Proposal Defense

Enhancing the Performance of Live Migration by Mitigating Redundant Memory Page Transfers in Pre-Copy Migration

Samindu R. Cooray - 20000251

Supervisor: Dr. Dinuni K. Fernando

Table of Contents

01

Background

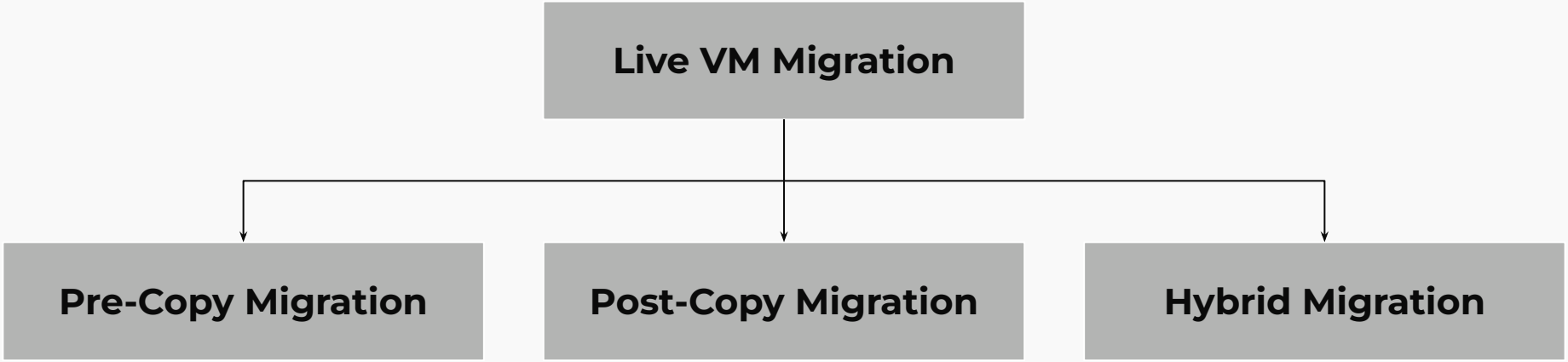
Background

Live VM Migration : Migrating a VM from a **Source Host Machine** to a **Destination Host Machine** while the **VM is in operating state**.

Goal : *Provide uninterrupted services to end-users.*

Background

Migration Techniques



Background

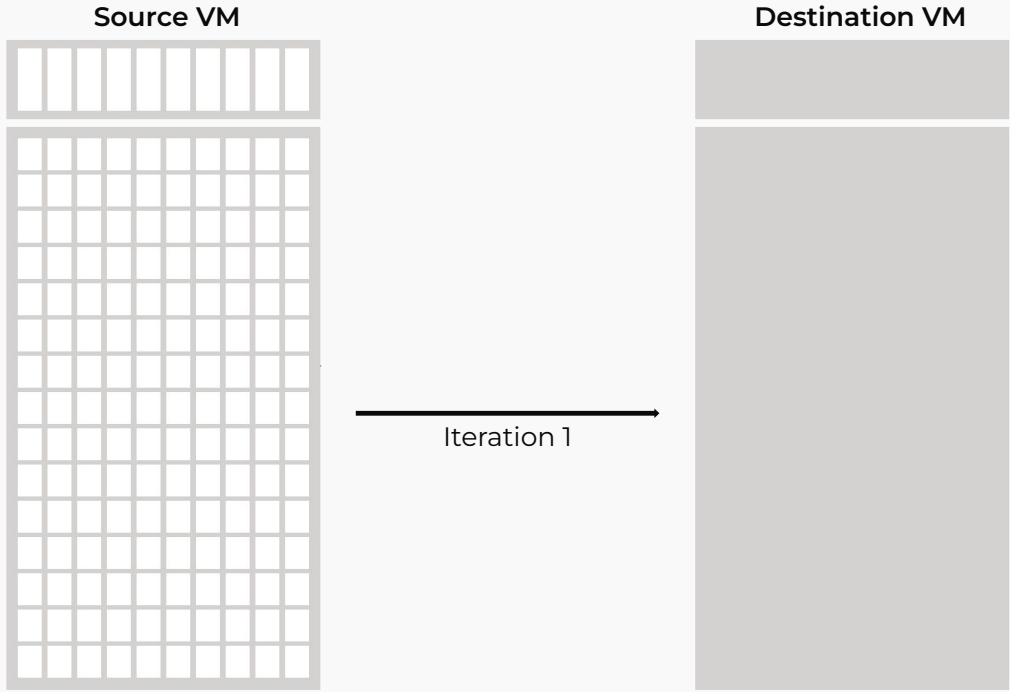
Pre-Copy Migration

Push Phase

1. Transfer all memory pages to the destination.
2. Iteratively transfer dirty pages to the destination until convergence point.

Stop-and-Copy Phase

3. At Convergence point, suspend the VM in the source and transfer the CPU and I/O state and the remaining memory pages to the destination.
4. Start VM in destination.



Background

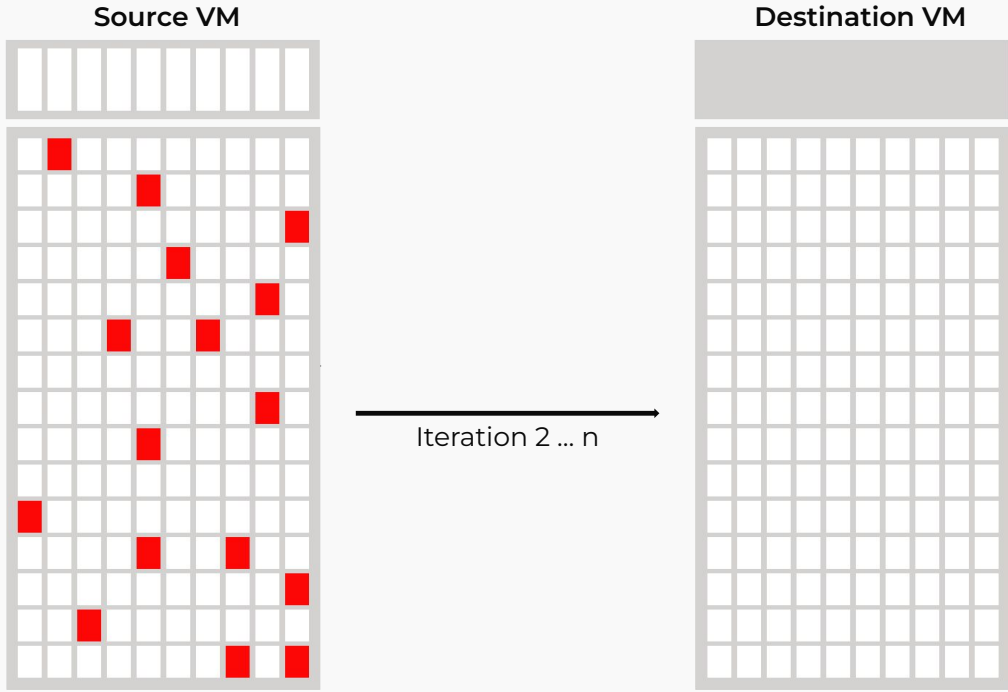
Pre-Copy Migration

Push Phase

1. Transfer all memory pages to the destination.
2. Iteratively transfer dirty pages to the destination until convergence point.

Stop-and-Copy Phase

3. At Convergence point, suspend the VM in the source and transfer the CPU and I/O state and the remaining memory pages to the destination.
4. Start VM in destination.



Background

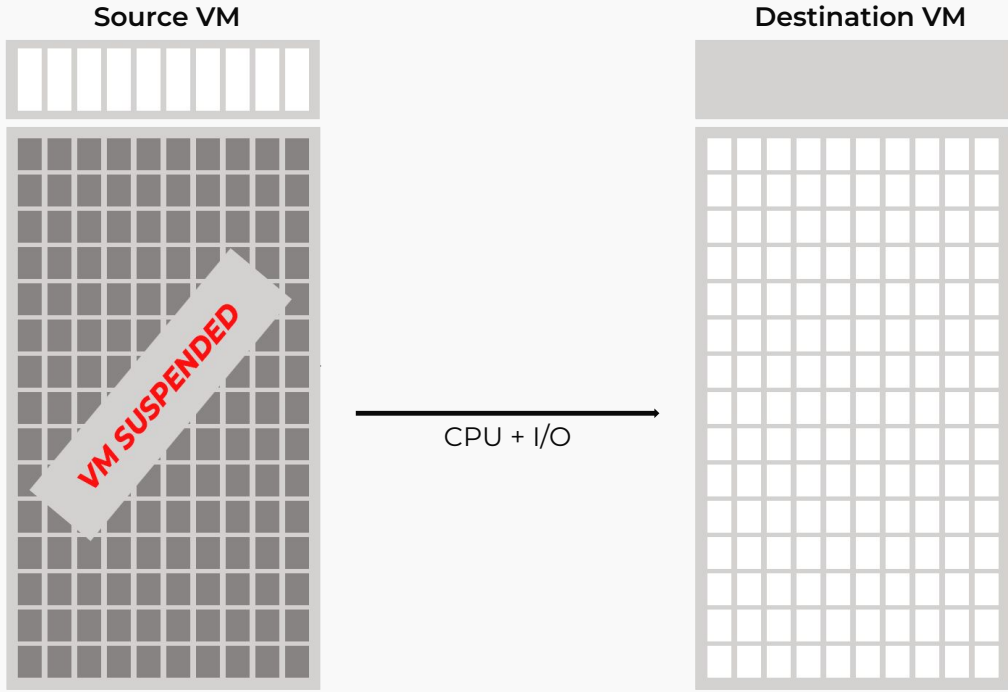
Pre-Copy Migration

Push Phase

1. Transfer all memory pages to the destination.
2. Iteratively transfer dirty pages to the destination until convergence point.

Stop-and-Copy Phase

3. At Convergence point, suspend the VM in the source and transfer the CPU and I/O state and the remaining memory pages to the destination.
4. Start VM in destination.



Background

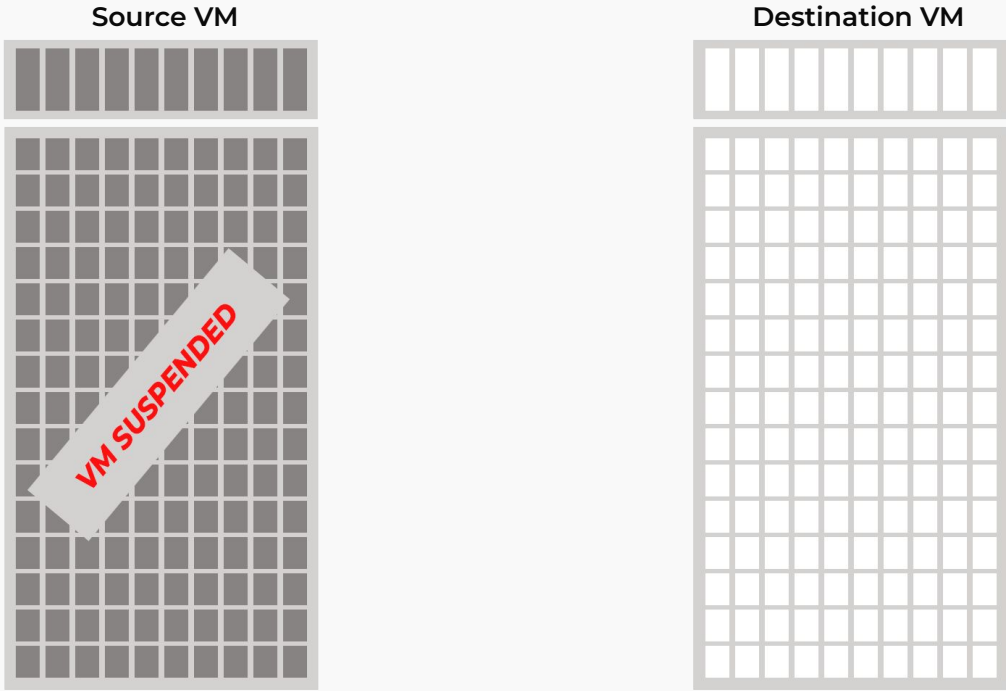
Pre-Copy Migration

Push Phase

1. Transfer all memory pages to the destination.
2. Iteratively transfer dirty pages to the destination until convergence point.

Stop-and-Copy Phase

3. At Convergence point, suspend the VM in the source and transfer the CPU and I/O state and the remaining memory pages to the destination.
4. Start VM in destination.



Background

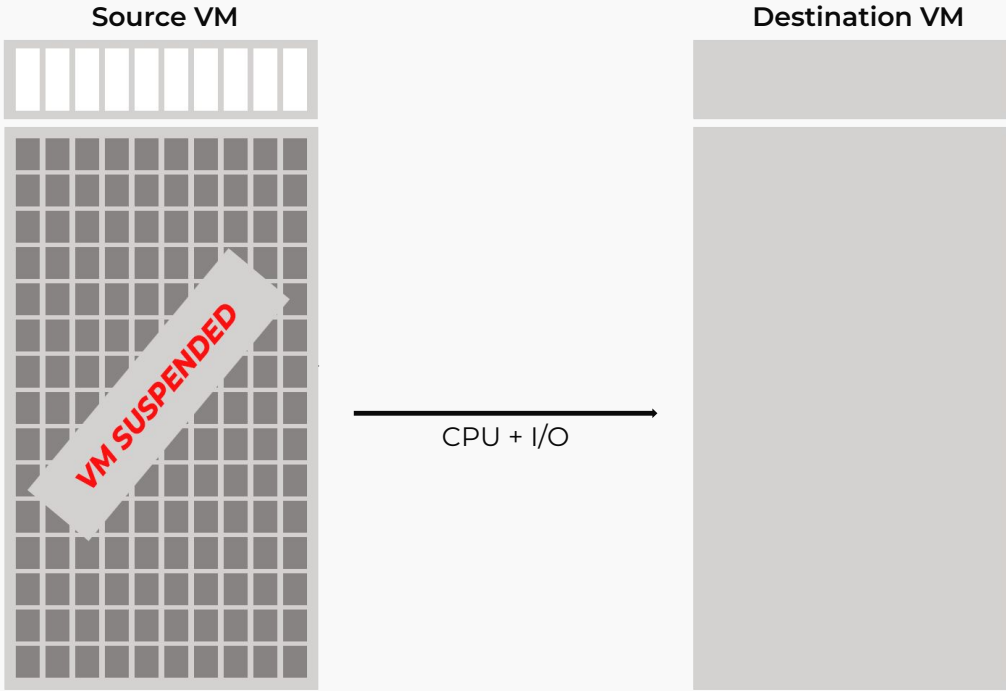
Post-Copy Migration

Stop-and-Copy Phase

1. Suspend the VM in the Source and transfer the CPU and I/O state to the destination.
2. Resume VM in destination.

Pull Phase

3. The source actively pushes pages to destination.
4. The destination fetches page faulted pages from source.



Background

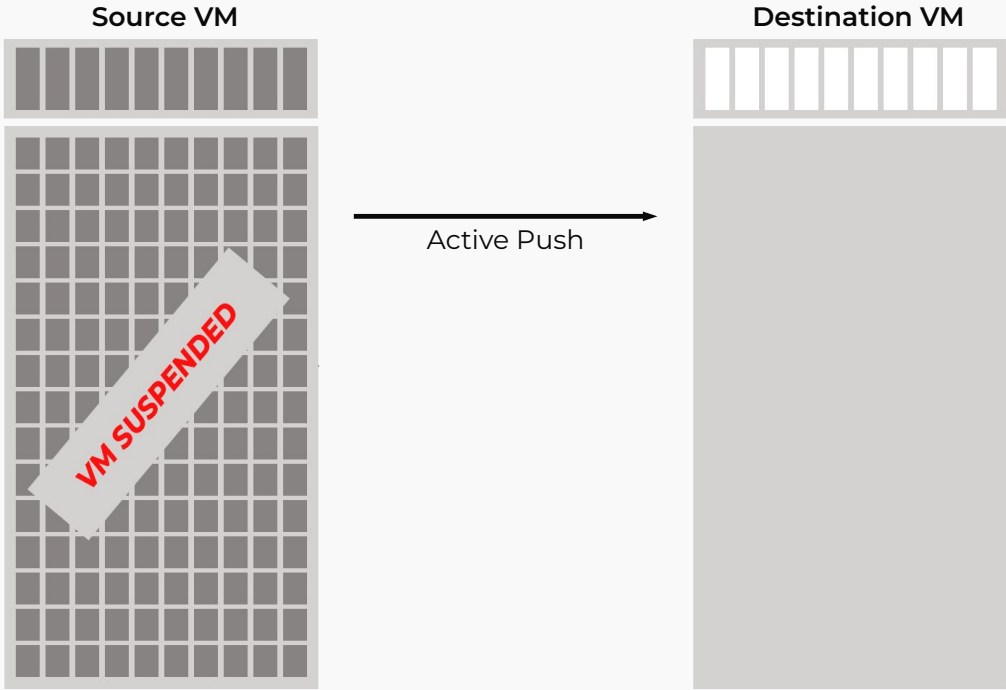
Post-Copy Migration

Stop-and-Copy Phase

1. Suspend the VM in the Source and transfer the CPU and I/O state to the destination.
2. Resume VM in destination.

Pull Phase

3. The source actively pushes pages to destination.
4. The destination fetches page faulted pages from source.



Background

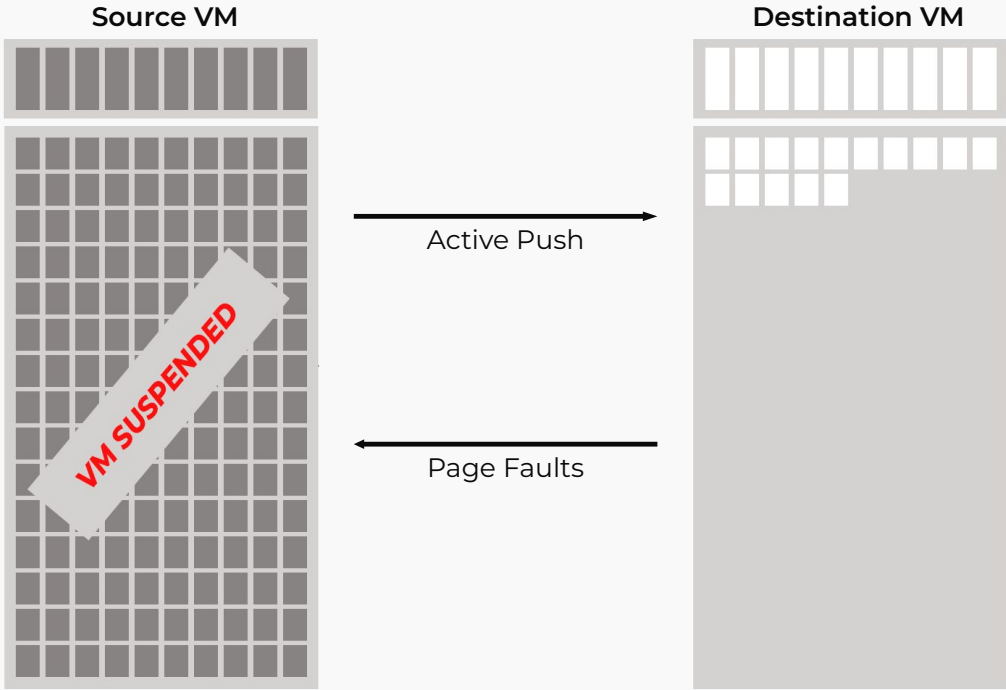
Post-Copy Migration

Stop-and-Copy Phase

1. Suspend the VM in the Source and transfer the CPU and I/O state to the destination.
2. Resume VM in destination.

Pull Phase

3. The source actively pushes pages to destination.
4. The destination fetches page faulted pages from source.



Background

Hybrid Migration

Push Phase

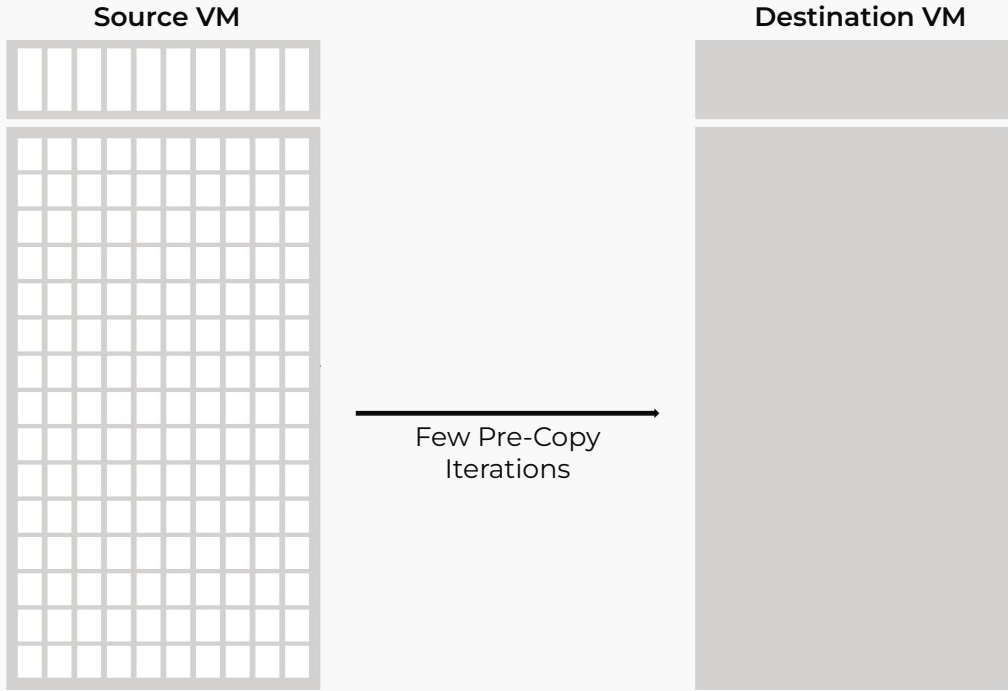
1. Transfer memory pages in few iterations

Stop-and-Copy Phase

2. Suspend the VM in the Source and transfer the CPU and I/O state to the destination.

Pull Phase

3. Resume VM in destination.
4. The destination fetches paging from the source.



Background

Hybrid Migration

Push Phase

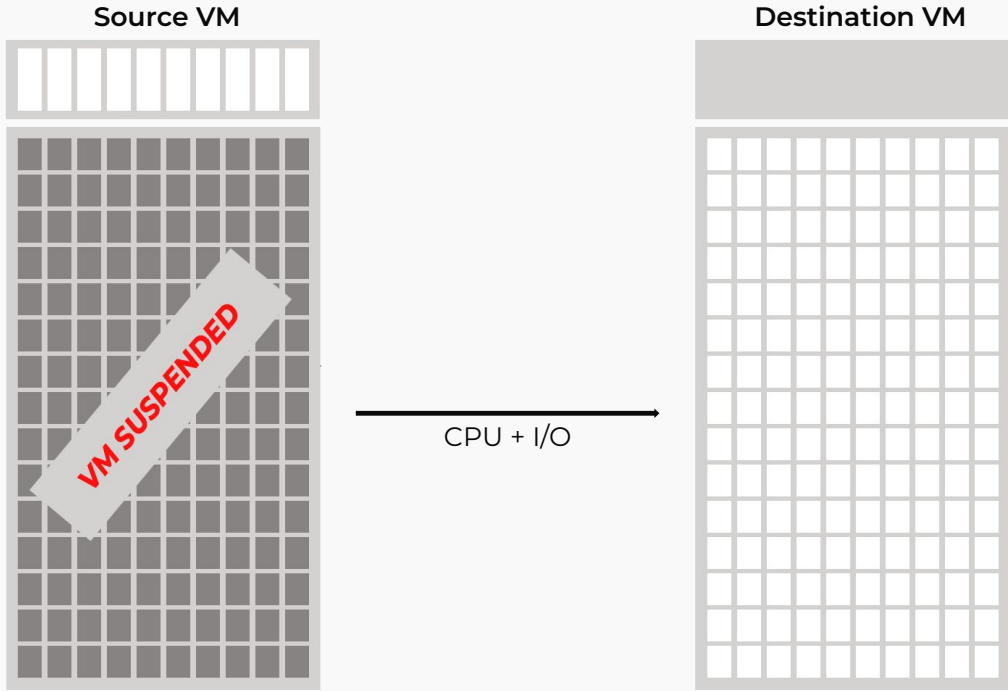
1. Transfer memory pages in few iterations

Stop-and-Copy Phase

2. Suspend the VM in the Source and transfer the CPU and I/O state to the destination.

Pull Phase

3. Resume VM in destination.
4. The destination fetches paging from the source.



Background

Hybrid Migration

Push Phase

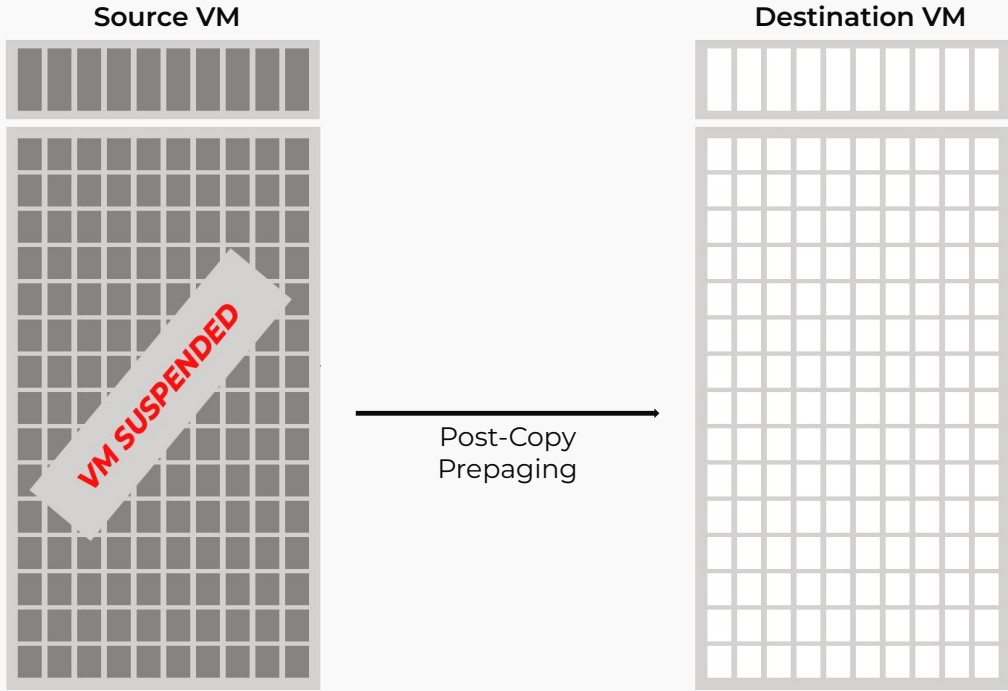
1. Transfer memory pages in few iterations

Stop-and-Copy Phase

2. Suspend the VM in the Source and transfer the CPU and I/O state to the destination.

Pull Phase

3. Resume VM in destination.
4. The destination fetches paging from the source.



Jul et al., Proceedings of the 2nd Symposium on Networked Systems Design and Implementation, 2005
Li et al., Future Generation Computer Systems, 2019
Elsaid et al., Distributed and Parallel Databases, 2022

Background

Challenges in Pre-Copy Migration

Primary Bottleneck in Pre-Copy : Total Migration Time.

Factors Affecting :

- Varying dirty page rates.
- Network transmission speed.

In extreme cases, rapid dirty page generation and low network bandwidth can lead to **prolonged total migration time** or even **migration failure**.

Background

Challenges in Pre-Copy Migration

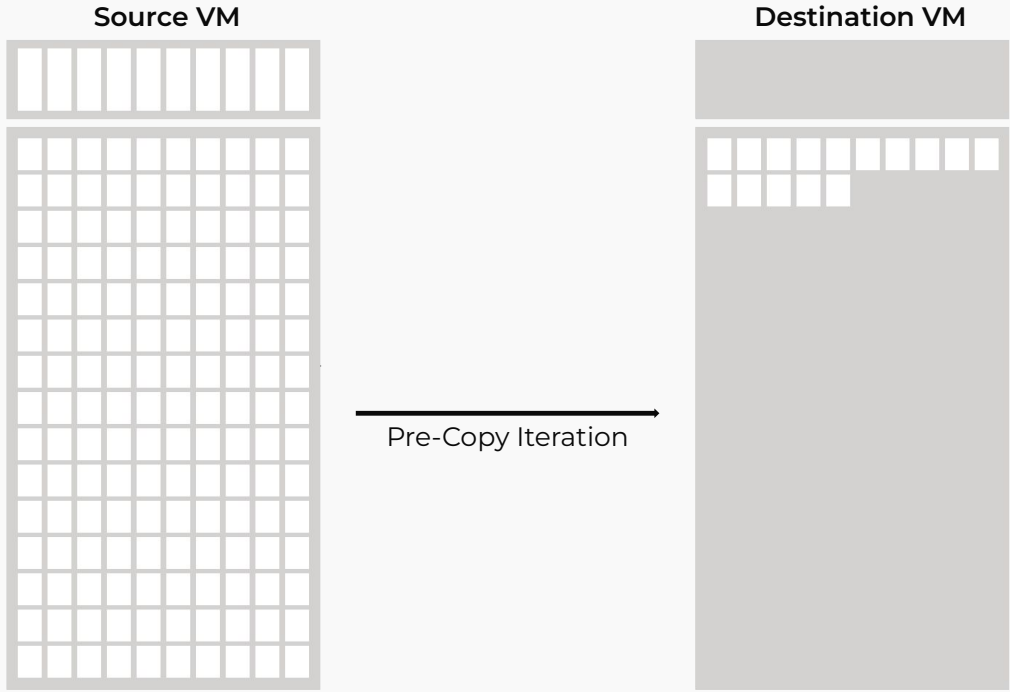
Interestingly we found that most of these **rapidly generated pages are Fake !** although this was initially found by Li et. al. in 2019 but not yet in plugged to streamline QEMU codebase!

What is a Fake Dirty Page ?

Background

Fake Dirty Page

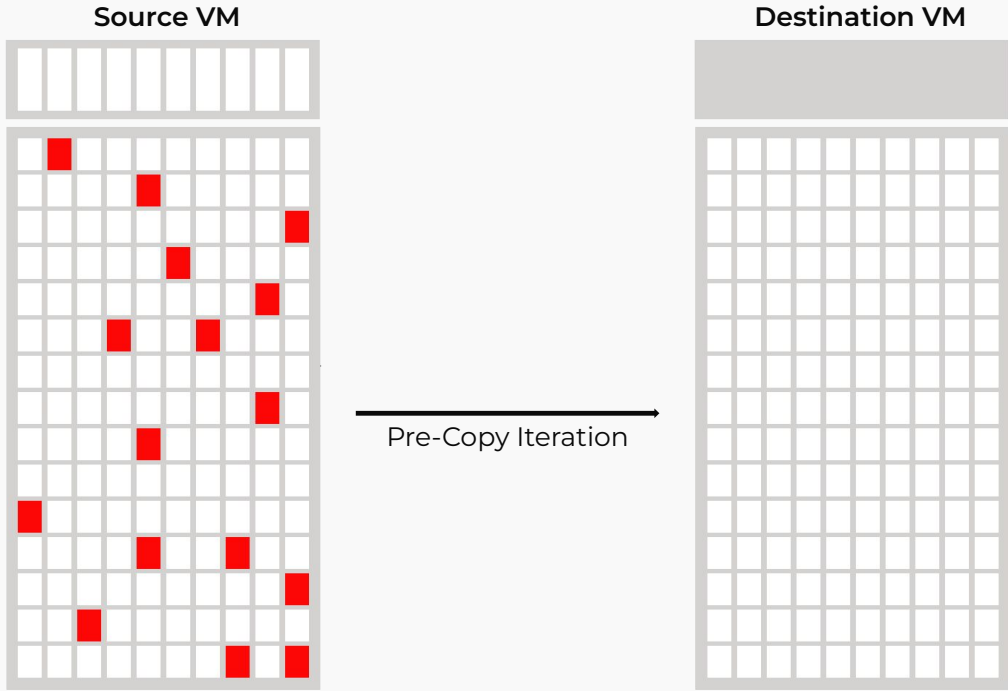
- During migration thread performs the migration the hypervisor keeps track of the pages that modified and mark that page as dirty in a bitmap called dirty bitmap.
- Then with the use of the dirty bitmap, only the pages that are dirtied in the previous iteration are considered to migrate in the current iteration .



Background

Fake Dirty Page

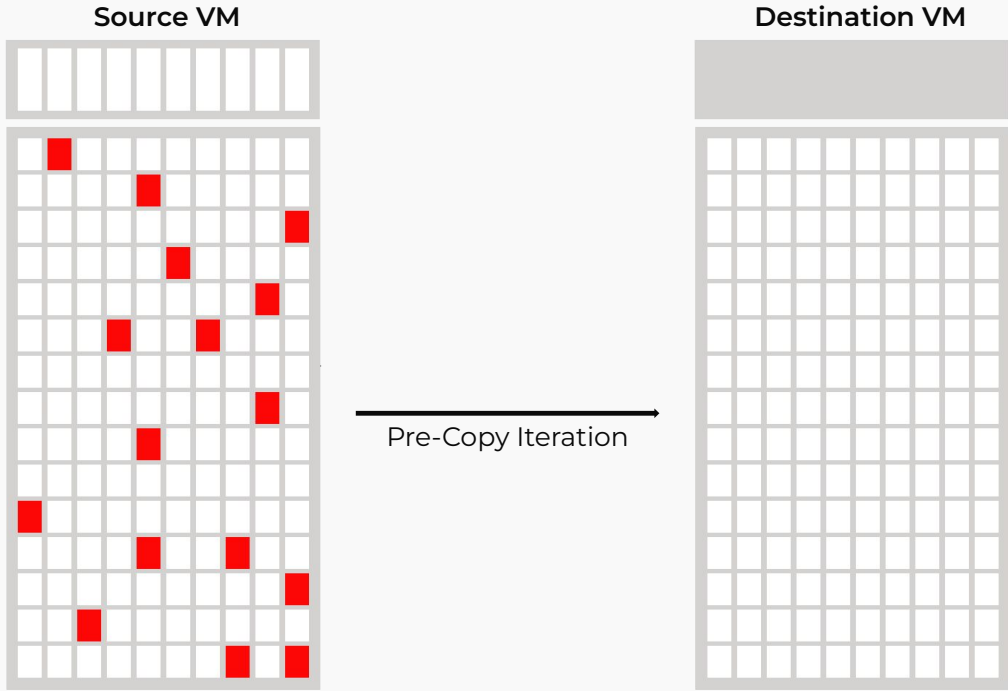
- During migration thread performs the migration the hypervisor keeps track of the pages that modified and mark that page as dirty in a bitmap called dirty bitmap.
- Then with the use of the dirty bitmap, only the pages that are dirtied in the previous iteration are considered to migrate in the current iteration .



Background

Fake Dirty Page

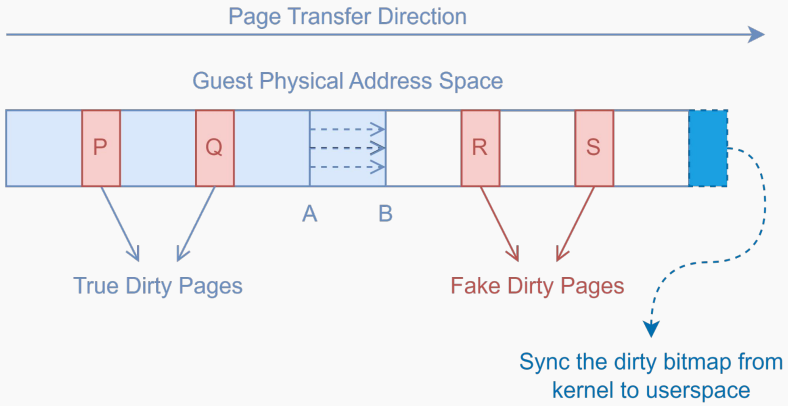
- In Reality, out of the pages marked as dirty in the previous iteration, there are some pages that **do not** have any actual **content change**. These pages are known as **Fake Dirty Pages**.
- This unnecessary duplication wastes **network bandwidth** and prolong **migration time**.



Background

Reasons for Fake Dirty Page Generation

- *“* ***Write-not-dirty*** *”* requests issued by *Silent Store Instructions*, *where existing value is written again to the memory address resulting no state change.*
- Defects in the ***Dirty Page Tracking Mechanism***.



02

Motivation

Motivation

- Cloud Computing (CC) offers efficient and reliable services to millions globally.
    - Eg:CC service providers include Microsoft Azure, Amazon Web Services (AWS), Alibaba Cloud, and Google Cloud Platform (GCP).
    - Ensure **uninterrupted services**
- Data centers use **virtualization** to offer services by hosting multiple virtual machines (VMs) on a single server.
- **Hardware** or **software failures** in servers, or essential **updates** in the servers can disrupt VM services.
- Live VM Migration techniques are employed moves VMs to another physical server. This approach supports **fault tolerance**, **load balancing**, **host maintenance**, and **server consolidation**.

Significance of Migration

1,000,000 Migrations Per Month

(Ruprecht, A., Jones, D., Shiraev, D., Harmon, G., Spivak, M., Krebs, M., Baker-Harvey, M. & Sanderson, T. (2018), 'Vm live migration at scale', ACM SIG-PLAN Notices 53(3), 45–56.)

03

Related Work

Related Work

There is **limited literature** available in this domain of research, highlighting the need for further investigation, which this study aims to address.

On Selecting the Right Optimizations for Virtual Machine Migration

(Nathan, S., Bellur, U. & Kulkarni, P. (2016), On selecting the right optimizations for virtual machine migration.)

- Delta Compression *(Zhang et al. 2010, Svärd et al. 2011)*
  - *The modifications done to the data are stored without storing the full data sets*
- Deduplication *(Deshpande et al. 2011, Wood et al. 2015)*
  - *Transfer only one copy of duplicate pages.*

Related Work

Efficient Live Virtual Machine Migration for Memory Write-intensive Workloads

(Li, C., Feng, D., Hua, Y. & Qin, L. (2019), 'Efficient live virtual machine migration for memory write-intensive workloads', Future Generation Computer Systems)

- Avoid Fake Dirty Pages transfer using secure hashes (SHA1).
- Algorithm Design :
  - Stores secure hashes of all the transferred pages in the initial iteration.
  - In the next iterations, the pages marked as dirty are compared with the previously stored respective secure hashes before transferring.

    If the new and the old hashes of a pages are:
    - Different - The page is transferred to the destination, and the hash of the new version replaces the former hash.
    - Similar - The page is a Fake Dirty Page where it is not transferred to the destination.

04

Research Gap

Research Gap

- A primary concern in VM migration is minimizing the **total migration time** to mitigate service interruptions.
  - In Pre-Copy, the total migration time depends on the number of **Pre-Copy rounds** (memory transfer iterations).
  - If these Pre-Copy rounds increase without converging, the tendency of migration failure increases.
- During Pre-Copy rounds, data exceeding the **VM's RAM size** is transmitted.
  - *For example, when migrating an 8GB VM, Pre-Copy rounds may transfer memory pages totaling more than 8GB*.
  - Reducing the number of pages transferred per Pre-Copy round can decrease the total migration time.
- Fake Dirty Pages, first identified in 2016 by Nathan et al. In 2019, Li et al. further investigated and proposed a solution based on **QEMU 2.5.1** in 2019.
  - The current version of **QEMU 8.1.2** reveal that there is no default handling of Fake Dirty Pages.

Research Gap

- According to Li et al., secure hashes (SHA1) were used to generate the hash of the page required in the algorithm.
    - Migration thread is paused until the hash of a page is generated and fake dirty pages are identified.
    - This study did not investigate the applicational overhead of hash computation and fake dirty page detection.
    - Overlooked the potential benefits of using **different hashing techniques over SHA1** to reduce the application overhead.
- Existing study has not considered applying different **optimization techniques** along with the hash-based fake dirty prevention algorithm for reducing total migration time.
    - Transferring a compressed page is more time effective than of a 4096 byte page over the network.
    - Application can enhance the efficiency of VM migration by minimizing total migration time and downtime.

05

Research Questions

Research Questions

01 How to mitigate redundant data transfers by identifying fake dirty pages?

02 How to reduce the data transferring load in Pre-Copy Migration to improve the performance of memory-intensive workloads?

06

Objectives

Objectives

01

To evaluate the performance improvement of using **different hashing mechanisms** to reduce redundant memory page transfers by considering Fake Dirty Pages.

02

To evaluate the performance improvement of **combining different optimization techniques** to reduce the data transferring load in Pre-Copy migration to improve the performance for memory-intensive workloads?

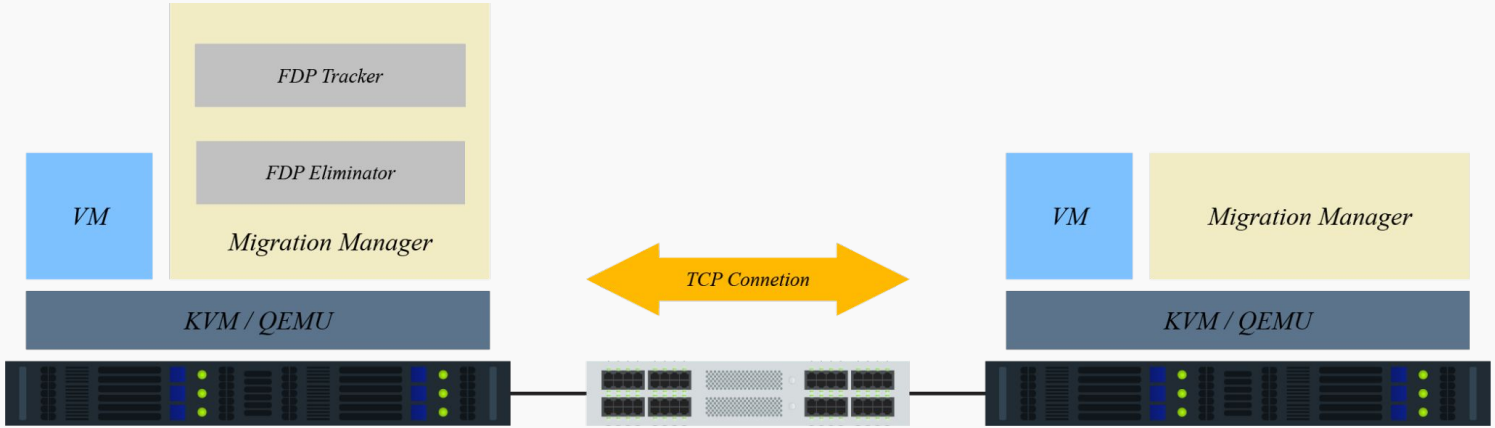
07

Preliminary Experiments

Preliminary Experiments

Testbed Setup

- Setuped two physical servers and a NFS server interconnected with Gigabit ethernet connection.
- Setup servers with QEMU/KVM version 8.1.2



Preliminary Experiments

Fake Dirty Detection Mechanism

1. Initialize an array to store hash values for each RAM block.
2. Compute the hash of each page before transfer and store it in the corresponding array .
3. In subsequent iterations, compute the hash of each selected dirty page and compare the computed hash with the stored previous hash.
    a. If the hashes are similar, increment the fake dirty count variable by one.
    b. Otherwise, replace the previous hash with the current hash and transfer the page to the destination.

Preliminary Experiments

Selecting Suitable Workloads

Workloads	Intensive Type	Description
Workingset	Memory	A benchmark that dirty pages to vary writable working set
Quicksort	CPU	A benchmark that repeatedly allocates random integers to an integer array of 1024 bytes and performs quicksort on the array.
Sysbench	CPU	A benchmark to assess the system performance of a machine planning on running a database under intensive load
Memcached	Multiple Resource	Memcached is an in-memory key-value store storing arbitrary data returned by a benchmark called Memaslap
YCSB	Multiple Resource	A Suite used to evaluate computer programs' retrieval and maintenance capabilities.

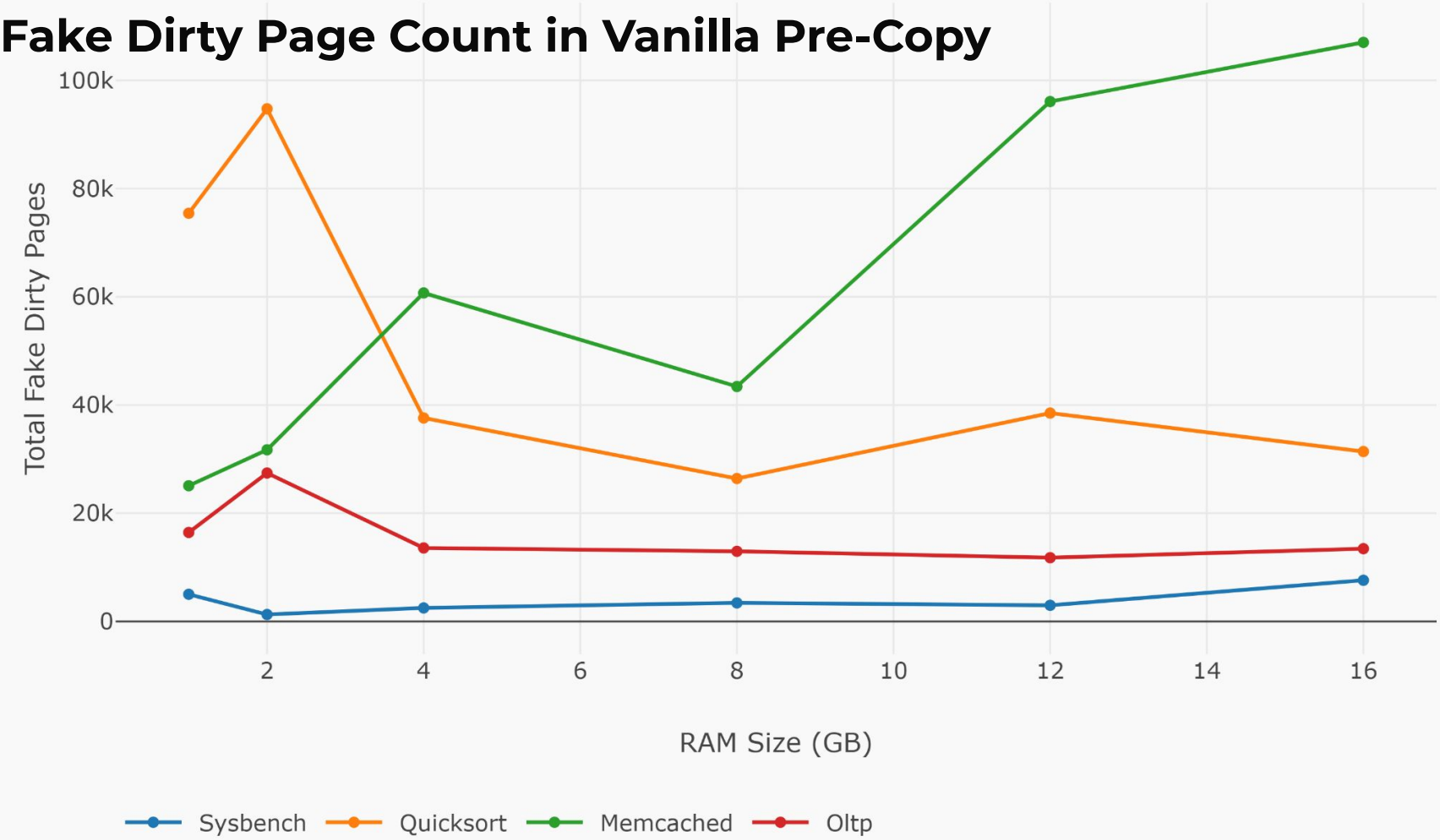
Preliminary Experiments

Fake Dirty Observation

- The initial experiment was to observe the presence of **Fake Dirty Pages** in Vanilla Pre-Copy Migration. The experiment was conducted for **6** VM memory sizes *(1GB, 2GB, 4GB, 8GB, 12GB, 16GB)*.
- Each data point is an average of 3 rounds of experiment.

Analysis in the Next Slide

Fake Dirty Page Count in Vanilla Pre-Copy



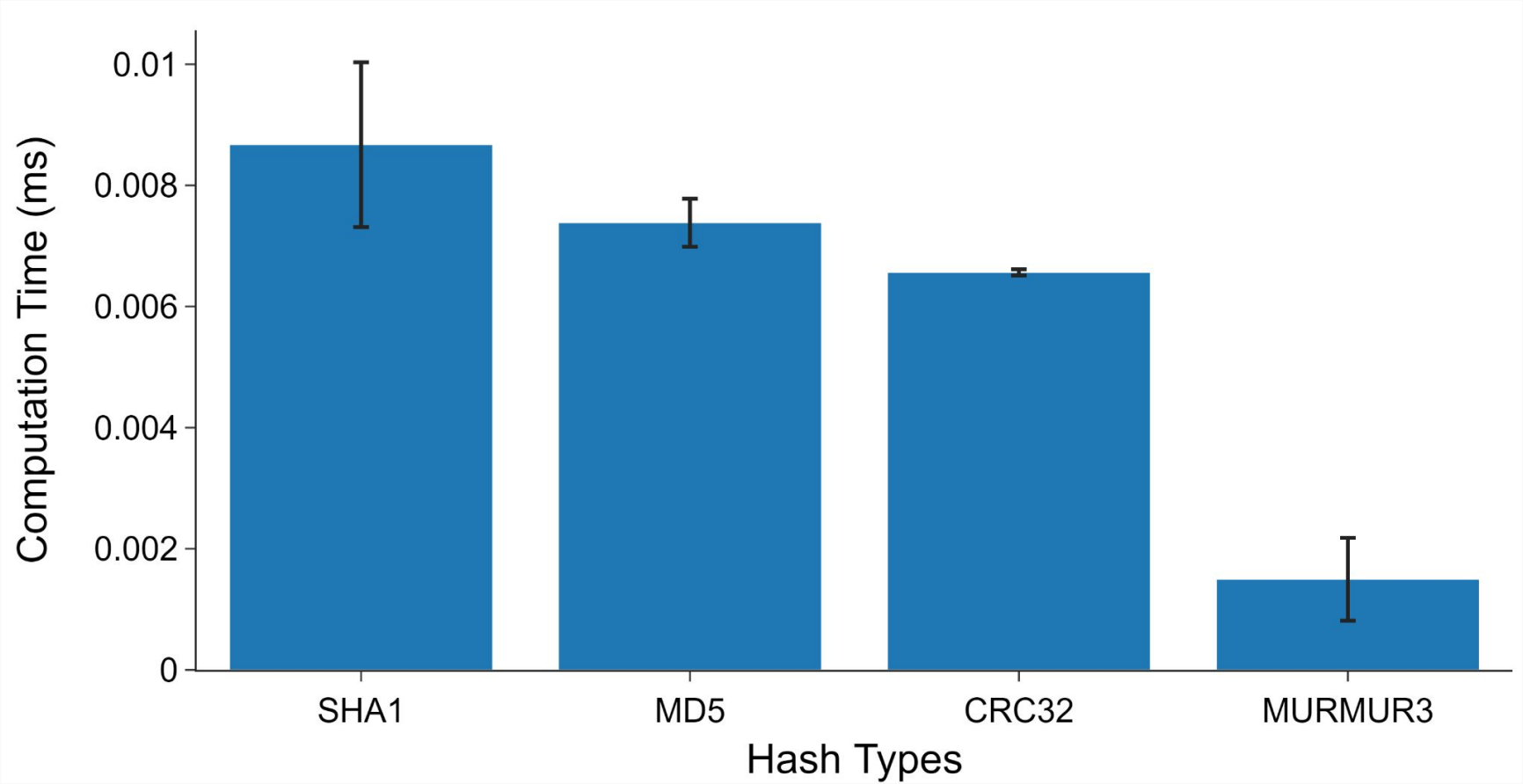
Preliminary Experiments

Hash Computation Time

- The next experiment was to observe the computation time for different hashing techniques.
- SHA1, MD5, CRC32, and Murmur3 hashing methods were used to conduct this experiment.
- The experiment was done in the C environment.
- In the experimental setup, a 4KB memory was allocated with random numbers and hashed using each hashing method, and the computation time was recorded.

Analysis in the Next Slide

Hash Computation Time



08

Scope

Scope

In Scope

Develop prototypes for eliminating Fake Dirty Pages using different hash techniques.

Incorporating different optimization techniques to further minimize total migration time

Primarily consider migrations performed in Linux Operating Systems in Local Area Network

09

Research Approach

Research Approach

Planned to follow **Design science** research methodology.

01 Testbed Setup

02 Selecting Suitable Workloads

03 Conducting Preliminary Experiments

04 Implementing Prototypes

05 Evaluation

Evaluation

- The developed prototypes with and without incorporating different optimization techniques would be evaluated against
  - **Vanilla Pre-Copy**,
  - **XBZRLE enabled Pre-Copy**
  - **the algorithm proposed by Li et al. (2019)** (Prototype developed with SHA1 hashes).

10

Progress

Progress So Far

01 Testbed Setup

02 Selecting Suitable Workloads

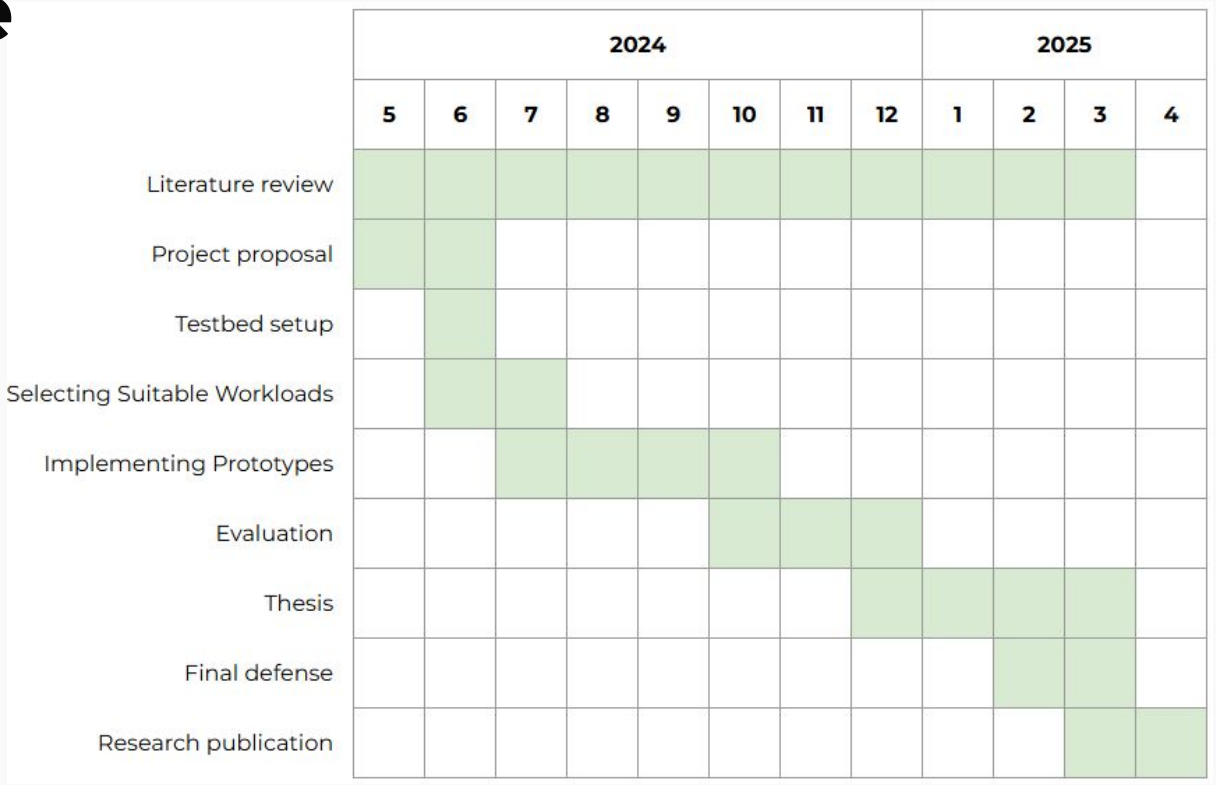
03 Conducting Preliminary Experiments

04 Implementing Prototype of the algorithm proposed by Li et al. (2019) using SHA1 hashes

05 Creating prototype variations using different hashing techniques replacing the SHA1 hashing technique.

06 Evaluation

Timeline


2024
2025
5
6
7
8
9
10
11
12
1
2
3
4
Literature review
Project proposal
Testbed setup
Selecting Suitable Workloads
Implementing Prototypes
Evaluation
Thesis
Final defense
Research publication

References

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. & Warfield, A. (2003), Xen and the art of virtualization, Vol. 37.

Elsaid, M. E., Abbas, H. M. & Meinel, C. (2022), 'Virtual machines pre-copy live migration cost modeling and prediction: a survey', Distributed and Parallel Databases 40.

Hines, M. R., Deshpande, U. & Gopalan, K. (2009), Post-copy live migration of virtual machines, Vol. 43, pp. 14–26.

Jul, E., Warfield, A., Clark, C., Fraser, K., Hand, S., Hansen, J. G., Limpach, C. & Pratt, I. (2005), Live migration of virtual machines. URL: https://www.researchgate.net/publication/220831959

Li, C., Feng, D., Hua, Y. & Qin, L. (2019), 'Efficient live virtual machine migration for memory write-intensive workloads', Future Generation Computer Systems 95.

Nathan, S., Bellur, U. & Kulkarni, P. (2016), On selecting the right optimizations for virtual machine migration.

Rayaprolu, A. (2024), 'How Many Companies Use Cloud Computing in 2024? All You Need To Know'. URL: https://techjury.net/blog/how-many-companies-use-cloud-computing/

Ruprecht, A., Jones, D., Shiraev, D., Harmon, G., Spivak, M., Krebs, M., Baker-Harvey, M. & Sanderson, T. (2018), 'Vm live migration at scale', ACM SIG-PLAN Notices 53(3), 45–56

Sahni, S. & Varma, V. (2012), A hybrid approach to live migration of virtual machines.

In Summary

- Live VM Migration is essential for maintaining **uninterrupted services**, achieving *fault tolerance*, *load balancing*, and *server consolidation* in cloud computing environments.
- This research focuses on enhancing pre-copy migration by addressing redundant memory page transfers, known as "**fake dirty pages**," which significantly impact total migration time.
- The study tries to mitigates these redundant transfers with the use of an optimal ashing technique, thereby optimizing migration efficiency and **reducing total migration time**.

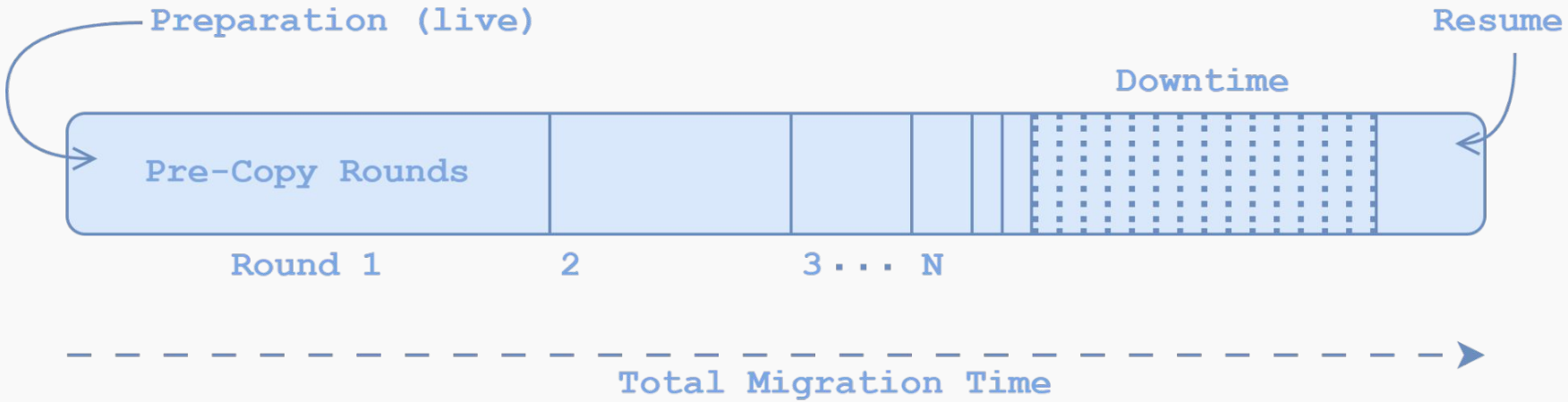
Thank you !

Thank you !

Jul et al. 2005

Background

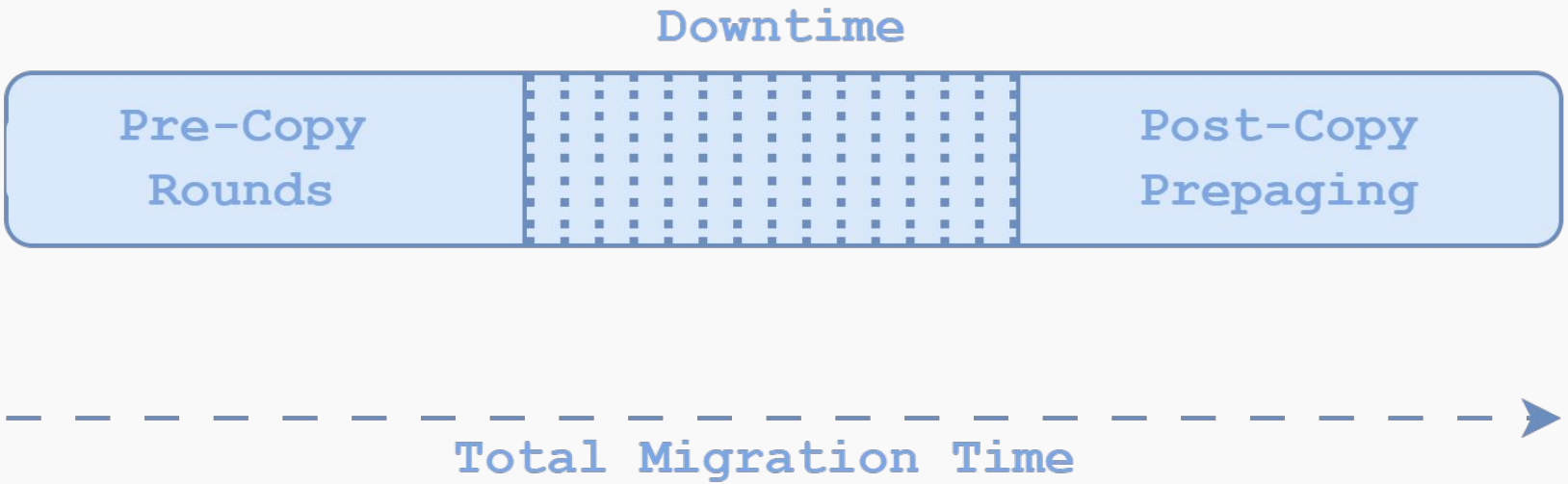
Pre-Copy Migration



Sahni & Varma 2012

Background

Hybrid Migration



Background

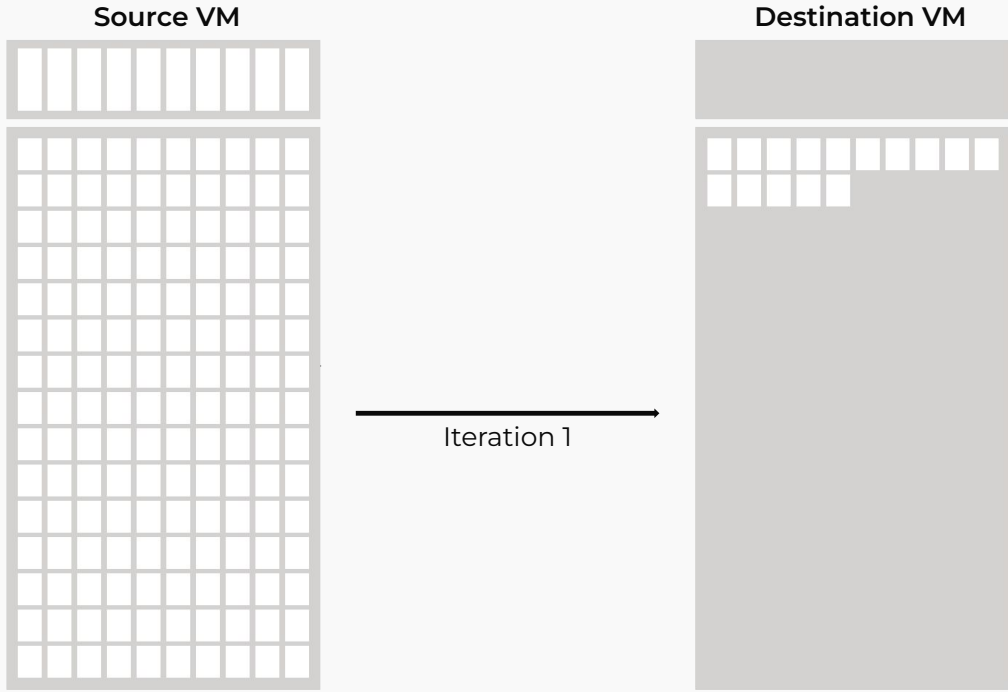
Pre-Copy Migration

Push Phase

1. Transfer all memory pages to the destination.
2. Iteratively transfer dirty pages to the destination until convergence point.

Stop-and-Copy Phase

3. At Convergence point, suspend the VM in the source and transfer the CPU and I/O state and the remaining memory pages to the destination.
4. Start VM in destination.



Background

Performance Metrics

Total Migration Time

Downtime

Service Degradation

Network Bandwidth Utilization

Network Traffic

Background

Performance Metrics

1. **Total Migration Time (TMT) :** The time duration from the start to the completion of a migration.
2. **Downtime (DT) :** the time duration from suspending the VM on the source host to successfully starting it on the destination host.
3. **Service Degradation :** The impact on the services executing in a VM due to migration.
4. **Network Traffic :** The total data flow through migration.
5. **Network Bandwidth Utilization :** The combined measure of TMT and Network Traffic.

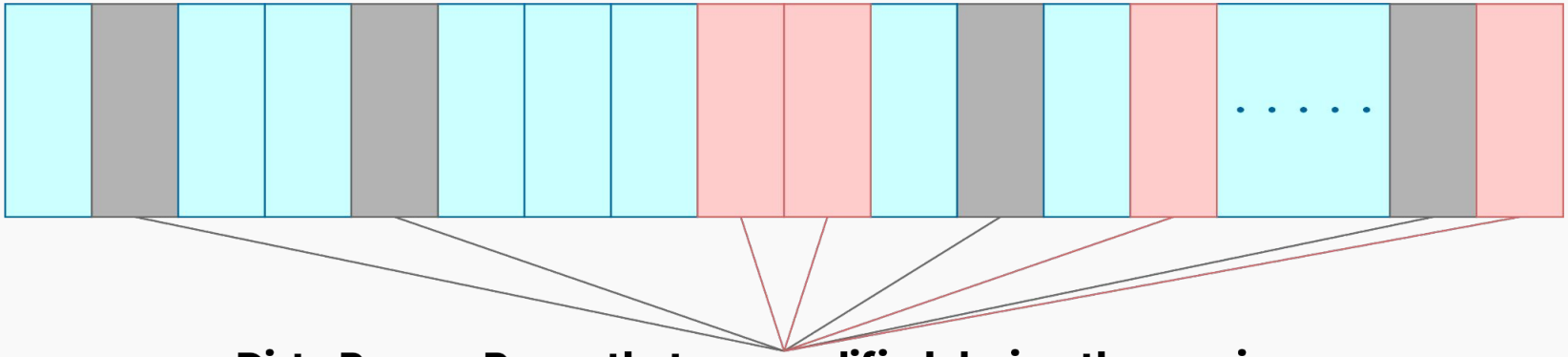
Motivation

- Cloud Computing (CC) is an essential technology in the modern world which provides efficient and reliable services.
- Cloud Computing (CC) Service Providers
  - Microsoft Azure
  - Amazon Web Services (AWS)
  - Alibaba Cloud
  - Google Cloud Platform (GCP)
- Millions of users worldwide
- Service Policy
  - Uninterrupted Services.

Nathan et al., ACM SIGPLAN Notices, 2016

Li et al., Future Generation Computer Systems, 2019

Background



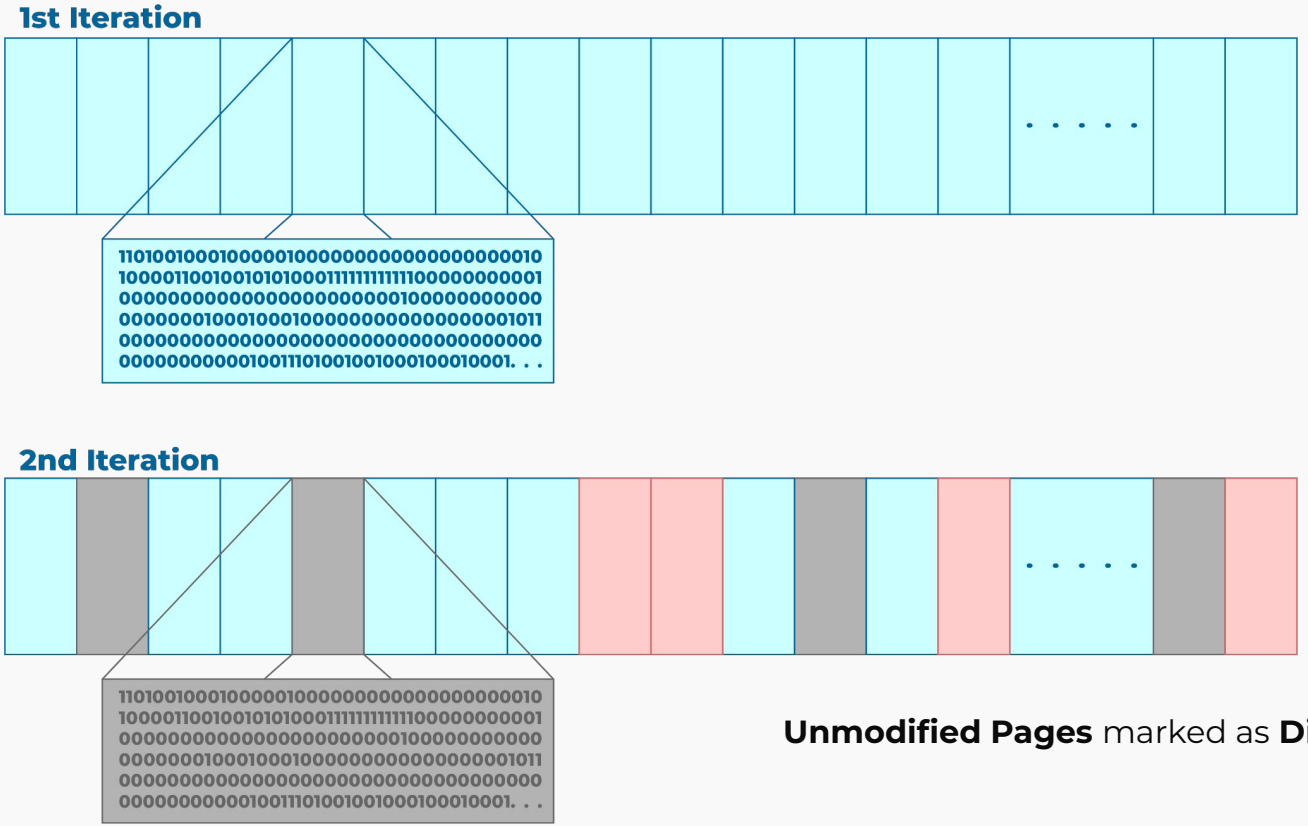
Dirty Pages : Pages that are modified during the previous

RAM in Pre-Copy Migration Second Iteration

Nathan et al., ACM SIGPLAN Notices, 2016

Li et al., Future Generation Computer Systems, 2019

Background



Research Gap

01 Overlooked the potential benefits of using different hashing techniques over SHA1 in hash-based fake dirty prevention algorithm.

02 No application different optimization techniques along with the hash-based fake dirty prevention algorithm

Motivation

- Data Centers embrace Virtualization to provide services.
- Data Centers maintain host multiple VMs in a single server.
- These servers can be subjected to various hardware or software failures.
    - This affects the interruptions to the services provided by the VMs in the failing server.

Solution :

- VMs in the server are migrated to another physical server by using ***Live VM Migration techniques***

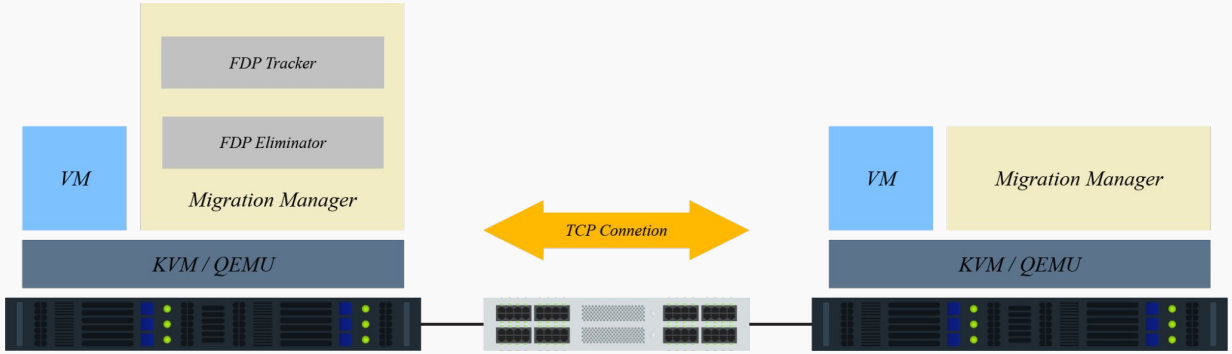
Usage of VM Migration :

- Fault Tolerance, Load Balancing, Host maintenance, Consolidation and etc.

Progress So Far

01. Testbed Setup

- Setuped two physical servers and a NFS server interconnected with Gigabit ethernet.
- Installed QEMU/KVM in Servers.



Progress So Far

02. Selecting Suitable Workloads

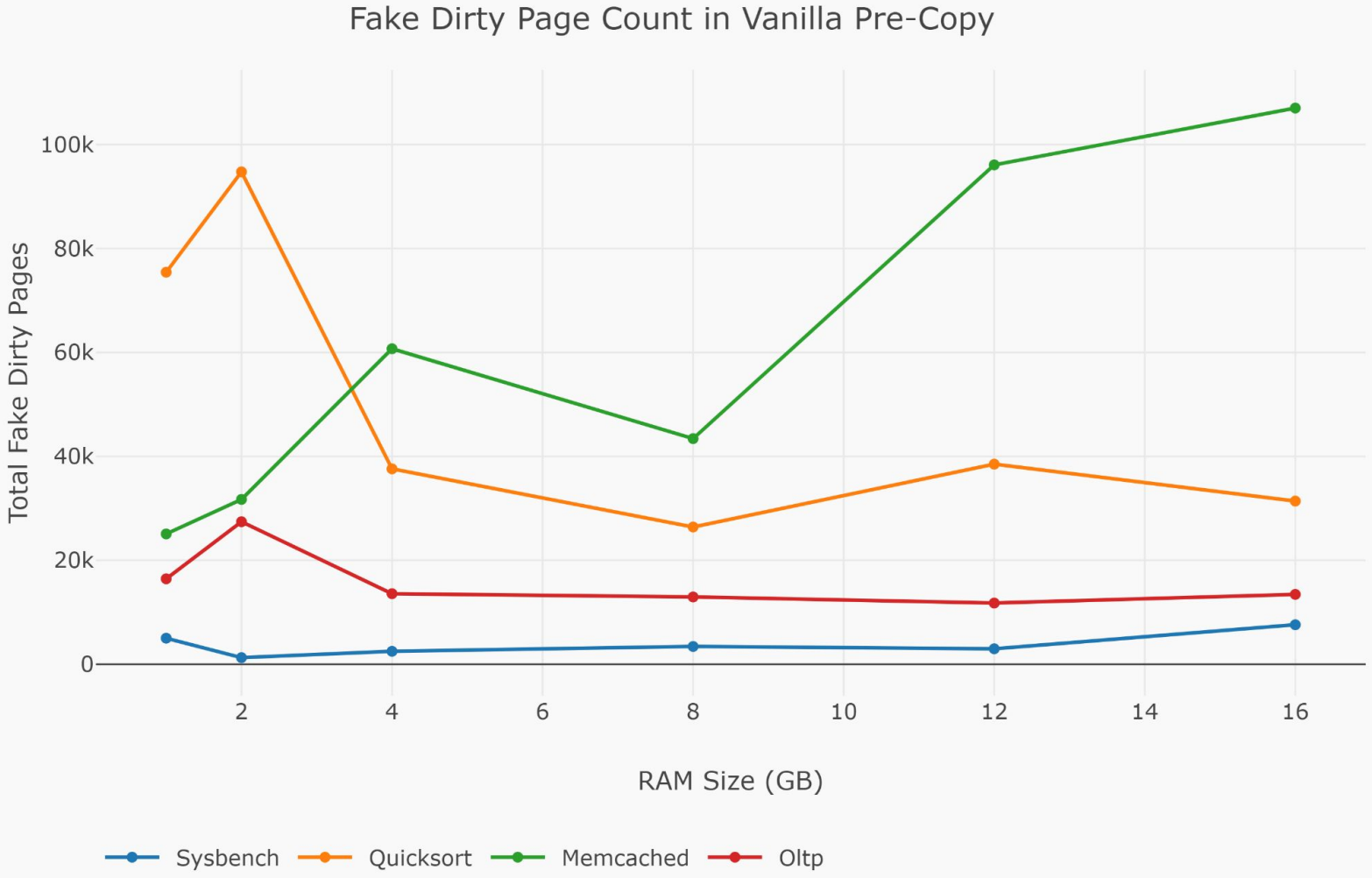
Workloads	Intensive Type	Description
Workingset	Memory	A benchmark that dirty pages to vary writable working set
Quicksort	CPU	A benchmark that repeatedly allocates random integers to an integer array of 1024 bytes and performs quicksort on the array.
Sysbench	CPU	A benchmark to assess the system performance of a machine planning on running a database under intensive load
Memcached	Multiple Resource	Memcached is an in-memory key-value store storing arbitrary data returned by a benchmark called Memaslap
YCSB	Multiple Resource	A Suite used to evaluate computer programs' retrieval and maintenance capabilities.

Progress So Far

03. Preliminary Experiments

- The initial experiment was to observe the presence of Fake Dirty Pages in Vanilla Pre-Copy Migration. The experiment was conducted for six VM memory sizes *(1GB, 2GB, 4GB, 8GB, 12GB, 16GB)*.
- The required readings were collected by executing four workloads in each Virtual Machine memory size.
- Each data point is an average of 3 rounds of experiment.

Analysis in the Next Slide

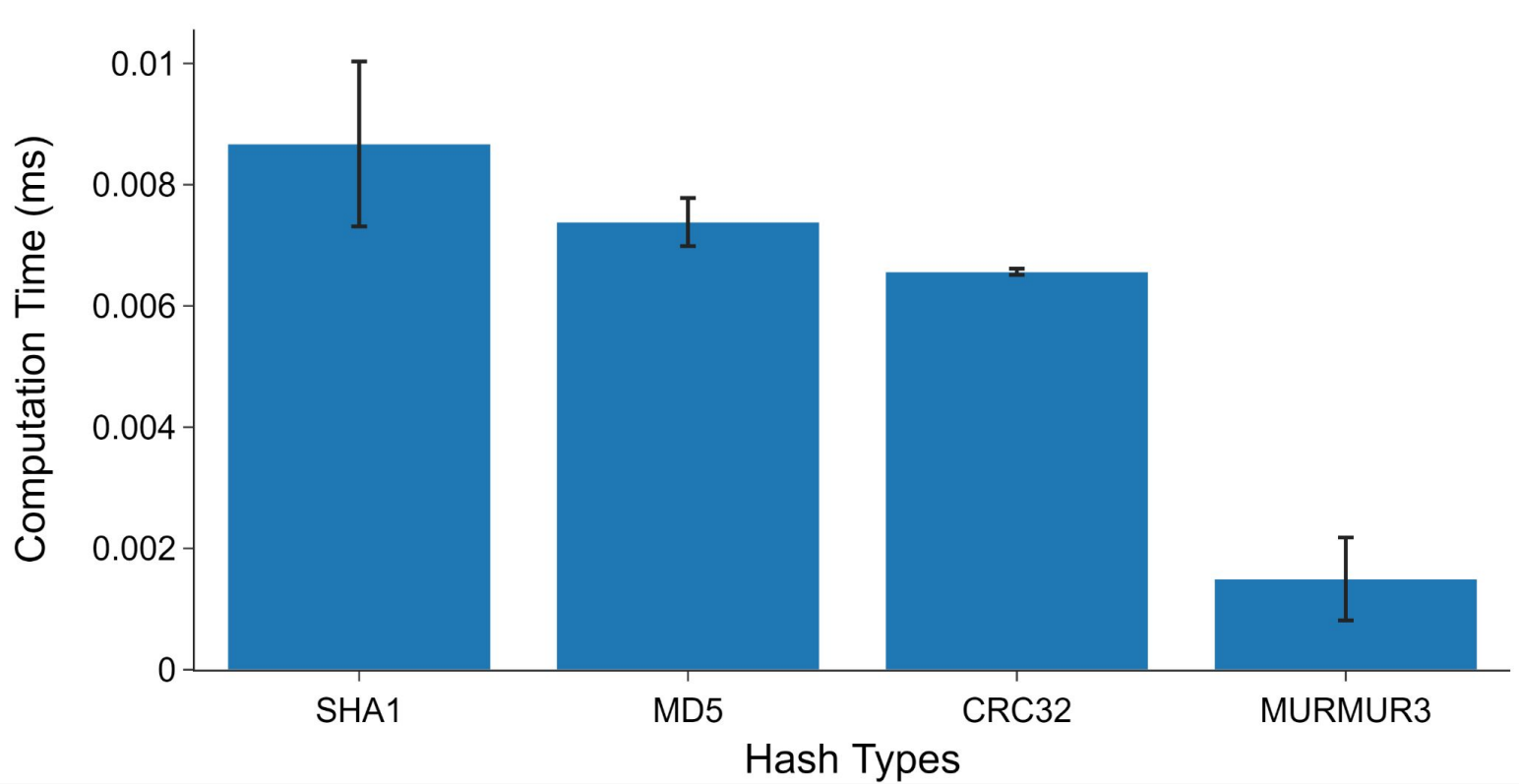

Fake Dirty Page Count in Vanilla Pre-Copy
Total Fake Dirty Pages
100k
80k
60k
40k
20k
0
2
4
6
8
10
12
14
16
RAM Size (GB)
Sysbench
Quicksort
Memcached
Oltp

Progress So Far

03. Preliminary Experiments Cont.

- The next experiment was to observe the computation time of different hashing methods.
- SHA1, MD5, CRC32, and Murmur3 hashing methods were used to conduct this experiment.
- The experiment was done in the C environment.
- In the experimental setup, a 4KB memory was allocated with random numbers and hashed using each hashing method, and the computation time was recorded.

Analysis in the Next Slide


Computation Time (ms)
0.01
0.008
0.006
0.004
0.002
0
SHA1
MD5
CRC32
MURMUR3
Hash Types

Scope

In Scope

Develop prototypes for eliminating Fake Dirty Pages using different hash techniques.

Incorporating different optimization techniques to further minimize total migration time

Out Scope

Non-Linux Operating Systems

WAN migration

Multiple VM Migration