Fast & Sharing-aware
Live Migration of Virtual Machines

Roja Eswaran, PhD

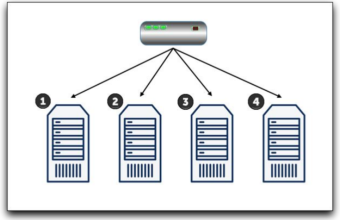
Agenda

- Motivation
- Problem Statement
- Background
- Inter-host
  - **T**emplate-aware **L**ive **M**igration of Virtual Machines (**TLM**)
  - **S**haring-aware **L**ive **M**igration of Virtual Machines (**SLM**)
- Intra-host TLM
- Future Research Directions
- Conclusion

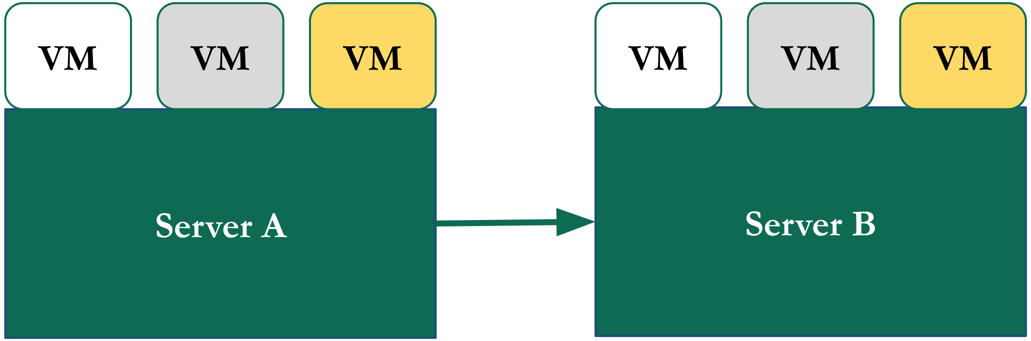
Motivation

Hardware Failure

Load Balancing



Maintenance



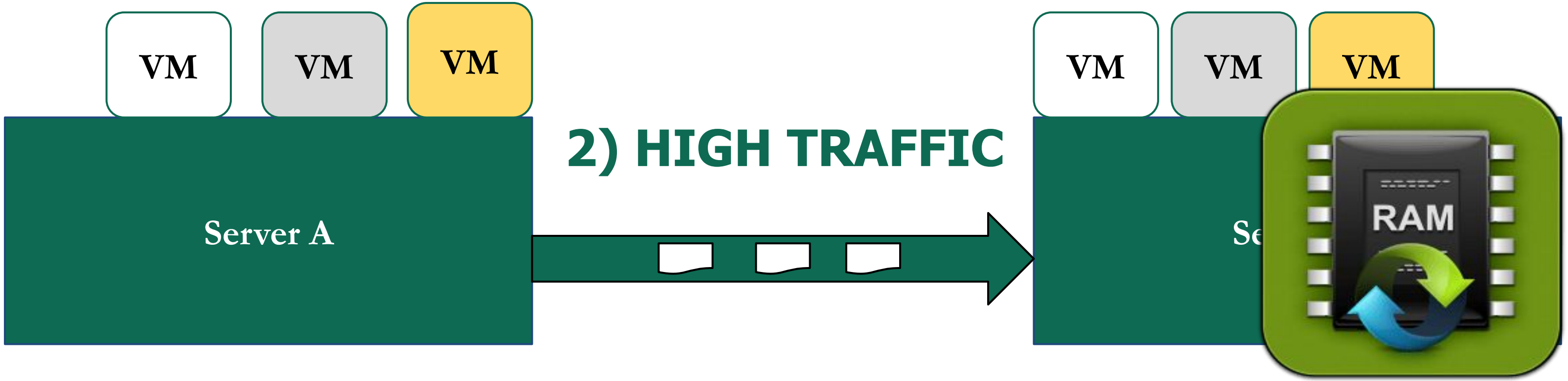
Replacing VM Manager

Live Migration is crucial technique for moving VMs around the datacenter

Problem Statement

DUPLICATE PAGES ARE THE *CULPRIT*

1) SLOW MIGRATION



2) HIGH TRAFFIC

3) MEMORY BLOWUP

em Statement

– Current Live Migration is unaware of the pre-existing **deduplication optimization**

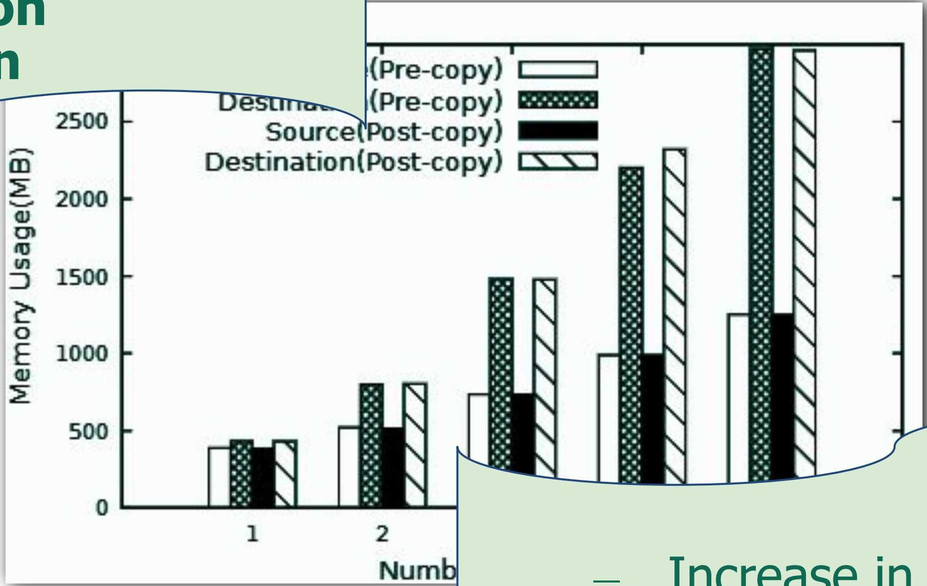


Fig: Memory Usage of V

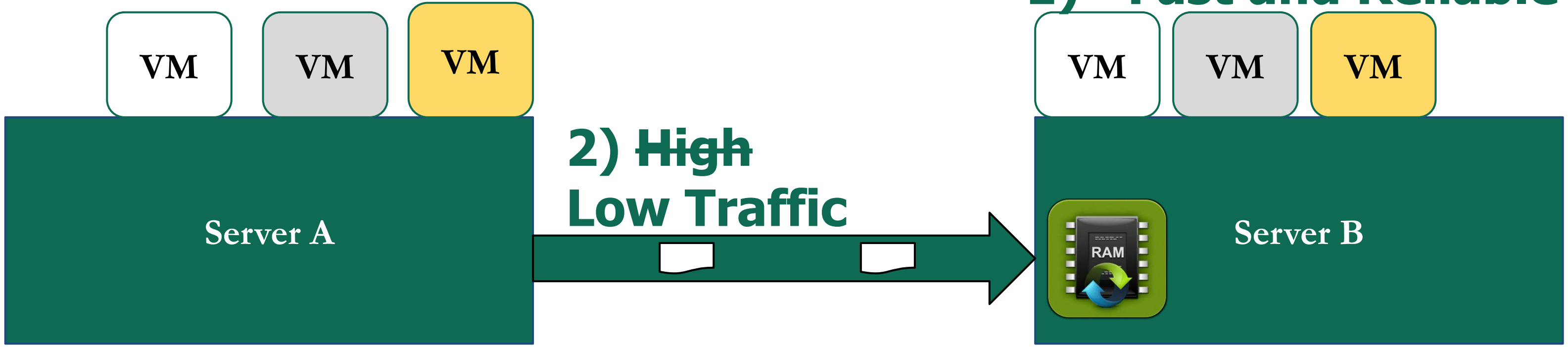
– Increase in migration time and network traffic

SOLUTION

Cognizant of Host/Hypervisor Page Sharing Optimization!

~~SLOW MIGRATION~~

1) Fast and Reliable



2) ~~High~~ Low Traffic

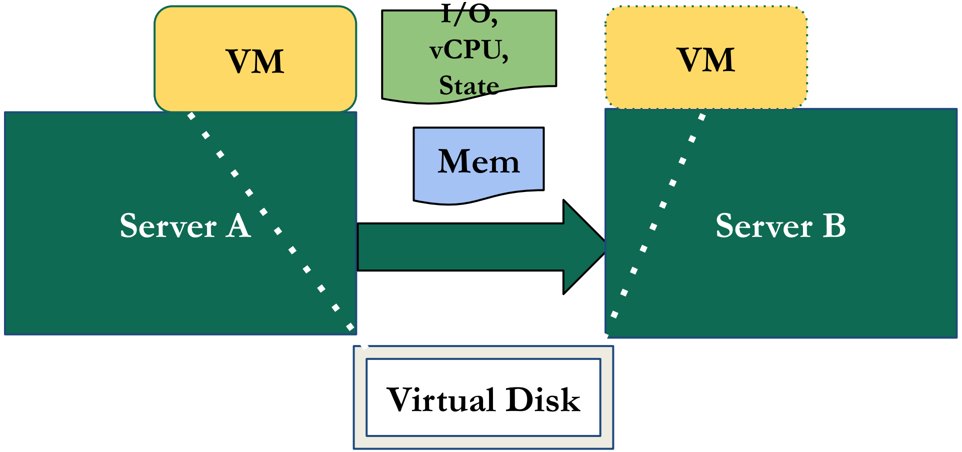
3) Efficient memory utilization (BONUS!)

Agenda

- Motivation
- Problem Statement
- **Background**

Inter-host Live Migration

- Moving VMs between two different hosts
  - Useful for load-balancing, maintenance, and hardware failures



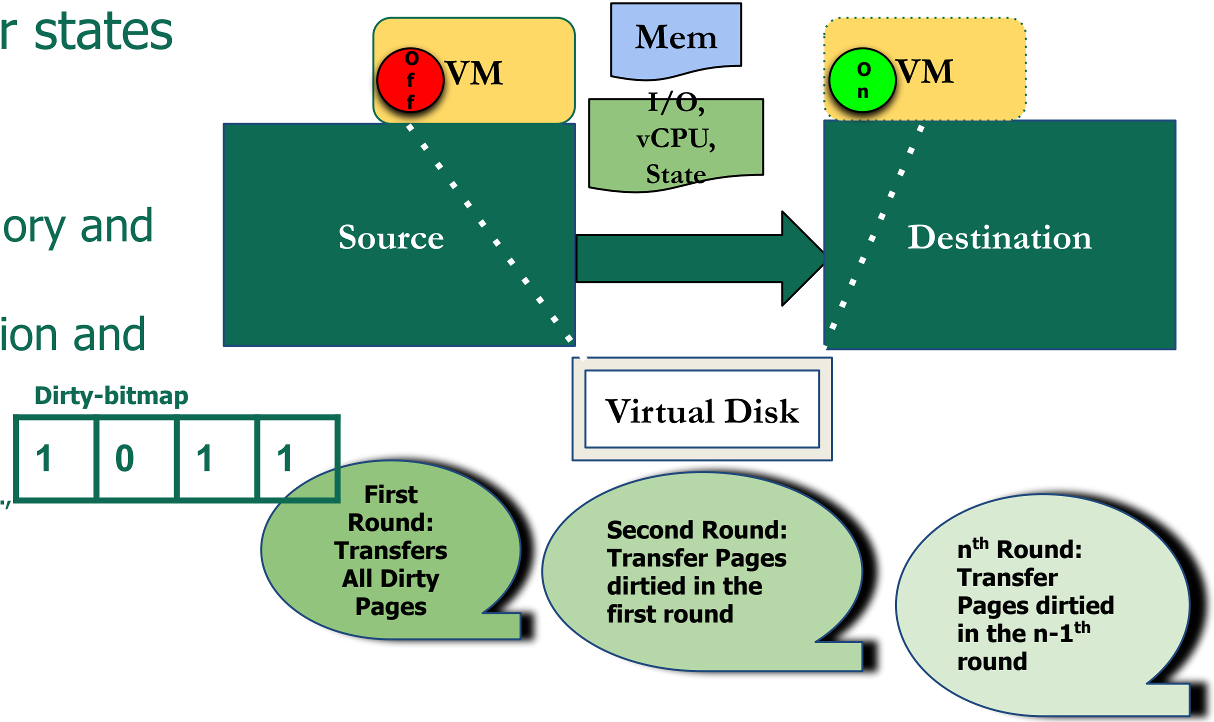
Background(1/6)

Pre-copy Live Migration

- Pre-copy
  - Transfer ***memory first*** and then transfer states

Downtime

- Transfer remaining memory and state
- **Resume** VM at destination and **stop** VM at source



Dirty-bitmap

1	0	1	1

First Round: Transfers All Dirty Pages

Second Round: Transfer Pages dirtied in the first round

nth Round: Transfer Pages dirtied in the n-1th round

Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I. and Warfield, A., 2005, May. *Live migration of virtual machines.* In Proceedings of conference on Symposium on Networked Systems Design & Implementation.

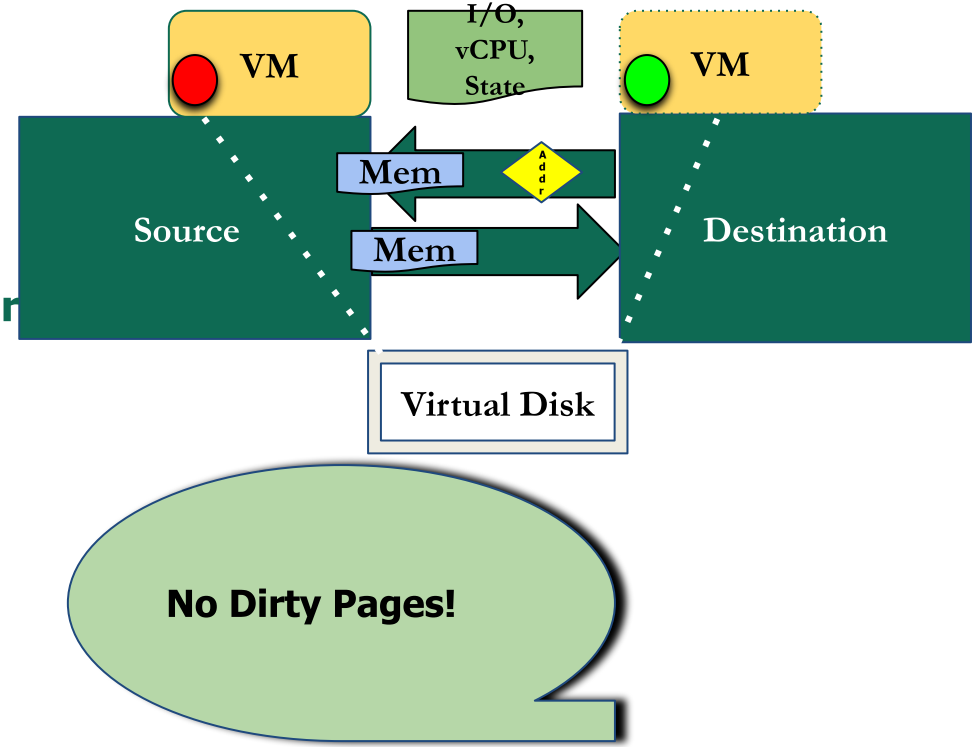
Post-copy Live Migration

Post-copy

- Transfer ***states first resume at destination*** and then transfer memory

Two mechanism for memory transfer

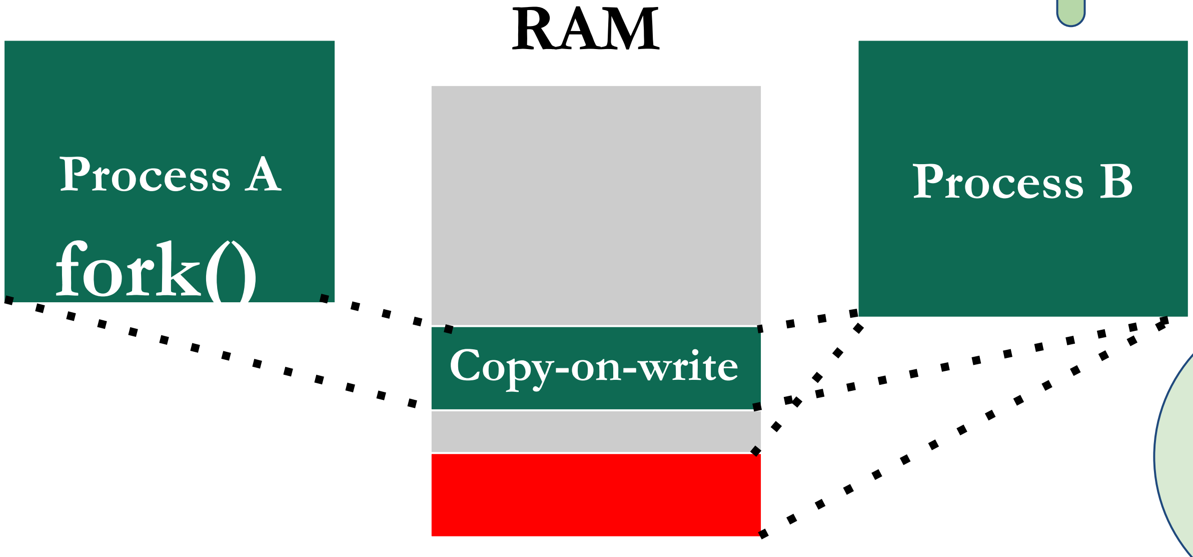
- Demand Paging
- Active Pushing



Hines, M.R., Deshpande, U. and Gopalan, K., 2009. *Post-copy live migration of virtual machines.* ACM SIGOPS operating systems review.

Fork

Simple Copy-on-Write Optimization



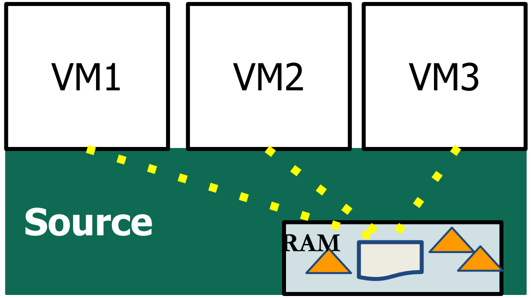
Faster Process Creation

Fork() creates a child process with ***copy-on-write*** sharing with the parent memory pages

Background(4/6)

VM Templating

- VM templating is a technique to quickly instantiate multiple lightweight VMs from **a shared *copy-on-write* (COW) template**



- The additional pages dirtied by the VMs are referred to as ***delta***

Advanced Copy-on-Write Optimization

Faster VM Creation

Kernel Samepage Merging (KSM)

Each virtual page has its physical page

Fig:1 Without KSM

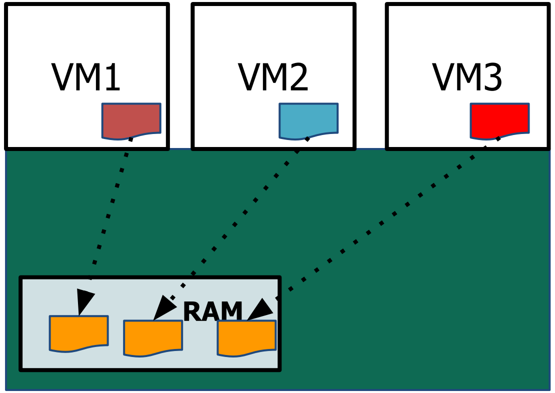
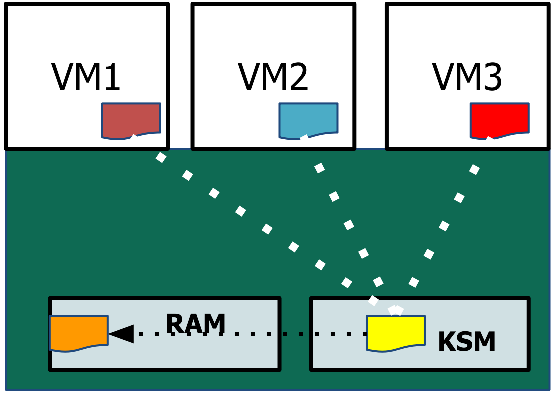


Fig:2 With KSM



KSM merges duplicate pages and create a single physical page with *copy-on-write* mapping

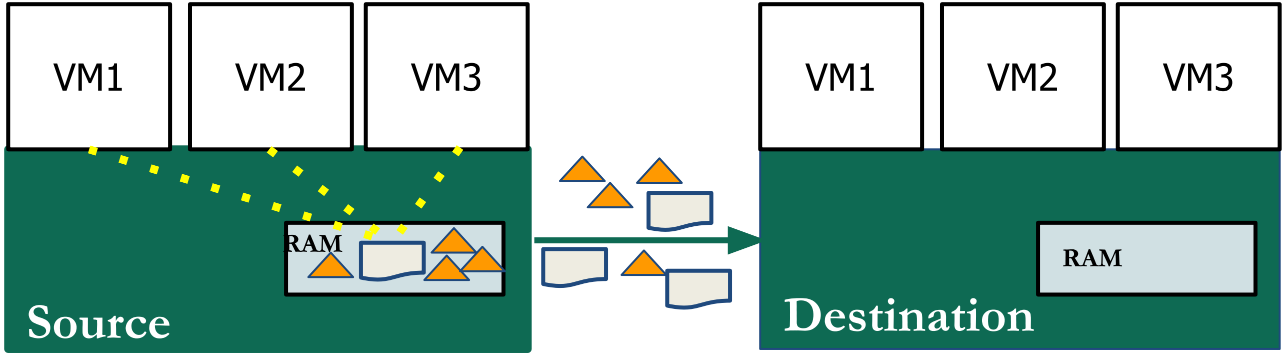
Arcangeli, A., Eidus, I. and Wright, C., 2009, July. *Increasing memory density by using KSM.* In Proceedings of the linux symposium.

Advanced Copy-on-Write Optimization

Agenda

- Motivation
- Problem Statement
- Background
- Inter-host
  - **T**emplate-aware **L**ive **M**igration of Virtual Machines (**TLM**) [Edgecomm: SEC’23]

Key Insight: TLM

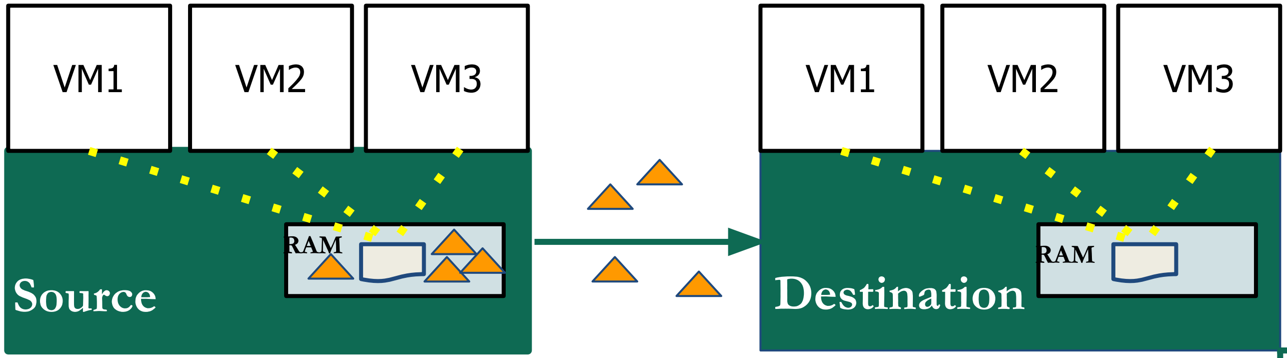


- Current live migration *lacks the awareness of the VM Template*
  - Transfers ***delta and the base template***
    - Breaks the template
  - Increases ***network-traffic***
  - Increases ***Total Migration Time***

Inefficient Live migration

Not fast and not Reliable
Not transferring fewer pages
Inefficient memory usage
Breaks the template

Design: TLM



Dirty-bitmap

0	0	0	0

1. At source, while booting up templated VMs, initialize dirty-bitmap tracking

2. Copy the template file to destination before migration

3. Transfer Delta by looking up the dirty-bitmap

Dirty-bitmap

1	0	1	1

Eswaran, R., Yan, M. and Gopalan, K., 2023, December. Template-Aware Live Migration of Virtual Machines. In 2023 IEEE/ACM Symposium on Edge Computing (SEC) (pp. 336-340). IEEE Computer Society.

Evaluation Setup

CPU: Intel Xeon E5-2620 v2 processors.

Mem: 128GB DRAM

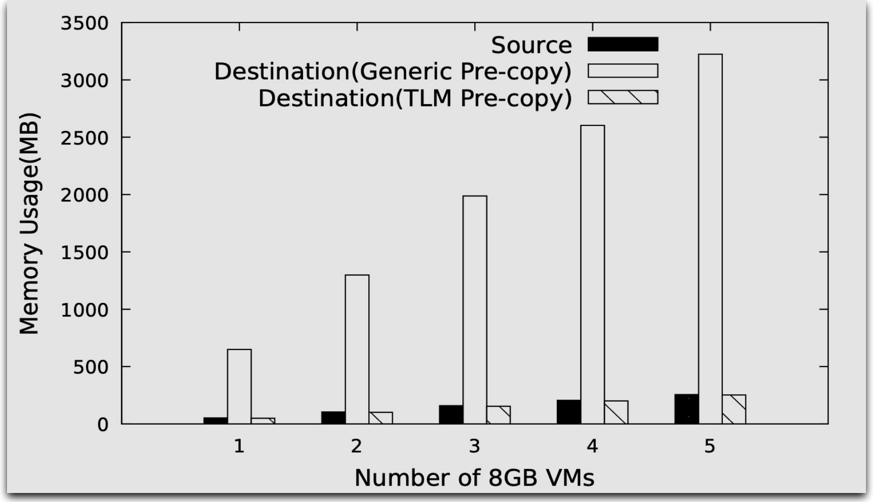
Hypervisor: QEMU/KVM

Host and Guest Kernel: linux-4.10.1

Live Migration technique: Pre-copy

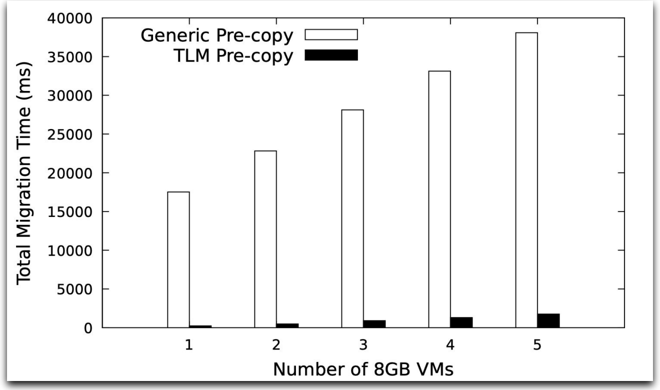
Generic - Traditional Pre-copy

Evaluation: Memory Footprint

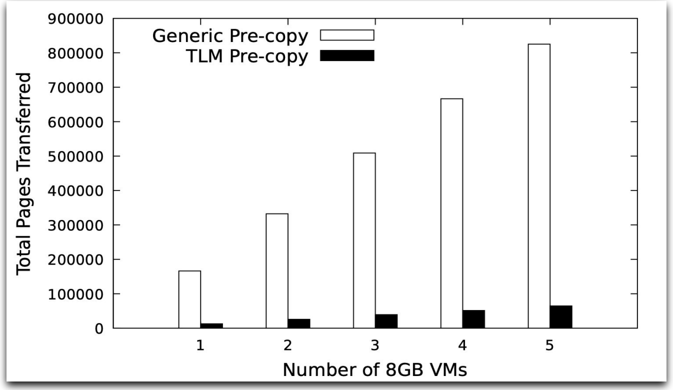


The source and destination have the same memory usage
retaining the templating benefits with the help of TLM

Evaluation: Total Migration Time & Total Pages Transferred



96% reduction in total migration time



93% reduction in total pages transferred

Summary

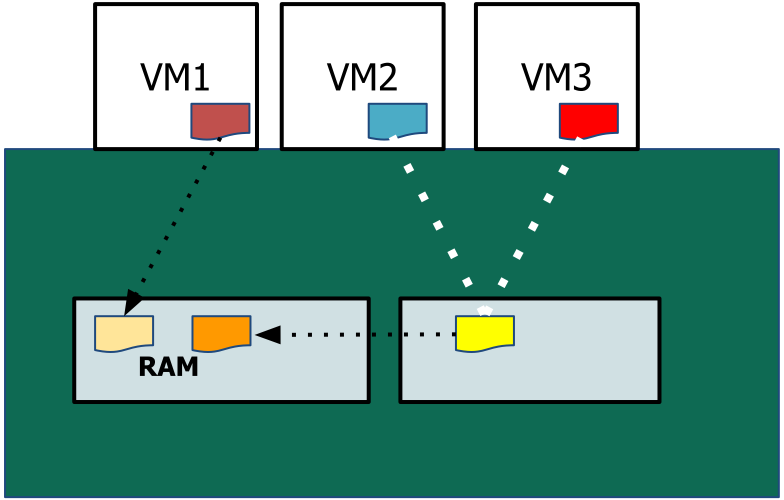
- Besides reducing the memory footprint, TLM
  - *Improves the total migration time and network-traffic*

- Limitations:
  - Works only for ***Templated VMs and unaware of KSM or Fork***

Agenda

- Motivation
- Problem Statement
- Background
- Inter-host
  - **T**emplate-aware **L**ive **M**igration of Virtual Machines (**TLM**) **[Edgecomm: SEC'23]**
  - **S**haring-aware **L**ive **M**igration of Virtual Machines (**SLM**) **[CCGRID'24]**

Key Insight: SLM



Page	Virtual Page Number (VPN)	Physical Frame Number (PFN)
🟦	0x11	**0xAA**
🟥	0x22	**0xAA**
🟫	0x33	0xBB

Merged pages have the ***same physical frame numbers***

Eswaran, R., Yan, M. and Gopalan, K., Tackling memory footprint expansion during live migration of virtual machines," in Proc. of International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2024

Existing Solution: Content-based Hashing

- ***Compute the hash*** of the page content to find the duplicate pages thereby avoiding their retransmission
  - *Compute-intensive*
  - *Unaware of existing COW Pages*

Deshpande, U., Wang, X. and Gopalan, K., 2011, June.

Live gang migration of virtual machines.

In Proceedings of international symposium on High performance distributed computing.

Zhang, X., Huo, Z., Ma, J. and Meng, D., 2010, September.

Exploiting data deduplication to accelerate live virtual machine migration.

In IEEE international conference on cluster computing.

Chiang, J.H., Li, H.L. and Chiueh, T.C., 2013.

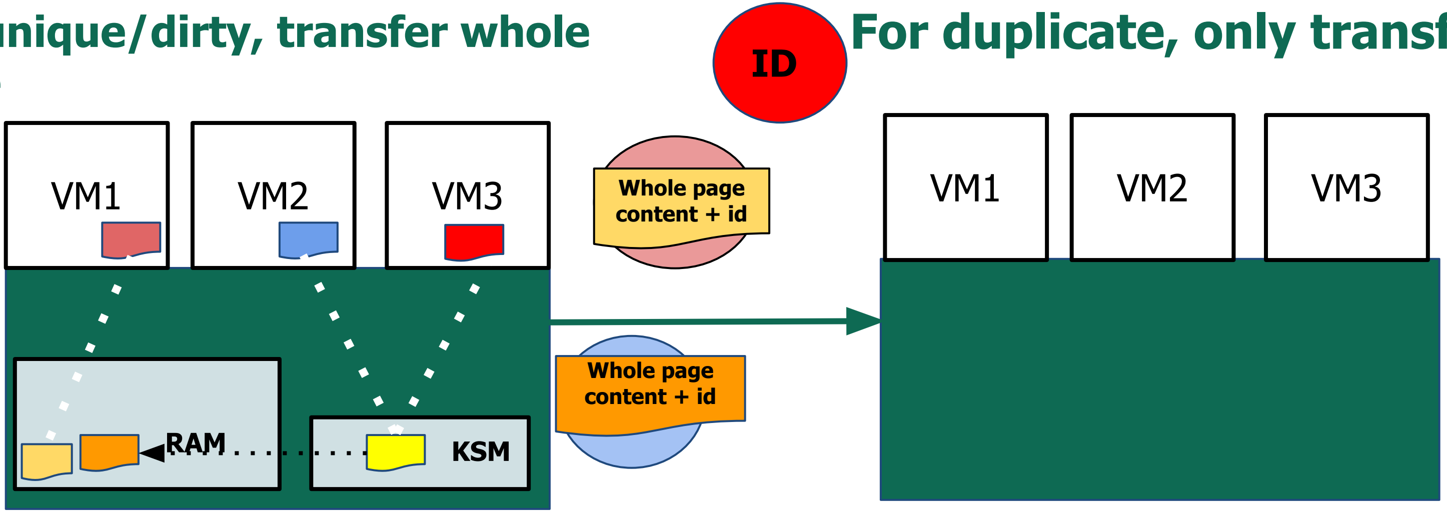
Introspection-based memory de-duplication and migration.

ACM SIGPLAN Notices

SLM Source

For unique/dirty, transfer whole page

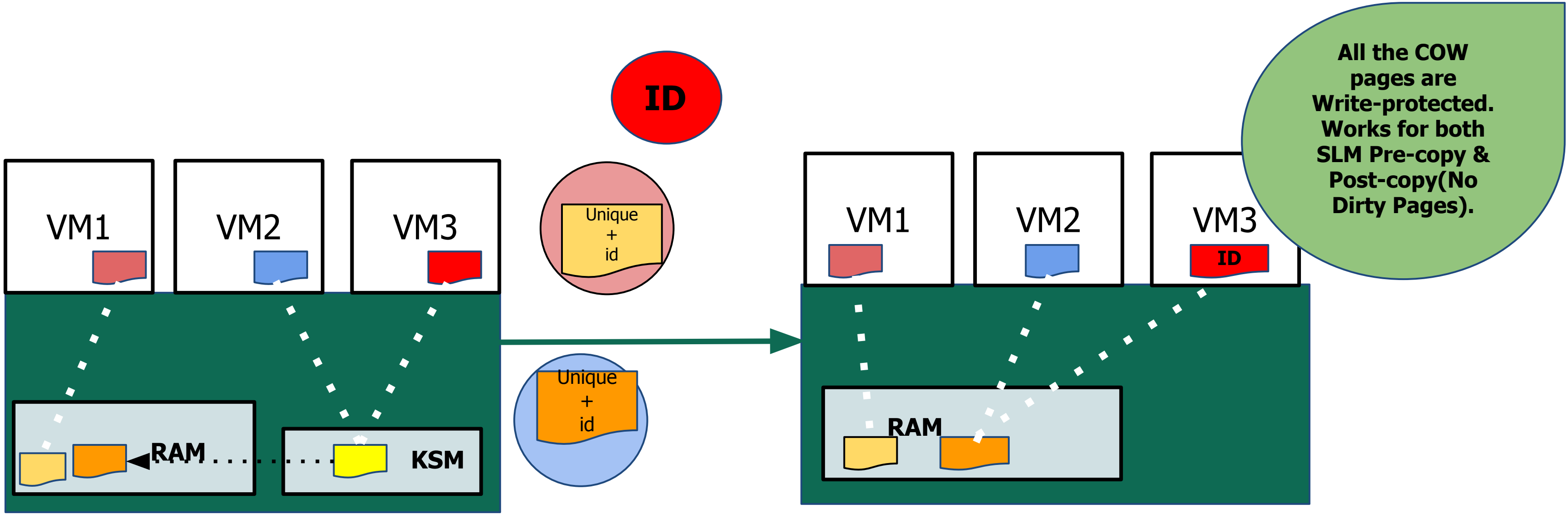
For duplicate, only transfer ID



Page	Virtual Page Number (VPN)	Physical Frame Number (PFN)	PageType
🟥	0x11	**0xAA** 🟨	**Unique**
🟦	0x22	**0xBB** 🟧	**Unique**
🟥	0x33	**0xBB** 🟧	**Duplicate**

SLM Destination

Unique/Dirty pages have unique RAM offsets to write the page content at the destination



All the COW pages are Write-protected. Works for both SLM Pre-copy & Post-copy(No Dirty Pages).

Duplicate pages COW map into the unique page content using the identifier

Evaluation Setup

CPU: Intel Xeon E5-2620 v2 processors

Mem: 128GB DRAM

Hypervisor: QEMU/KVM

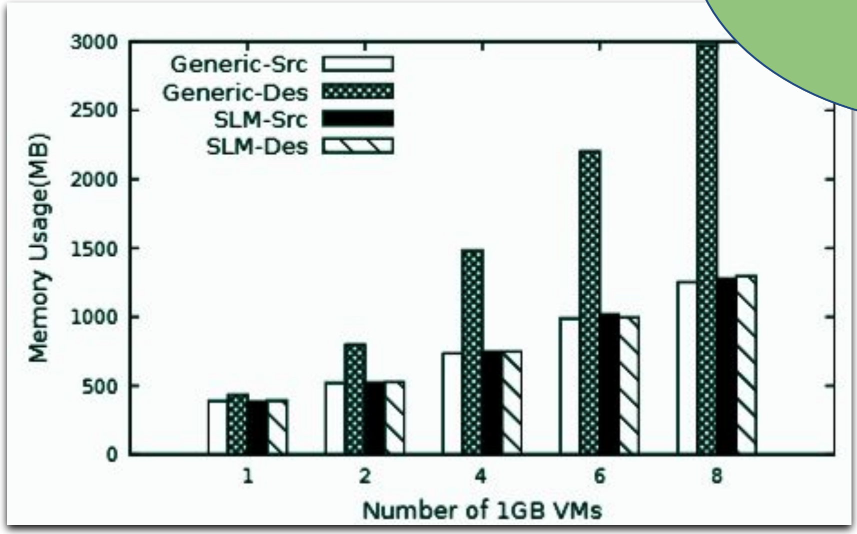
Host and Guest Kernel: linux-4.10.1

Live Migration technique: Pre-copy, Post-copy

Generic - Traditional Pre/Post copy

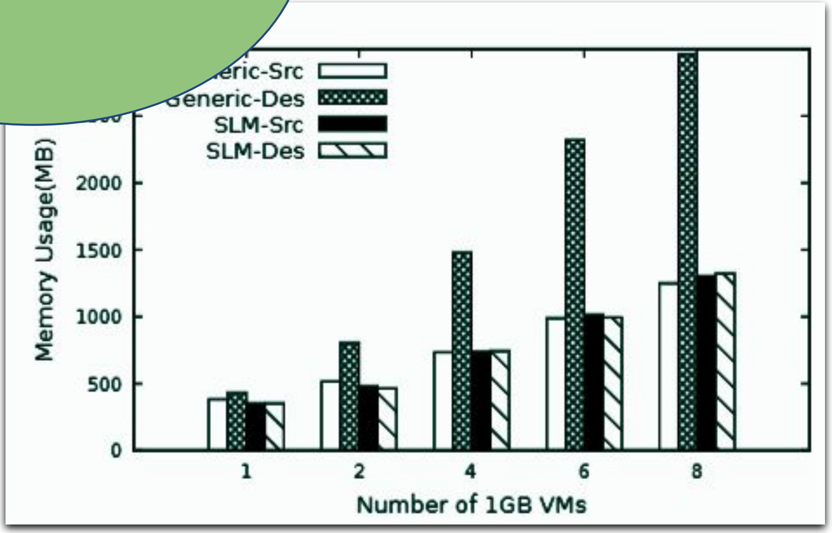
Evaluation: Memory Footprint

Fig 1: Pre-copy

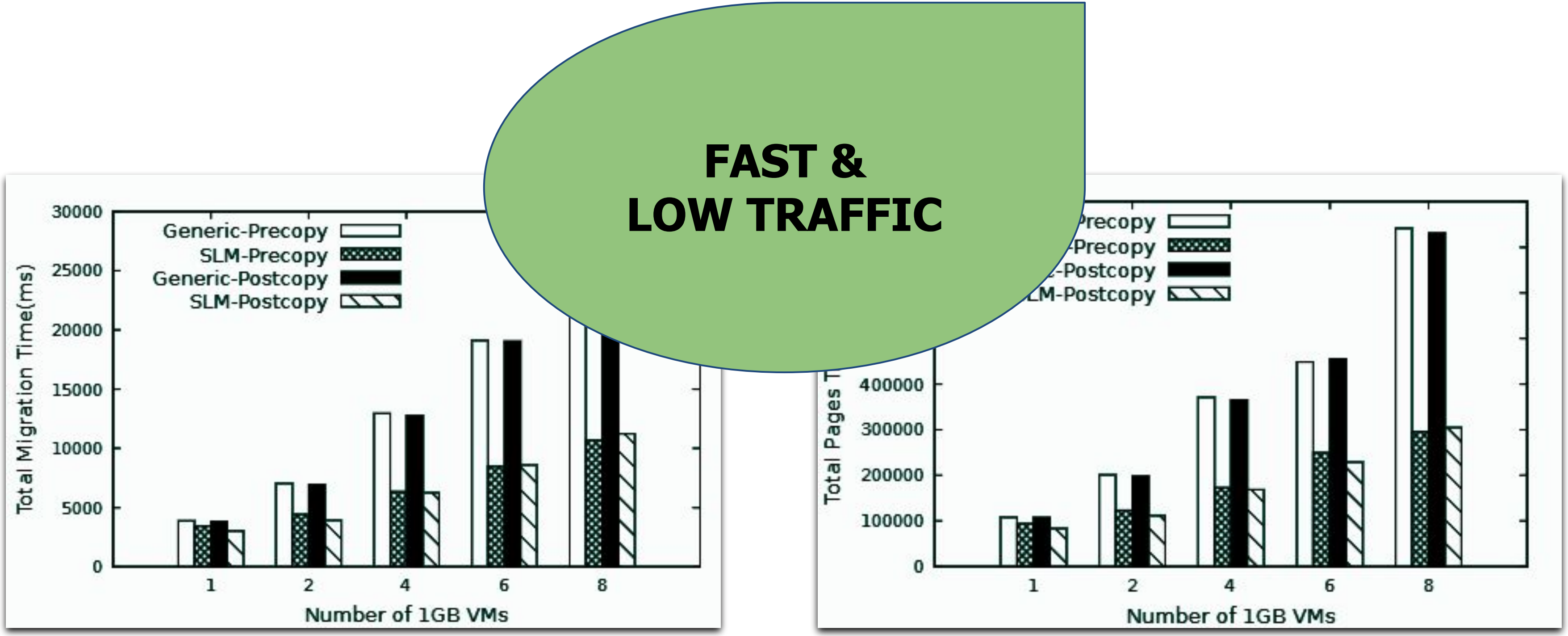


NO MEMORY BLOW-UP

2: Post-copy



Evaluation: Total Migration Time & Total Pages Transferred



FAST &
LOW TRAFFIC

59% reduction in total migration time

62% reduction in total pages transferred

Summary

- SLM retains the ***COW shared pages irrespective of mechanisms***
  - ***Fork, KSM*** and others
  - ***Works for both pre-copy and post-copy live migrations***

- SLM works for both ***regular non-templated and templated VMs***
  - besides reducing the total migration time and total pages transferred

- Both Iperf and Redis shows that ***SLM doesn't introduce any additional performance overhead***
  - Infact, compared to the baseline it finishes the migration faster due to less page transfer.

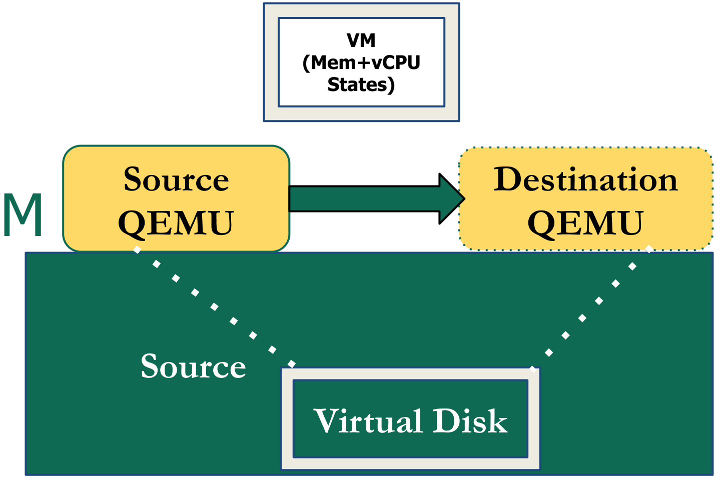
Agenda

- Motivation
- Problem Statement
- Background
- Inter-host
  - Template-aware Live Migration of Virtual Machines (TLM) [Edgecomm: SEC’23]
  - Sharing-aware Live Migration of Virtual Machines (SLM) [CCGRID’24]
- Intra-host TLM **[TPDS’24 under-review]**

Intra-host Live Migration

- Moving VMs from one QEMU to another within the same host

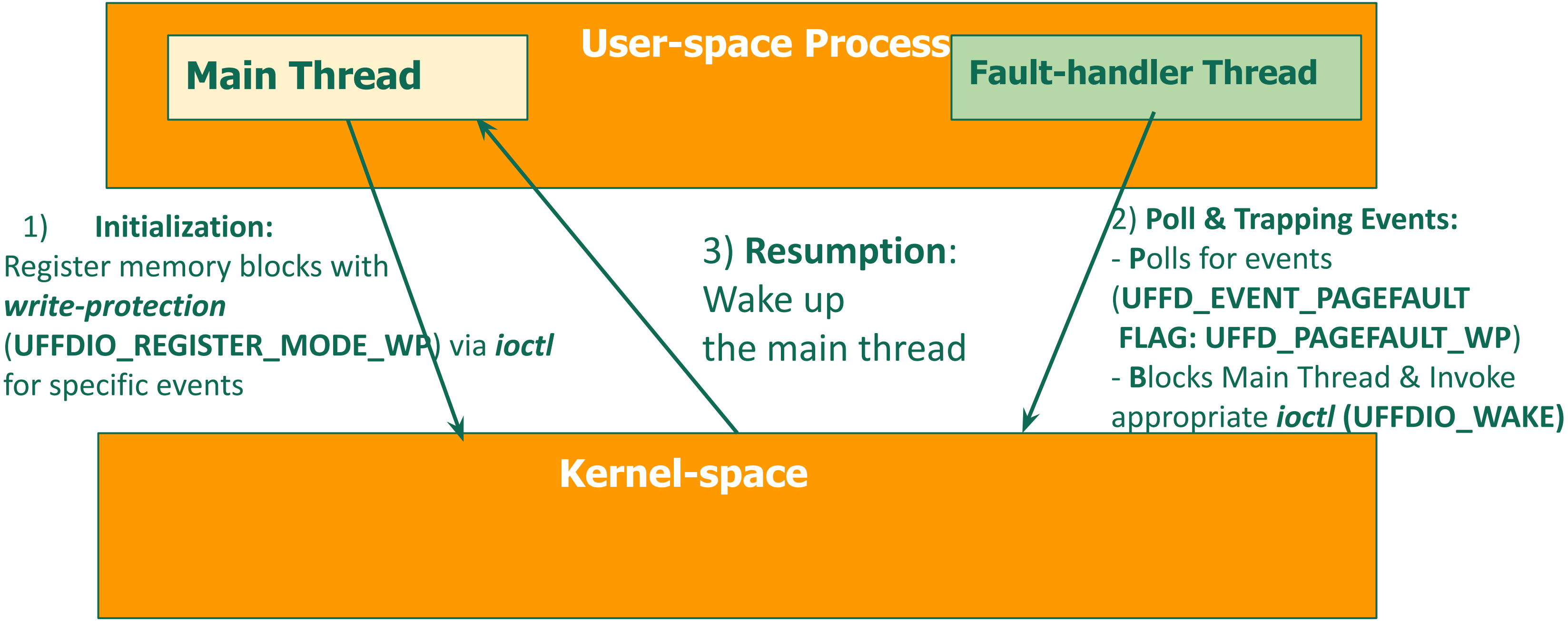
- Useful for live replacement of VM Manager process (***QEMU***)
  - New Features and Functionalities
  - Compatibility updates
  - Performance improvements



BACKGROUND (1/2)

Userfaultfd

- Userfaultfd mechanism lets the user-space process (**via fault-handler thread**) handles memory-related events especially pagefault without ***kernel intervention***



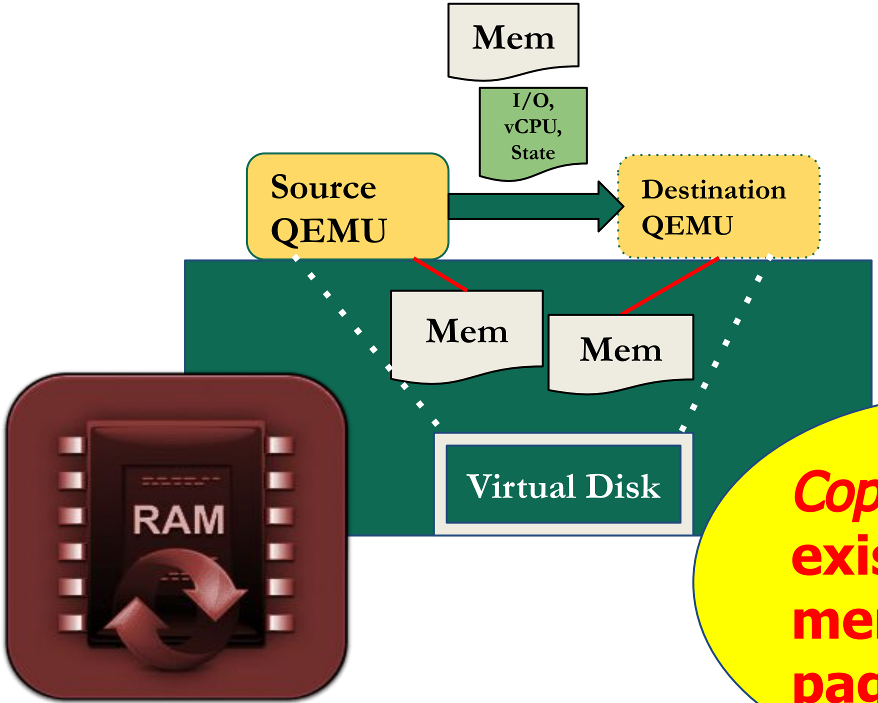
1) **Initialization:**
Register memory blocks with
write-protection
(**UFFDIO_REGISTER_MODE_WP**) via ***ioctl***
for specific events

3) **Resumption:**
Wake up
the main thread

2) **Poll & Trapping Events:**
- **P**olls for events
(**UFFD_EVENT_PAGEFAULT**
FLAG: UFFD_PAGEFAULT_WP)
- **B**locks Main Thread & Invoke
appropriate ***ioctl*** **(UFFDIO_WAKE)**

BACKGROUND
(2/2)

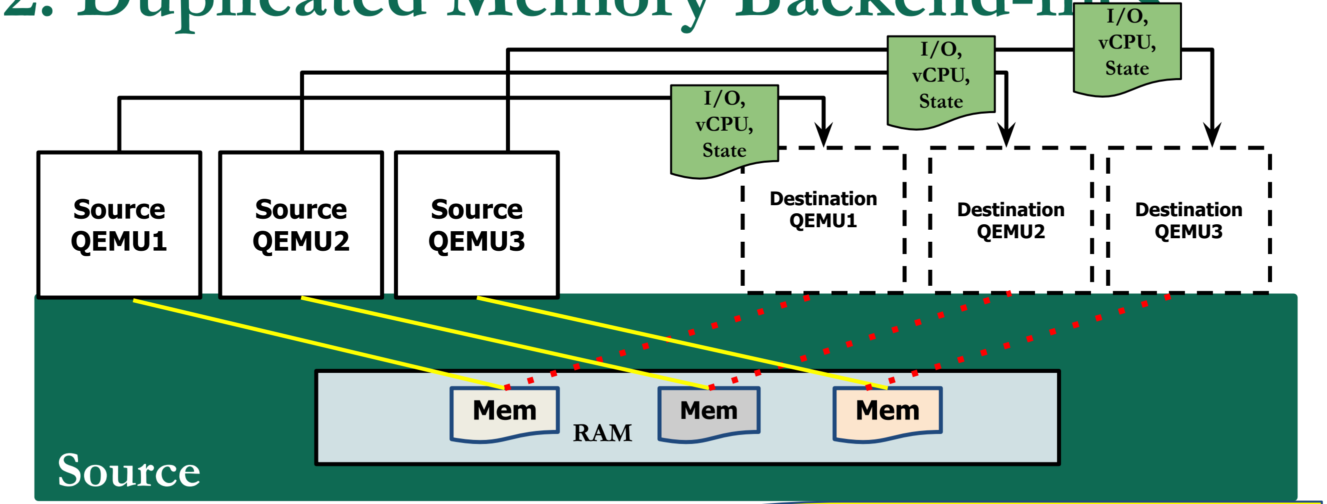
Problem 1: Copying Existing Memory Pages



Copying **existing memory pages again!**

PROBLEM (1/3)

Problem 2: Duplicated Memory Backend-files



Using "*by-pass*" flag, the transfer of memory can be eliminated

- High ***Memory Usage*** due to duplicated back-end file
- Essential to ***integrate VM Templating*** for efficient memory usage

PROBLEM (2/3)

Problem 3: Unwanted Copying of Delta Pages

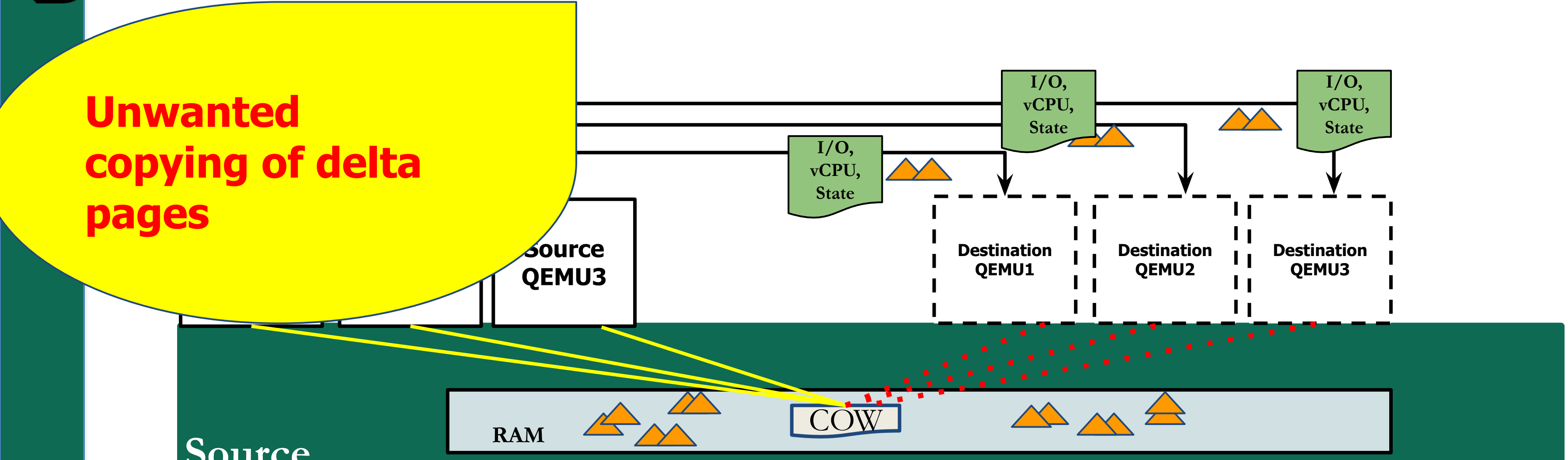


Fig:1 Intra-host VM Templating

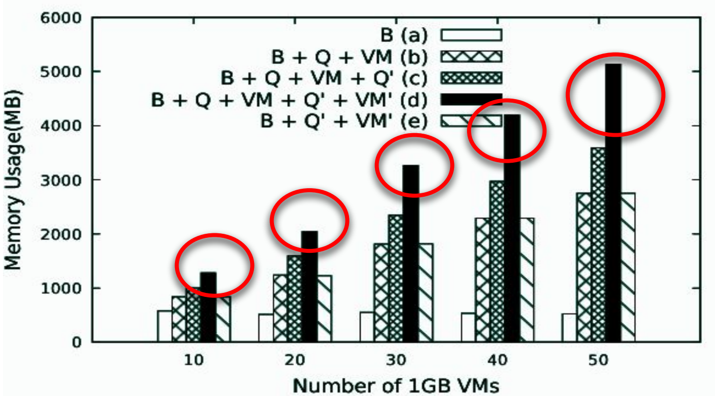
a:*Base* System memory Usage

b:a+*Source QEMU+VM*

c: b+*Destination QEMU (Listenin*
Mode)

d: c+*VM'* **(After Migration)**

e - a+*Destination QEMU+VM'* **(After Killing Source)**



PROBLEM (3/3)

Intra-host TLM Design: Phase 1 - Initialization

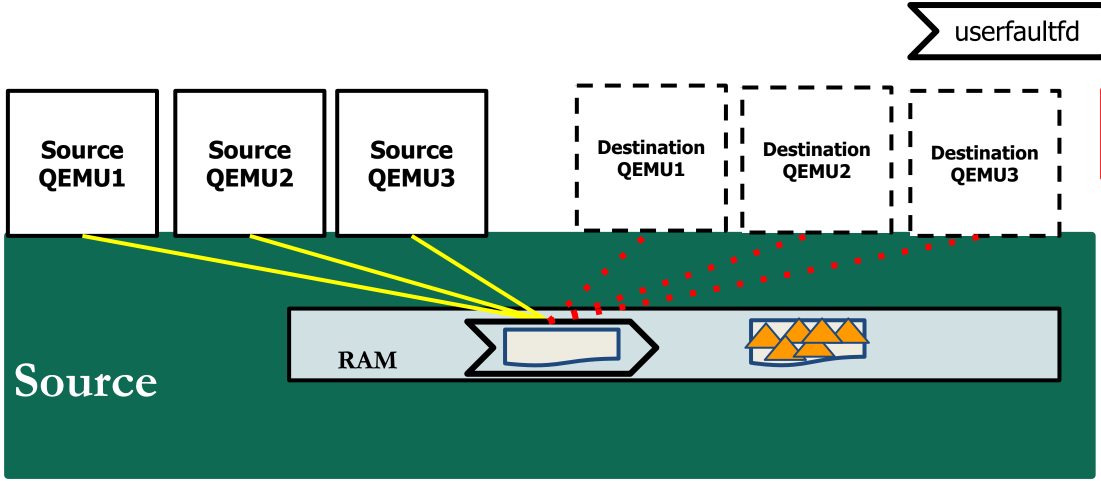


Fig1*:* Userfaultfd ***traps and redirects*** all the writes to the dedication location

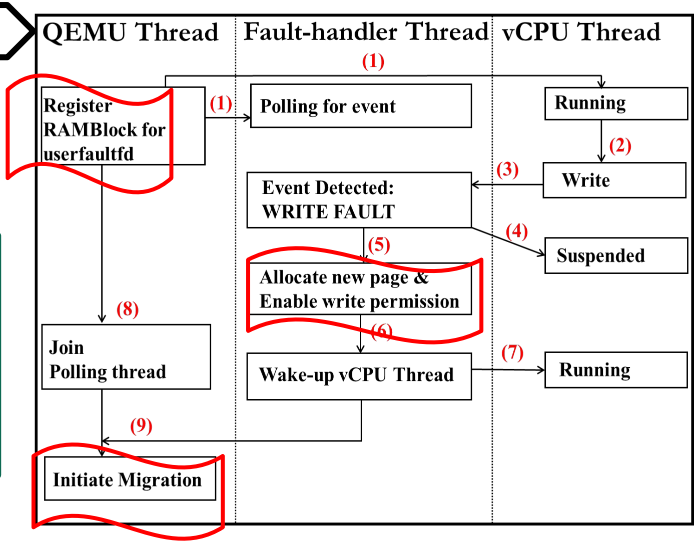


Fig2*:* Workflow of Initialization Phase

Intra-host TLM
(1/6)

Intra-host TLM Design: Phase 2 - Migration

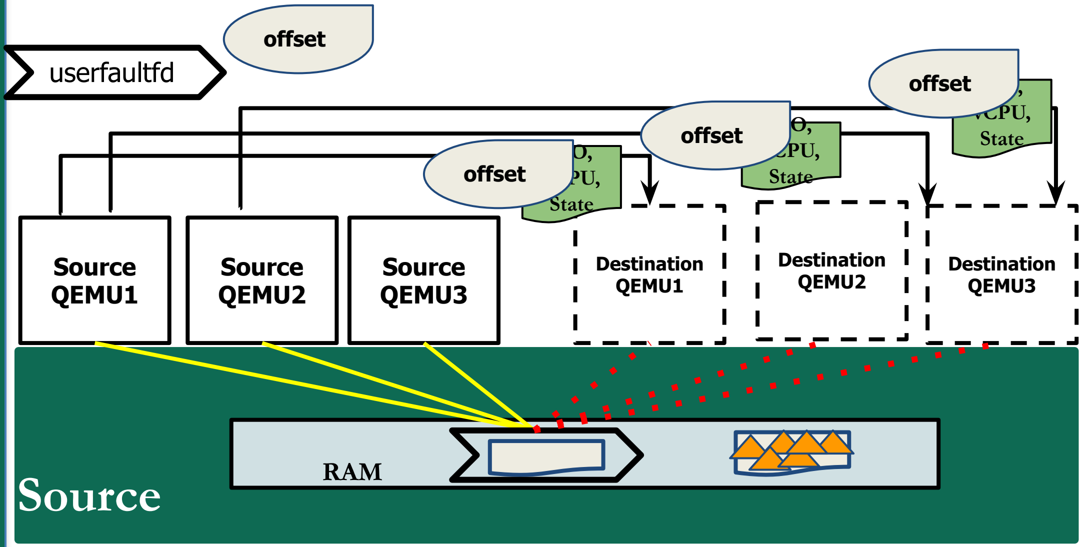


Fig1*:* Intra-host TLM ***transfers the ownership of pages*** instead of copying them again by only sending the ***vCPU states and offsets***

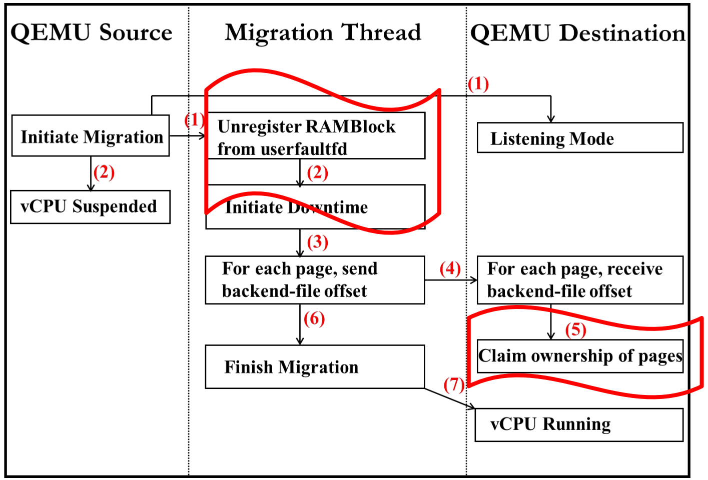


Fig2*:* Workflow of Migration

Performance Evaluation: Intra-host TLM

Fig 1a: Memory Usage - Generic TLM

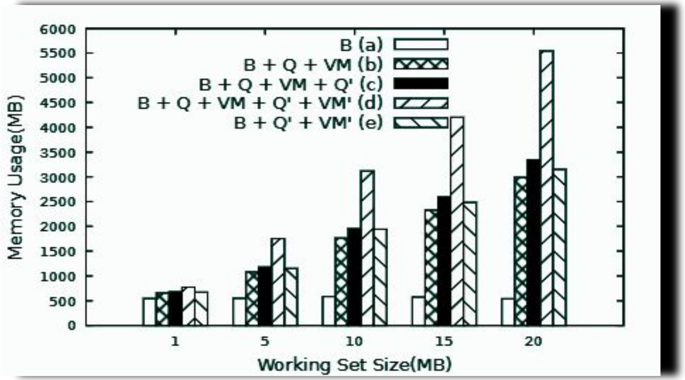
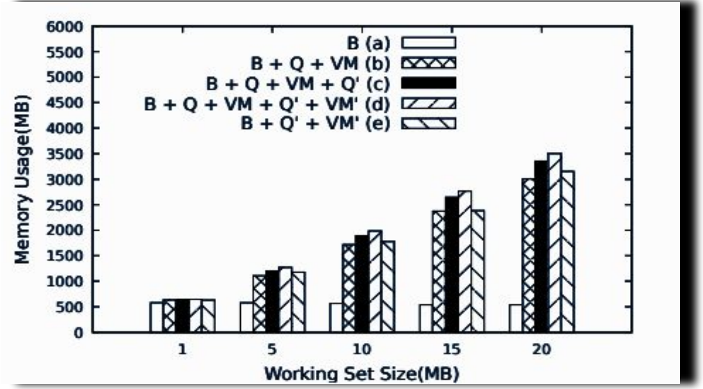


Fig 1b: Memory Usage - Intra-host TLM



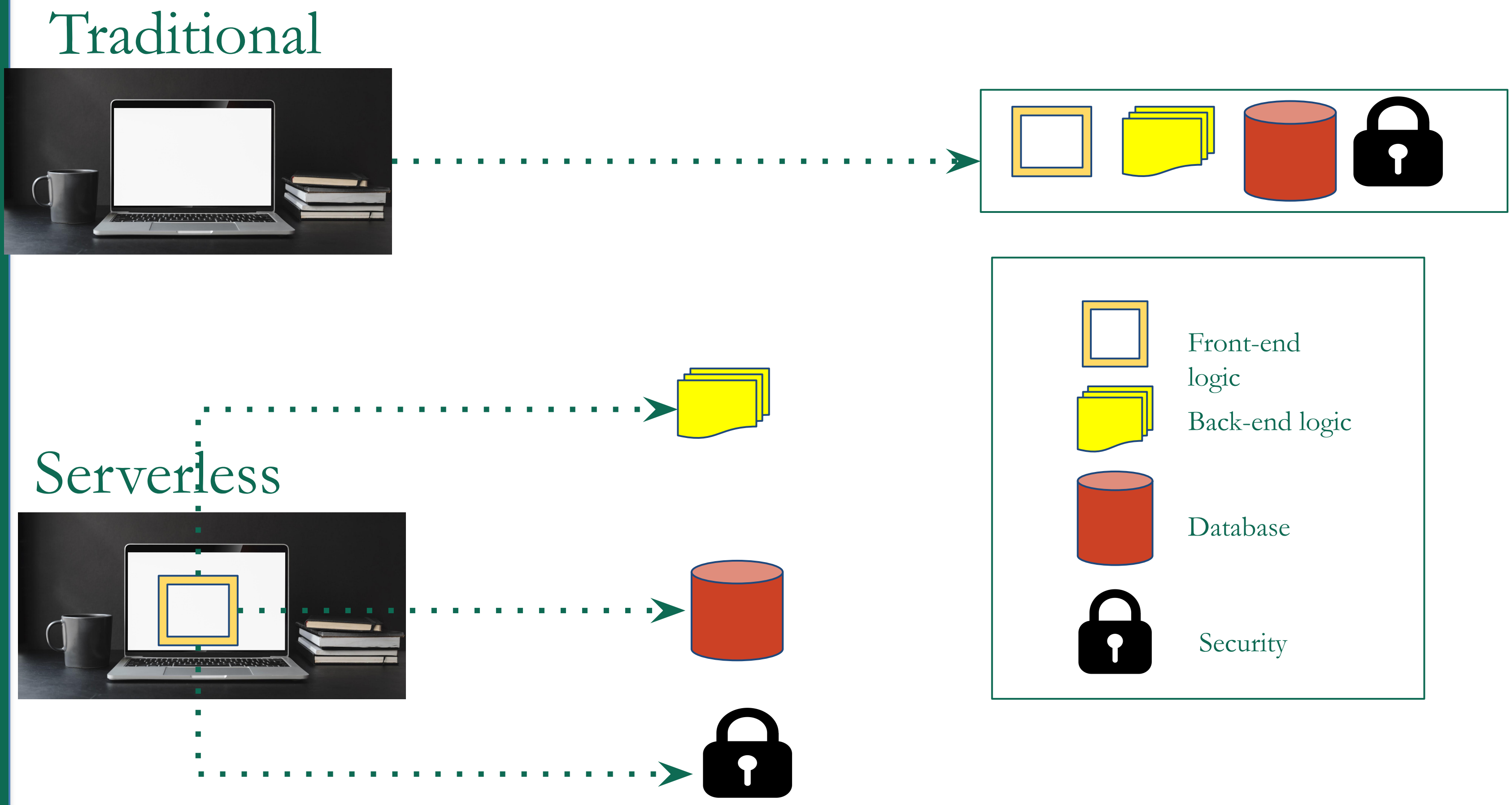
Intra-host TLM **eliminates the intermediate memory bump** thereby reducing the memory usage

Intra-host TLM (4/6)

Agenda

- Motivation
- Problem Statement
- Background
- Inter-host
  - Template-aware Live Migration of Virtual Machines (TLM) [Edgecomm: SEC’23]
  - Sharing-aware Live Migration of Virtual Machines (SLM) [CCGRID’24]
- Intra-host TLM [TPDS’24 under-review]
- Future Research Directions

Traditional vs Serverless



Traditional

Serverless

Front-end logic

Back-end logic

Database

Security

Serverless Functions and Limitations

- Serverless functions (Stateful/Stateless), due to their scalability and lightweight nature, are deployed on edge nodes.
  - Functions can be wrapped within a Webassembly(WASM)/container/VMs

- Existing stateful serverless function has following limitations
  - Doesn’t support migration
  - Cold-start penalty
  - Thin isolation
  - Lacking awareness of hardware extensions

Future Work

- Using AI/ML Infrastructure tools to decide the optimal isolation strategy for serverless functions

- To investigate the live migration implications for stateful serverless functions wrapped within VMs/containers

- Inter-communication latency for co-located serverless functions wrapped within wasm/containers/VMs

Conclusion

- We identified the limitation of current state-of-art inter/intra host live migration techniques
  - We introduced ***Inter-host TLM*** that retains ***COW benefits from VM templating*** across the machines during live migration
  - we identified the limitation of ***Inter-host TLM*** , and implemented a more generic technique called ***SLM,*** that retains COW shared pages from ***any COW memory optimization techniques*** and works for ***all types of VMs***

Conclusion

- Finally, we also introduced a more generic technique, ***Intra-host TLM*** for efficient live migration of all VM types within the same machines ***avoiding unnecessary copy of pages*** and concluded with future research directions

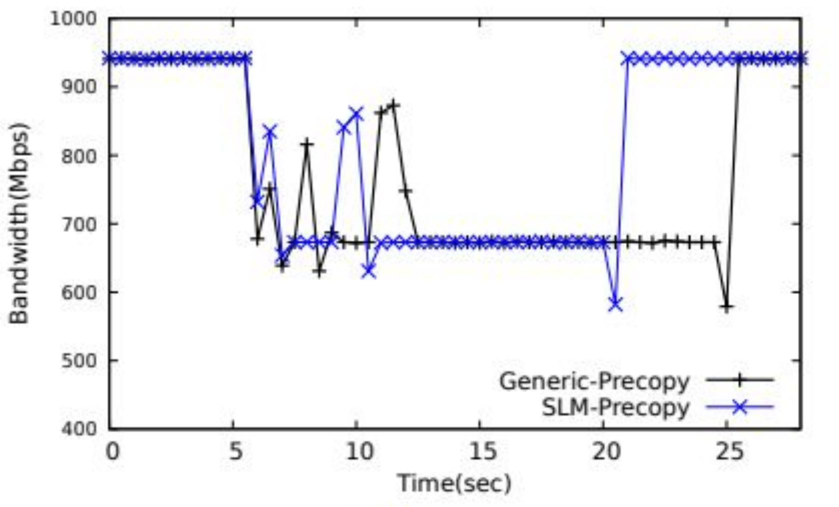
Thank you for the attention!

Questions?

MISC SLIDES

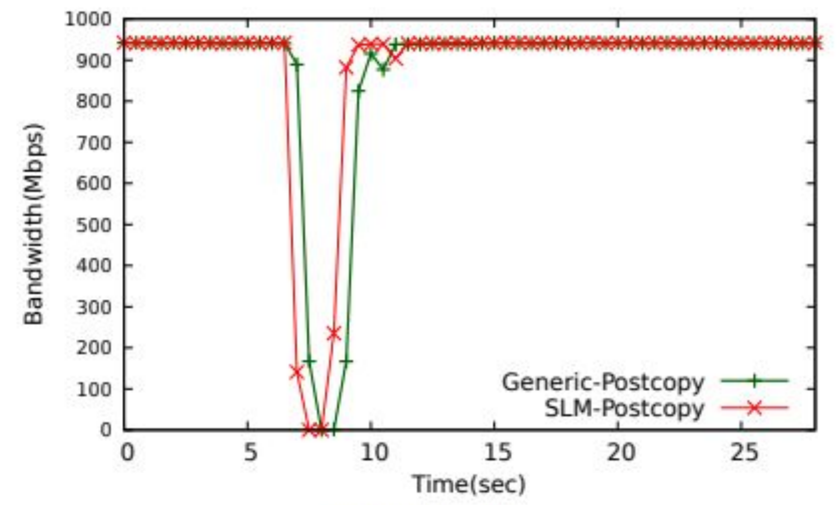
Evaluation: Iperf

Fig 1: Pre-copy



(a) Pre-copy

Fig 2: Post-copy



(b) Post-copy

Evaluation: Total Migration Time & Total Pages Transferred

Fig:1

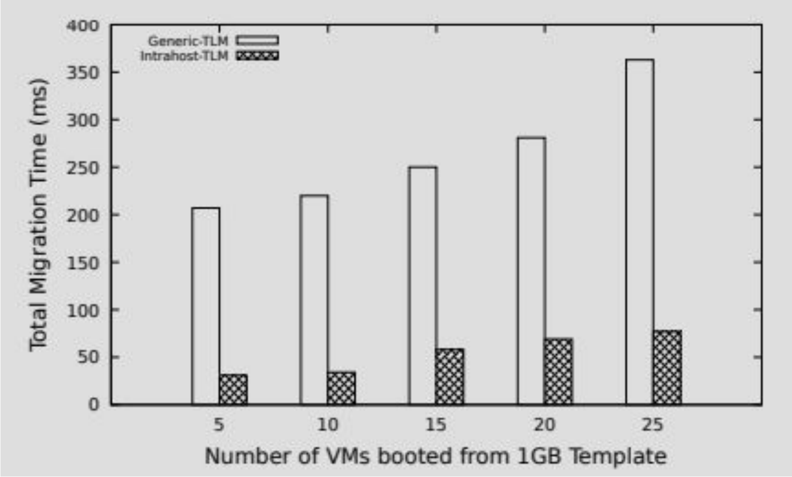
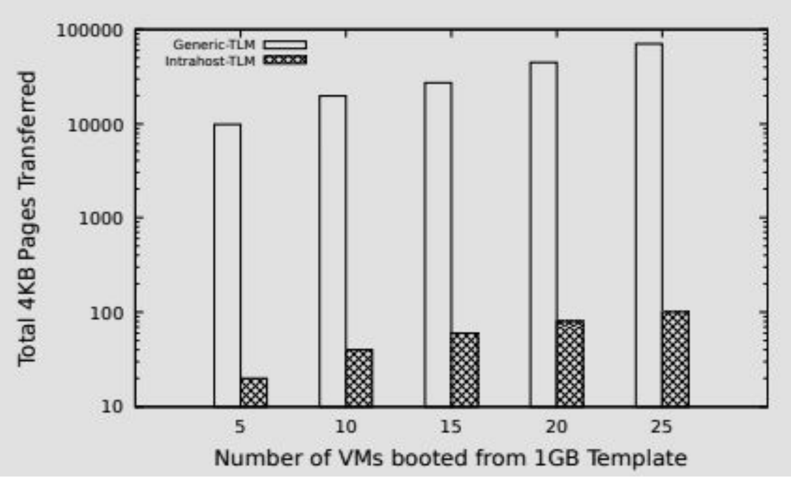
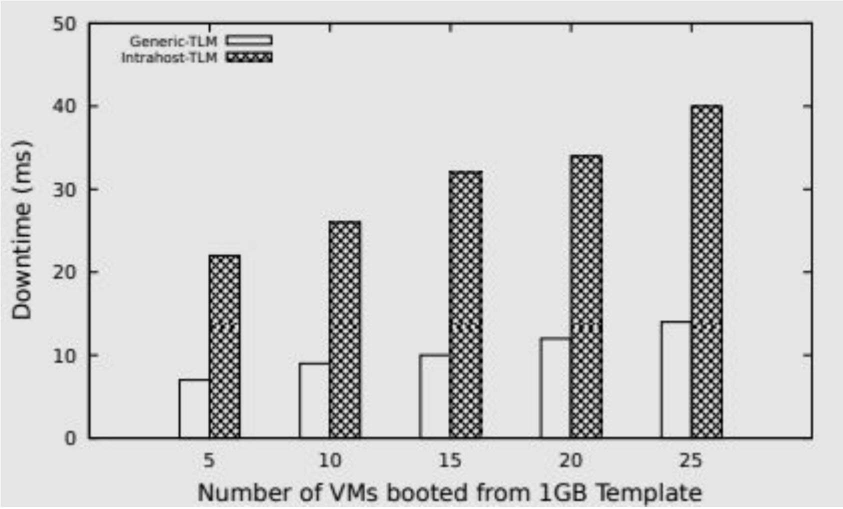


Fig:2



Evaluation: Downtime

- Downtime Overhead is due to transfer of backend-file offsets.



Evaluation: Memory Footprint

Fig:1

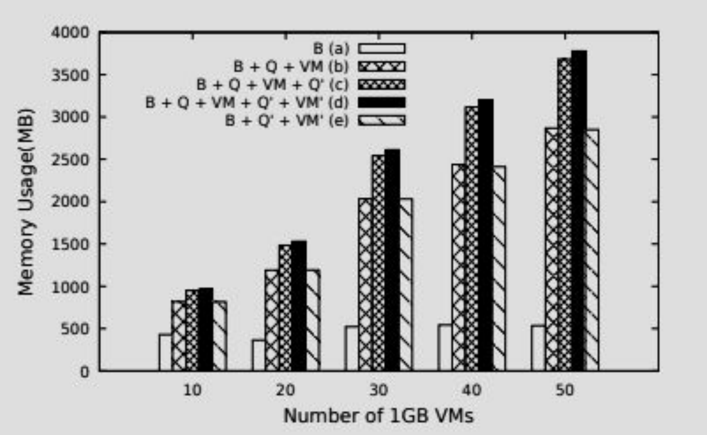
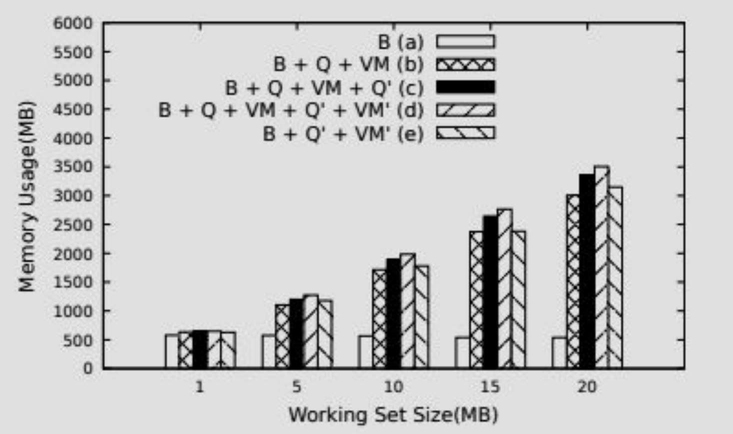
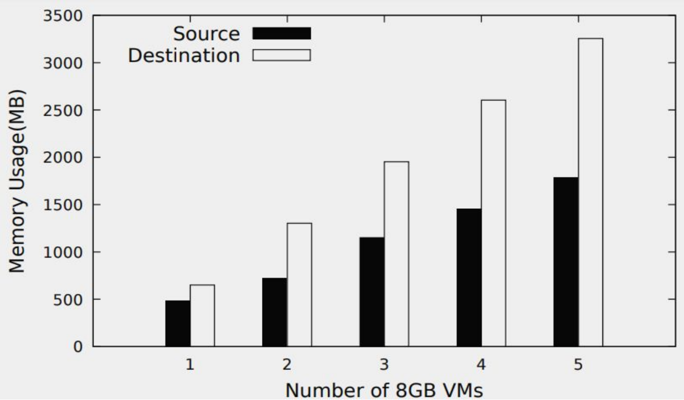


Fig:2



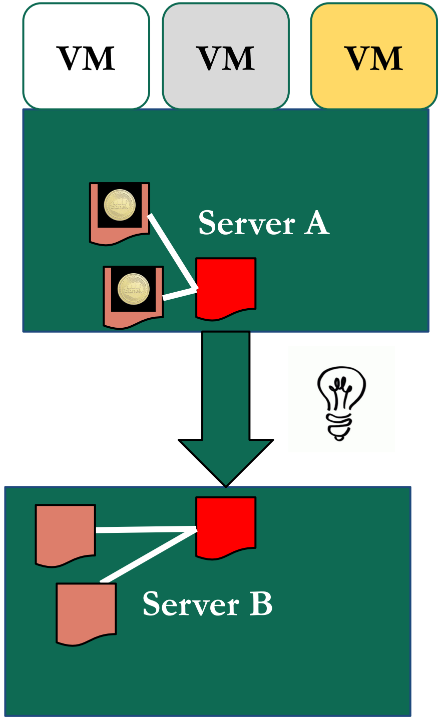
VM Templating & Live Migration

- Current live migration is not aware of VM templates
  - Transfer shared base template pages increasing
    - Total Migration Time
    - Network Traffic

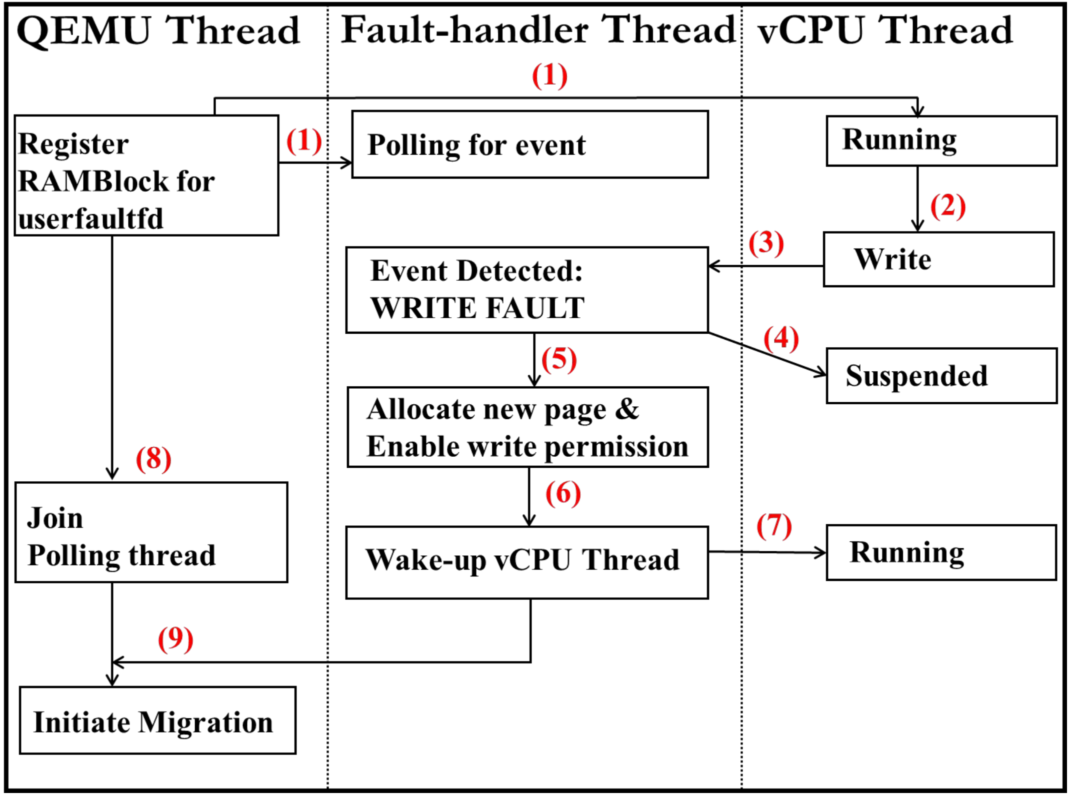


Key Insight: TLM

- *Improving live migration awareness* of duplicated pages using existing host/hypervisor *Copy-on-Write (COW)* optimization
  - Reducing Total Migration Time
  - Reducing Network Traffic

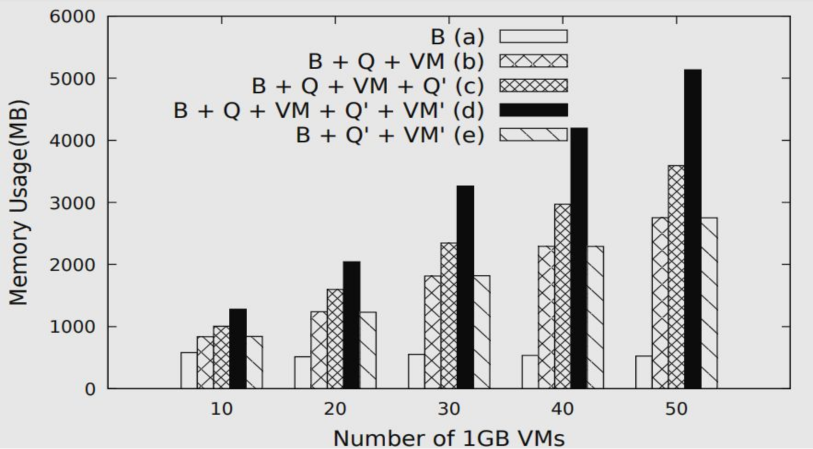


Design: Source



Key Insight: Intrahost TLM

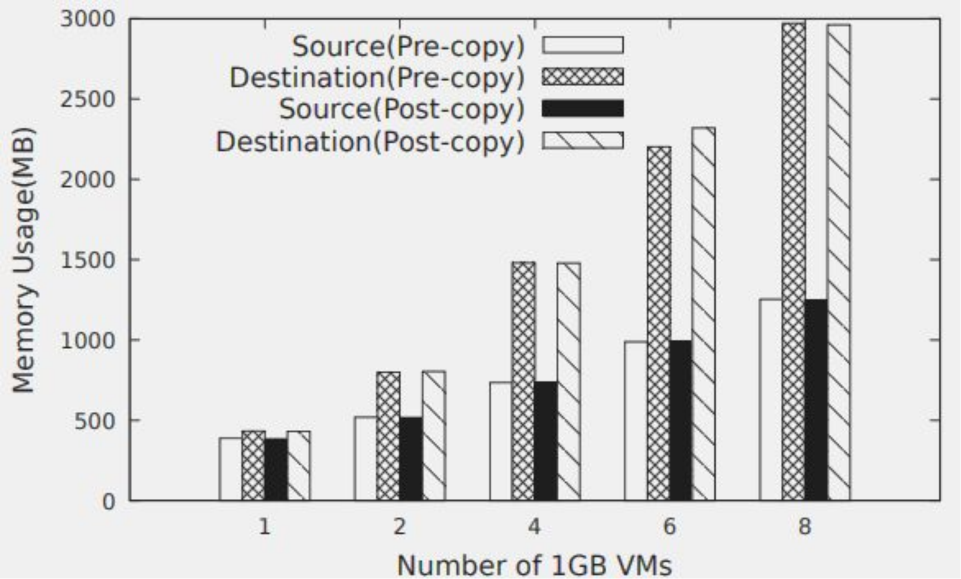
- Current Pre-copy is inefficient for migration within the same host because it copies the pages twice
  - Increasing memory footprint
  - Total Migration Time



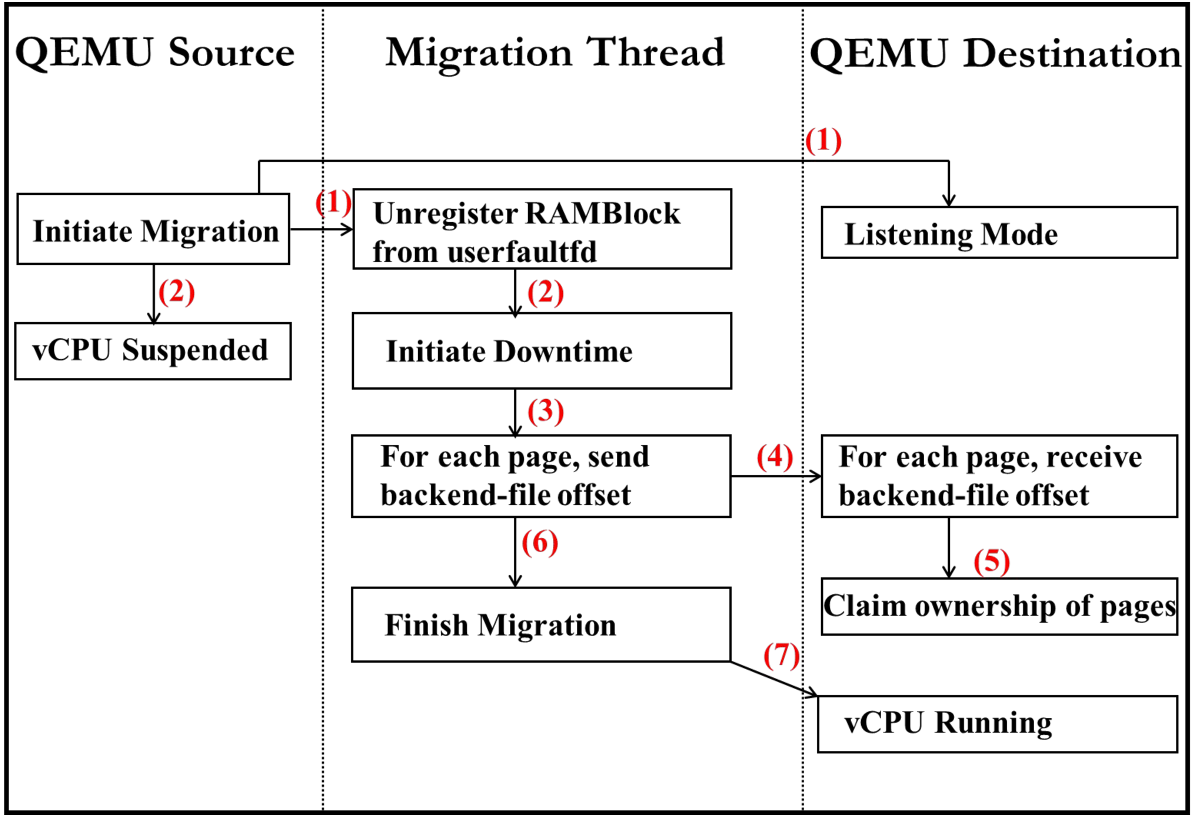
Key Insight: SLM

Current live migration is not *aware of COW shared pages* irrespective of the underlying memory optimization, increasing

- Total migration time
- Network Traffic



Design: Destination



Future Work

- Serverless functions, due to their scalability and lightweight nature, is deployed on edge nodes.
- For strong isolation, they needed to be encapsulate inside a VM.

- Our Future work focuses on addressing:
  - The invocation latency of serverless functions within the VM.
  - Inter-VM communication latency for the co-located serverless functions within the same node.