migration

- Stage 1
  - Set up 2 TCP channels
  - Start dirty bit tracking



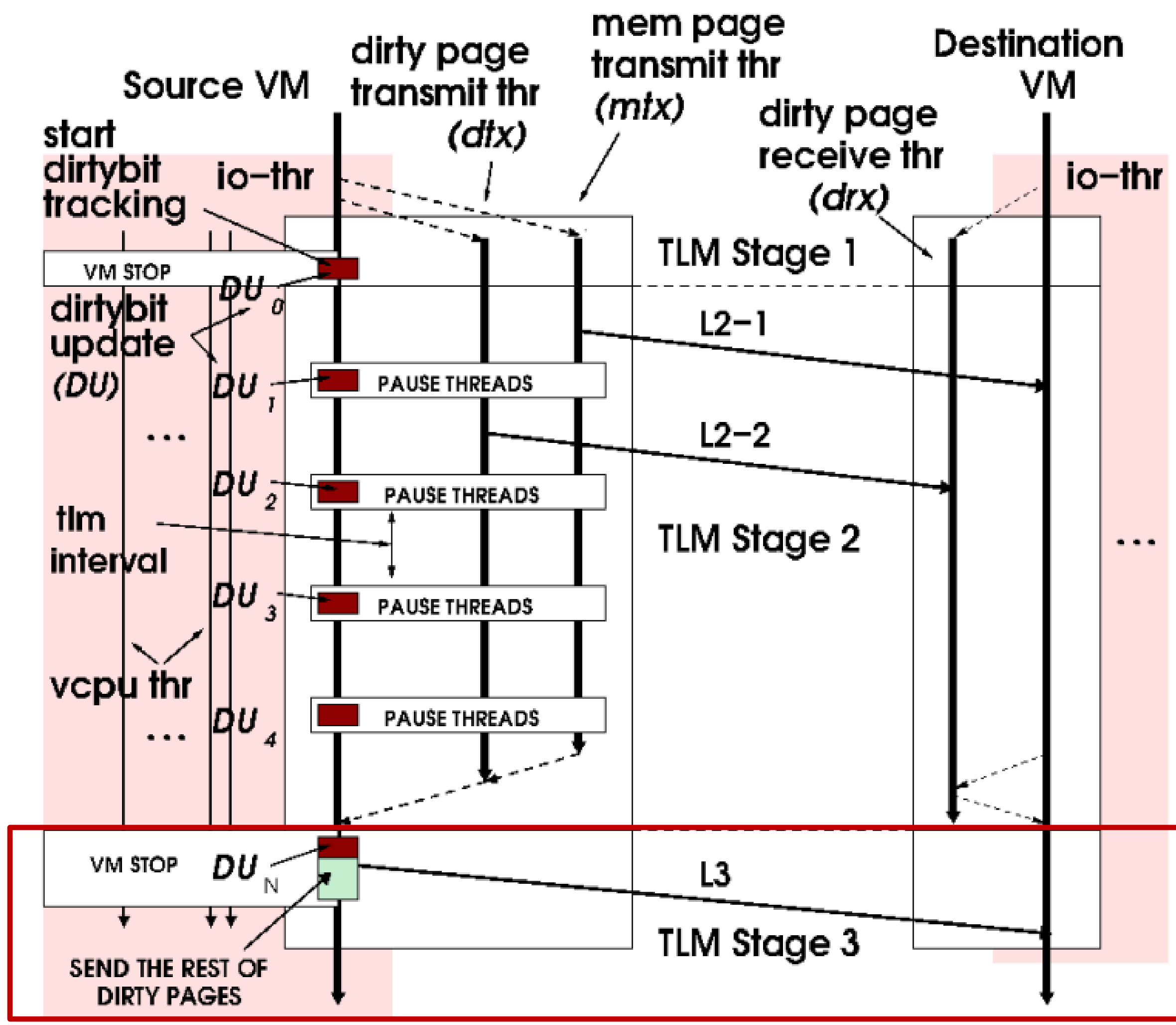
Thread-based Live Migration

- Stage 1
  - Set up 2 TCP channels
  - Start dirty bit tracking
- Stage 2
  - Mtx transfers Ram from first to last page
  - Dtx transfers dirty pages
  - Mtx skips transferring new dirty pages



Thread-based Live Migration

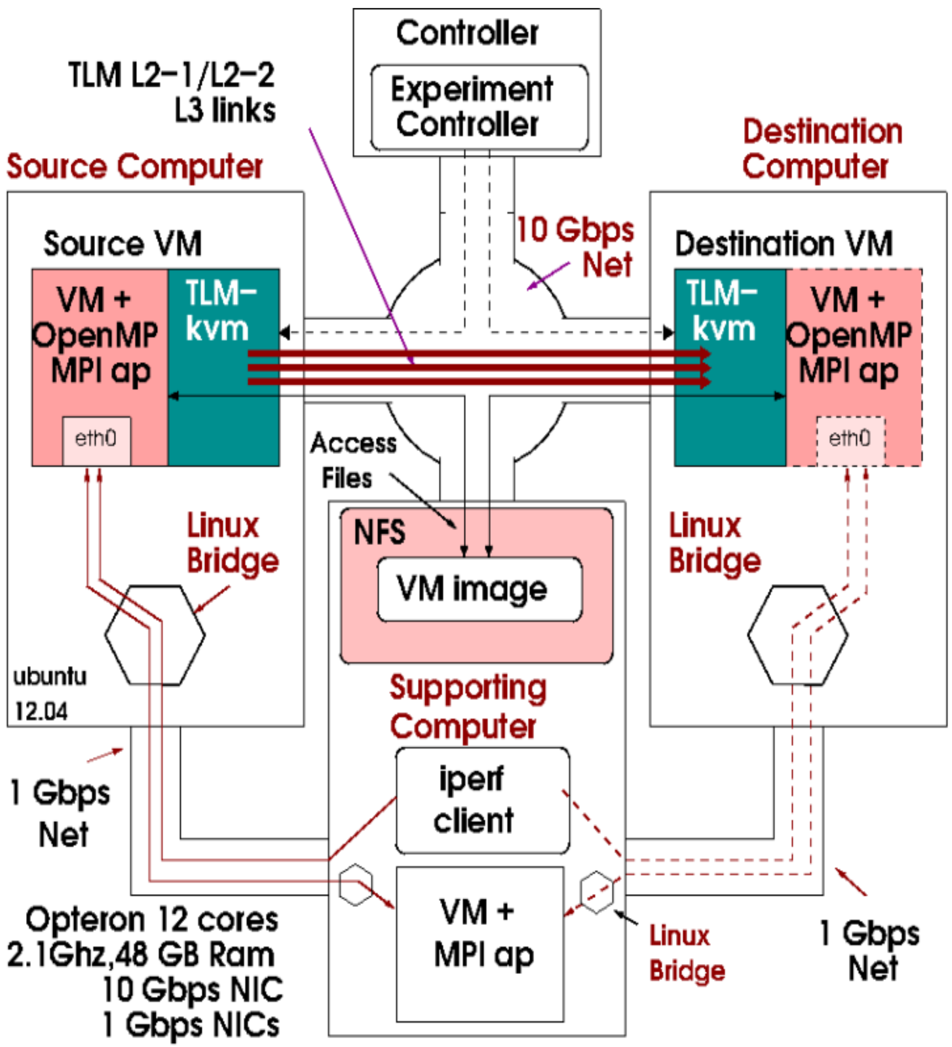
- Stage 1
  - Set up 2 TCP channels
  - Start dirty bit tracking
- Stage 2
  - Mtx transfers Ram from first to last page
  - Dtx transfers dirty pages
- Stage 3
  - Stop VM
  - Transfer the rest of dirty pages



Outline

- Introduction
- Virtual Machine Migration
- Thread-based Live Migration Overview
- **Experimental Results**
- Conclusion

Thread-based Live Migration



- NAS Parallel Benchmark v3.3
- OpenMP Class D
- VM 8 vcpu originally
- VM with Kernel MG
  - 36GB Ram, 27.3GB WSS
- VM with Kernel IS
  - 36GB Ram, 34.1GB WSS
- VM with Kernel MG
  - 16GB Ram, 12.1GB WSS
- VM with Kernel MG
  - 16GB Ram, 11.8GB WSS

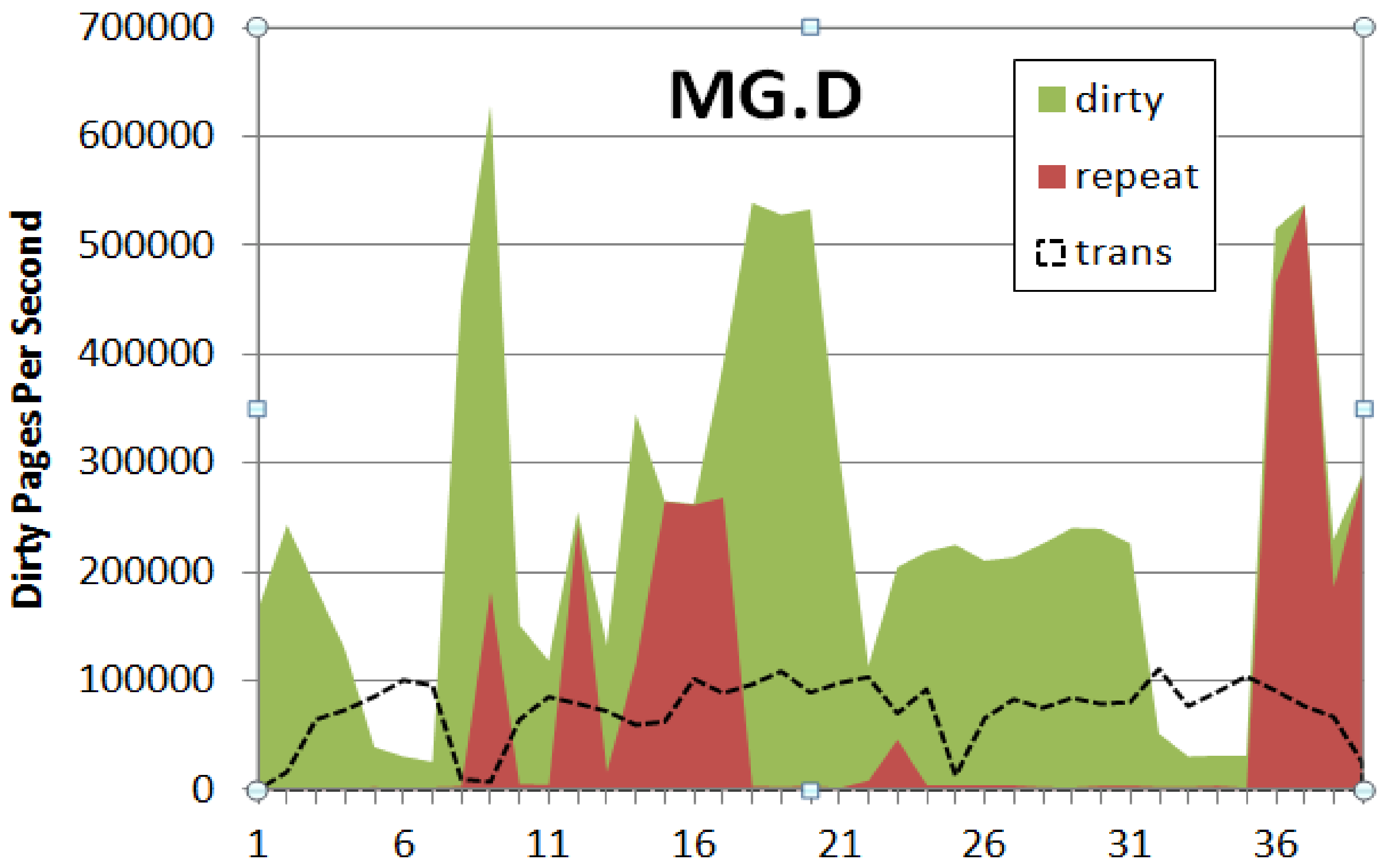
Notations

- Live Migrate: Time to perform live migration where the migration is performed during VM computation
- Downtime: Time the VM stop to transfer the last part of VM state

Notations

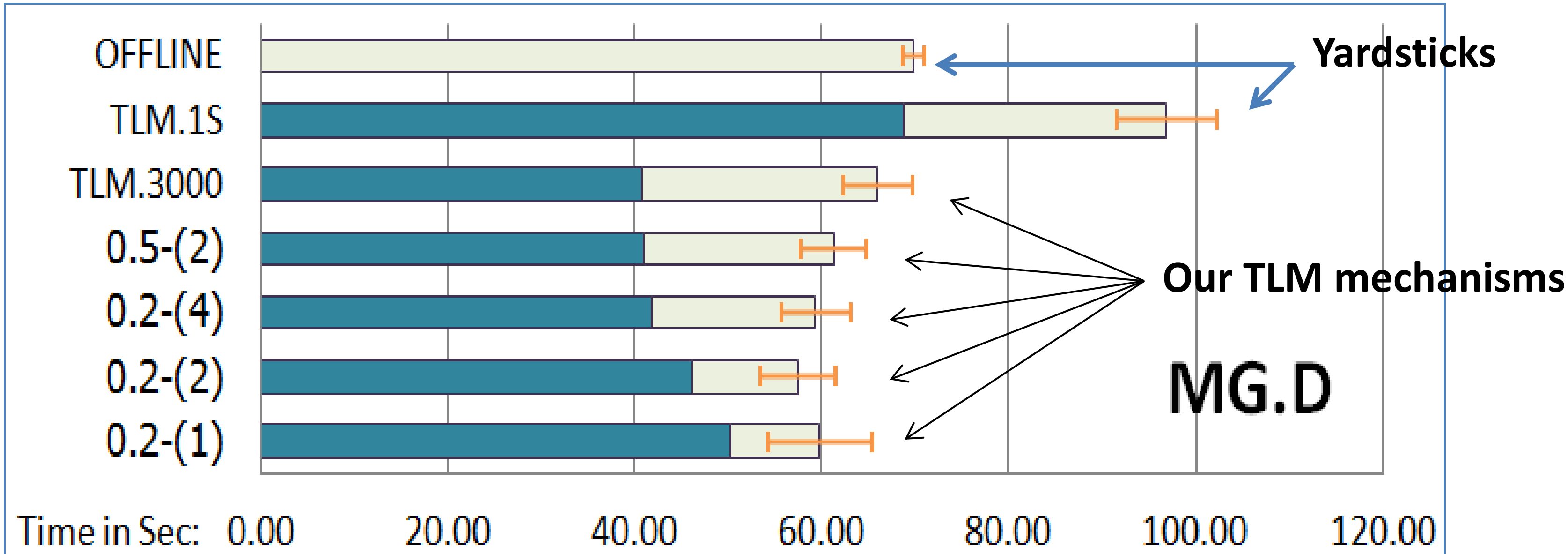
- Migration Time = Live Migrate + Downtime
- Offline: Time to migrate by stop VM & Transfer
- TLM.1S: Like TLM but let Stage 3 transfer all dirty pages
- TLM.3000: Migration Time of TLM
- 0.5-(2): Over-commit VM's 8 vcpus (from 8 host cores) on **2 host cores** after **50%** of live migration (mtx)

Experimental Results



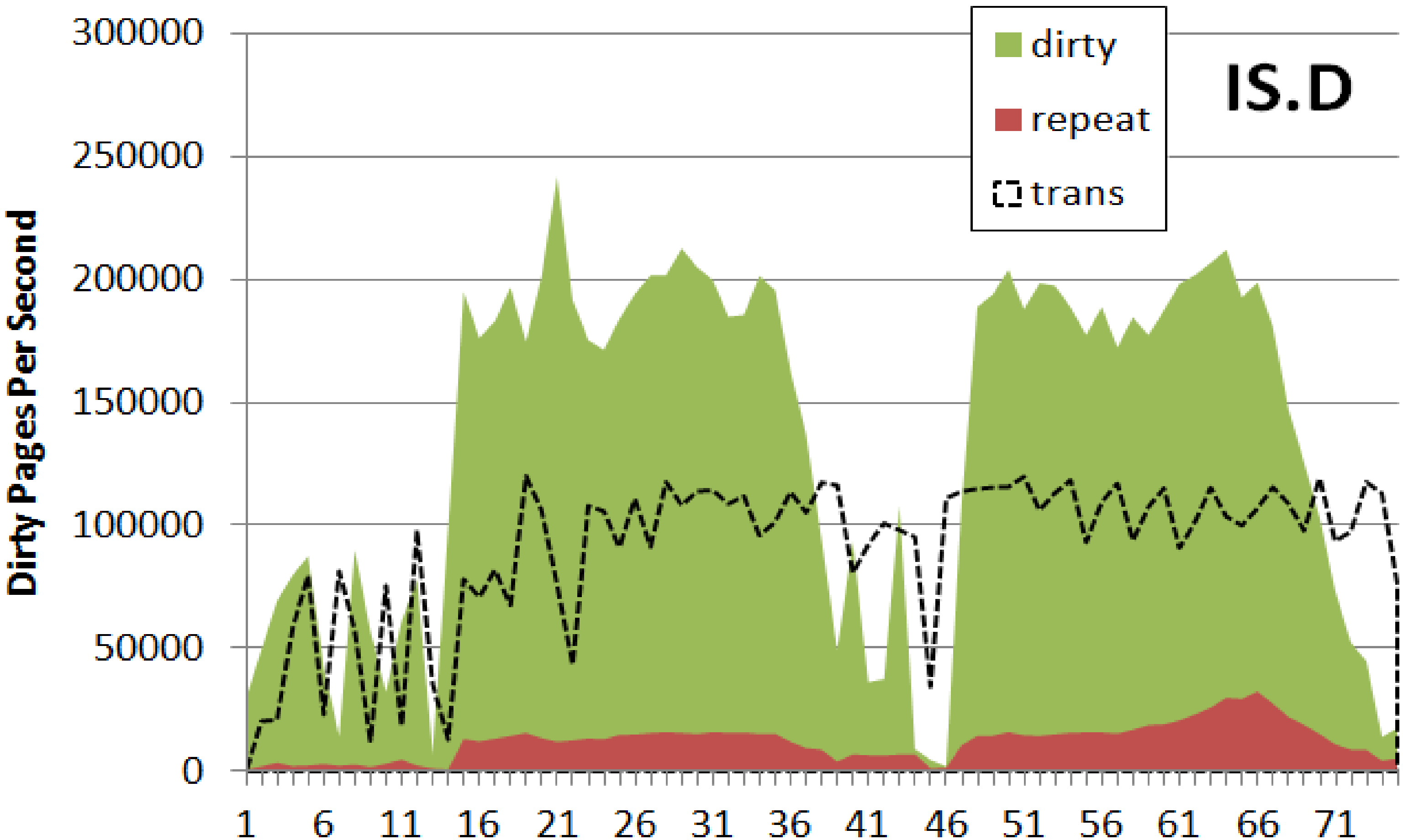
Very High Memory Update, Low Locality, Dtx Transfer rate << Dirty rate

Experimental Results



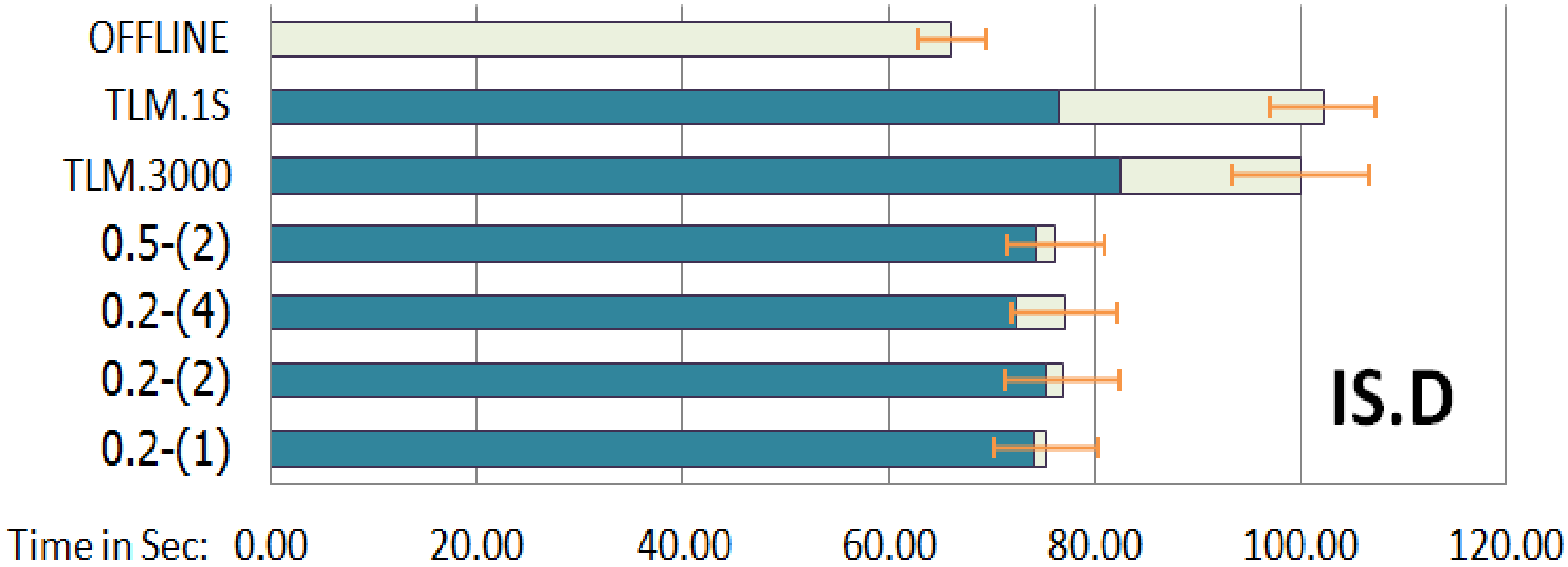
	0.2-(1)	0.2-(2)	0.2-(4)	0.5-(2)	TLM.3000	TLM.1S	OFFLINE
Live Migrate	50.34	46.21	41.82	41.00	40.72	68.77	0.00
Downtime	9.41	11.23	17.52	20.30	25.27	28.09	69.86

Experimental Results



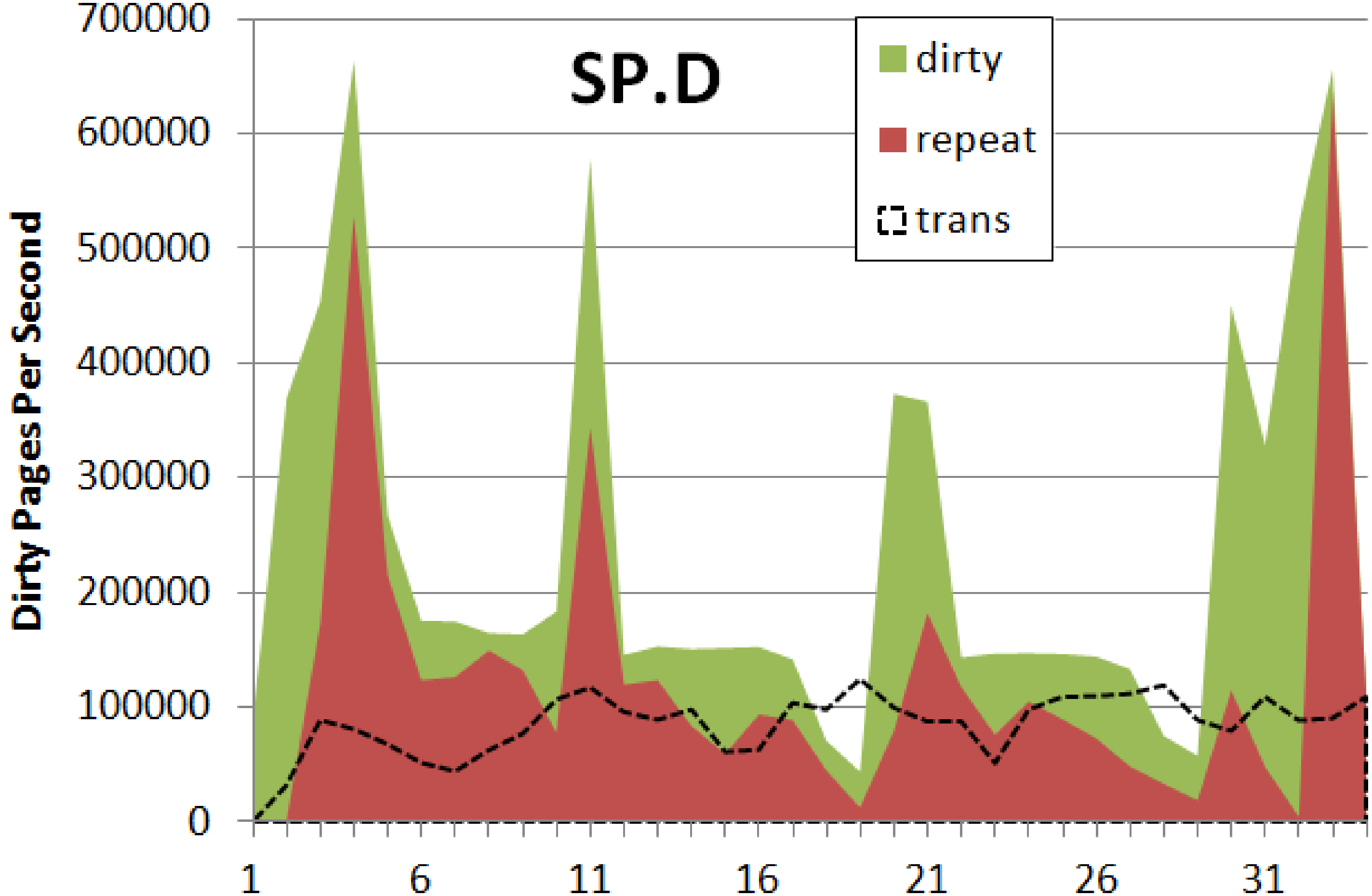
High Memory Update, Low Locality, Dtx Transfer rate = 2 x Dirty rate

Experimental Results



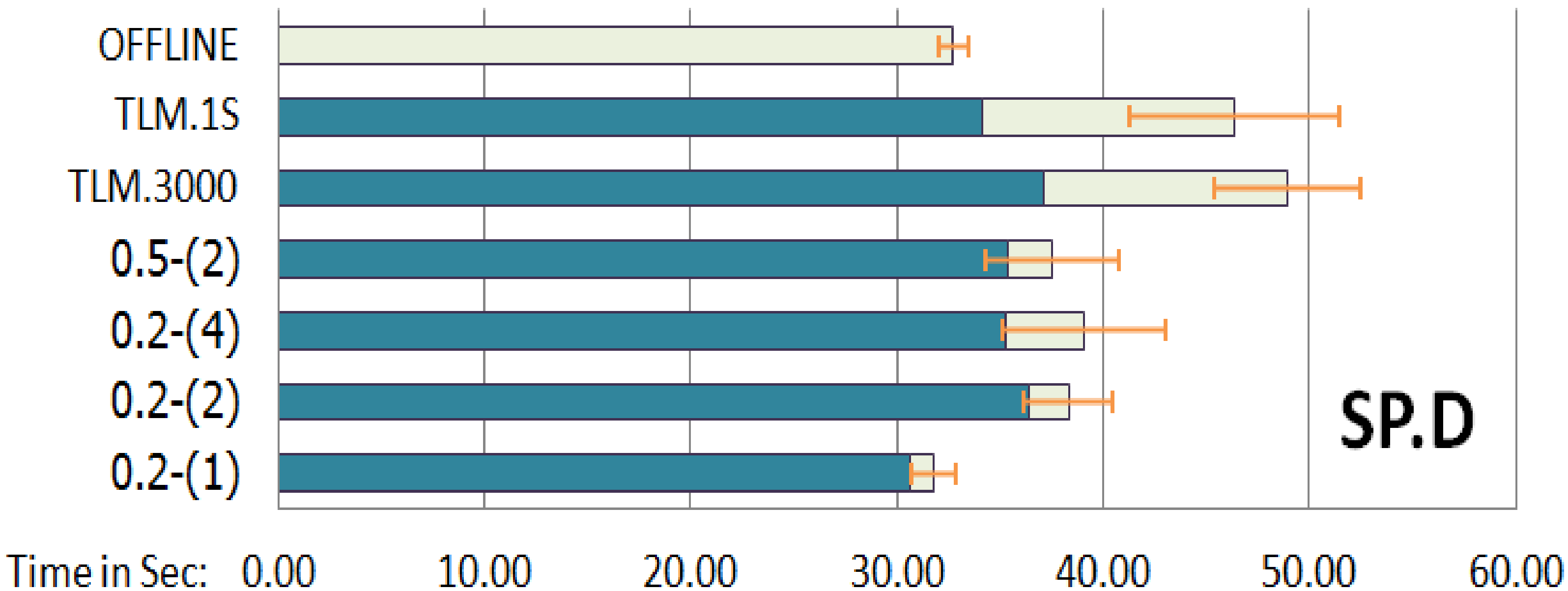
	0.2-(1)	0.2-(2)	0.2-(4)	0.5-(2)	TLM.3000	TLM.1S	OFFLINE
Live Migrate	74.14	75.28	72.39	74.22	82.51	76.67	0.00
Downtime	1.26	1.68	4.72	2.00	17.51	25.60	66.14

Experimental Results



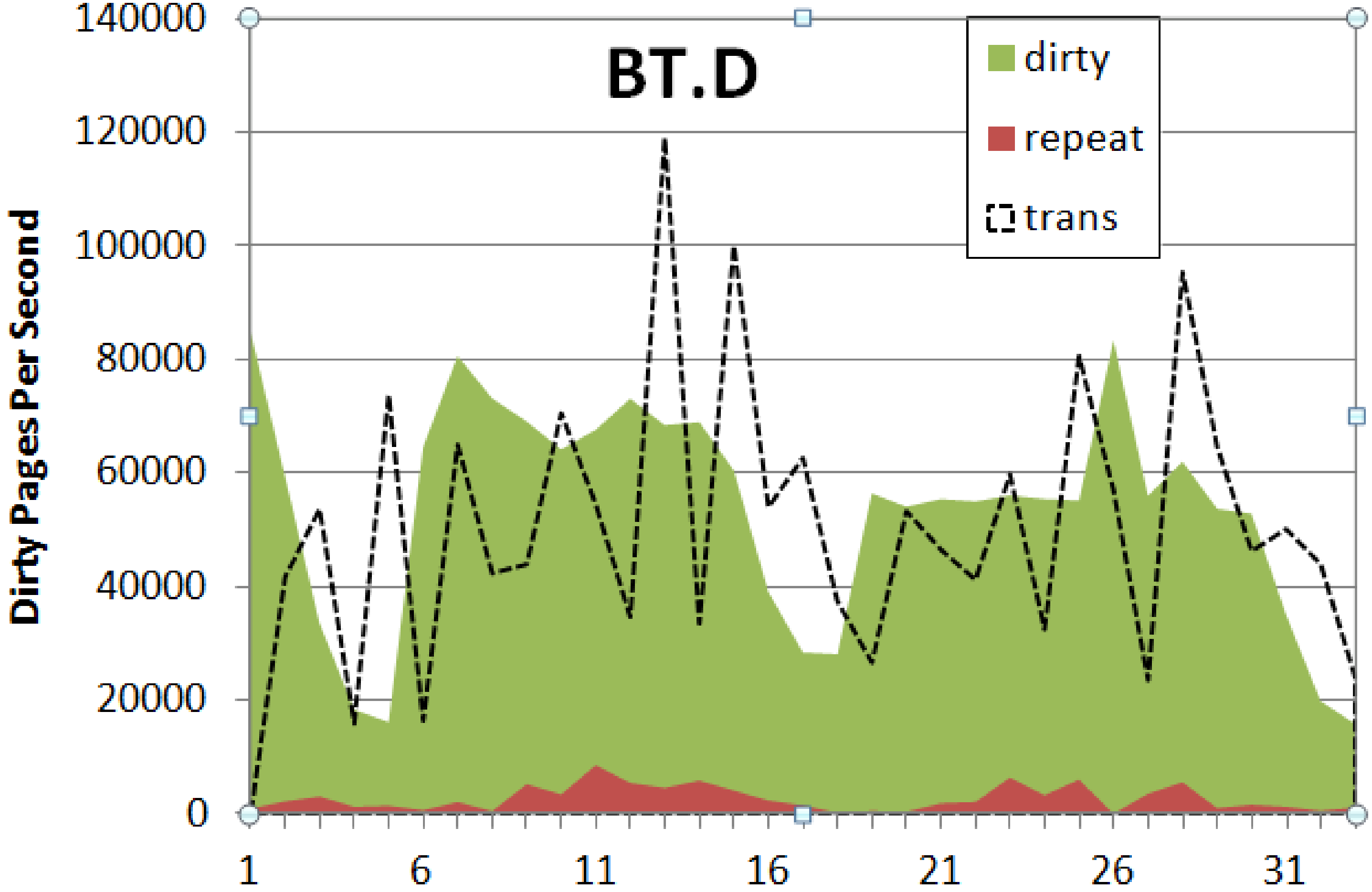
High Memory Update, High Locality, Dtx Transfer rate << Dirty rate

Experimental Results



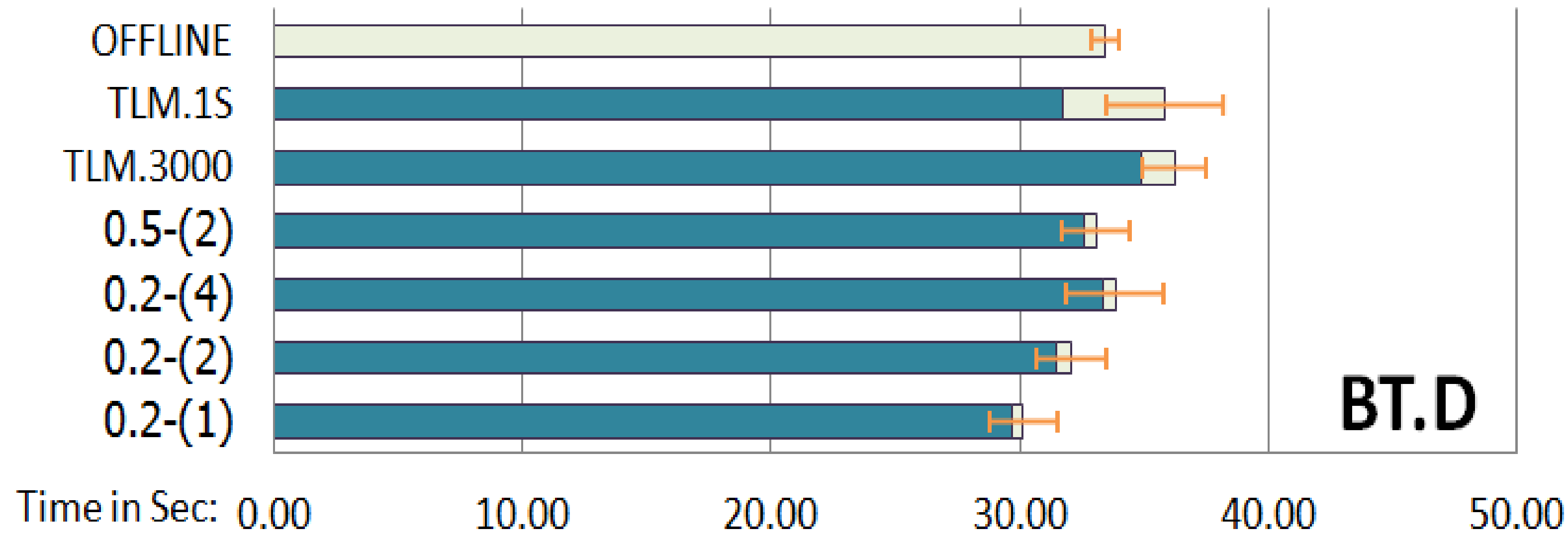
	0.2-(1)	0.2-(2)	0.2-(4)	0.5-(2)	TLM.3000	TLM.1S	OFFLINE
Live Migrate	30.66	36.38	35.29	35.36	37.07	34.15	0.00
Downtime	1.16	1.96	3.76	2.17	11.86	12.24	32.75

Experimental Results



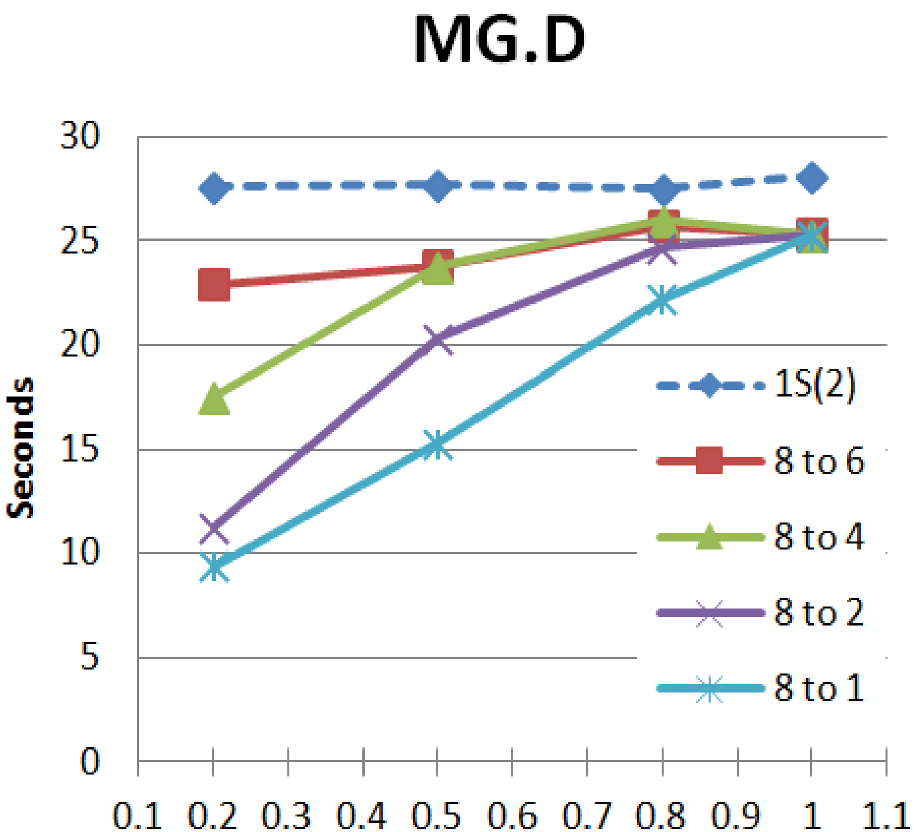
Medium memory Update, Low Locality, Transfer rate = Dirty rate

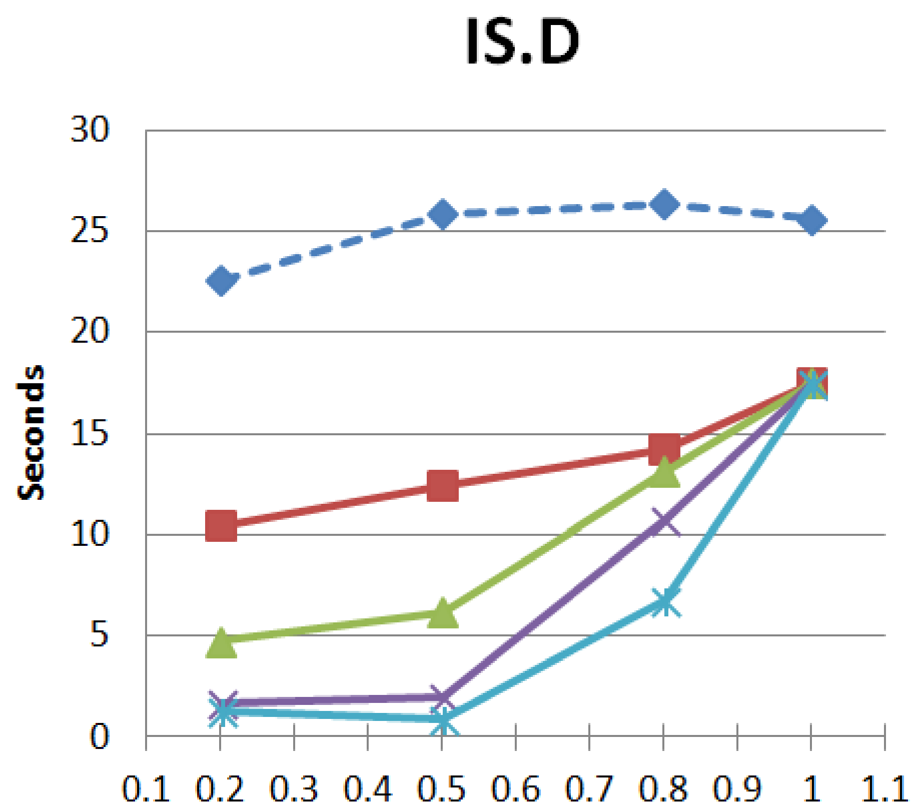
Experimental Results



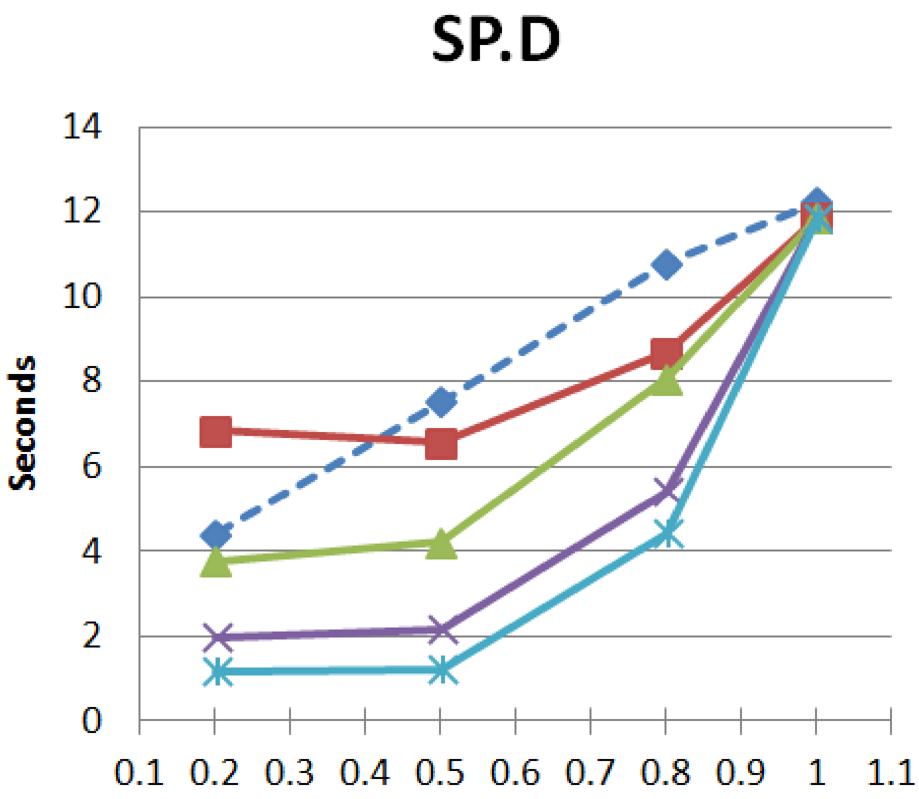
	0.2-(1)	0.2-(2)	0.2-(4)	0.5-(2)	TLM.3000	TLM.1S	OFFLINE
Live Migrate	29.70	31.48	33.33	32.56	34.90	31.74	0.00
Downtime	0.45	0.60	0.51	0.50	1.33	4.11	33.45

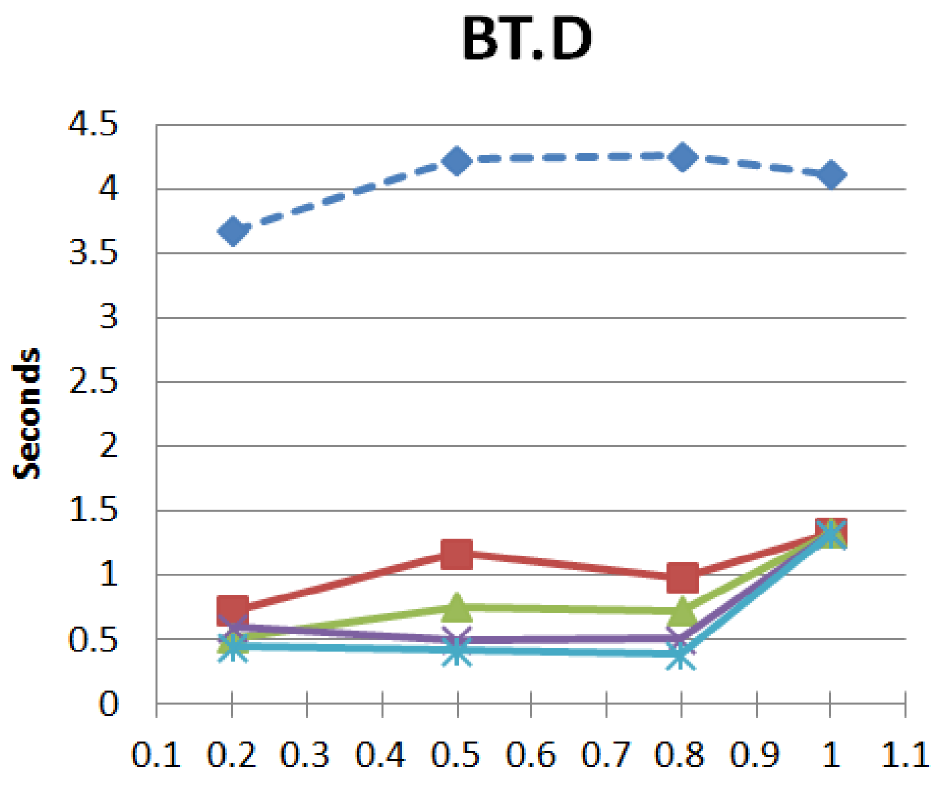
Downtime Minimization using CPU over-commit



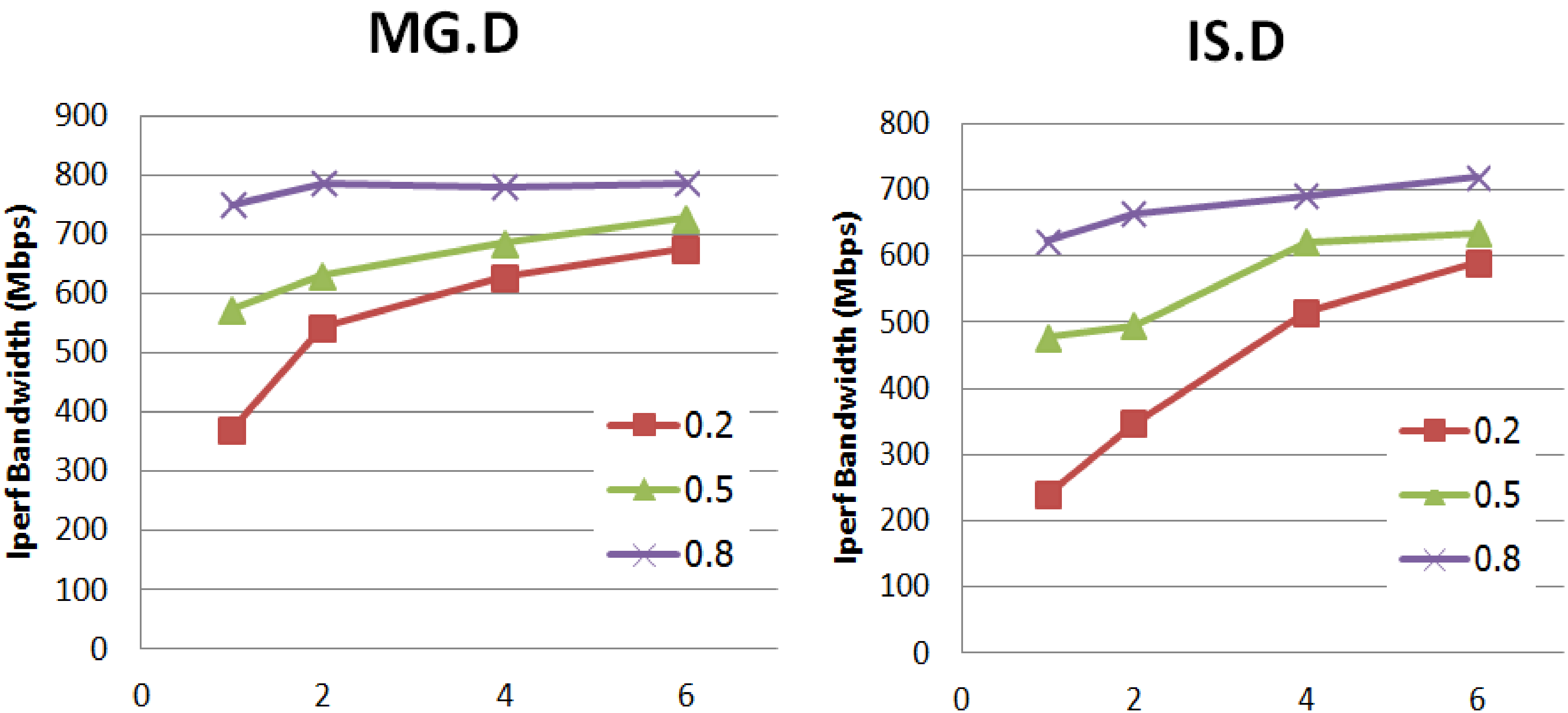


Downtime Minimization using CPU over-commit

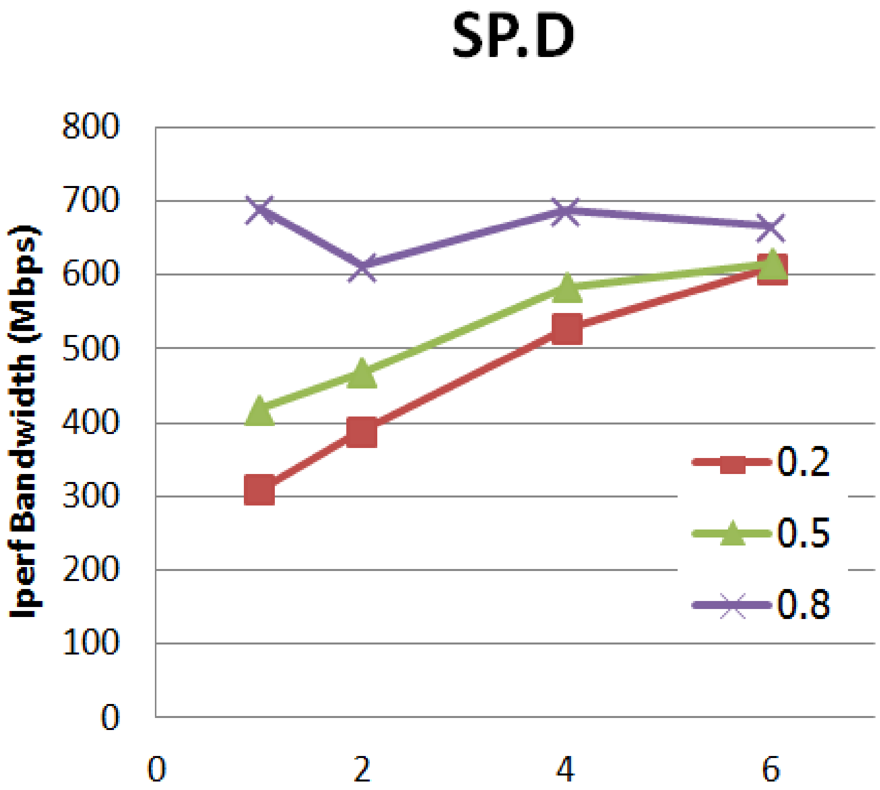


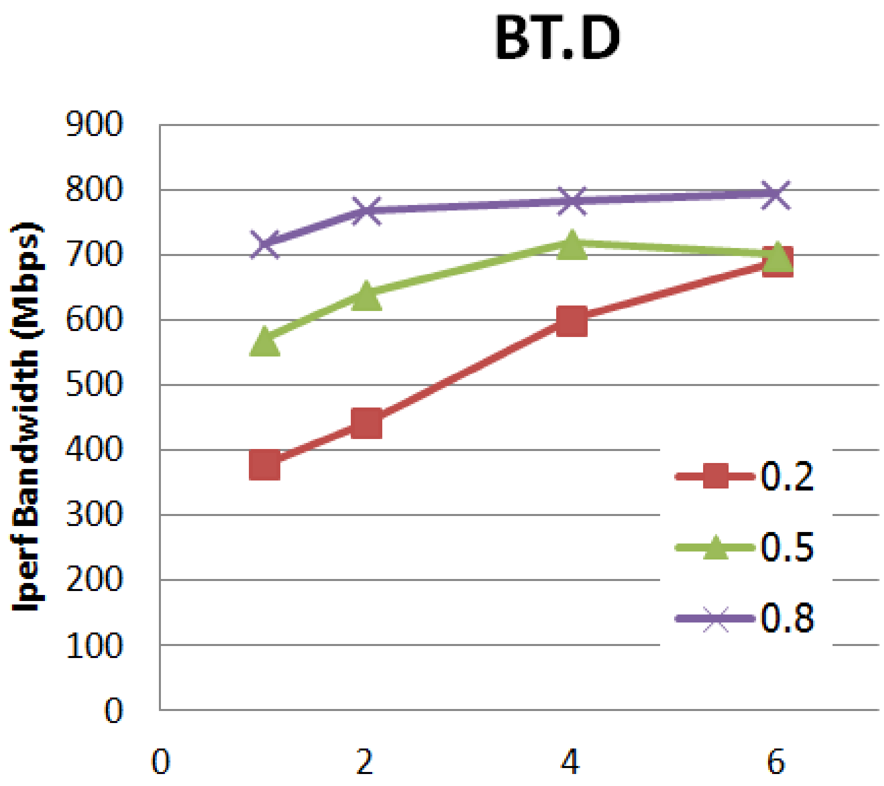


Bandwidth Reduction when applying CPUover-commit



Bandwidth Reduction when applying CPUover-commit





Other Results

- We tested TLM on MPI NPB benchmarks.
- We compared TLM to qemu-1.6.0 (released in August).
  - Developed at the same time with our approach
  - Qemu-1.6.0 has a migration thread
  - It has auto-convergence feature to periodically "stun" CPU when migration does not converge

Other Results

- Our solution takes less total migration time than that of qemu-1.6.0
  - 0.25 to 0.5 time that of qemu-1.6.0, the most recent (best) pre-copy migration mechanism
- Our solution can achieve low downtime comparable to that of qemu-1.6.0

Outline

- Introduction
- Existing Solutions
- TLM Overview
- Experimental Results
- **Conclusion**

Conclusion

- We have invented the TLM mechanism that can handle VMs with CPU and Memory intensive workloads
- TLM is Time-Bound
- Use Best Efforts to Transfer VM State
- Over-commit CPU to reduce downtime
- Better than existing pre-copy migration
- Provide basic for live Checkpointing Mechanism
- Thank you. Questions?

Time-Bounded, Thread-Based Live Checkpointing of Virtual Machines

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

Outline

- **Introduction**
- Thread-based Live Checkpointing with remote storage
- Experimental Results
- Conclusion

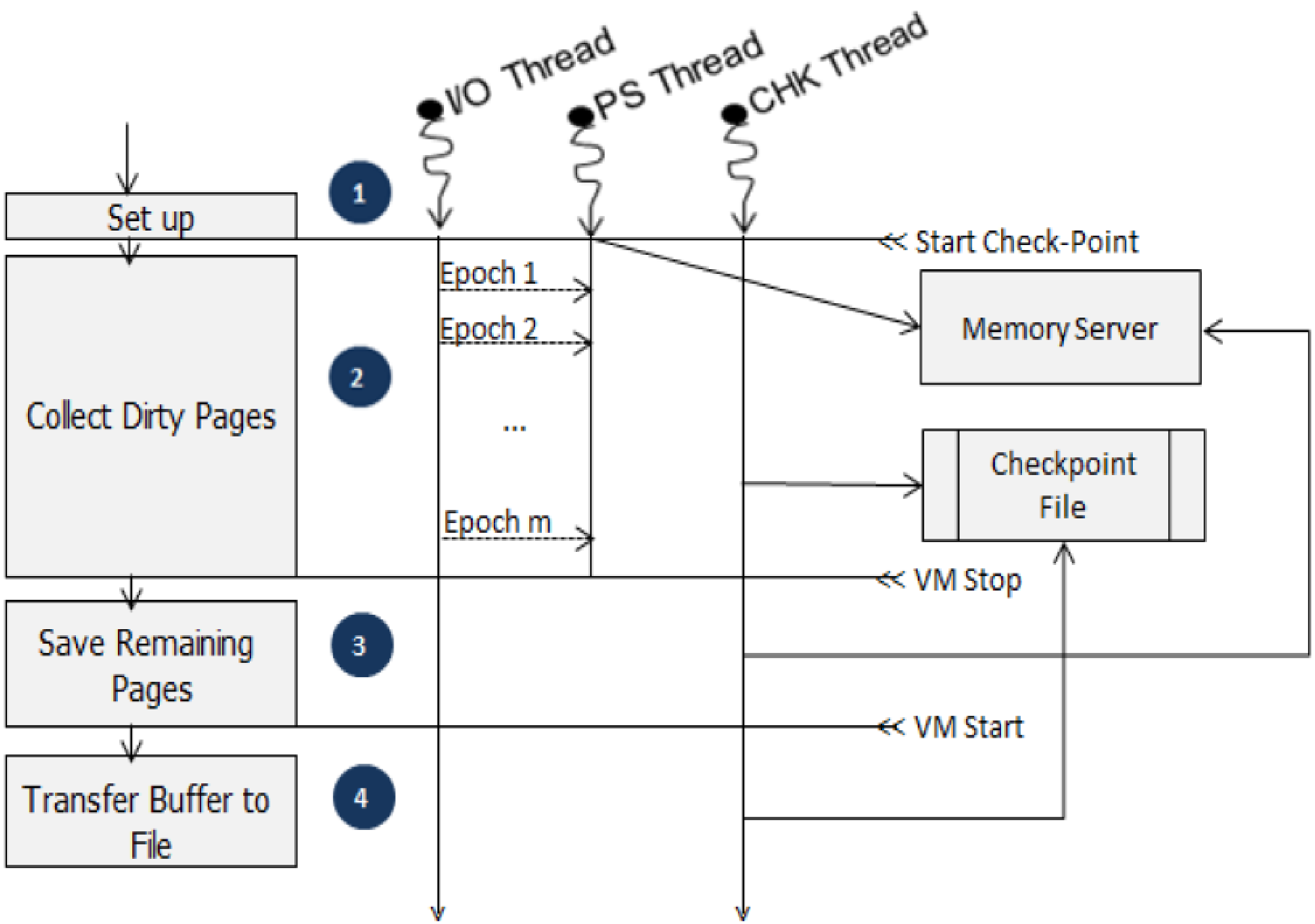
Introduction

- Checkpointing is a basic fault-tolerant mechanism for HPC applications
- Checkpointing a VM saves state of all applications running on the VM
- Checkpointing is costly
  - Collect State information
  - Save State to Remote or Local Persistent Storage
  - Hard to handle a lot of checkpoint information at the same time

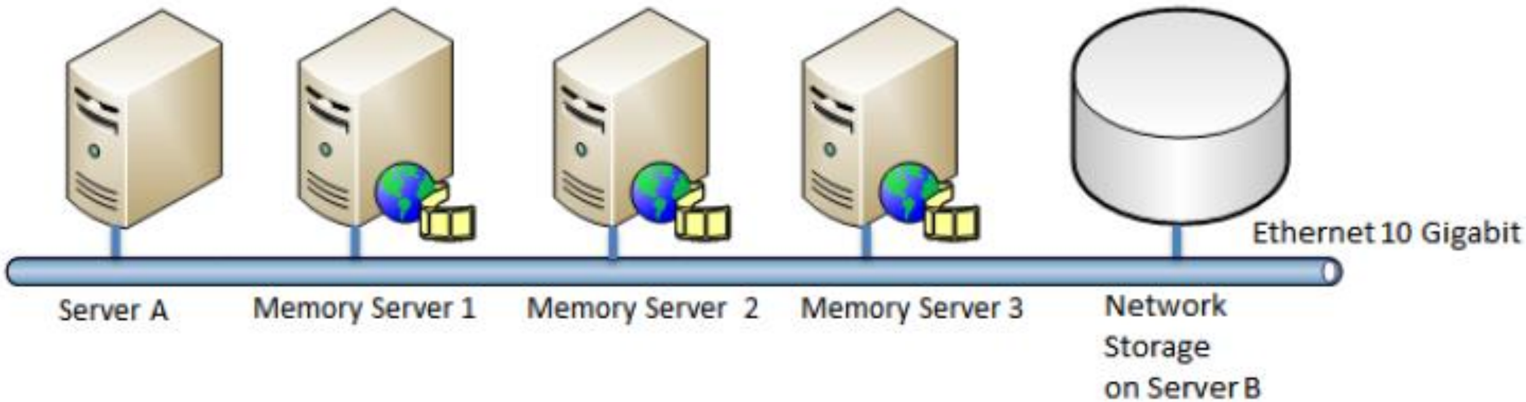
Time-bound, Thread-based Live Checkpointing

- Leverage the Time-Bound, Thread-based Live Migration approach
  - Short checkpoint time/Low downtime
- Use remote memory servers to help perform checkpointing

Time-bound, Thread-based Live Checkpointing

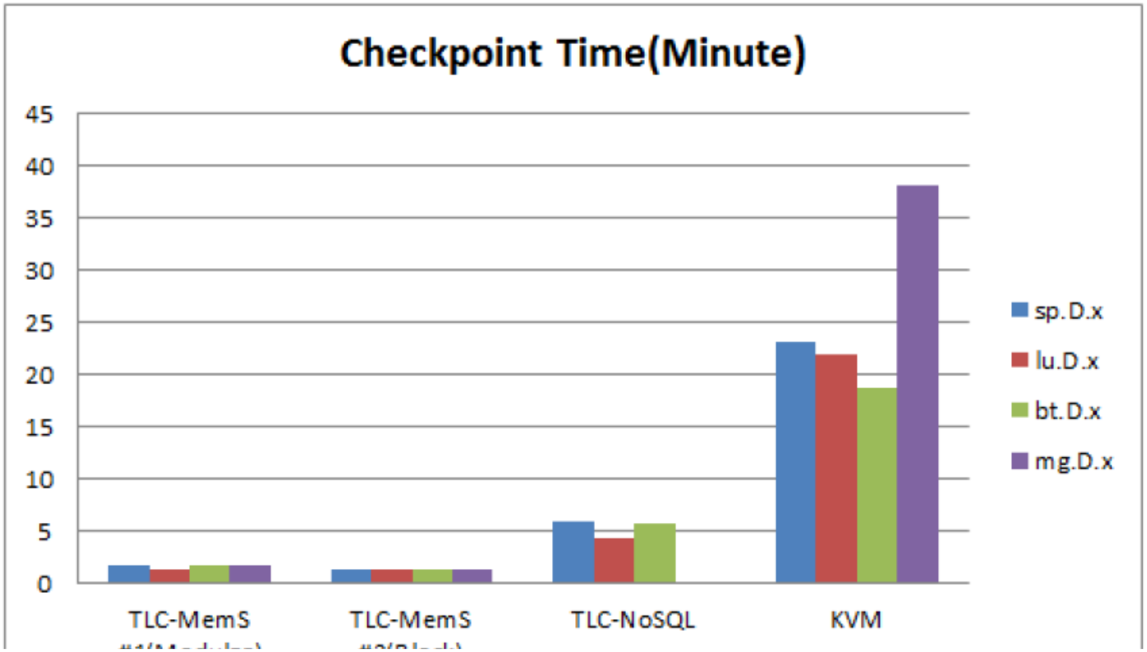


Experimental Setup



Checkpoint Time

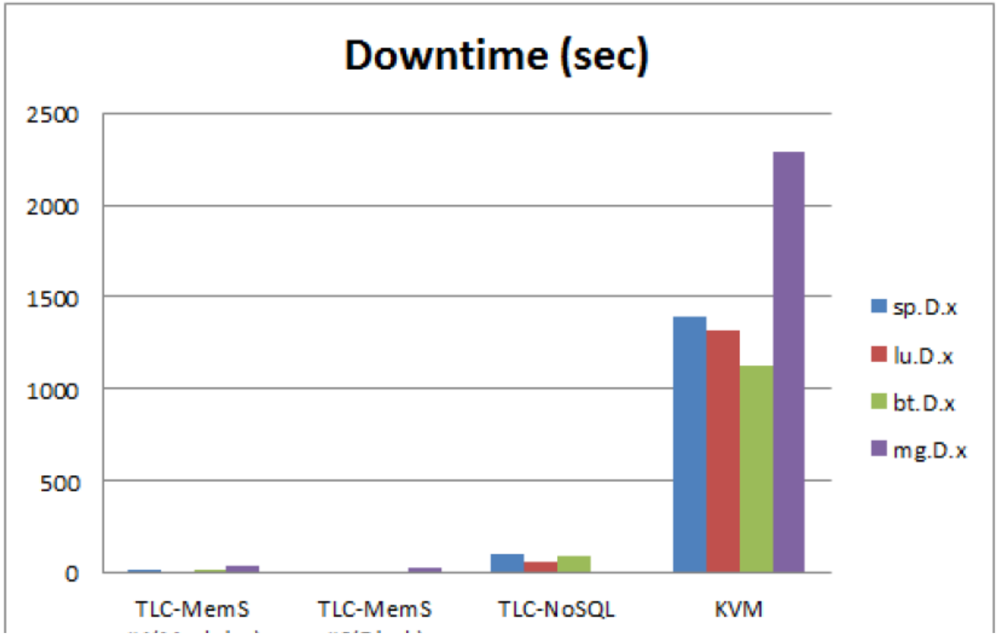
Checkpoint Time



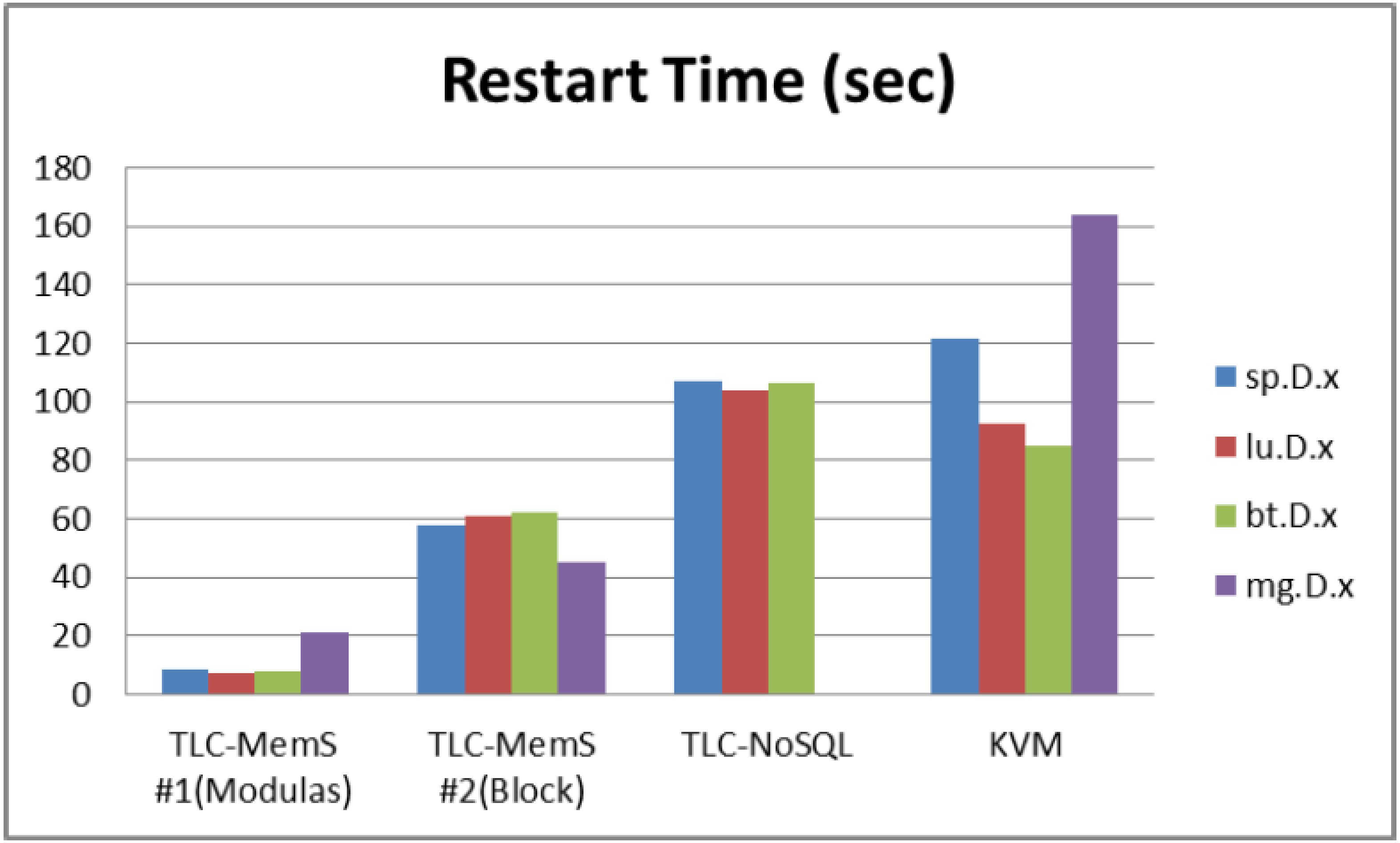
Benchmark	Checkpoint Time (Minutes)								
	TLC-MemS #1(Modulas)	TLC-MemS #2(Block)	TLC-NoSQL	KVM	TLC-MemS#1 / TLC-NoSQL	TLC-MemS#2 / TLC-NoSQL	TLC-NoSQL/ KVM	TLC-MemS#1/ KVM	TLC-MemS#2/ KVM
sp.D.x	1.72	1.34	5.90	23.20	0.29	0.23	0.25	0.07	0.06
lu.D.x	1.44	1.41	4.35	21.91	0.33	0.32	0.20	0.07	0.06
bt.D.x	1.71	1.36	5.74	18.72	0.30	0.24	0.31	0.09	0.07
mg.D.x	1.75	1.44	*	38.20	*	*	*	0.05	0.04

Downtime

Downtime (sec)



Downtime(sec)									
Benchmark	TLC-MemS #1(Modulas)	TLC-MemS #2(Block)	TLC-NoSQL	KVM	TLC-MemS#1 /TLC-NoSQL	TLC-MemS#2 /TLC-NoSQL	TLC-NoSQL/ KVM	TLC-MemS#1 /KVM	TLC-MemS#2/ KVM
sp.D.x	15.19	8.84	107	1,393.20	0.142	0.083	0.08	0.011	0.006
lu.D.x	2.63	2.13	61	1,314.60	0.043	0.035	0.05	0.002	0.002
bt.D.x	19.4	4.16	95	1,123.20	0.204	0.044	0.08	0.017	0.004
mg.D.x	38.68	29.3	*	2,292.00	*	*	*	0.017	0.013



Restart Time (sec)
180
160
140
120
100
80
60
40
20
0
TLC-MemS #1(Modulas)
TLC-MemS #2(Block)
TLC-NoSQL
KVM
sp.D.x
lu.D.x
bt.D.x
mg.D.x

Science Cloud: TU OpenStack Private Cloud

Kasidit Chanchio, Vasinee Siripoon

Vasabilab

Dept of Computer Science,

Faculty of Science and Technology,

Thammasat University

http://vasabilab.cs.tu.ac.th

vasabiLab

VIRTUALIZATION ARCHITECTURE AND SCALABLE INFRASTRUCTURE LABORATORY

DEPARTMENT OF COMPUTER SCIENCE THAMMASAT UNIVERSITY

TU science

Science Cloud

- A Pilot Project for the Development and Deployment of a Private Cloud to support Scientific Computing in the Faculty of Science and Technology, Thammasat University
- Study and develop a private cloud.
- Provide the private cloud service to researchers and staffs in the Faculty of Science and Technology.

Resources

- 5 servers
- 34 CPUs
- 136GB Memory
- 2.5TB Disk

OpenStack: Cloud Operating System

- Latest version: Grizzly
- Components:
  - Keystone
  - Glance
  - Nova
  - Neutron (Quantum)
  - Dashboard

Deployment

- Usage from July, 2013
- 17 users
- 20 active instances