

Enhancing Convergence of Live VM Migration with Dynamic Workloads

NAME: D.N.NETHMINI EPA | 20000499

SUPERVISOR: DR. DINUNI K. FERNANDO

CO-SUPERVISOR: DR. JEROME DINAL HERATH

• Migration techniques perform differently with workloads

- **“...even moderately write-intensive workloads can reduce precopy’s effectiveness during migration”**

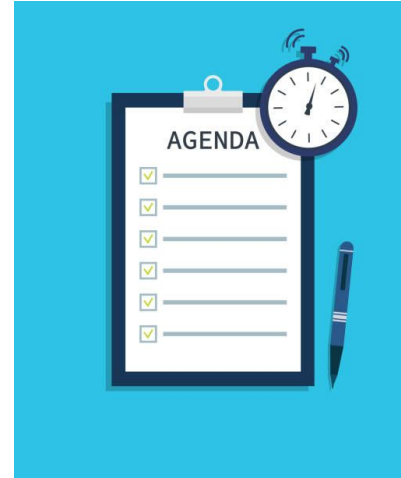
- Hines, M. R., Deshpande, U. & Gopalan, K. (2009), ‘Post-copy live migration of virtual machines’, ACM SIGOPS operating systems review 43(3), 14–26.

“... post-copy doesn’t perform well with read intensive loads. A read intensive VM will lead to an increase in the number of page faults ”

Sahni, S. and Varma, V., 2012, October. A hybrid approach to live migration of virtual machines. In 2012 IEEE international conference on cloud computing in emerging markets (CCEM) (pp. 1-5). IEEE.

Agenda

1. Background
2. Motivation
3. Related Work
4. Research Gap
5. Research Questions
6. Objectives
7. Scope
8. Research Approach and Progress
9. Evaluation



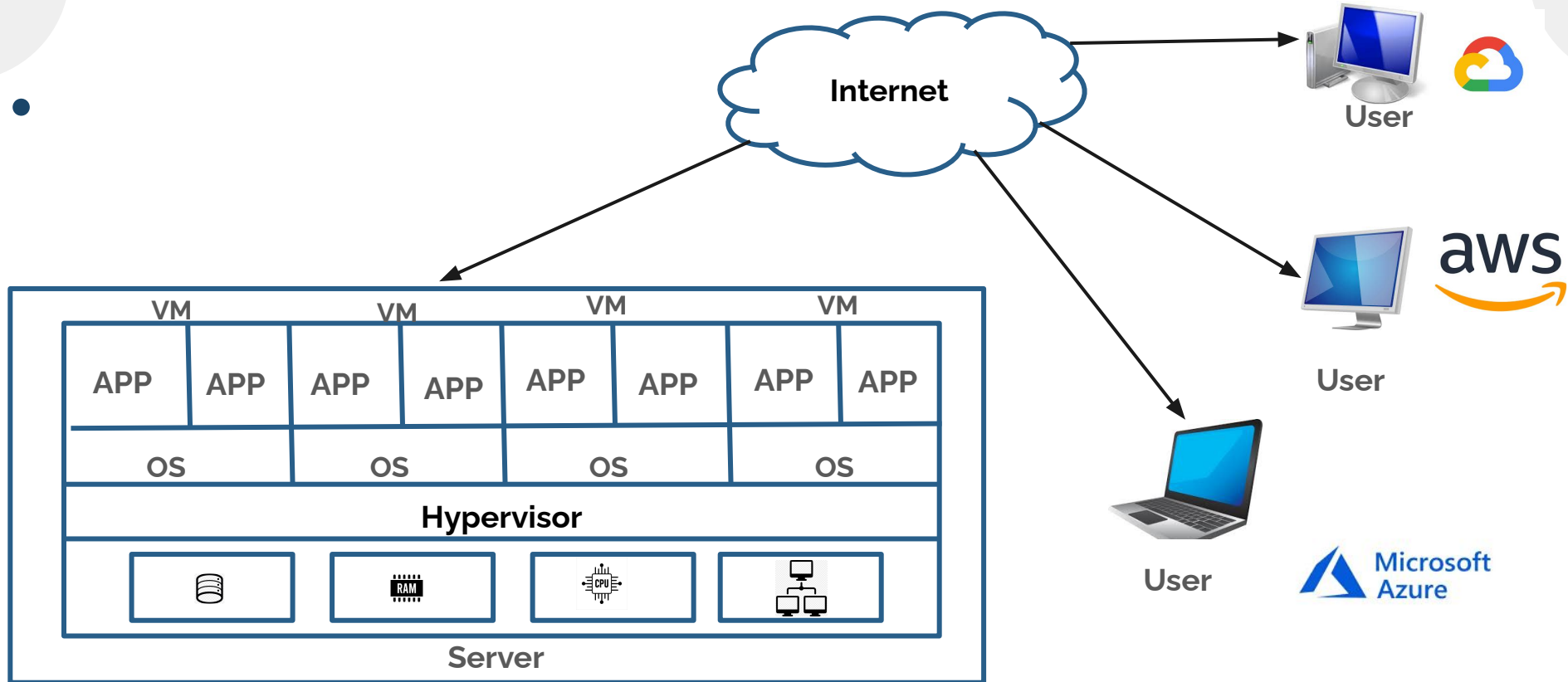
Background

Servers



Cloud Data Center

Background



Failures in Google Data Centers

- Failure of a server can occur at any time.
 - Failures due to hardware malfunctions
 - Unexpected power loss impacting the host server
 - Bugs or crashes in the hypervisor software(KVM)



"In each cluster's first year, it's typical that 1,000 machine failures will occur; one power distribution unit will fail, bringing down 500 to 1,000 machines; 20 racks will fail, each time causing 40 to 80 machines to vanish from the network; there is about a 50 percent chance that the cluster will overheat, taking down most of the servers in less than 5 minutes"

(R. Miller, "Failure rates in google data centers," <https://www.datacenterknowledge.com/archives/2008/05/30/failure-rates-in-google-data-centers>, 2008)

VM Migration

- **VM migration** is essential in virtualized and cloud environments. It allows for the transfer of virtual machines from one physical host to another with minimal downtime.
- Used for **fault tolerance, host maintenance, load balancing and consolidation** as well.

VM Migration Techniques

Live VM Migration

VM continues to operate, and its services remain available during the migration process.

Pre-Copy Migration

Hybrid Migration

Post-Copy Migration

Non-Live VM Migration

VM is usually powered off during the migration process

During Migration

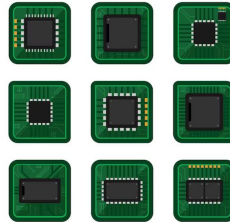


Virtual machine

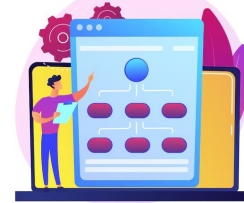
Memory Content



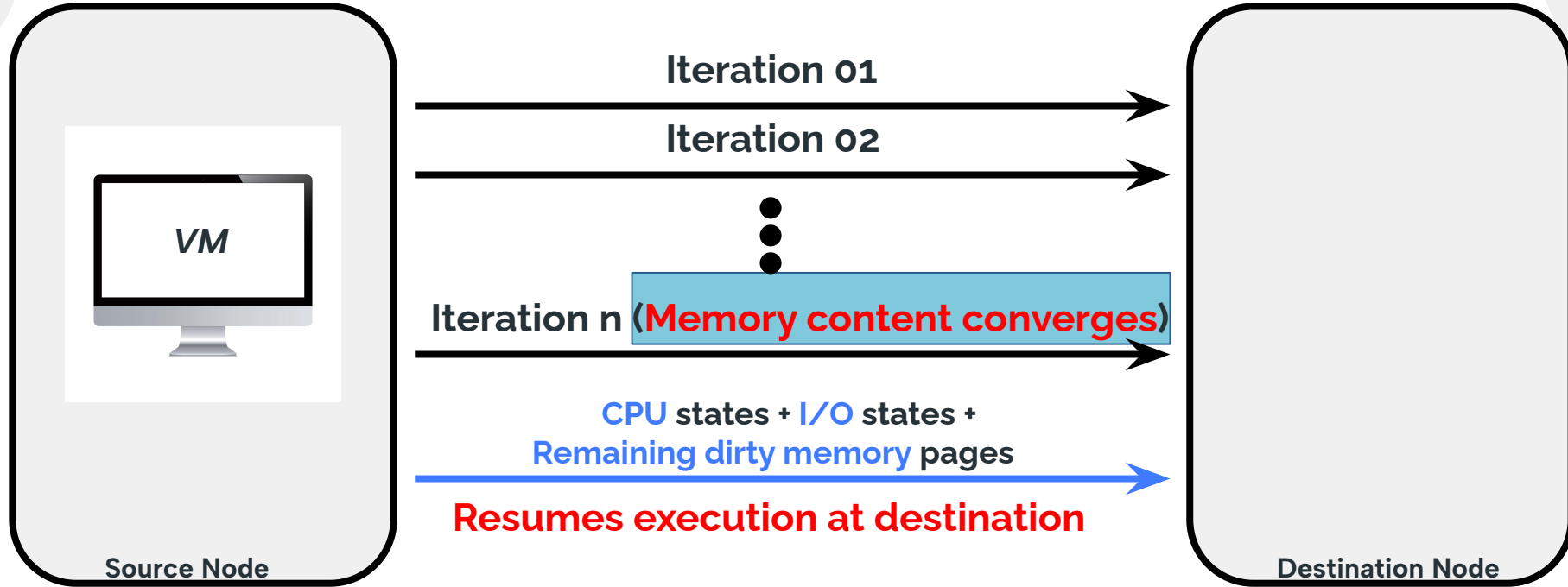
CPU State



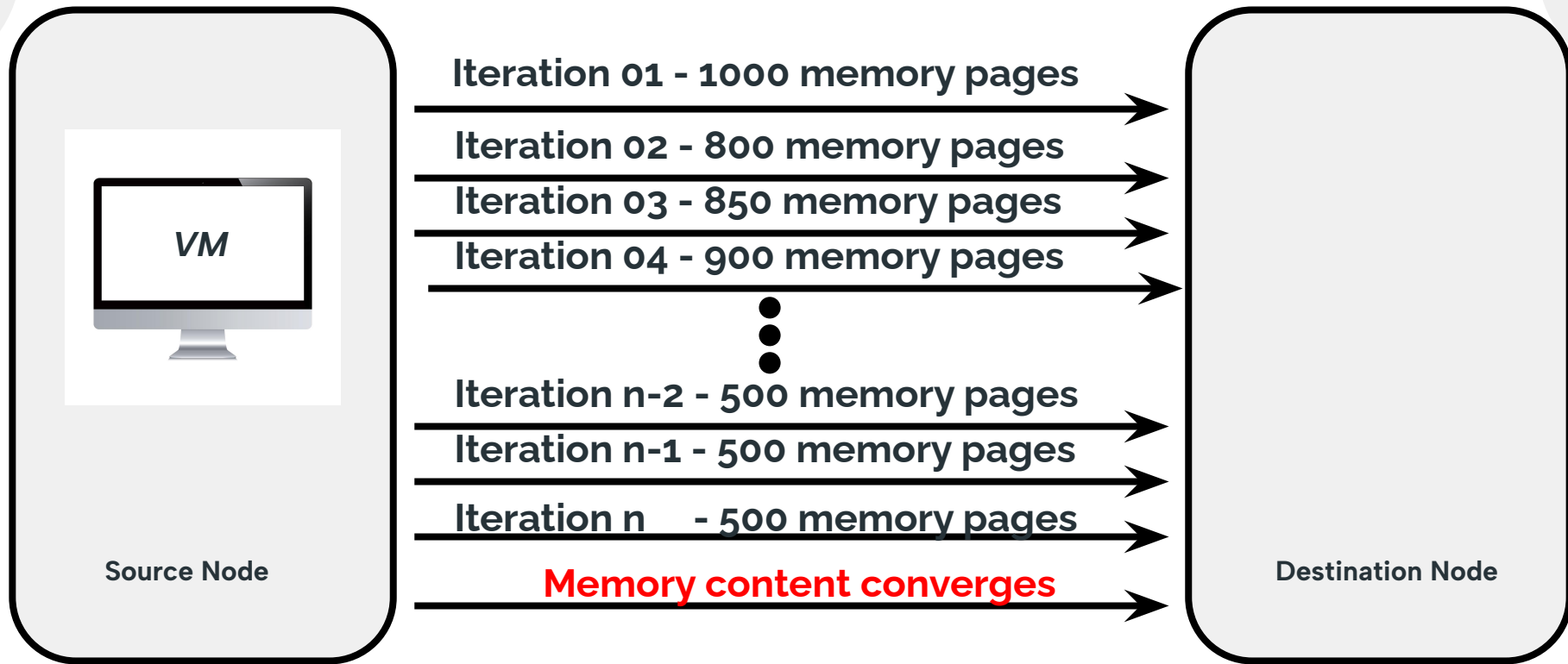
I/O State



Pre-Copy Migration



Pre-Copy Migration- Convergence point

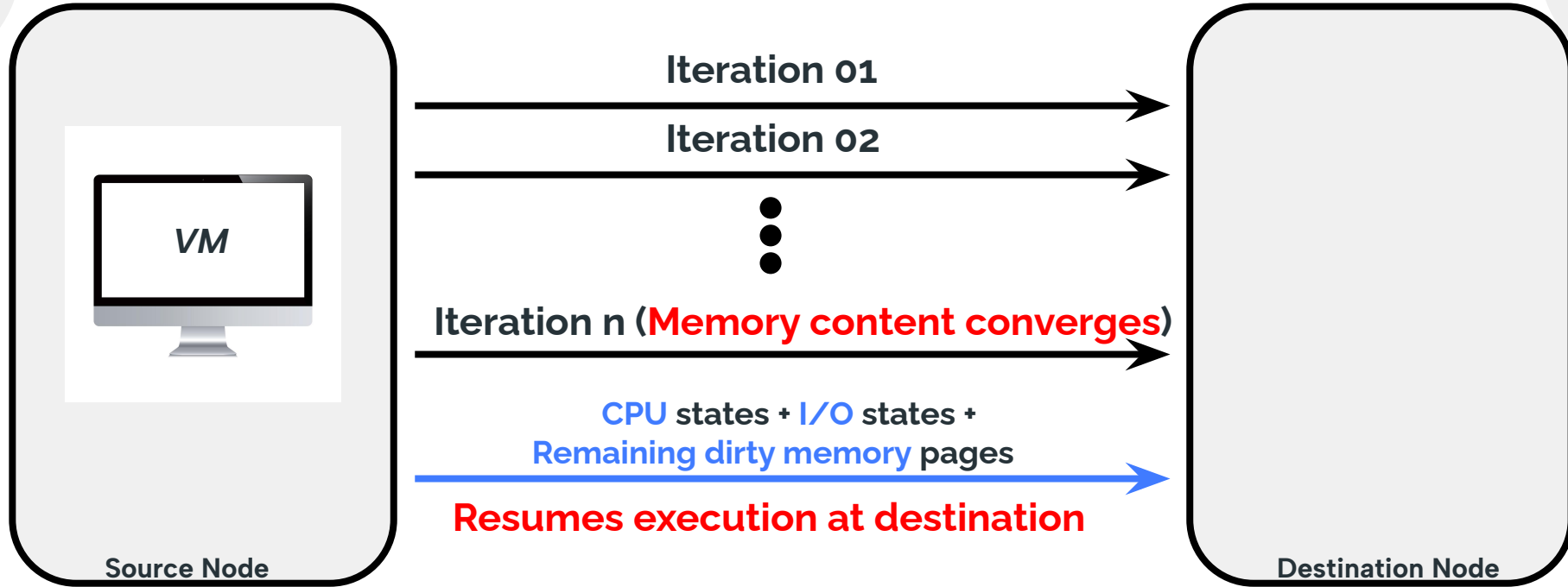


Identifying the Convergence point is important

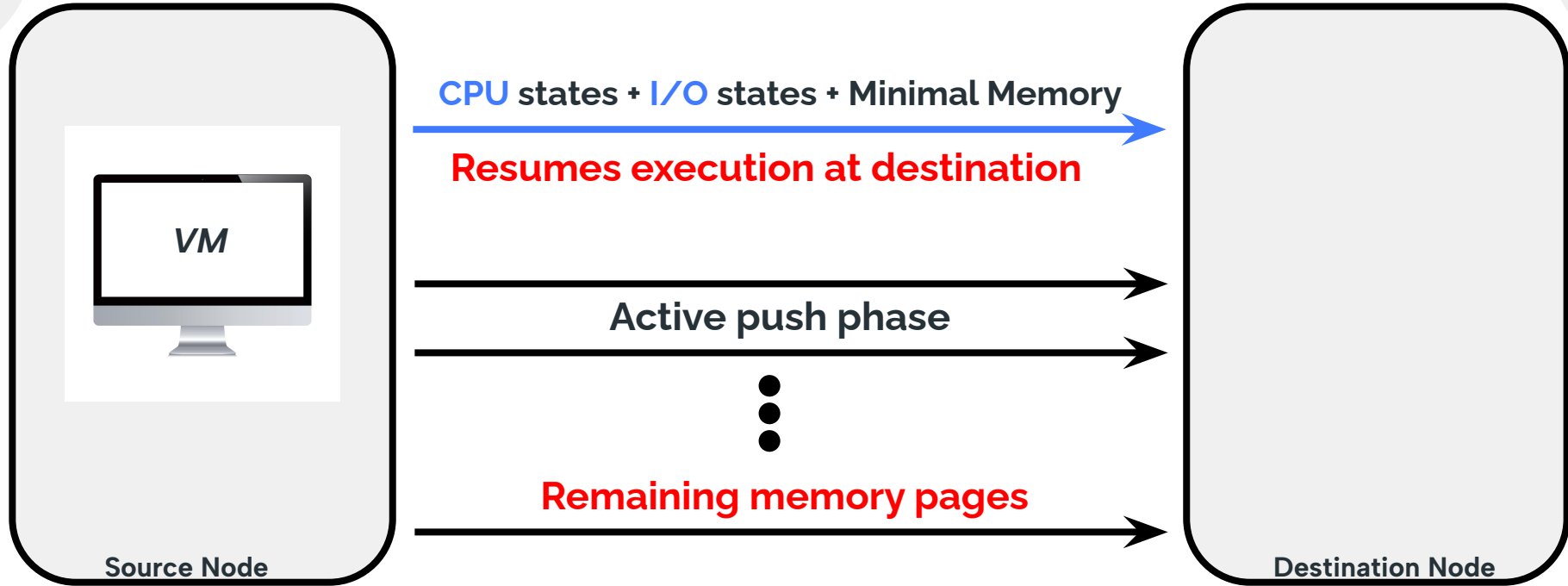
- It will provide provisions to deterministically predict how long migration lasts upon a triggering event.
- Provides a seamless switchover with **minimal downtime** and no data inconsistencies.



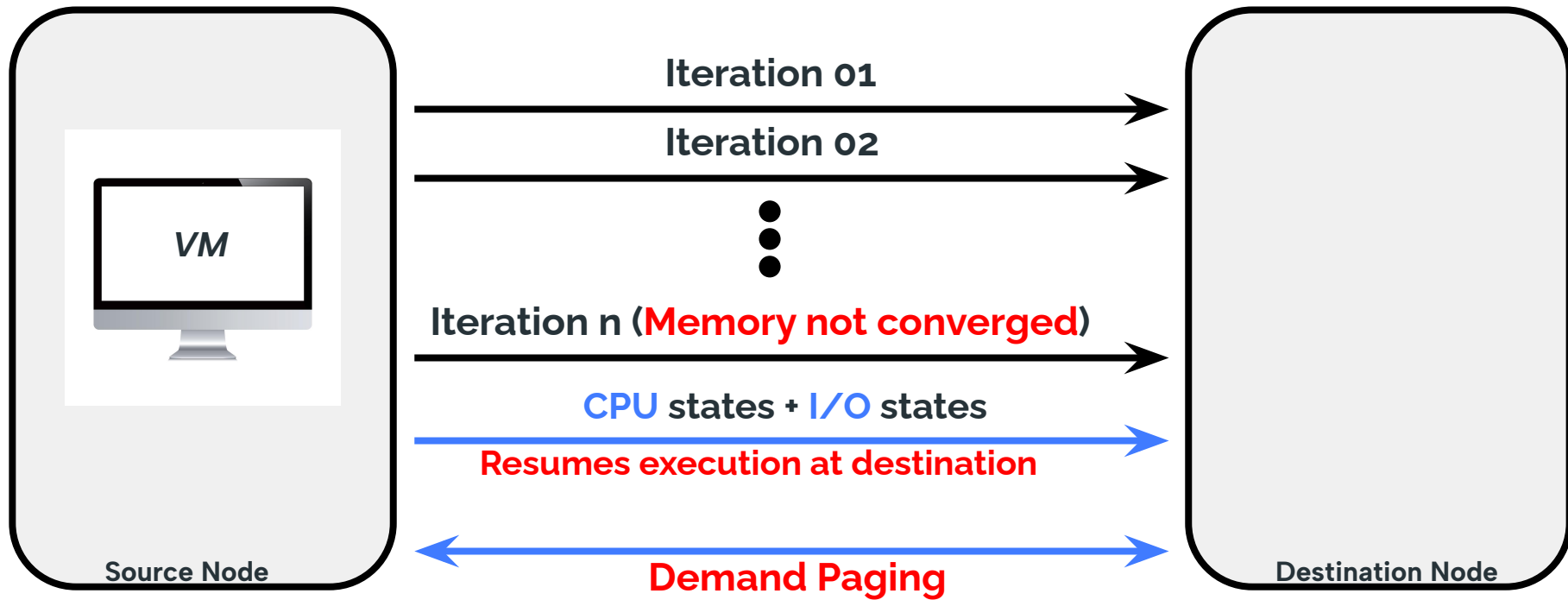
Pre-Copy Migration



Post-Copy Migration



Hybrid Migration



Performance Metrics

Total Migration Time

→ Duration between the start and end of migration

Downtime

→ The amount of time the VM is unavailable during the migration process.

Application Performance Degradation

→ The effect of the migration on the performance of migrating VM

Workload

The amount of time and computing resources a system or network takes to complete a task or generate a particular output.

Static Workloads

Consists of constant amount of computer resources for a long period of time

CPU
intensive
workloads

Memory
intensive
workloads

Network
intensive
workloads

Dynamic Workloads

Change the workload behavior very frequently

Persistent
Workloads

Real Time
Workloads

Serverless
Workloads

Dynamic Workloads

Change the workload behavior very frequently

Persistent
Workloads

Database Management
Systems (DBMS): **MySQL**,
PostgreSQL, **MongoDB**



YCSB Benchmark

Real Time
Workloads

Video Streaming: **Youtube**,
Netflix



Stress Tool

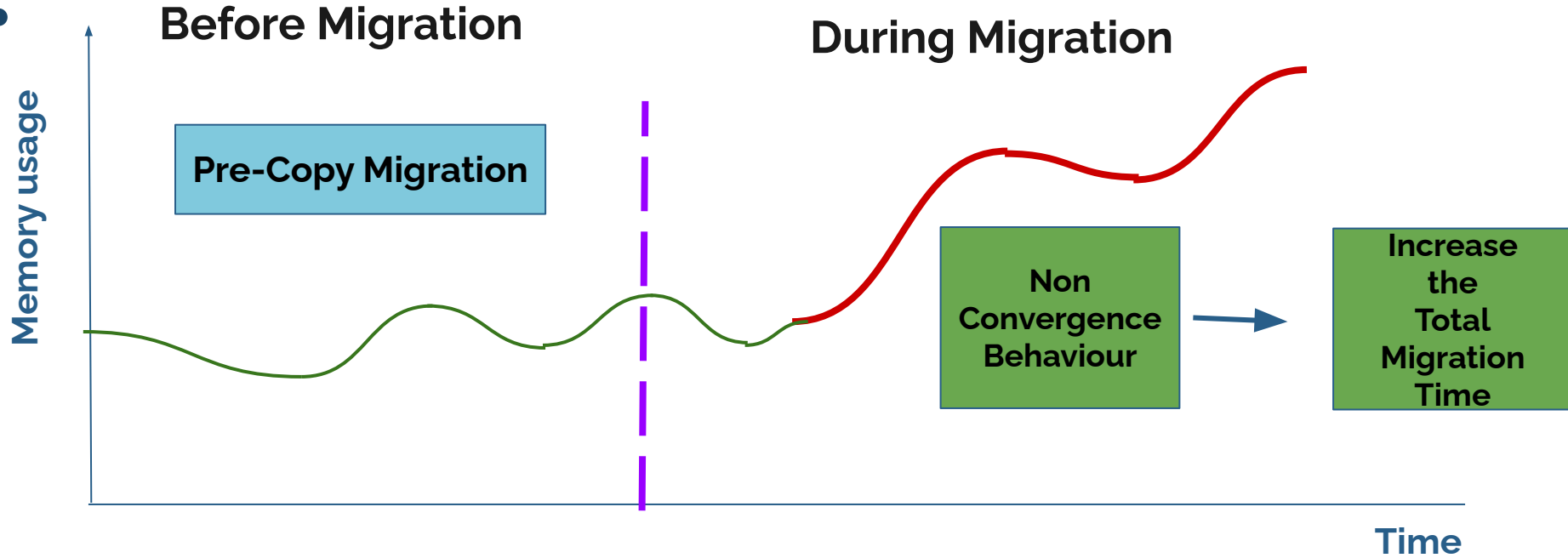
Serverless
Workloads

Event-Triggered Data
Processing: **AWS**
Lambda, **Google**
Cloud Functions



Motivation

Dynamic workload



Traditional approach

In the traditional vanilla Pre-Copy algorithm, if convergence is not achieved within 50 iterations, it switches to Post-Copy, creating a hybrid approach.

If we anticipate high memory intensity in the workload, we can switch to Post-Copy earlier, reducing the number of Pre-Copy iterations and decreasing the Total Migration Time (TMT).

It lowers the risk of server failure before migration completes.

How do we predict the future workload?

Workload Prediction techniques

Machine learning techniques

Recurrent
Neural
Networks

Support
Vector
Regression

Supervised
Learning

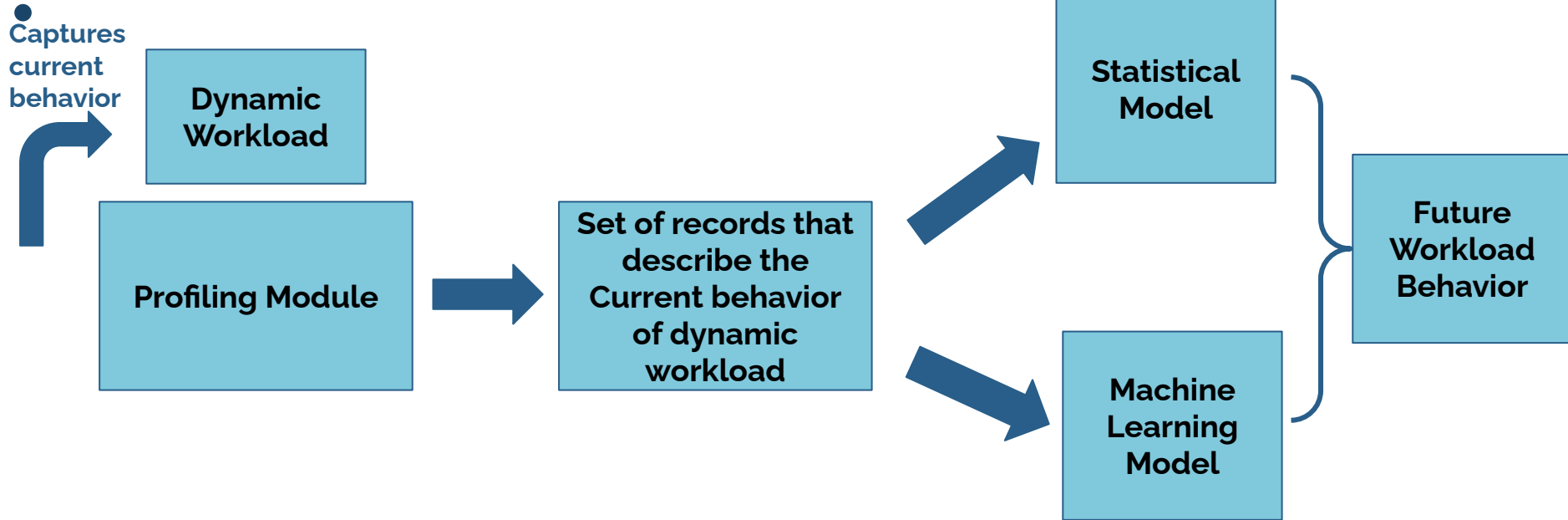
Statistical techniques

Time Series
Analysis

Markov
Chains

- To predict the future behavior these techniques require capturing and collecting records of the current behavior of the workload to predict future behaviors.
- For that process, a well-known technique called **profiling** can be used.

Workflow



What is the best workload prediction approach?

Machine Learning Techniques



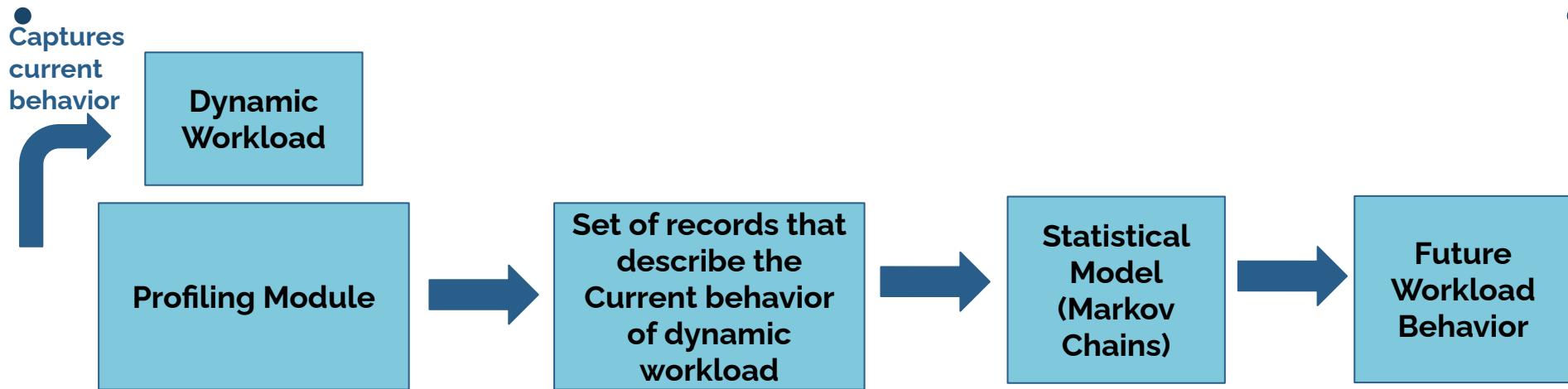
OR

Statistical Techniques



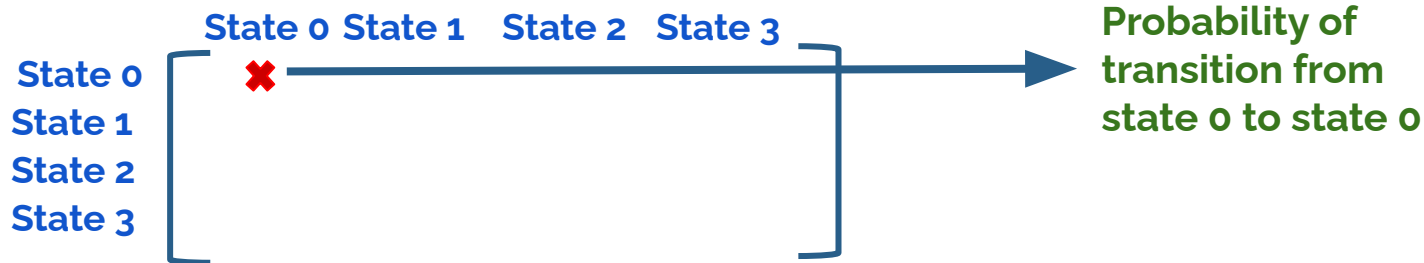
- Lower computational resource requirements
- Need of less offline data
- No need to predict the exact values

Workflow



Markov Chains

- Maps a workload into set of possible states.
- Example-:
 - **Resources:** CPU usage , Memory Usage
 - **Usage Levels:** less than 50%, greater than or equal to 50%
- All possible states - 4 states
 - **State 0:** CPU <50% and Memory <50%
 - **State 1:** CPU <50% and Memory >50%
 - **State 2:** CPU >50% and Memory <50%
 - **State 3:** CPU >50% and Memory >50%
- Represent state transitions using a matrix.



Calculate the probabilities

Initial count matrix

	State 0	State 1	State 2	State 3
State 0	0	0	0	0
State 1	0	0	0	0
State 2	0	0	0	0
State 3	0	0	0	0

Initial transition matrix

	State 0	State 1	State 2	State 3
State 0	0.25	0.25	0.25	0.25
State 1	0.25	0.25	0.25	0.25
State 2	0.25	0.25	0.25	0.25
State 3	0.25	0.25	0.25	0.25

Transition from (state 0 -> state 1)

Updated count matrix

	State 0	State 1	State 2	State 3
State 0	0	1	0	0
State 1	0	0	0	0
State 2	0	0	0	0
State 3	0	0	0	0

Updated transition matrix

	State 0	State 1	State 2	State 3
State 0	0/1	1/1	0/1	0/1
State 1	0.25	0.25	0.25	0.25
State 2	0.25	0.25	0.25	0.25
State 3	0.25	0.25	0.25	0.25

Predicting the future states using transition matrix

- Current resource usage of the VM: CPU 25%, memory 25%

So, the current state is: state 0

Initial state vector : [1 0 0 0]

State vector for second state = Initial state vector x transition matrix

$$\begin{aligned} &= [1 \ 0 \ 0 \ 0] \times \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix} \\ &= [0 \ 1 \ 0 \ 0] \end{aligned}$$

Next state :

Take the maximum
probability

Next State
is State 01

Predicting the future states using transition matrix

- State after the next state : state vector for previous state x transition matrix

$$:[0 \ 1 \ 0 \ 0] \times \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}$$

:(initial state vector x transition matrix) x transition matrix

:initial state vector x [transition matrix]²

So, the state after **n** steps is,

$$\text{Initial state vector} \times [\text{transition matrix}]^n$$



Related Works

Dynamic workload migration

Naik, K. J. (2022), 'An adaptive push-pull for disseminating dynamic workload and virtual machine live migration in cloud computing'

- Use an Adaptive Push-Pull algorithm to migrate a dynamic workload.
- The VM Manager manages workload within the same server, and the CRM (Central Resource Manager) oversees workload distribution across all servers.

Glap:Khelghatdoust, M., Gramoli, V. & Sun, D. 2016 IEEE International Conference on Cluster Computing (CLUSTER)

- Developed a fully distributed and threshold-free Dynamic Virtual Machine Consolidation algorithm.
- The solution relies on an unstructured gossip-based protocol and Q-Learning .
- Migrate the VM using 2 a two phase algorithm

Research Gap

Research Gap

- Most of the researchers has used ML techniques and complex mathematical models handle and migrate a dynamic workload.
- Training and deploying ML models can requires
 - A long period of training time
 - A huge set of data requirements
 - A high computation power
- If we can get the same accuracy with a low resource intensive solution then it will be positively impact the performance of the VM.

So, this research aims to propose a less resource intensive statistical technique(Markov chains) to migrate a dynamic workload.

Research Questions

1. How to identify the convergence point of dynamic workloads in live migration with workload-specific behavior?
2. How can the performance of vanilla pre-copy be improved upon detection of convergence point in terms of total migration time, downtime, and application performance?

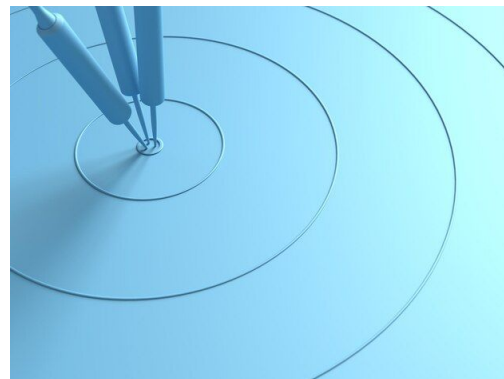



Research Objectives

1. Design and develop a method to predict the future behavior of a dynamic workload.
2. Design and develop an algorithm to select the best migration strategy to migrate a dynamic workload.
3. Identify the convergence point of a dynamic workload.
4. Evaluate the performance of migration using total migration time, downtime, and application performance metrics by incorporating industrial accepted benchmarks such as Sysbench, YCSB (Yahoo cloud serving benchmark), Stress, Memcached, etc.

Scope

- **Live VM migration with dynamic workloads**
- **Single VM migration**
- **Focus on Ubuntu host OS .**
- **Migrate VMs with Linux guest OS**
- **Implementing a working prototype**
- **QEMU-KVM Hypervisor will be used for the prototype implementation**

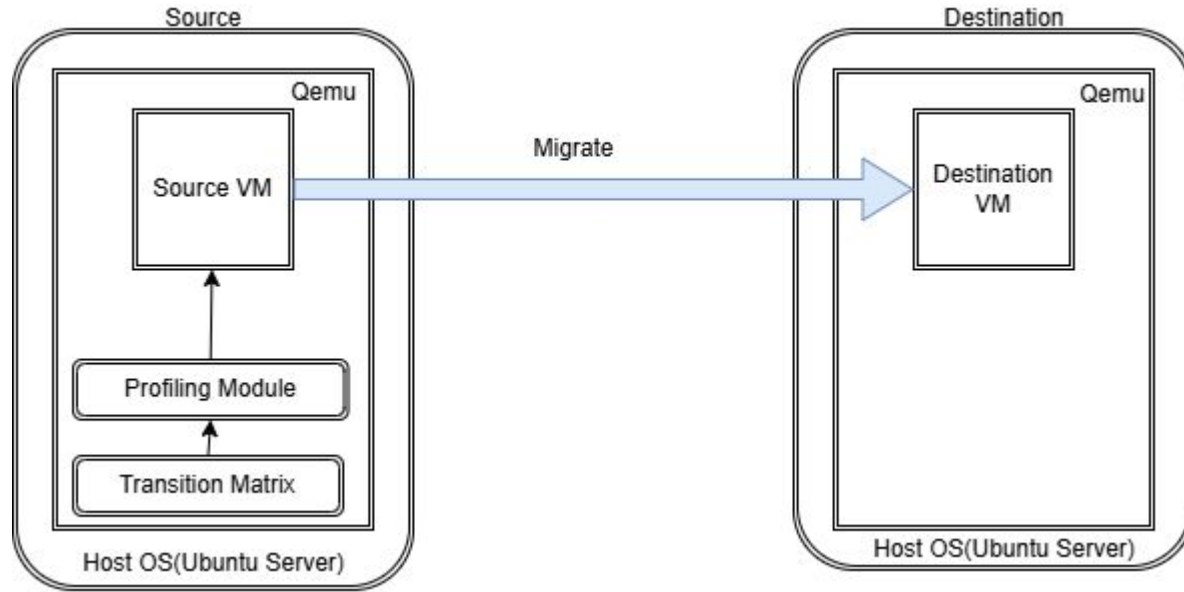




Research Approach and Progress

Research Approach and Progress

- 1. **Testbed setup:** Two physical servers interconnected with Gigabit ethernet will be used. (8 GB RAM, 1000 Mbps Bandwidth and 500+GB HDD)



Research Approach and Progress

2. Implementation of the transition matrix and count matrix:

- **Resources:** CPU, Memory, Incoming Network and Outgoing Network
- **Resource Usage Levels:** Less than 25%, Between 25% to 50%, greater than 50%
- **Total no of states** = $3 \times 3 \times 3 \times 3 = 81$ states
- **So the matrix contains 81x 81 state transitions.**

CPU Usage	Memory Usage	Incoming Network	Outgoing Network
Low	Low	Low	Low
Low	Low	Low	Moderate
Low	Low	Low	High
Low	Low	Moderate	Low
Low	Low	Moderate	Moderate
Low	Low	Moderate	High
Low	Low	High	Low

Low: <25%

Moderate: Between 25% and 50%

High: >50%

Research Approach and Progress

- **3.Implementation of the profiling module:** A profiling module was implemented inside the source host to capture the resource usage of the VM.

This profiling module will capture the resource usage of the VM for each second and map it with the associative state.

CPU usage - using **mpstat** command

Memory usage - using **free** command

Network usage - using **ifstat** command

Research Approach and Progress

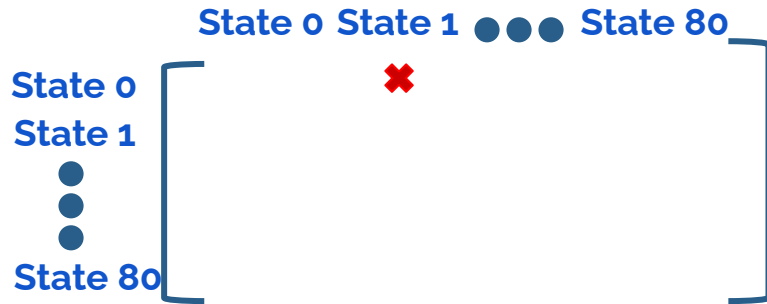
3. Implementation of the profiling module:

Ex-: CPU:12% memory:45% incoming_network:12% outgoing_network:5%

This will mapped to : low, moderate, low, low state → State 1

Previous state : low, low, low, low state → state 0

State Transition: state 0 → state 1



According to the current state the count matrix and the transition matrix will be updated.

Research Approach and Progress

- 4. **Design and development of the algorithm:** An algorithm was implemented to map all possible states to the most suitable migration decision.

To map the states with the migration decision the following conditions will be used.

1. If a VM is memory-intensive, it is migrated using **post-copy**
2. For a network-intensive workload with mostly outgoing packets, the VM is migrated in **post-copy**
3. If there are mostly incoming packets, they are migrated in **pre-copy**.
4. All other VM workloads are migrated adapting the **hybrid** method.

Research Approach and Progress

- 4. **Design and development of the algorithm:** An algorithm was implemented to map all possible states to the most suitable migration decision.

Decide convergence during migration

- If the selected migration decision is **post-copy** then no need to decide the convergence point.
- If the selected migration decision is **pre copy** then the dirty page rate(**calc-dirty-rate command in QEMU**) will be captured to identify a behavior of **exponential decay**.



Research Approach and Progress

4. Design and development of the algorithm

How we can identify an exponential decay using the dirty page rate.

Iteration 01 : 1000 pages per second
Iteration 02: 800 pages per second
Iteration 03: 900 pages per second



Iteration 01 : 1000 pages per second
Iteration 02: 800 pages per second
Iteration 03: 600 pages per second



Research Approach and Progress

4. Design and development of the algorithm

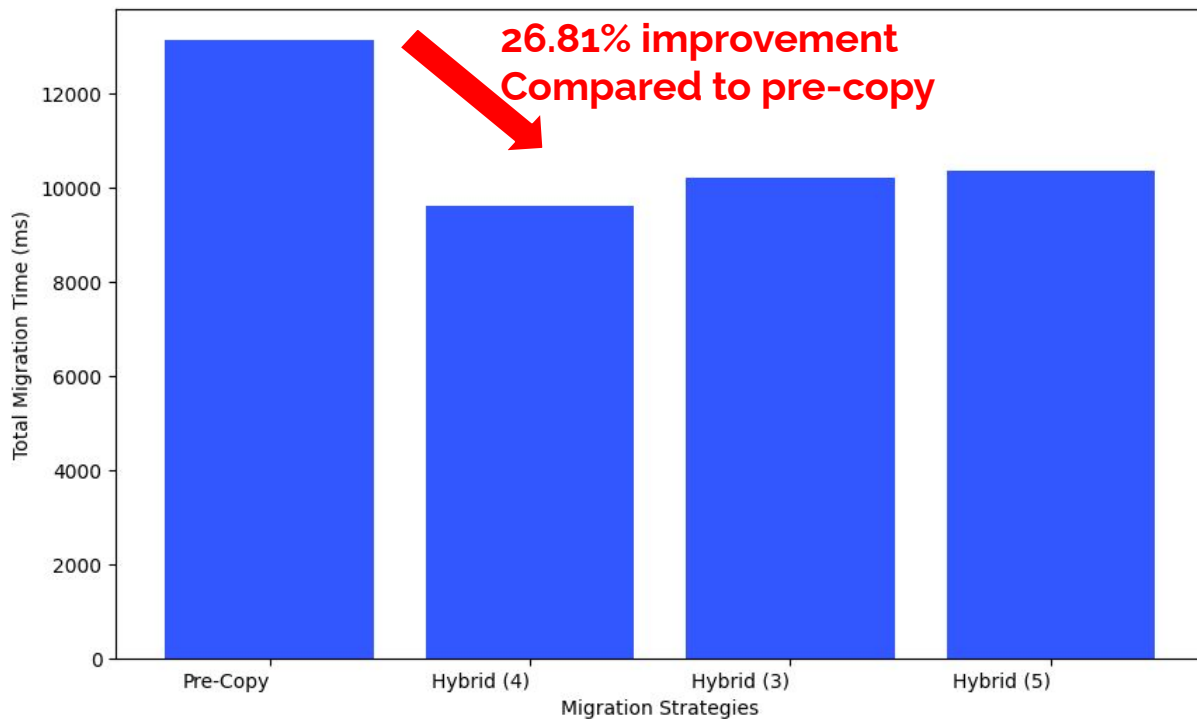
If the workload does not show a pattern of exponential decay and the dirty page rate increases rapidly then,

It suggests that the workload should switch to the post copy migration technique to complete the migration.

To handle this kind of situations a **maximum no if iteration count** will be used.

This will be a **workload specific parameter**.

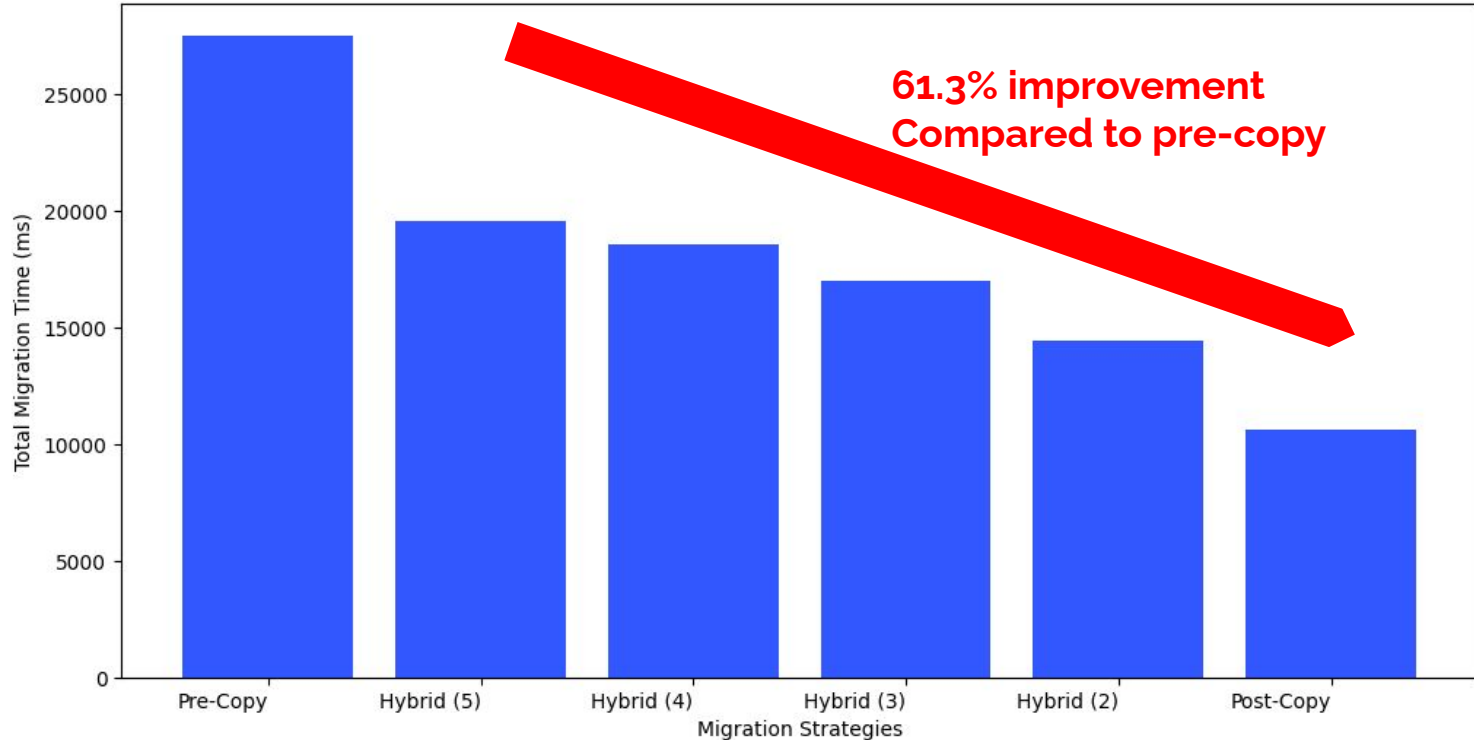
Experiments to Decide the Maximum No of Iterations- (Persistent Workloads)



hybrid(4) has,

- 5.6% improvement compared to Hybrid(3)
- 7.2% improvement compared to Hybrid(5)

Experiments to Decide the Maximum No of Iterations- (Real time Workloads)



Algorithm

Algorithm 1 Migration Decision Algorithm

```
1: Input: Max_Iterations =  $m$ , Look_ahead_states =  $n$ 
2: Initialize: Current_Migration_Decision  $\leftarrow$  Get the migration decision based
   on the current state
3: Next_behavior  $\leftarrow$  Calculate the behavior of the upcoming  $n$  states using the
   transition matrix
4: if Current_Migration_Decision == post_copy then
5:   No switching during the migration
6: else if Current_Migration_Decision == pre_copy then
7:   if Next_behavior is memory-intensive and (Iteration_count =  $m$  or Dirty
     page rate shows an exponential decay) then
8:     Switch to post_copy
9:   else
10:    Vanilla pre_copy
11:   end if
12: else if Current_Migration_Decision == Hybrid then
13:   if Iteration_count =  $m$  or Dirty page rate shows an exponential decay then
14:     Switch to post_copy
15:   end if
16: end if
```

Algorithm

Algorithm 1 Migration Decision Algorithm

```
1: Input: Max Iterations =  $m$ . Look ahead states =  $n$ 
2: Initialize: Current_Migration_Decision  $\leftarrow$  Get the migration decision based
   on the current state
3: Next_behavior  $\leftarrow$  Calculate the behavior of the upcoming  $n$  states using the
   transition matrix
4: if Current_Migration_Decision == post_copy then
5:   No switching during the migration
6: else if Current_Migration_Decision == pre_copy then
7:   if Next_behavior is memory-intensive and (Iteration_count =  $m$  or Dirty
     page rate shows an exponential decay) then
8:     Switch to post_copy
9:   else
10:    Vanilla pre_copy
11:   end if
12: else if Current_Migration_Decision == Hybrid then
13:   if Iteration_count =  $m$  or Dirty page rate shows an exponential decay then
14:     Switch to post_copy
15:   end if
16: end if
```

Algorithm

Algorithm 1 Migration Decision Algorithm

```
1: Input: Max_Iterations =  $m$ , Look_ahead_states =  $n$ 
2: Initialize: Current_Migration_Decision  $\leftarrow$  Get the migration decision based
   on the current state
3: Next_behavior  $\leftarrow$  Calculate the behavior of the upcoming  $n$  states using the
   transition matrix
4: if Current_Migration_Decision == post_copy then
5:   No switching during the migration
6: else if Current_Migration_Decision == pre_copy then
7:   if Next_behavior is memory-intensive and (Iteration_count =  $m$  or Dirty
     page rate shows an exponential decay) then
8:     Switch to post_copy
9:   else
10:    Vanilla pre_copy
11:   end if
12: else if Current_Migration_Decision == Hybrid then
13:   if Iteration_count =  $m$  or Dirty page rate shows an exponential decay then
14:     Switch to post_copy
15:   end if
16: end if
```



Evaluation and Next Steps

Evaluation

1. Reducing Total Migration Time (TMT) & Downtime

- Reduce TMT and downtime compared to the baseline (vanilla Pre-Copy algorithm).
- Run multiple migration trials and compare results of the proposed method with the Pre-Copy approach.

2. Effect of Look-Ahead Parameter defined in the algorithm

- Assess impact of the look-ahead parameter on prediction accuracy and migration performance.
- Test different parameter values to optimize predictive accuracy and resource efficiency.

Evaluation

3. Initial Transition Matrix Configuration

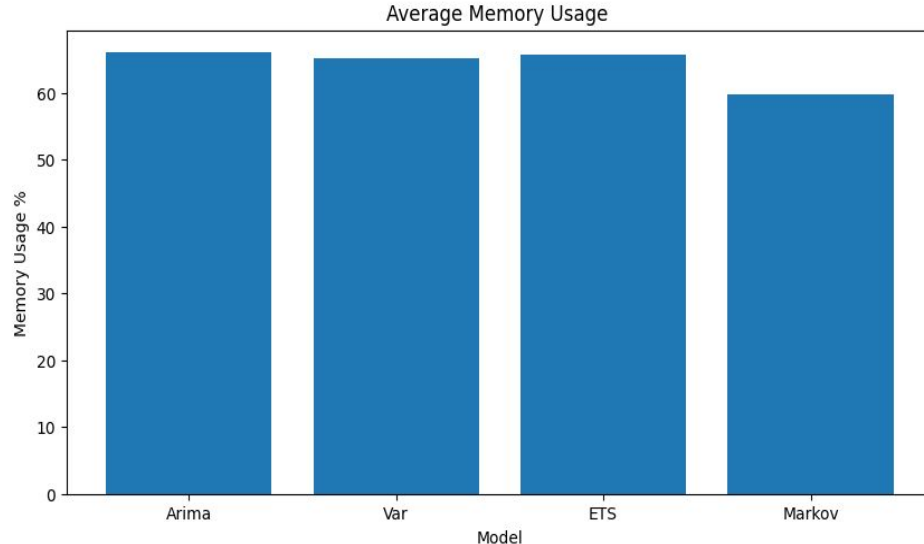
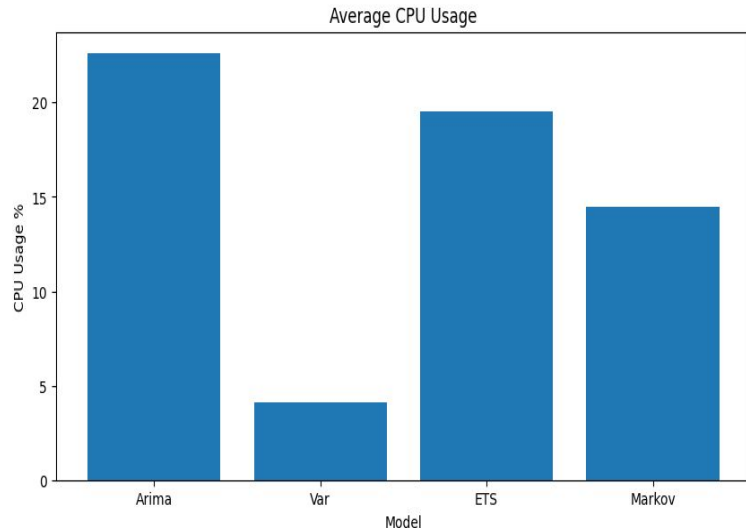
- **Equal Probability for All States:** Neutral starting point with equal transition probabilities.
- **Workload-Specific Matrix:** Use a predefined matrix tailored to the workload
- Compare performance to determine impact on migration efficiency and predictive accuracy.

4. Measure the overhead on host and co-located VMs.

- The overhead on the host and co-located VMs will be measured and compared with the overhead reported in existing related works.

Next Steps

- Reduce the CPU consumption of the Markov chains method.



- Plug the implemented algorithm into the migration process to decide best migration strategy based on the workload nature.

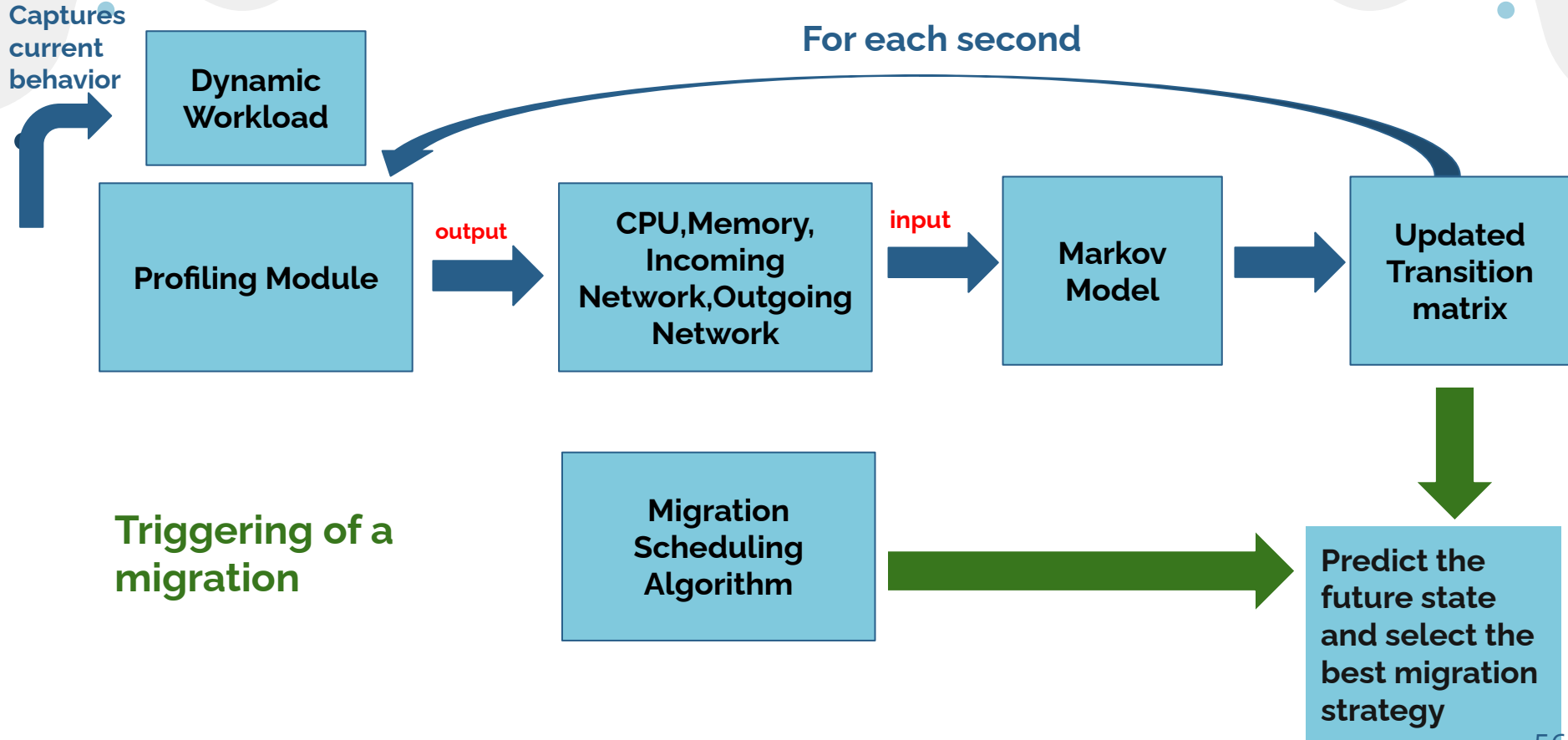
Next Steps

- **Evaluate the whole system according to the criterias given in the evaluation section.**
 - Measure the total migration time and downtime when using the Markov model, and compare it with the vanilla pre-copy algorithm.
 - Analyze the impact of the Look-Ahead Parameter by varying its value.
 - Compare the prediction accuracy between the matrices initialized with equal probabilities for all states and those using predefined matrices.
 - Measure the overhead on the host and co-located VMs when utilizing the Markov model.

Timeline

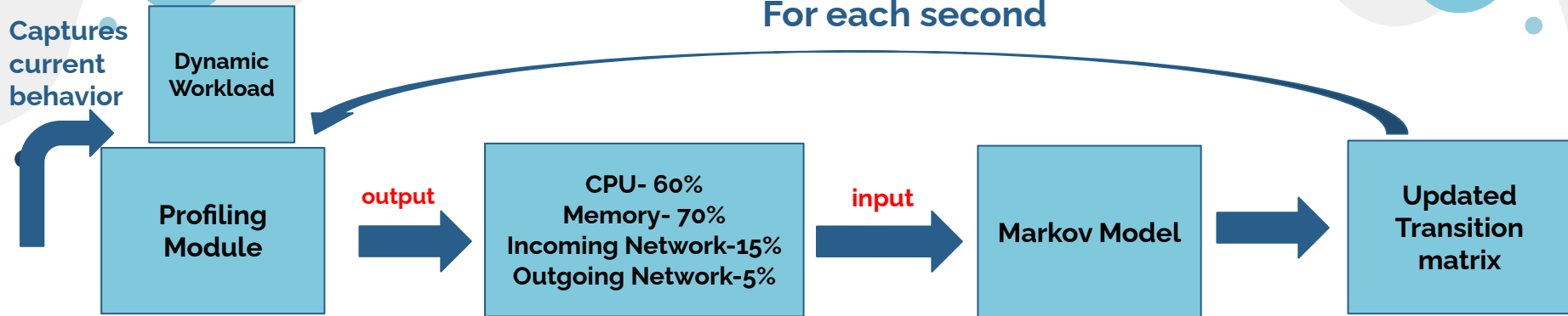
	June	July	August	September	October	November	December	January	February	March	April
Literature Survey											
Background research											
Project proposal											
Getting familiar with profiling tools											
Experiments to find the best statistical approach											
create the pseudo code to map states in markov chain with migration decisions											
Implementation of the transition matrix											
Implementation of the algorithm and connect it with the transition matrix to make the final solution											
performance evaluation											
Thesis writing											
Research publication											

Summary

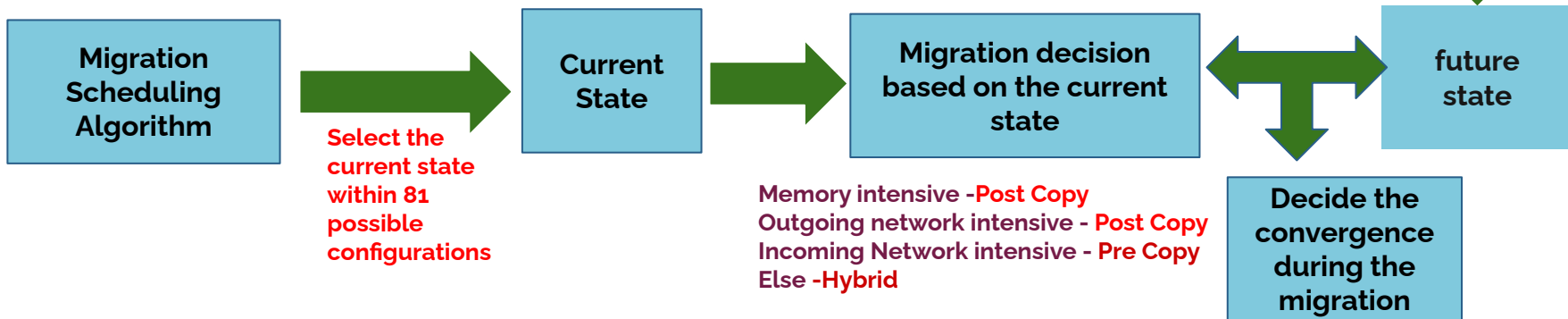


Summary

For each second



Triggering of a migration



References

- Wu, Q. & Wolf, T. (2008), Dynamic workload profiling and task allocation in packet processing systems, in '2008 International Conference on High Performance Switching and Routing', IEEE, pp. 123–130.
- Ye, K., Wu, Z., Wang, C., Zhou, B. B., Si, W., Jiang, X. & Zomaya, A. Y. (2014), 'Profiling-based workload consolidation and migration in virtualized data centers', IEEE Transactions on Parallel and Distributed Systems 26(3), 878–890.
- Han, J., Hong, Y. & Kim, J. (2020), 'Refining microservices placement employing workload profiling over multiple kubernetes clusters', IEEE access 8, 192543–192556.
- Banerjee, P., Roy, S., Modibbo, U. M., Pandey, S. K., Chaudhary, P., Sinha, A. & Singh, N. K. (2023), 'Optidjs+: A next-generation enhanced dynamic johnson sequencing algorithm for efficient resource scheduling in distributed overloading within cloud computing environment', Electronics 12(19), 4123.
- Calheiros, R. N., Masoumi, E., Ranjan, R. & Buyya, R. (2014), 'Workload prediction using arima model and its impact on cloud applications' qos', IEEE transactions on cloud computing 3(4), 449–458.

References

- Lu, H., Xu, C., Cheng, C., Kompella, R. & Xu, D. (2015), vhaul: Towards optimal scheduling of live multi-vm migration for multi-tier applications, in '2015 IEEE 8th International Conference on Cloud Computing', IEEE, pp. 453–460.
- Khelghatdoust, M., Gramoli, V. & Sun, D. (2016), Glap: Distributed dynamic workload consolidation through gossip-based learning, in '2016 IEEE International Conference on Cluster Computing (CLUSTER)', IEEE, pp. 80–89.



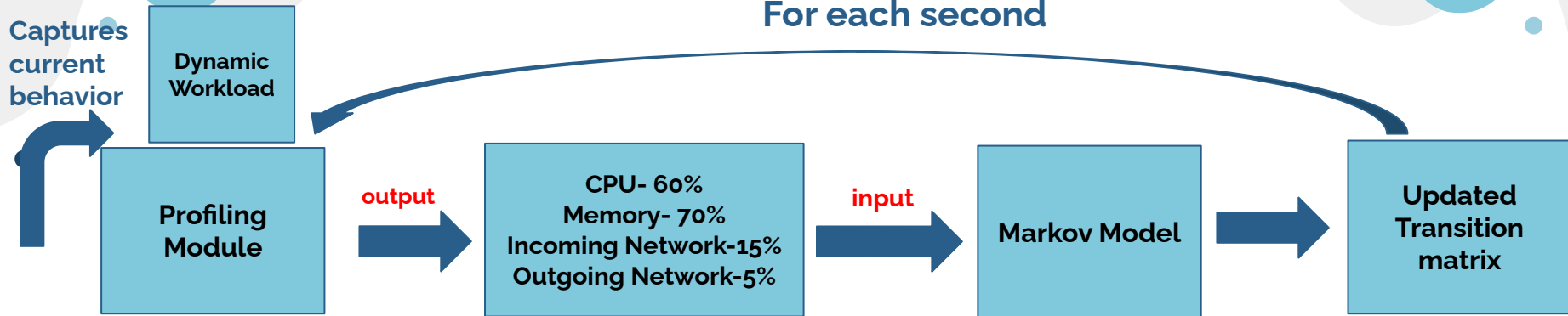
Thank you!



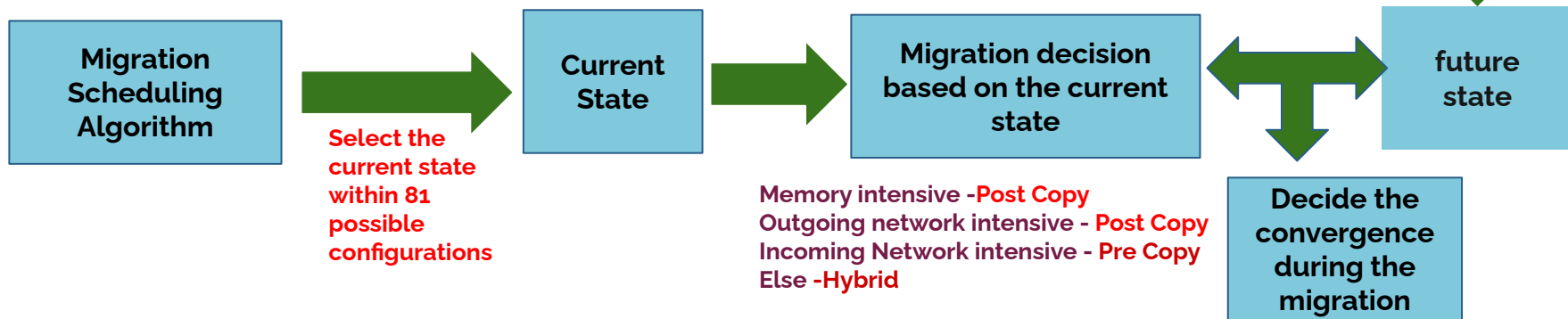
Q & A

Summary

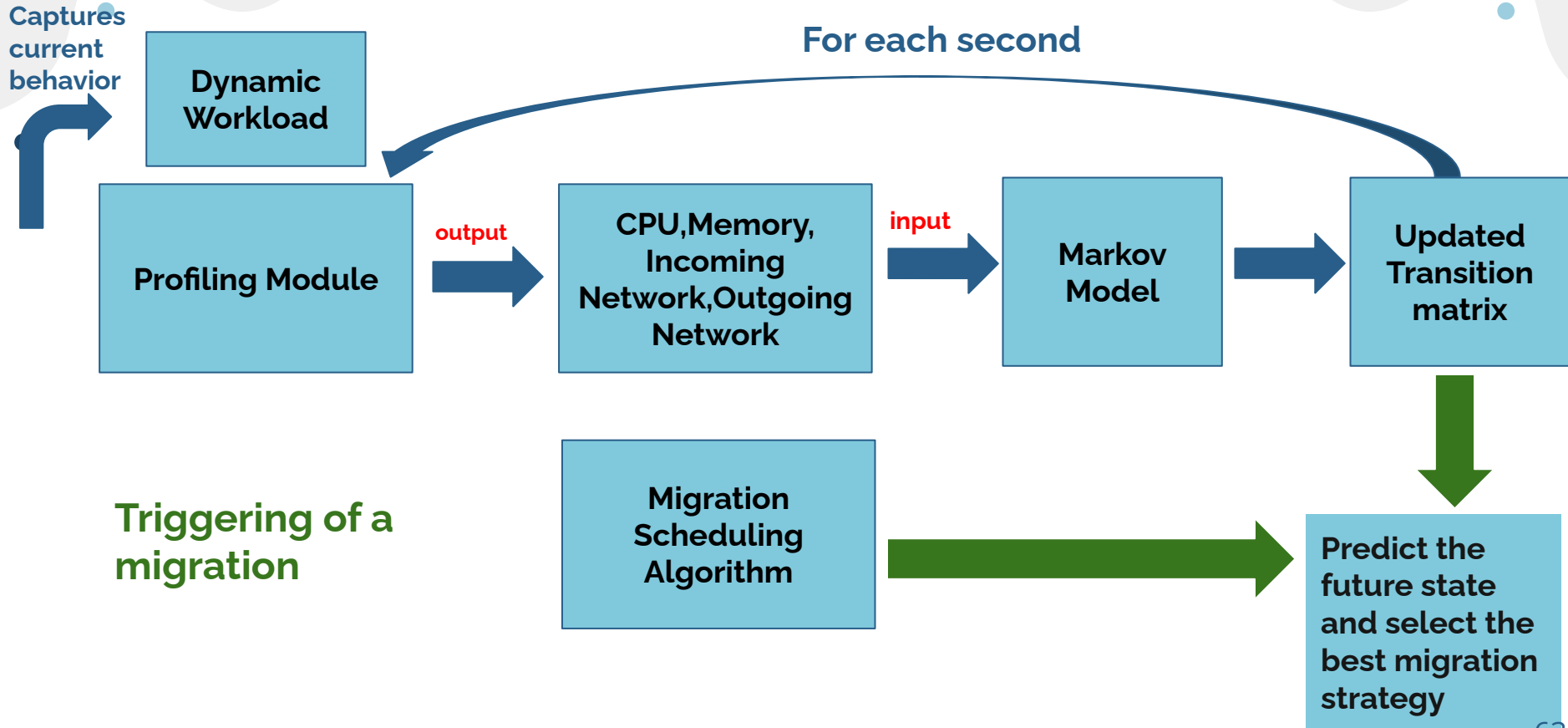
For each second



Triggering of a migration



Summary



Persistent Workloads

```
java -jar benchbase.jar -b ycsb -c config/postgres/readHeavy.xml --create=true --load=true  
--execute=true -s 1
```

A	B	C	D	E	F	G
Workload Type	Read	Insert	Scan	Update	Delete	Read/Modify/Write
Read Heavy	90	2	2	2	2	2
Read/modify/write	20	10	10	10	10	40
Insert Heavy	10	70	5	5	5	5
Scan Heavy	10	10	60	10	5	5
Delete Heavy	10	10	5	10	60	5
Update Heavy	10	10	5	60	10	5

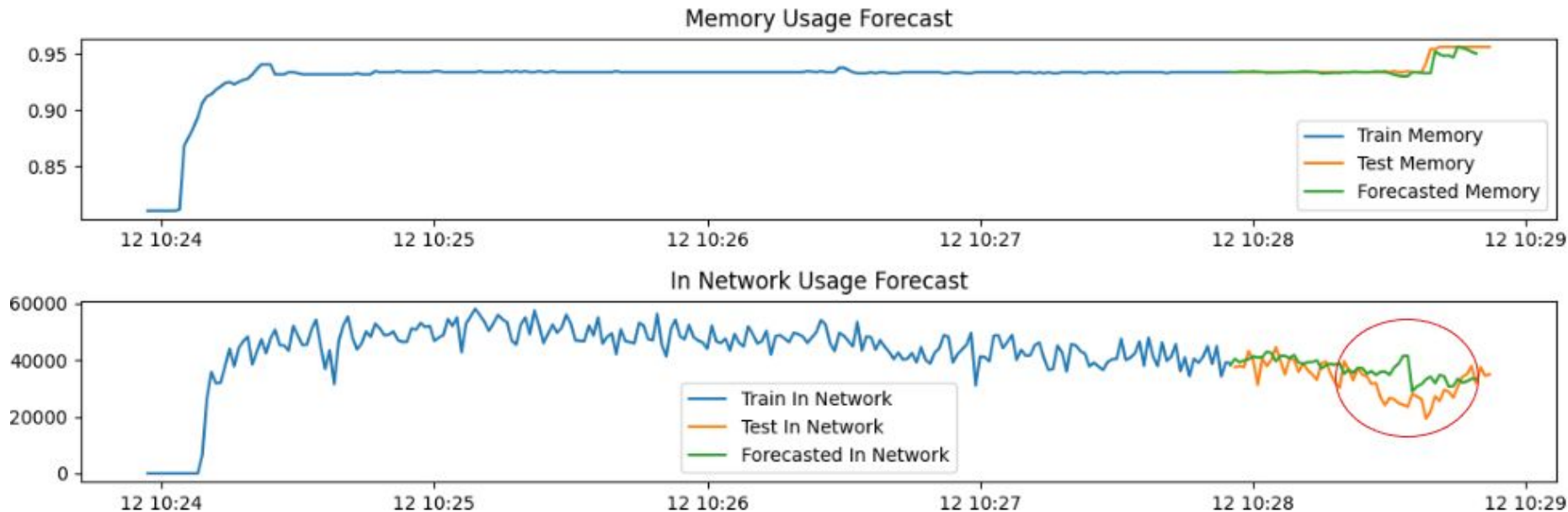
Scan Heavy Workloads

CPU- ARIMA
Memory- ETS

Update Heavy Workloads

CPU- VAR
Memory- ARIMA

Selection between Time Series Analysis and Markov Chains



Real Time Workloads

- **Stress Tool is used.**
- **Installation**
 - `sudo apt install stress`
- **Generate a dynamic workload**
 - `stress --cpu 2 --vm 2 --vm-bytes 512M --io 1 --timeout 60`

Progress So Far

S06

```
0.00 0.00
18:21:20 all 72.00 0.00 28.00 0.00 0.00 0.00 0.00
0.00 0.00
18:21:21 all 66.00 0.00 34.00 0.00 0.00 0.00 0.00
0.00 0.00
18:21:22 all 72.00 0.00 28.00 0.00 0.00 0.00 0.00
0.00 0.00
18:21:23 all 67.68 0.00 32.32 0.00 0.00 0.00 0.00
0.00 0.00
18:21:24 all 68.32 0.00 31.68 0.00 0.00 0.00 0.00
0.00 0.00
Average: all 69.72 0.00 30.25 0.00 0.00 0.02 0.02 0.00
0.00 0.00
root@workingset:~#
```

S06 (t)

Broadcast X

```
-/+ buffers/cache: 840M 1.1G
Swap: 8.0G 0B 8.0G

total used free shared buffers cached
Mem: 2.0G 975M 1.0G 14M 51M 325M
-/+ buffers/cache: 599M 1.4G
Swap: 8.0G 0B 8.0G

total used free shared buffers cached
Mem: 2.0G 1.2G 758M 14M 51M 325M
-/+ buffers/cache: 865M 1.1G
Swap: 8.0G 0B 8.0G
```

S06

```
-----
Current State: 1 0.628514 0 0
CPU Level:2,Memory Level:2,Incoming Network Level:0,Outgoing Network Level:0
Current State: 72
Previous State: 63
Transition Matrix updated: 63 -> 72
```

```
Current State: 1 0.509371 0 0
CPU Level:2,Memory Level:2,Incoming Network Level:0,Outgoing Network Level:0
Current State: 72
Previous State: 72
Transition Matrix updated: 72 -> 72
```

```
-----
Current State: 1 0.438587 0 0
CPU Level:2,Memory Level:1,Incoming Network Level:0,Outgoing Network Level:0
```

Activate Windows
Go to Settings to activate Windows.

Profiling Tools

```
nadeesha@Nadeesha:~$ mpstat 1 100
Linux 6.8.0-49-generic (Nadeesha)
```

11/21/2024

_x86_64_

(1 CPU)

09:09:15 PM	CPU	%usr	%nice	%sys	%iowait	%irq	%soft	%steal	%guest	%gnice	%idle
09:09:16 PM	all	4.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	95.79
09:09:17 PM	all	7.37	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	92.63
09:09:18 PM	all	5.15	0.00	1.03	0.00	0.00	1.03	0.00	0.00	0.00	92.78
09:09:19 PM	all	8.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	91.67
09:09:20 PM	all	8.51	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	91.49
09:09:21 PM	all	5.15	0.00	1.03	0.00	0.00	0.00	0.00	0.00	0.00	93.81
09:09:22 PM	all	4.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	95.79
09:09:23 PM	all	10.53	0.00	2.11	0.00	0.00	0.00	0.00	0.00	0.00	87.37

```
nadeesha@Nadeesha:~$ ifstat
```

enp0s3

KB/s in KB/s out

0.00 0.00

0.00 0.00

0.00 0.00

0.00 0.00

0.00 0.00

0.00 0.00

0.00 0.00

```
nadeesha@Nadeesha:~$ free -h -s 1
```

	total	used	free	shared	buff/cache	available
Mem:	3.7Gi	767Mi	1.6Gi	33Mi	1.3Gi	2.7Gi
Swap:	2.6Gi	0B	2.6Gi			

	total	used	free	shared	buff/cache	available
Mem:	3.7Gi	767Mi	1.6Gi	33Mi	1.3Gi	2.7Gi
Swap:	2.6Gi	0B	2.6Gi			

	total	used	free	shared	buff/cache	available
Mem:	3.7Gi	767Mi	1.6Gi	33Mi	1.3Gi	2.7Gi
Swap:	2.6Gi	0B	2.6Gi			

	total	used	free	shared	buff/cache	available
Mem:	3.7Gi	767Mi	1.6Gi	33Mi	1.3Gi	2.7Gi
Swap:	2.6Gi	0B	2.6Gi			

	total	used	free	shared	buff/cache	available
Mem:	3.7Gi	767Mi	1.6Gi	33Mi	1.3Gi	2.7Gi
Swap:	2.6Gi	0B	2.6Gi			

	total	used	free	shared	buff/cache	available
Mem:	3.7Gi	767Mi	1.6Gi	33Mi	1.3Gi	2.7Gi
Swap:	2.6Gi	0B	2.6Gi			