Memory Flipping: A threat to NUMA virtual machines in the Cloud

Djob Mvondo, LIG, University of Grenoble Alpes
Boris Teabe, IRIT, University of Toulouse
Alain Tchana, I3S, University of Nice
Daniel Hagimont, IRIT, University of Toulouse,
Noel De Palma, LIG, University of Grenoble Alpes

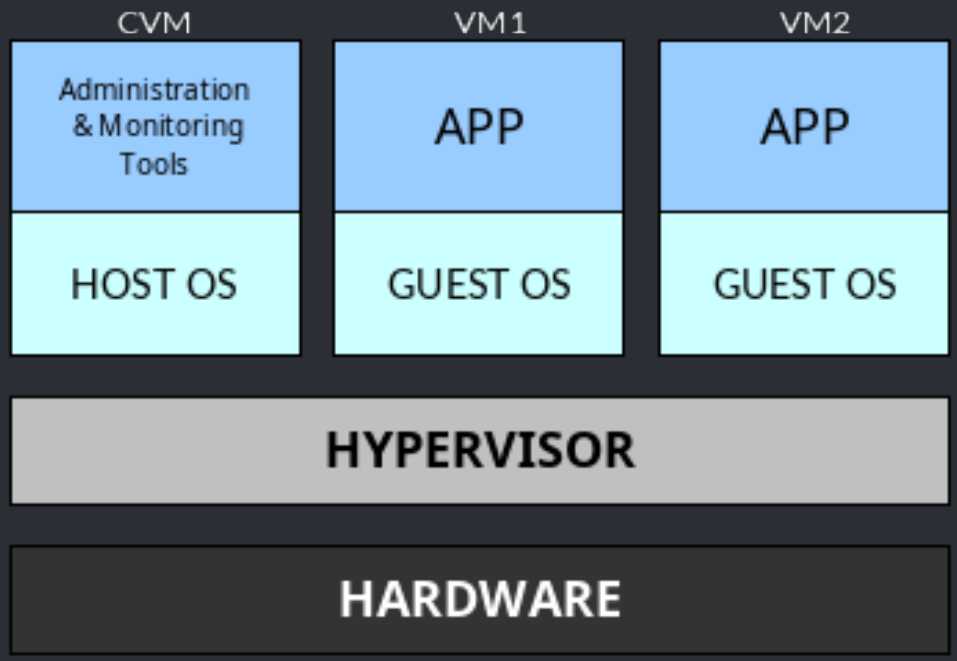
Context

System Virtualization

System virtualization enables several operating systems (Oses) to run on a physical server. These Oses run in black boxes referred to as virtual machines (VMs).

The hypervisor is in charge of :

- VM administration
- Block devices
- Network devices
- Scheduling
- Memory management

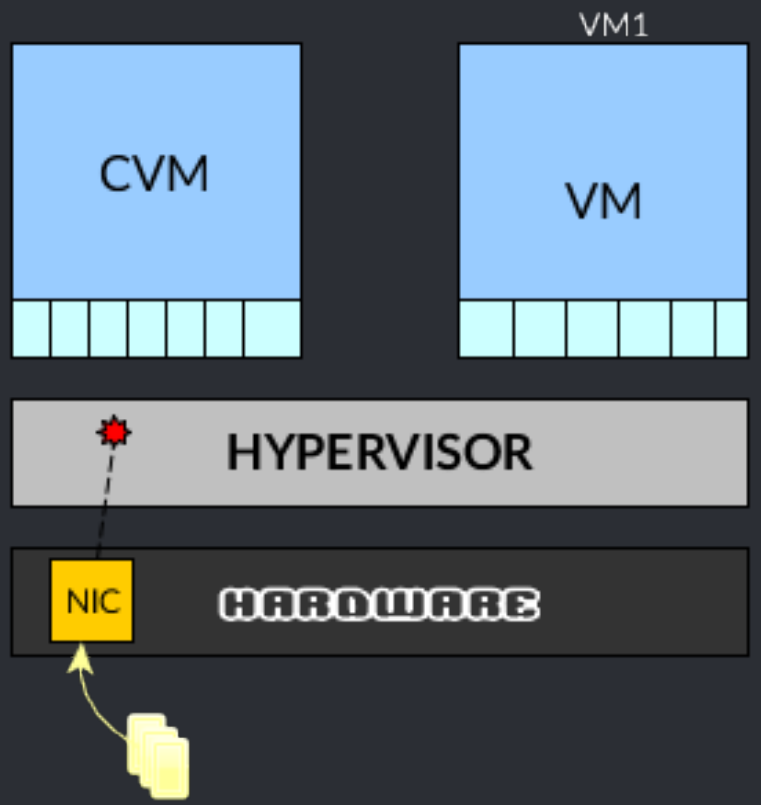


Context

System Virtualization - Network

The hypervisor handles incoming and outgoing network packets to/from VMs. In general, when a packet arrives on a NIC :

- a hardware interrupt is raised and
- caught by the hypervisor.

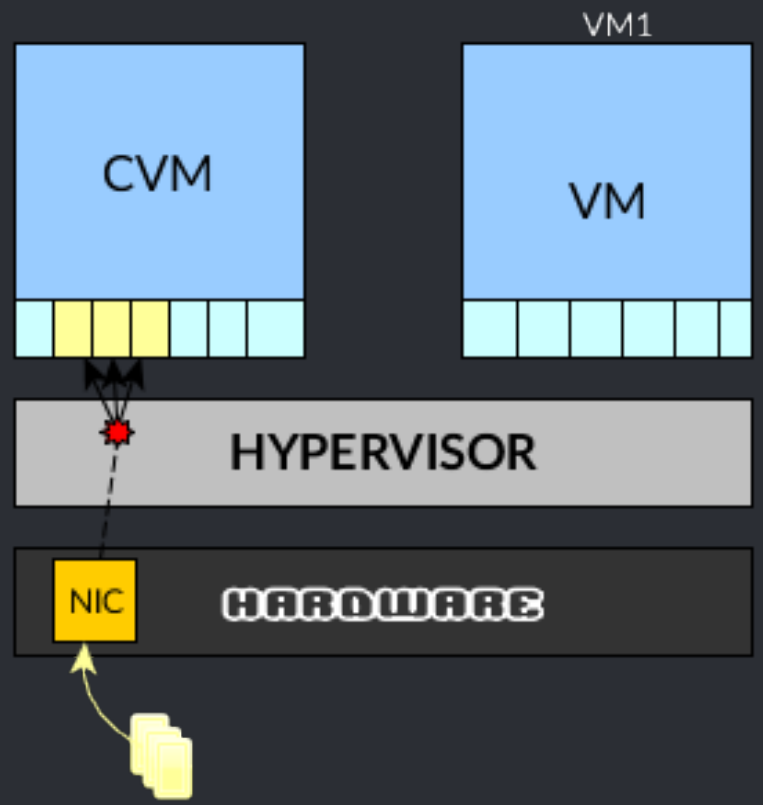


Context

System Virtualization - Network

The hypervisor handles incoming and outgoing network packets to/from VMs. In general, when a packet arrives on a NIC :

- a hardware interrupt is raised and
- caught by the hypervisor.
- The packet is then reconstructed in the hypervisor memory (or CVM[1] memory)

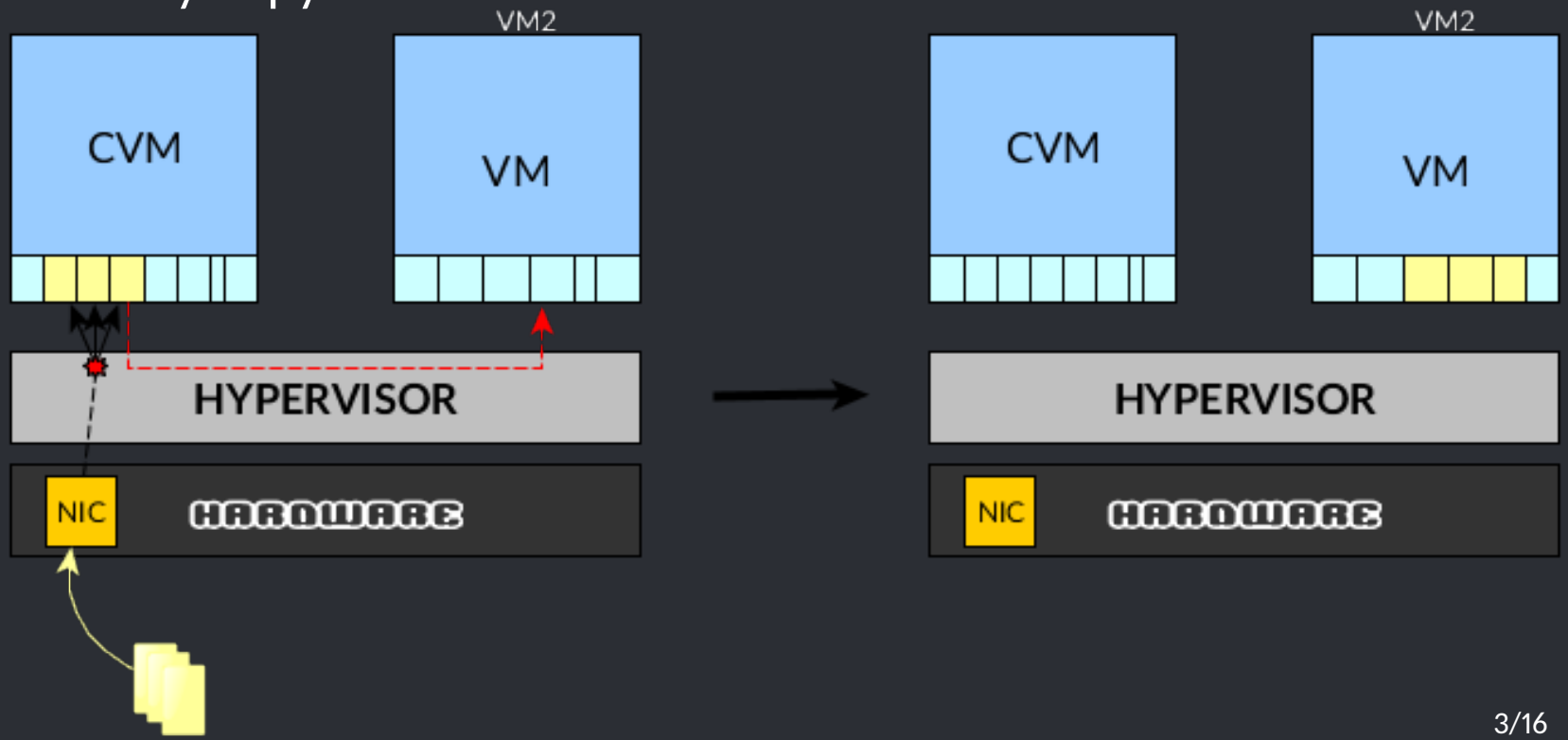


Context

System Virtualization - Network

Now, the hypervisor must enable the concerned VM to access the packet in his memory space.

- Memory copy



Context

System Virtualization - Network

Now, the hypervisor must enable the concerned VM to access the packet in his memory space.

- ~~Memory copy~~(Too costy)
- **Memory flipping**

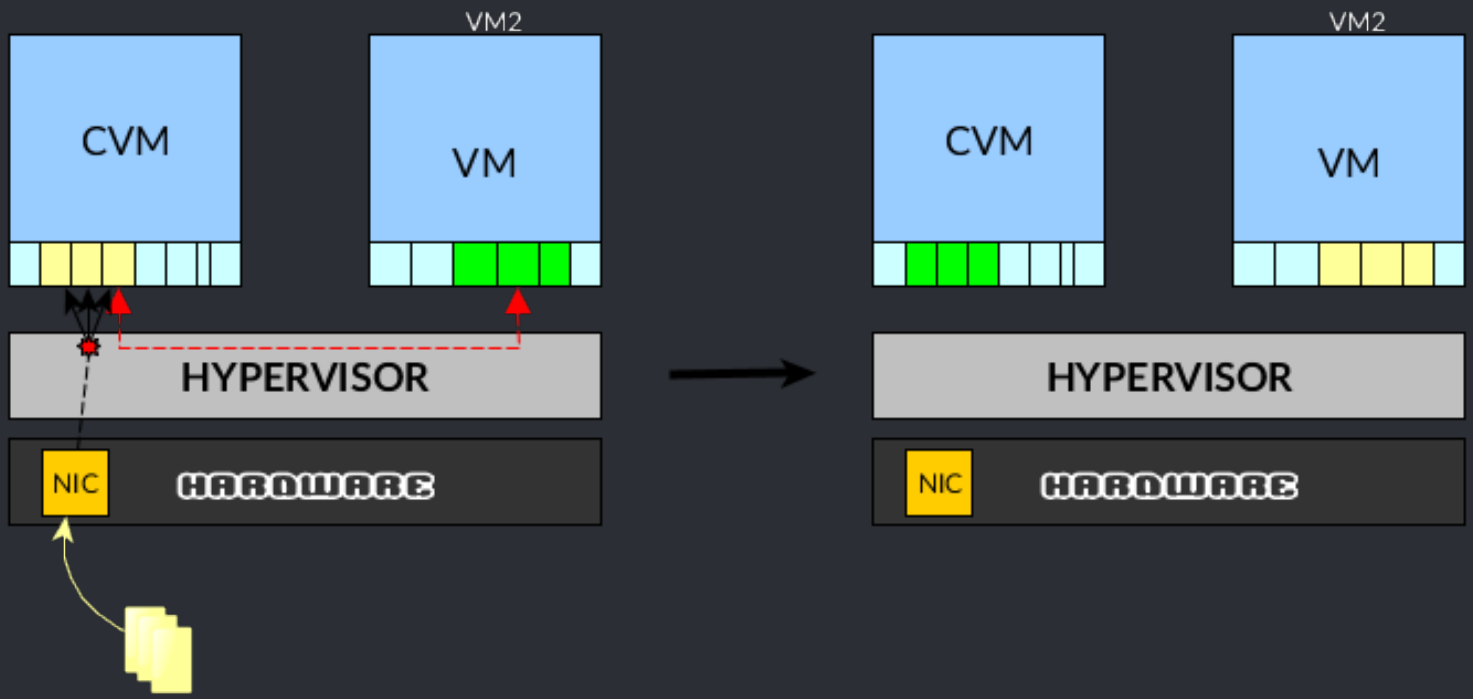
Definition

Memory flipping is the process where the hypervisor gives ownership grants/rights on the pages (storing the packet data) to the concerned VM. To counterbalance, the VM offers free pages for the CVM.

Context

System Virtualization - Network - Memory Flipping

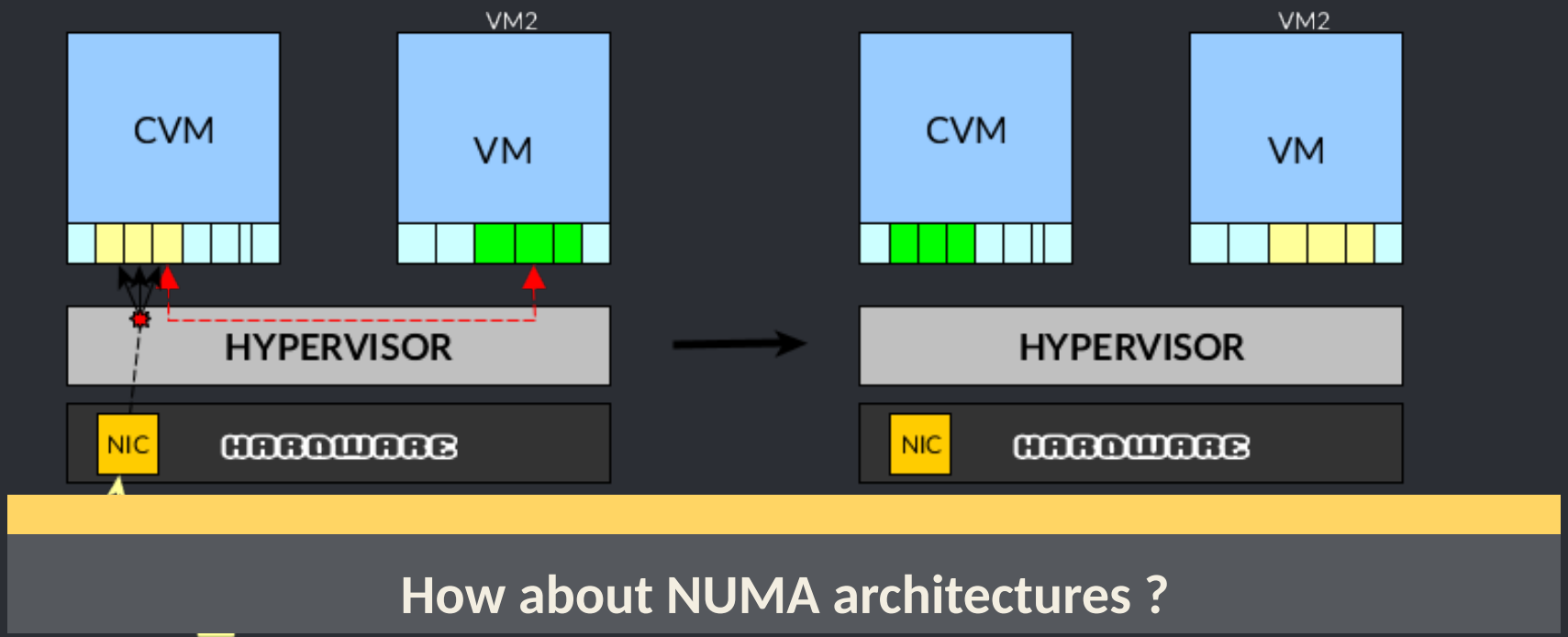
- Better throughput than memory copy
- Works well on uniform memory architectures



Context

System Virtualization - Network - Memory Flipping

- Better throughput than memory copy
- Works well on uniform memory architectures

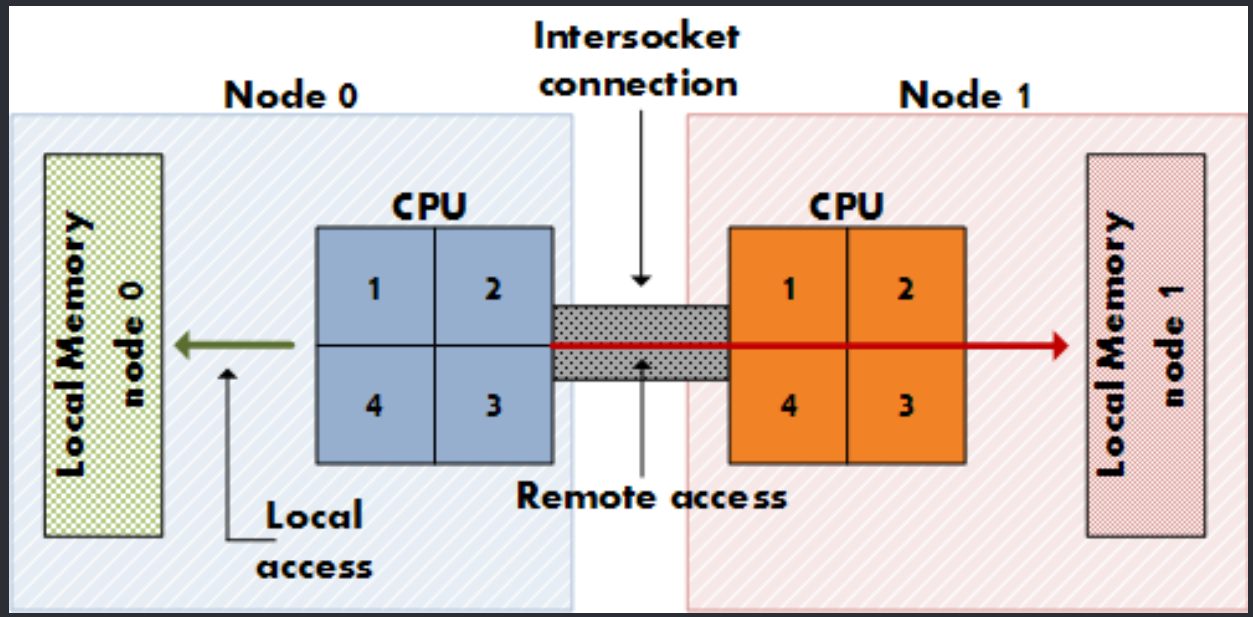


How about NUMA architectures ?

Context

System Virtualization - NUMA (Recall)

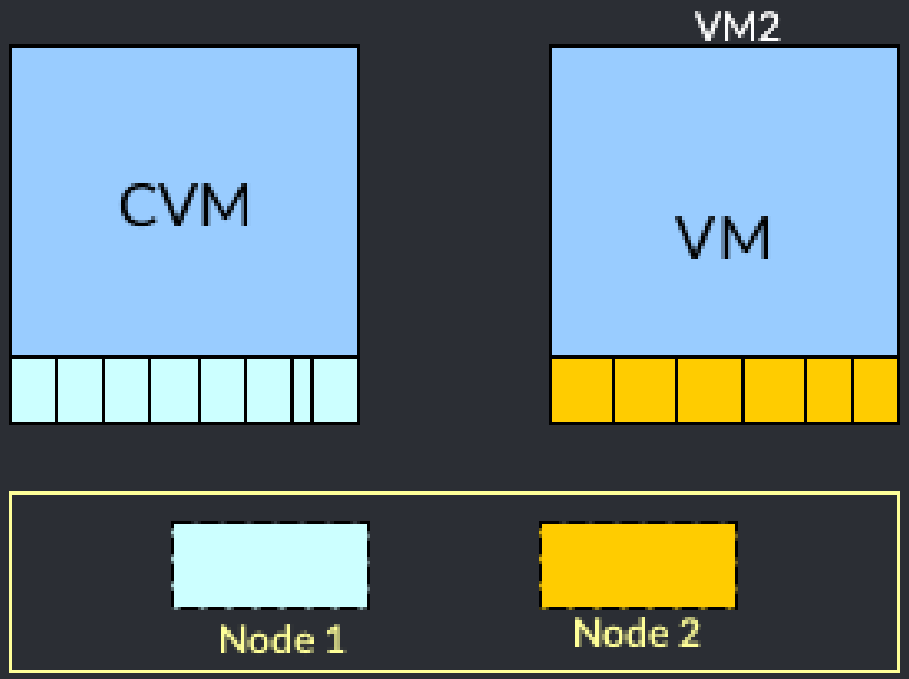
- Remote memory access **is costlier** compared a local one.
- Modern Oses updated their memory allocation and scheduling policies to take into account NUMA.



Problematic

System Virtualization - Network - NUMA

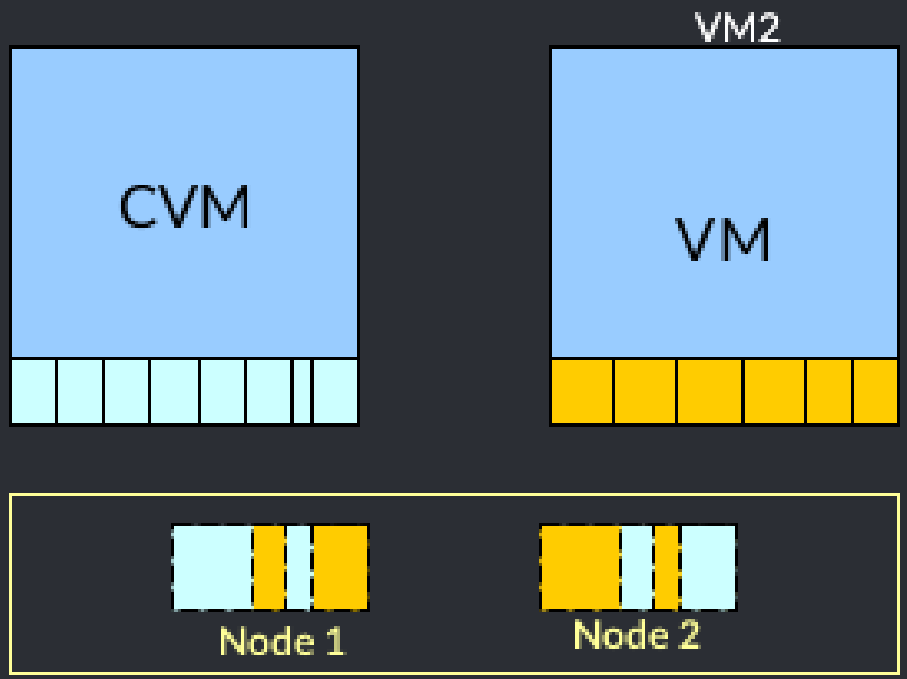
In a virtualized NUMA environment, the trend is to allocate a whole node for the CVM. Hence, *the CVM's memory is usually on a different NUMA node than those of VMs*.



Problematic

System Virtualization - Flipping - NUMA

With this layout, repeated memory flipping operations leads to the VM's ***transparent memory migration*** from one node to another.

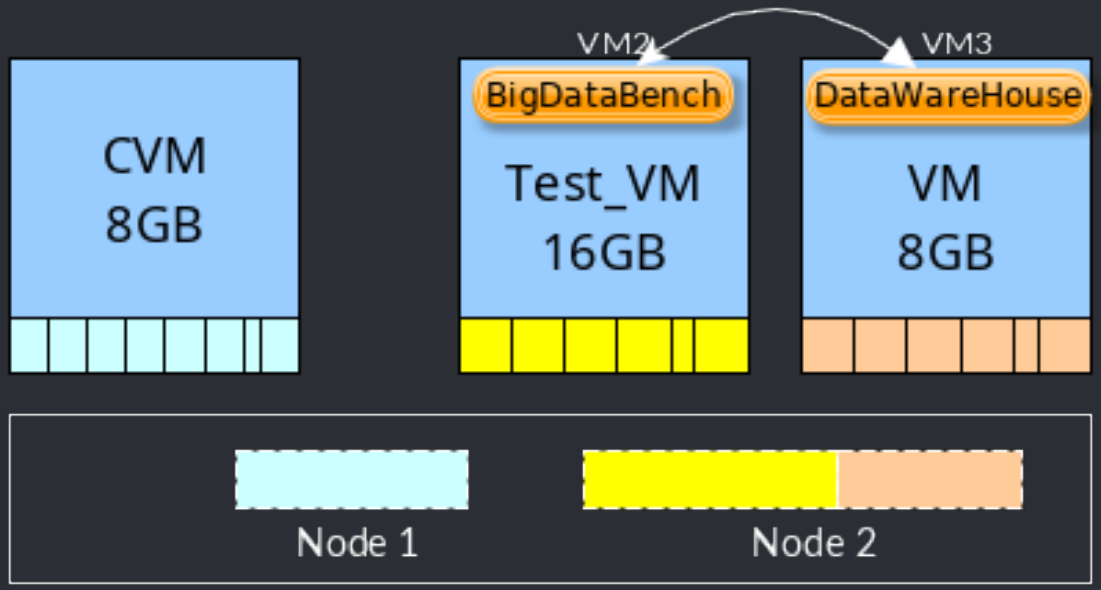


Problematic

System Virtualization - Flipping - NUMA

To confirm this hypothesis, we ran an E-Commerce benchmark from the **BigDataBench suite (Eight TPC-DS Web Queries)**.

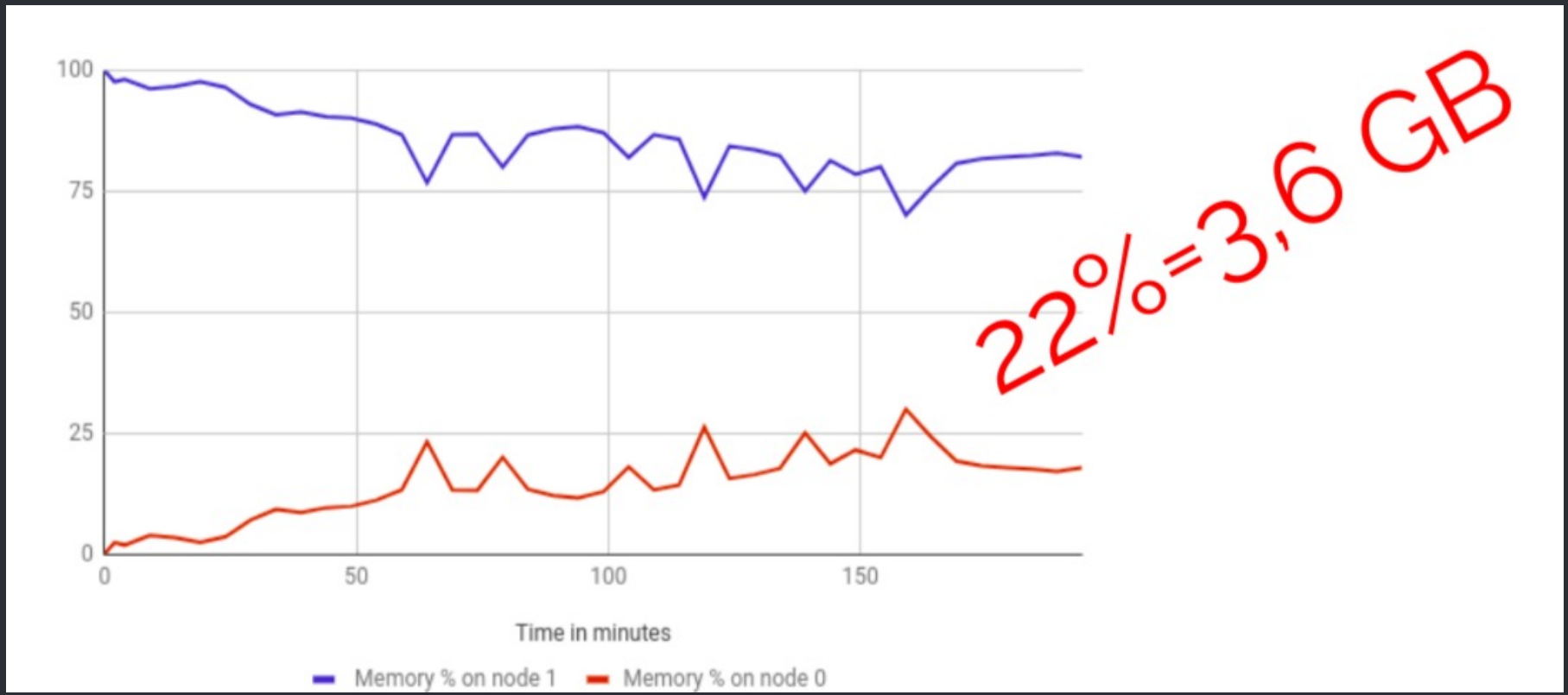
We monitor VM2's mem % on each node during the experiment



Problematic

System Virtualization - Flipping - NUMA

At the end of the experiment, up to **22% ≈ 3, 6GB** of the VM memory becomes **remote**.



Consequences

Effects of Flipping on NUMA

Is this *really* a problem ?

Imagine a sprinter preparing for a 200m race.

On the race day, suddenly it is no more **200m but a 2km marathon.**

It is clear that he/she will not perform as well as if it was a 200m race.

Consequences

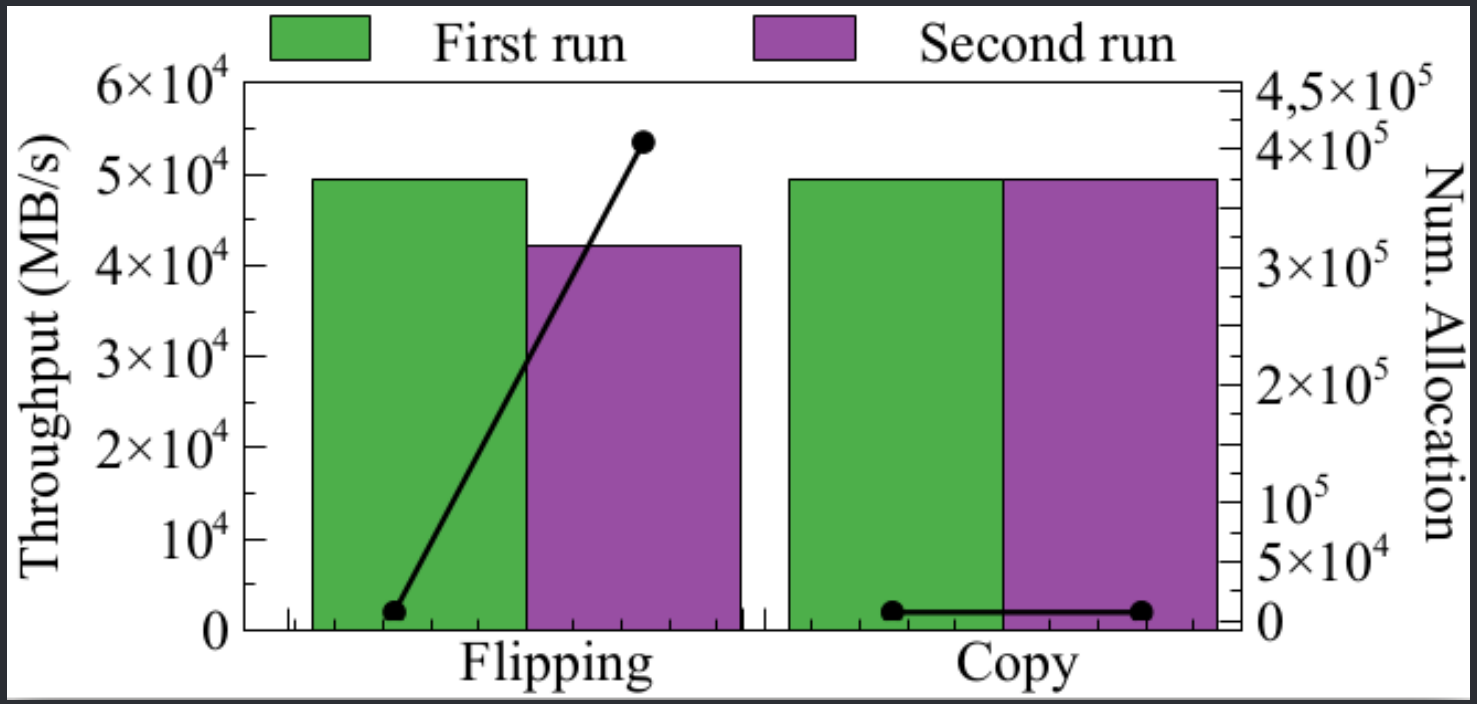
Effects of Flipping on NUMA

Analogy to Virtualization

Consequences

Effects of Flipping on NUMA

Analogy to Virtualization : STREAM before & after flipping



Our approaches

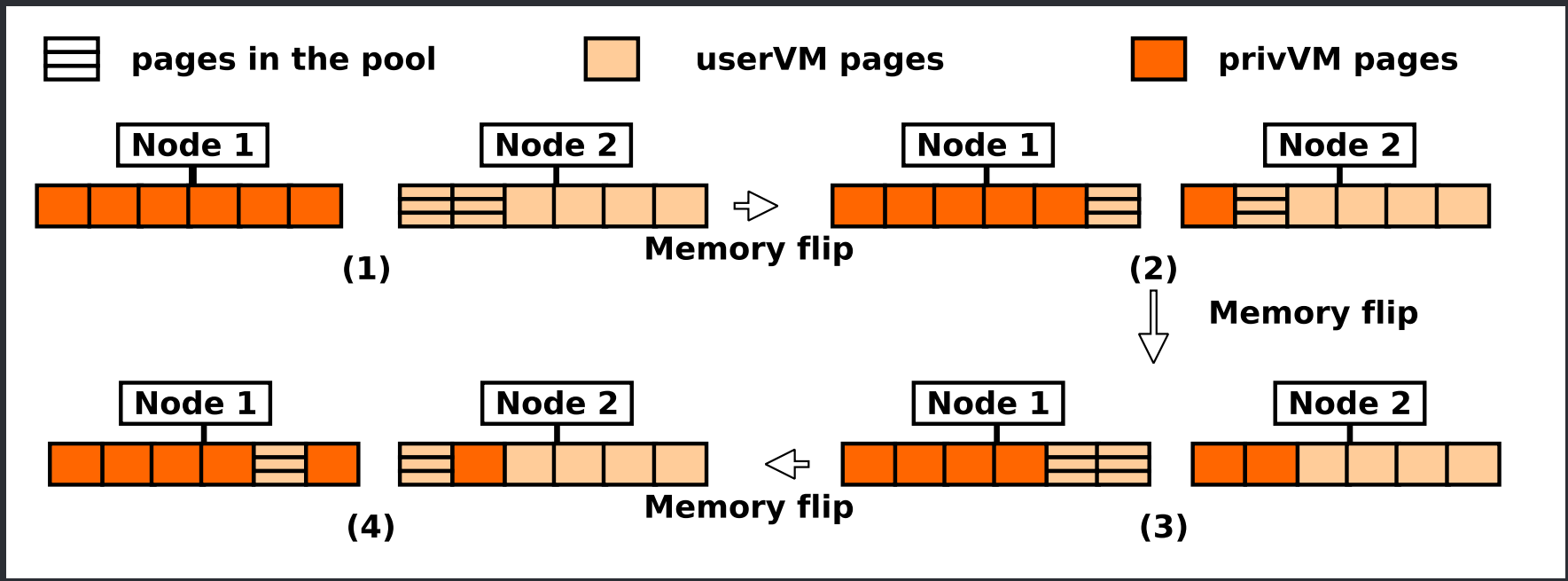
Dedicated page pool

- Maintain a pool of pages acquired from previous flipping.
- For each flipping operation, pick pages in this pool (if no available pages, then do *malloc*)
- These pages can’t be used by other kernel processes.

Our approaches

Dedicated page pool

- Maintain a pool of pages acquired from previous flipping.
- For each flipping operation, pick pages in this pool (if no available pages, then do *malloc*)
- These pages can’t be used by other kernel processes.



Our approaches

Dedicated page pool - Pool size

What about the pool size ?

Our approaches

Dedicated page pool - Pool size

What about the pool size ?

- Define a static pool size by calibration.

Our approaches

Dedicated page pool - Pool size

What about the pool size ?

- Dynamic pool size

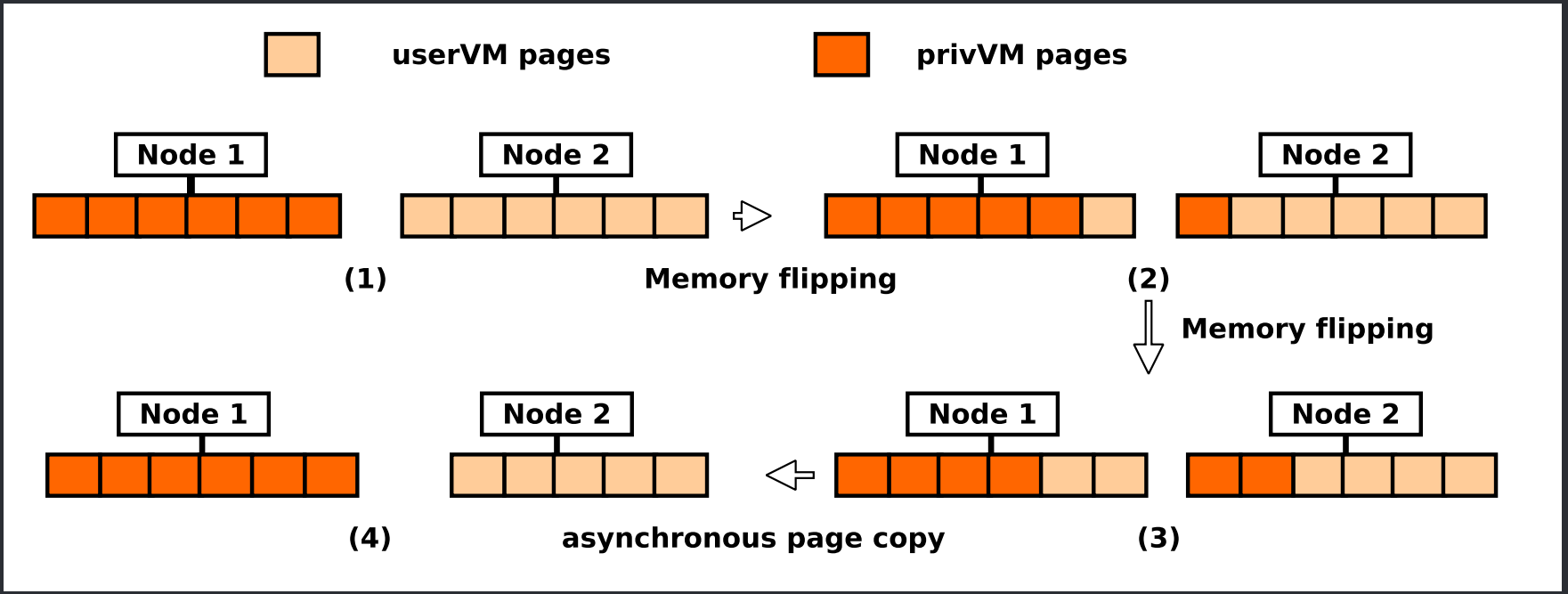
Algorithm 2 Dynamic poolSize estimation algorithm

```
1: compute used_poolSize
2: if used_poolSize > 90% then
3:     add 10% of poolSize
4: else if used_poolSize < 20% then
5:     remove 10% of poolSize
6: end if
7: set new poolSize value
```

Our approches

Asynchronous memory migration

Asynchronously migrate all pages which became remote due to memory flipping.



Our approches

Asynchronous memory migration

But, *when should we migrate ?*

Our approches

Asynchronous memory migration

But, *when should we migrate ?*

- **Periodically**
  We define a period and for each corresponding epoch, migrate all pages which became remote.

Our approches

Asynchronous memory migration

But, *when should we migrate ?*

- **Periodically**
  We define a period and for each corresponding epoch, migrate all pages which became remote.
- **Based on a threshold value**
  We define a threshold value which corresponds to the maximum remote memory authorized. Once this threshold is reached, the asynchronous migration is triggered.

Evaluations

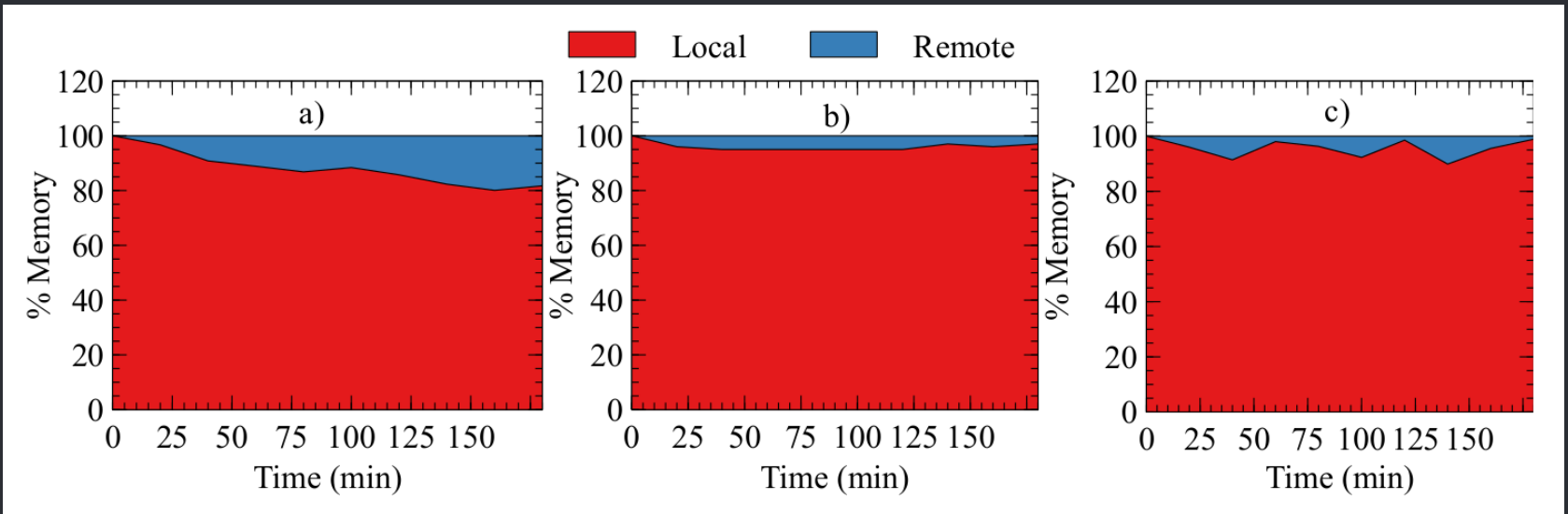
Does our solutions mitigate the issue ?

We implemented our solutions in the **Xen hypervisor 4.9.0** and introduced modest changes in the **Linux Kernel 4.14.2**.

Evaluations

Does our solutions mitigate the issue ?

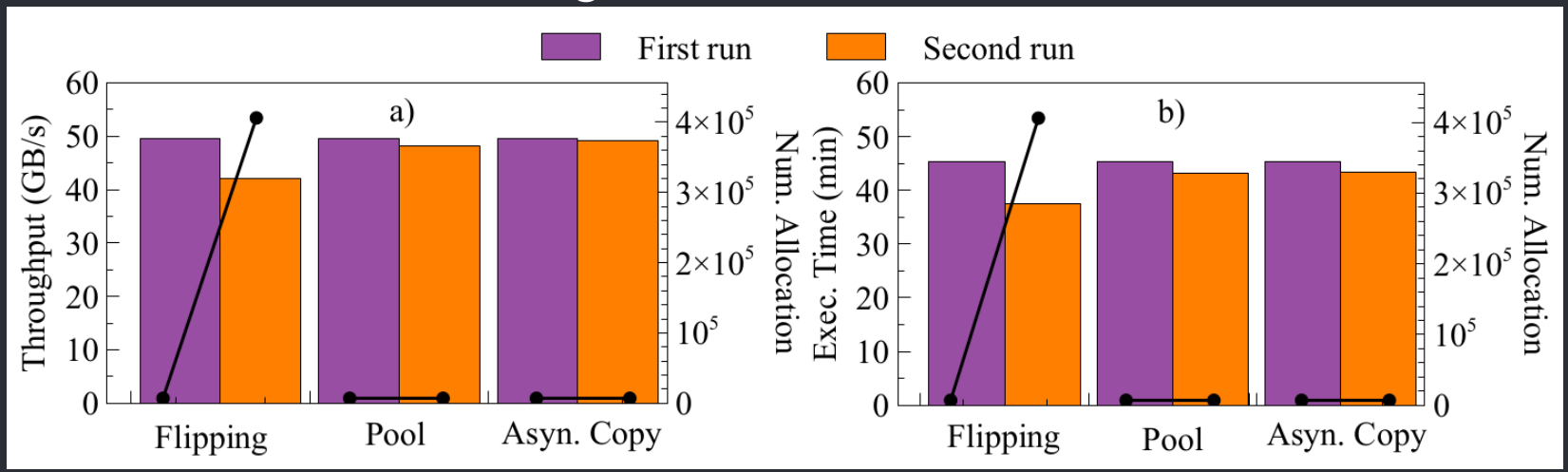
We implemented our solutions in the **Xen hypervisor 4.9.0** and introduced modest changes in the **Linux Kernel 4.14.2**.



Evaluations

Does our solutions mitigate the issue ?

We implemented our solutions in the **Xen hypervisor 4.9.0** and introduced modest changes in the **Linux Kernel 4.14.2**.



Conclusion

Thank you for your attention!!!
Feel free to ask any question.