Virtflex: Automatic adaptation to NUMA topology change for OpenMP applications

Runhua Zhang, Alan L. Cox, Scott Rixner

Rice University

IWOMP 2020

RICE UNIVERSITY

NUMA virtualization

Rack-scale computers

- Share-memory, each discrete device is a NUMA node (large number, up to ~50)
- Complex underlying NUMA topology
- Requires dynamic elastic guest NUMA topology to achieve high utilization rate

Current NUMA machine

- Small number of NUMA nodes 1~4
- Simple static guest NUMA topology
- VMs are forced to be pinned into a NUMA node or a small set of NUMA nodes

Lack of support for elastic guest NUMA topology

Hypervisor

- Lack of control for the amount of resources on each guest node
- Express dynamic guest NUMA topolgoy

Linux kernel

- Static ACPI table
  - Guest NUMA topolgoy is obtained at boot time and never changes
- AutoNUMA

  Only suitable in cases where most memory is in the right place. Slow to adapt to topology changes

User application

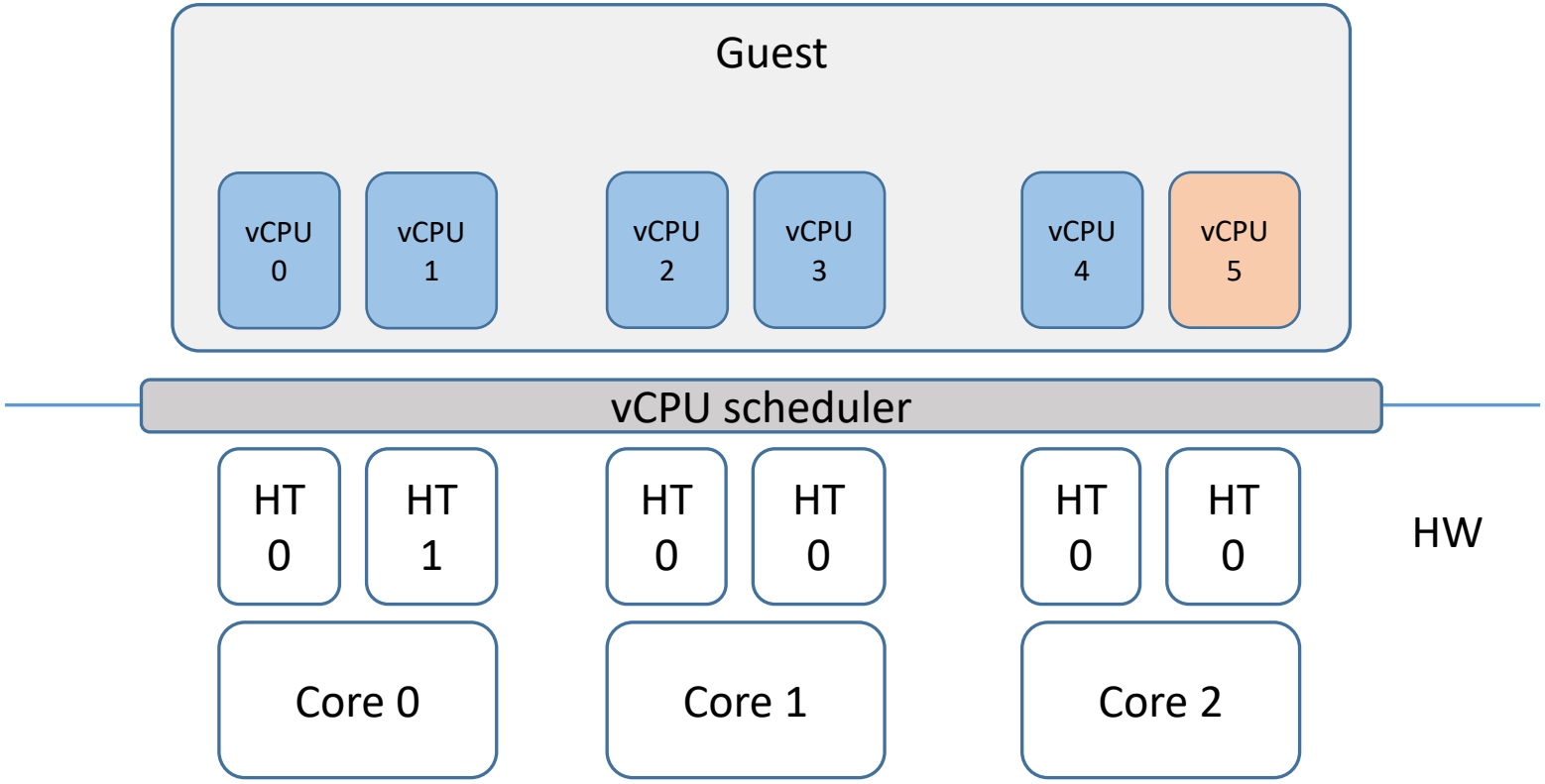
- Libnuma place memory and threads in a static way, e.g. using specific node #
- Need to rewrite source code to fit dynamic topology

Virtualization background

- vCPU and vCPU hotplug
- Two dimentional address translation
- Ballooning

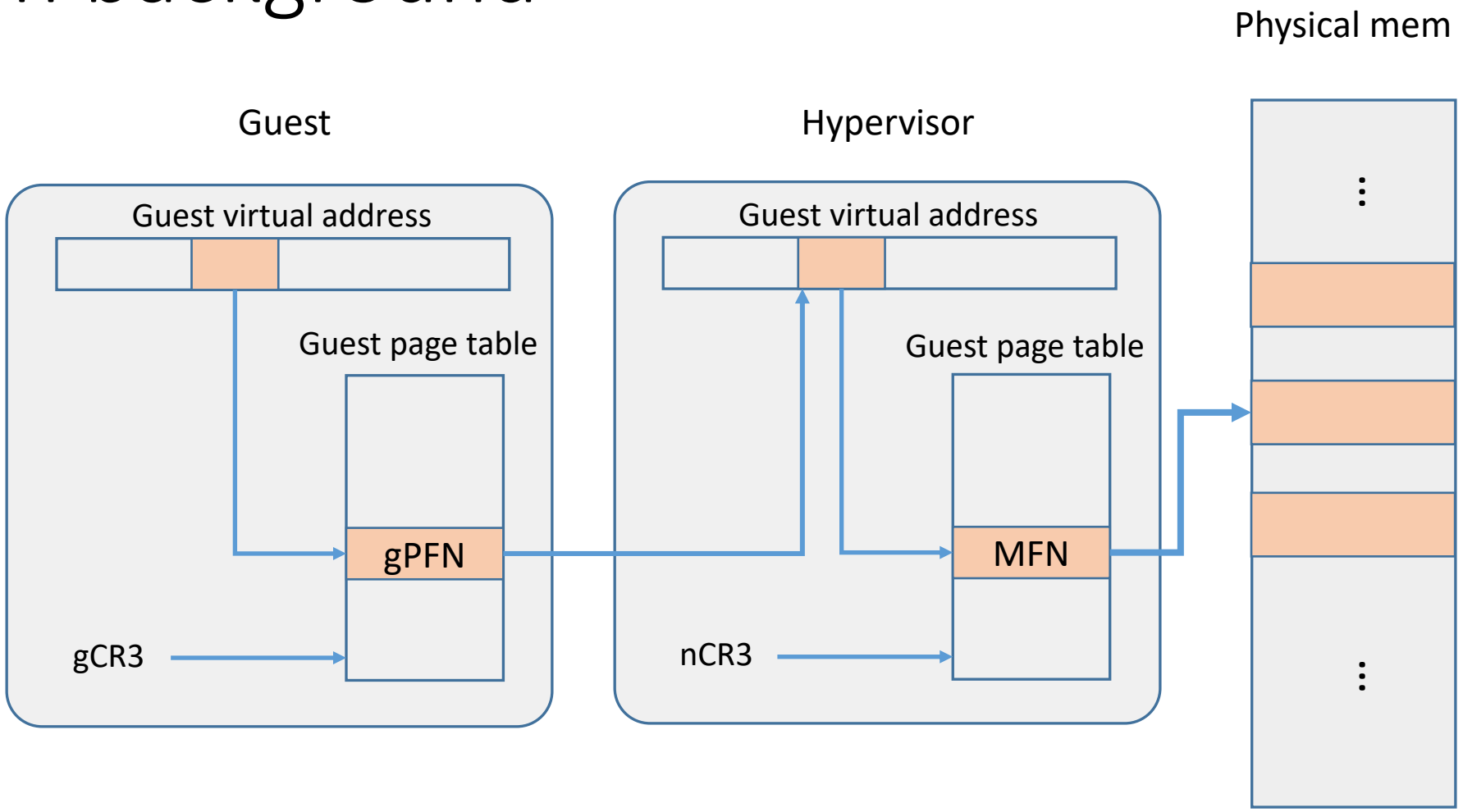
Virtualization background

- vCPU and vCPU hotplug
- Two dimentional address translation
- Ballooning



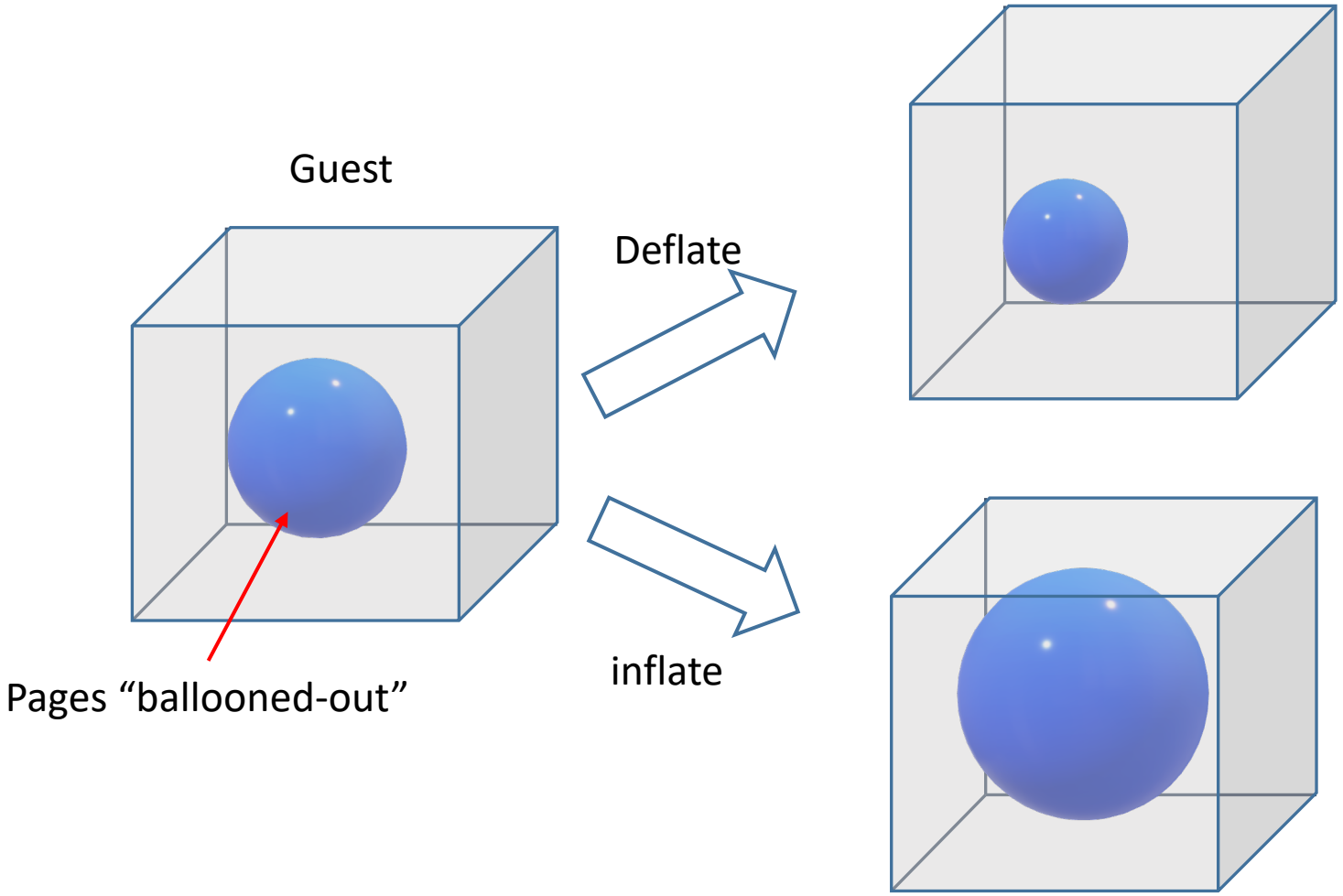
Virtualization background

- vCPU and vCPU hotplug
- Two dimentional address translation
- Ballooning



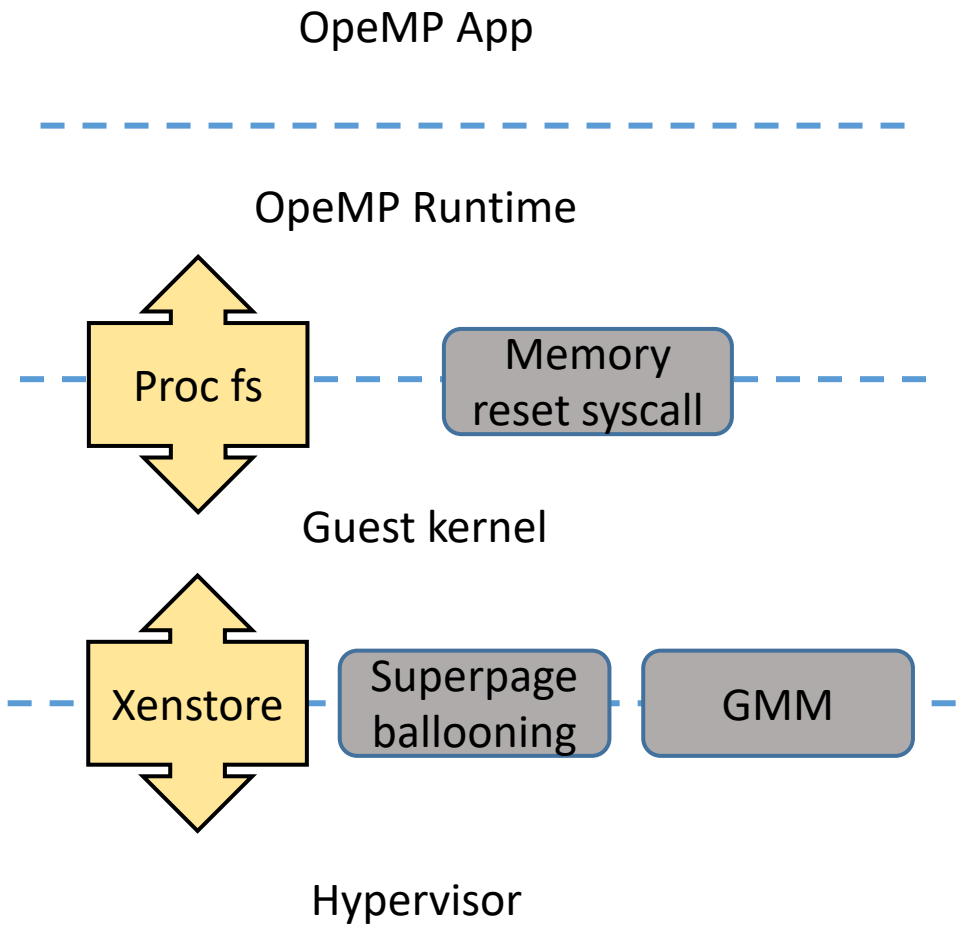
Virtualization background

- vCPU and vCPU hotplug
- Two dimentional address translation
- Ballooning



Virtflex

Virtflex

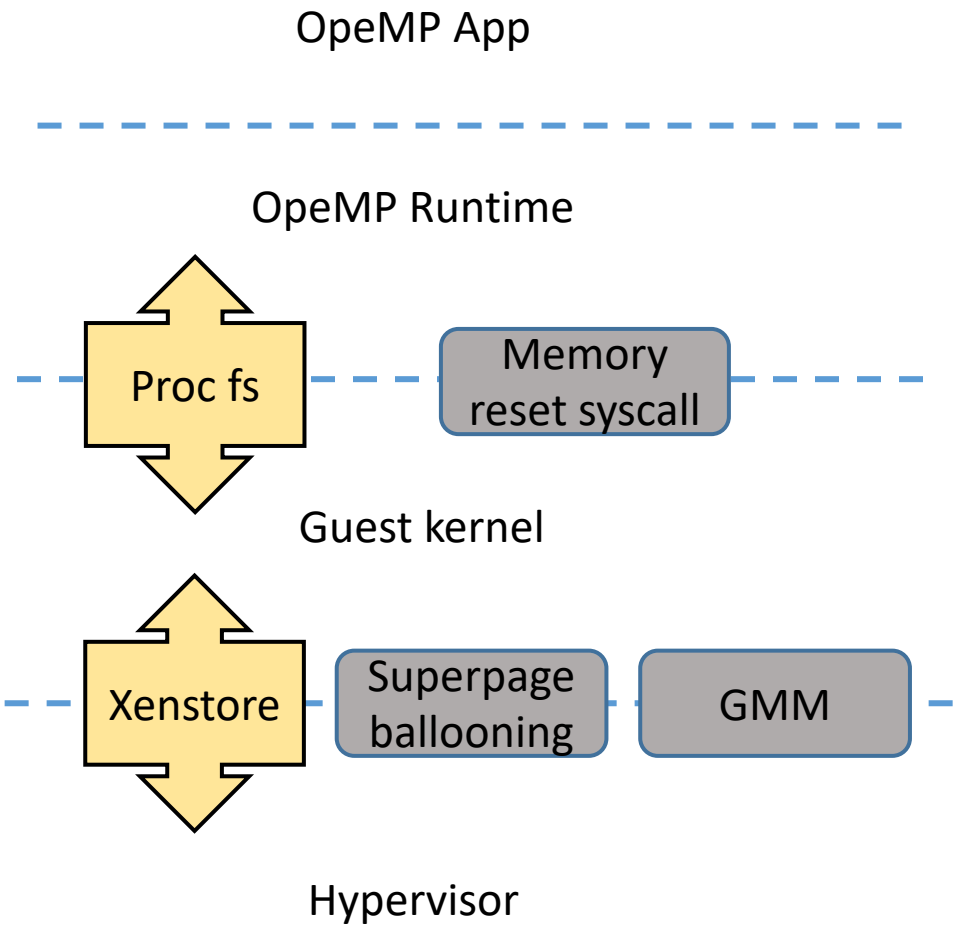


- Virtflex is a multilayered system for enabling unmodified OpenMP applications to adapt automatically to NUMA topology changes.
- Enhancement across the Xen hypervisor, Linux kernel and OpenMP runtime

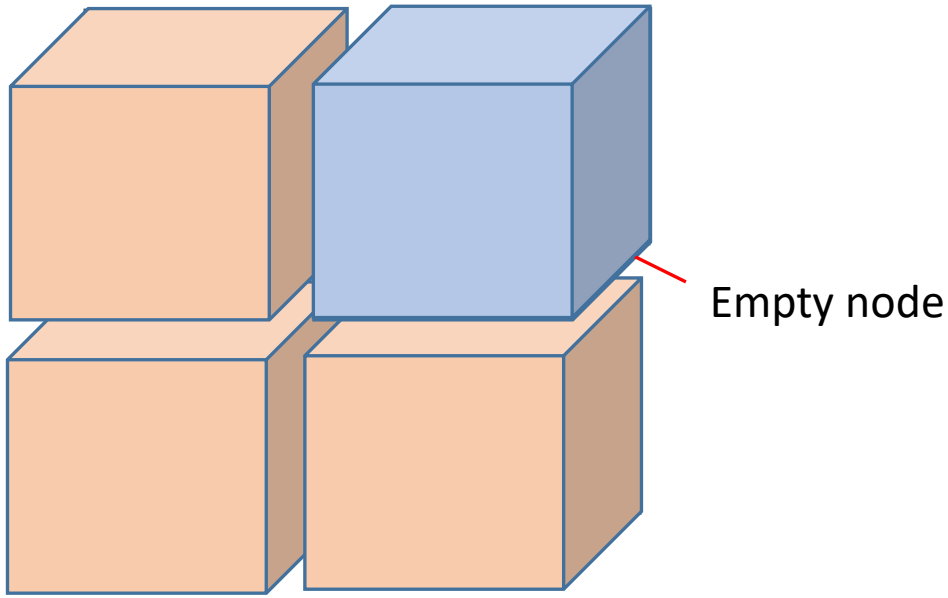
Related works

Virtualization on NUMA machines	Hiding NUMA topology from the guests	Rao [9], Rao [10], Liu [11], Wu [12] improve NUMA vCPU scheduling or memory placement at the hypervisor level.
	Exposing guest NUMA topology	Bui et al [14] abandoned the ACPI interface and propose another interface that allow guest kernel to get notification when topology changes
OpenMP optimization on NUMA machine	Olivier [1], Durand [2], Muddukrishna[3] improved the loop/task scheduler on NUMA machines Broquedis et al. [8] further developed interfaces to maintain thread-memory affinity for OpenMP applications on NUMA machines. Uses next-touch to migrate memory to the correct NUMA node, but it requires guidance from the programmer	

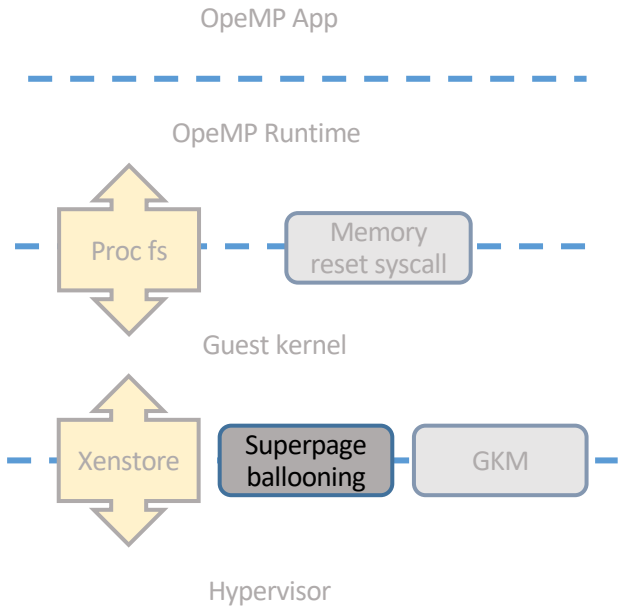
Elasticity of guest NUMA topology

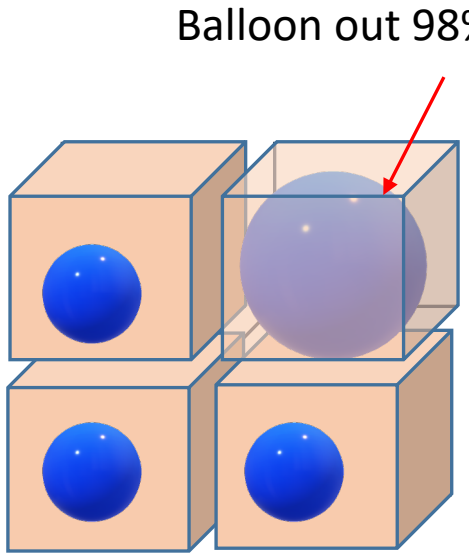


- Efficient adding /removing node



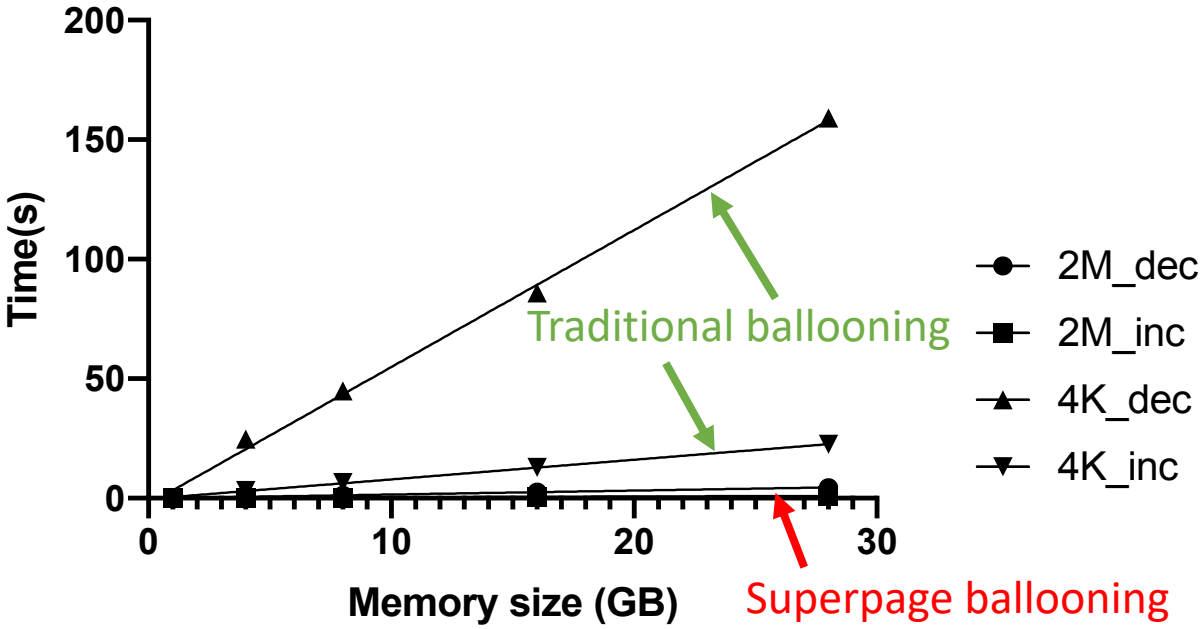
Make topology change fast (NUMA-aware Superpage ballooning)





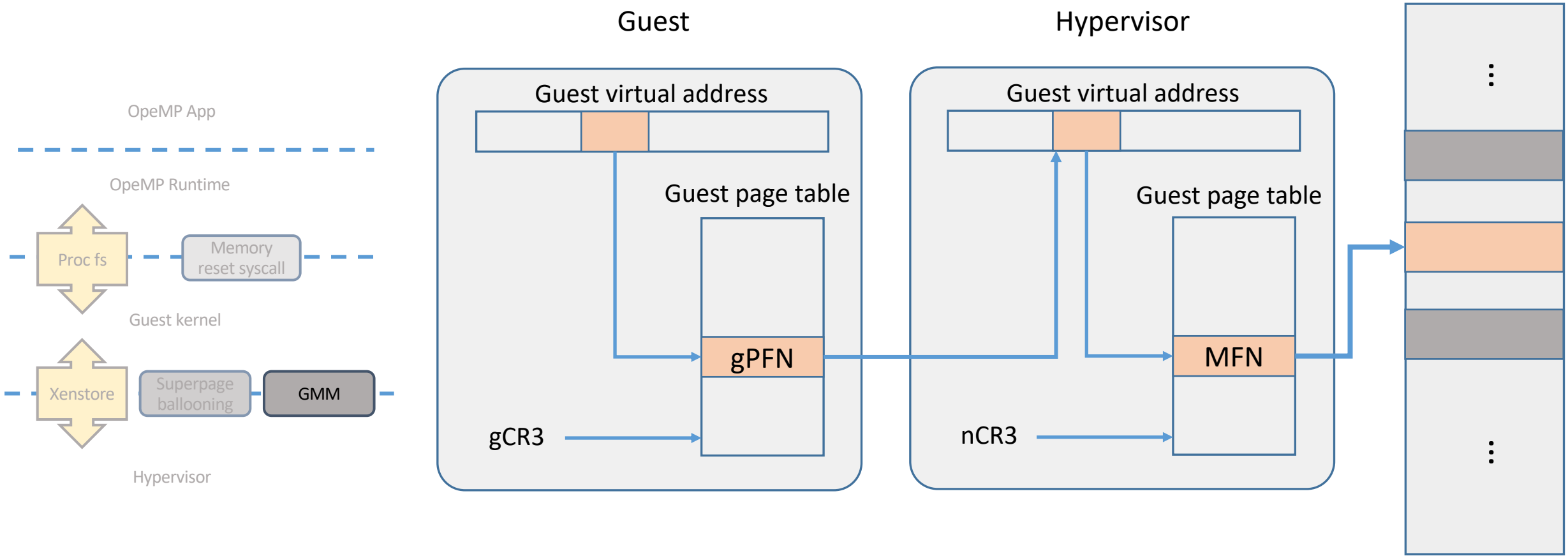
- ~~Not NUMA aware~~
- NUMA awareness of ballooning
  - Each node contains one balloon
- ~~Slow~~
- Superpage ballooning
  - Use superpage for inflation and deflation

Superpage ballooning performance

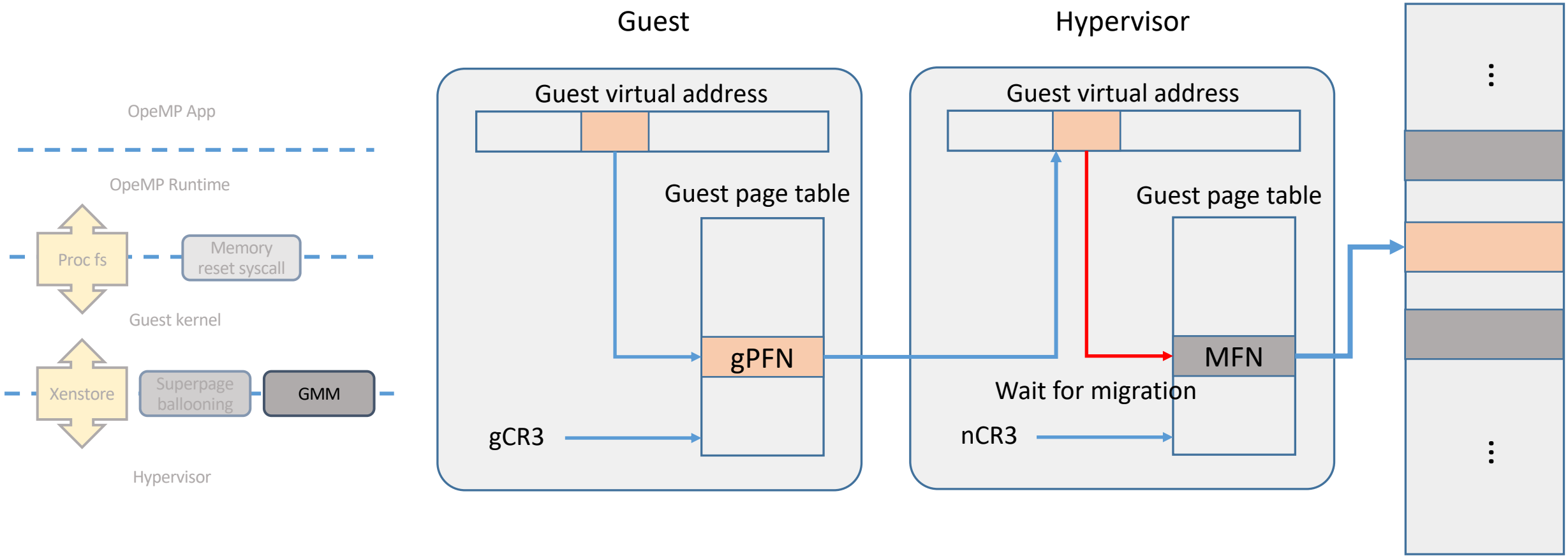


- Superpage ballooning outperform ballooning by up to 30X
- Decrease reservation more expensive than increase reservation

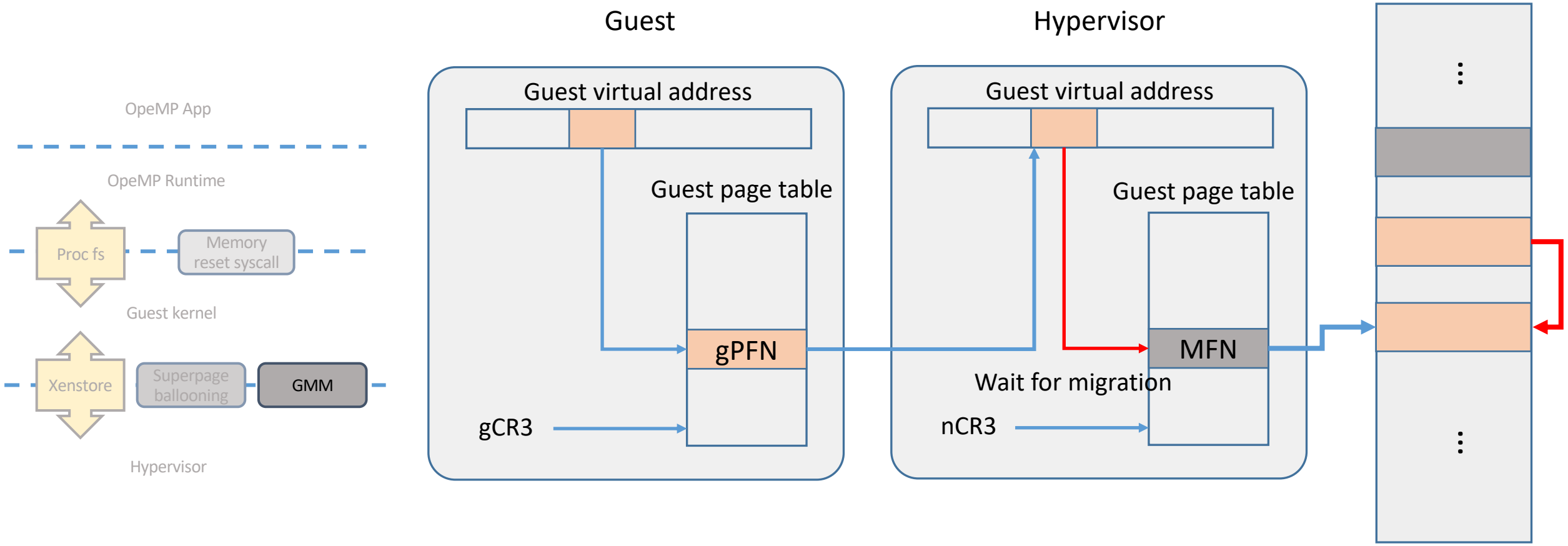
Make topology change complete (Guest Memory Migraion GMM)



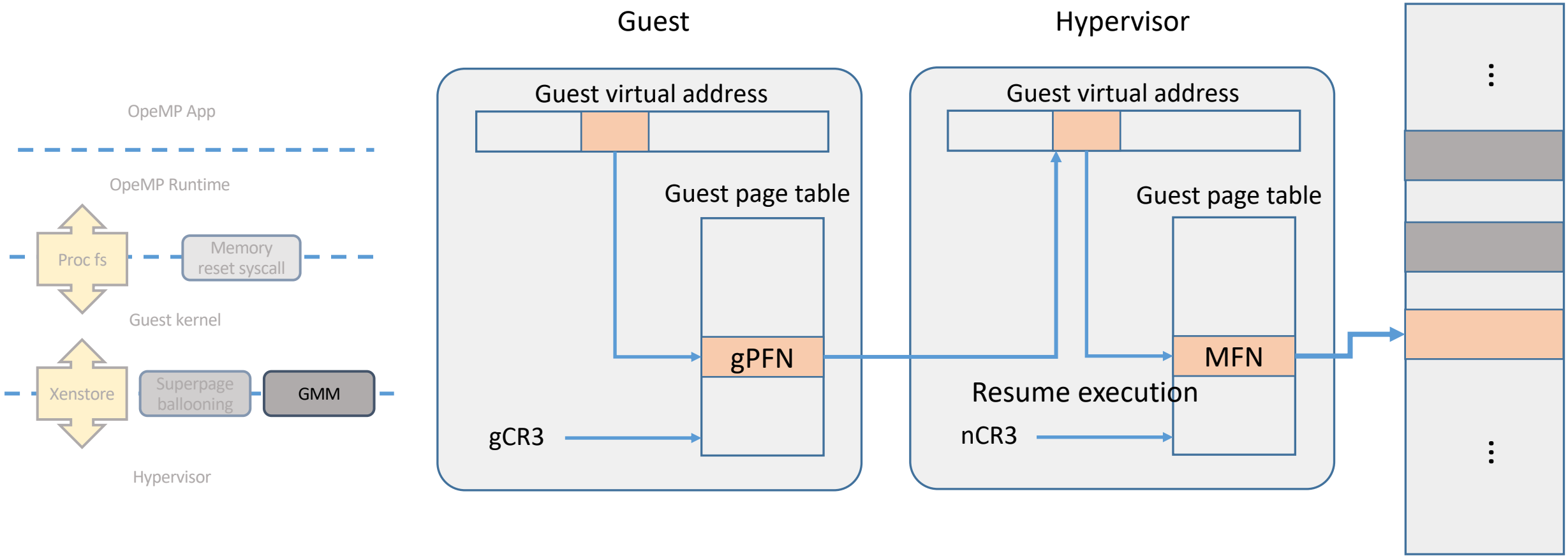
Make topology change complete (Guest Memory Migraion GMM)



Make topology change complete (Guest Memory Migraion GMM)



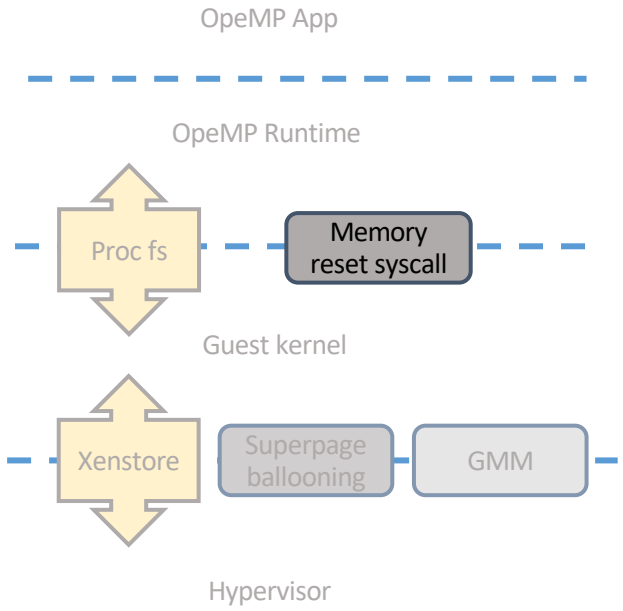
Make topology change complete (Guest Memory Migration GMM)

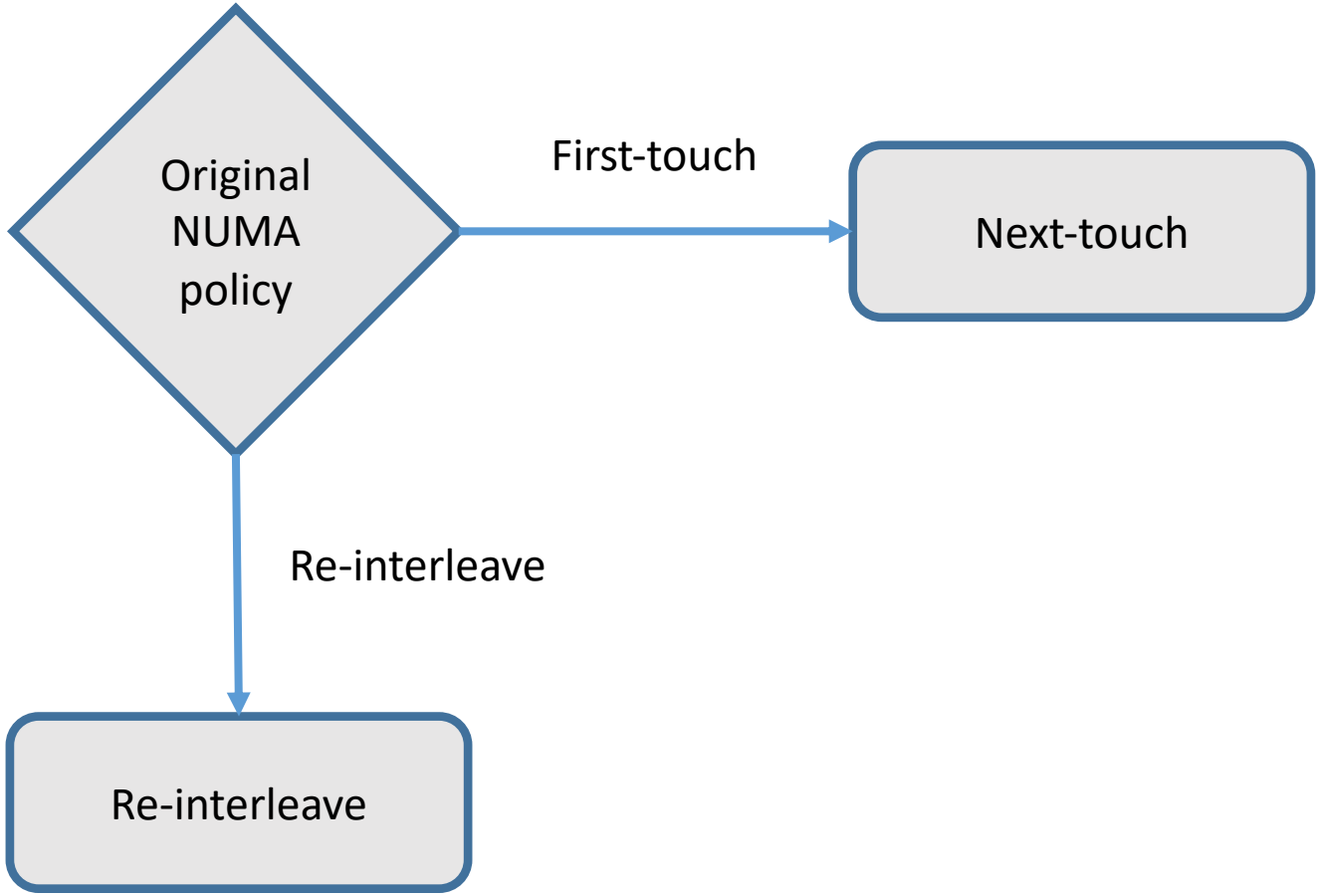


Changes to OpenMP runtime (~200 loc)

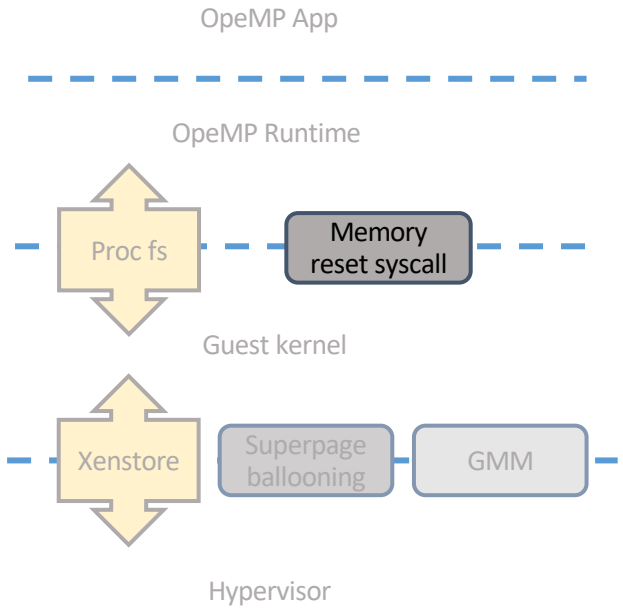
- Topology change notification:
  - Check topology version at the beginning of each parallel session.
- Thread adaptation:
  - Uses OMP_DYNAMIC to dynamically change number of threads
  - Reassign thread affinity on OMP_PLACES if necessary
- Memory adaptation:
  - Issues memory reset system call before launching threads

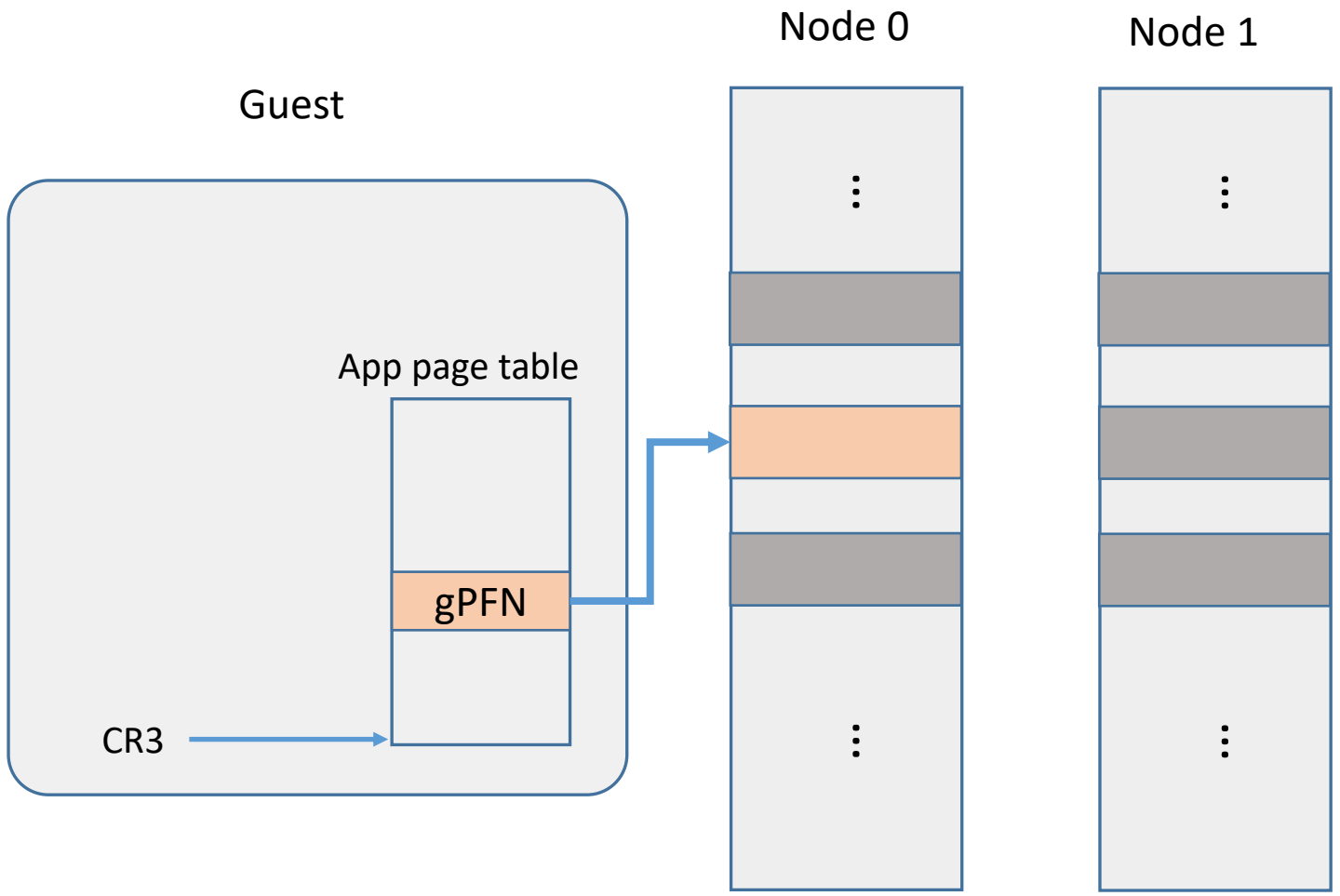
Application adapt to topology change (Memory reset syscall)



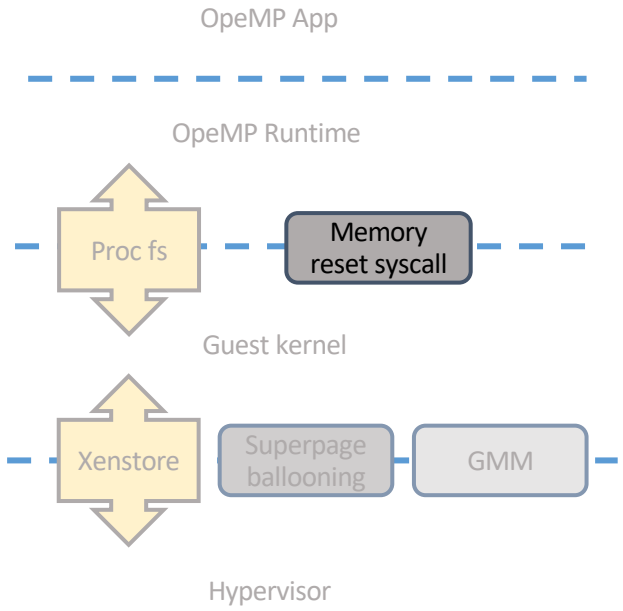


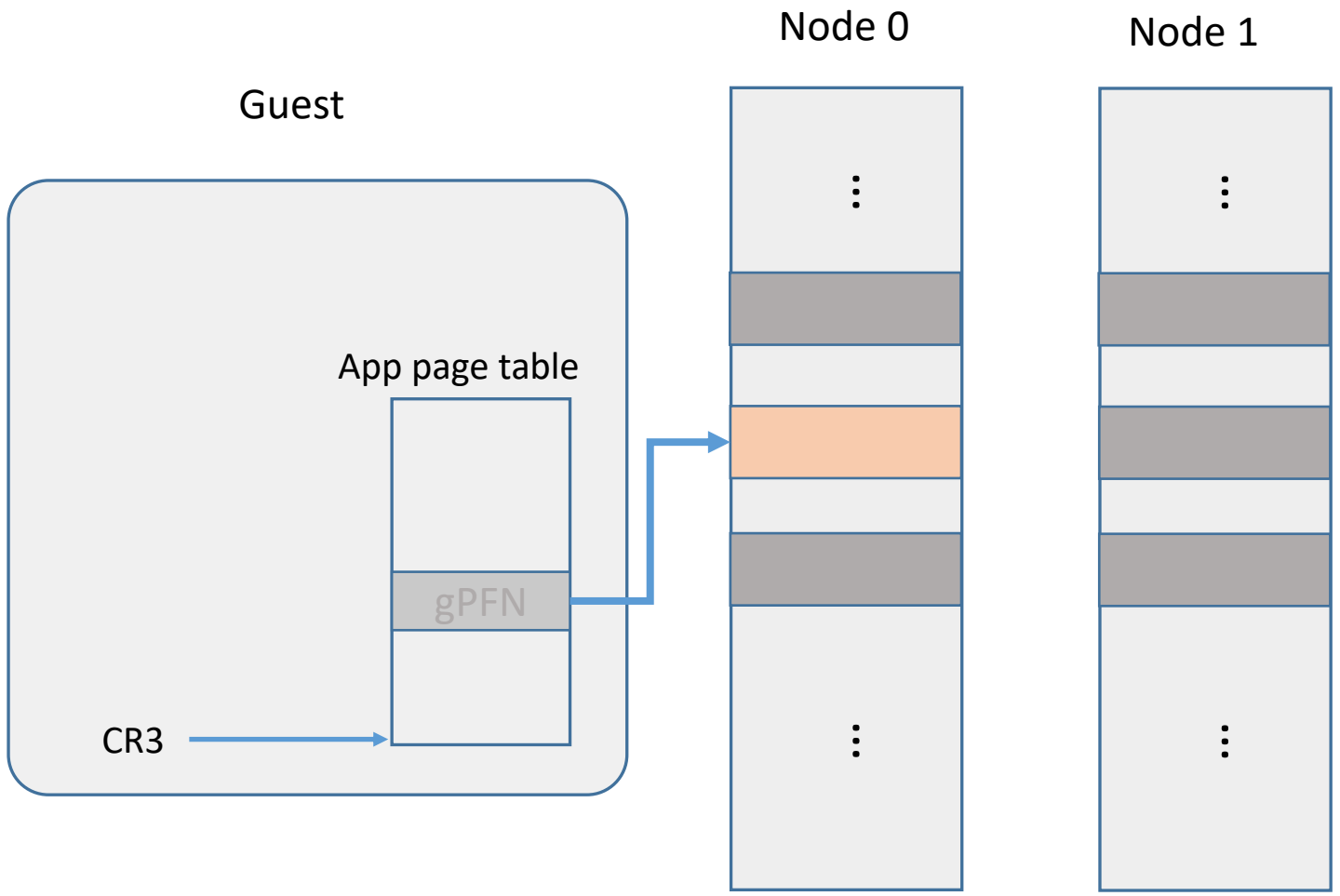
Application adapt to topology change (Next-touch)



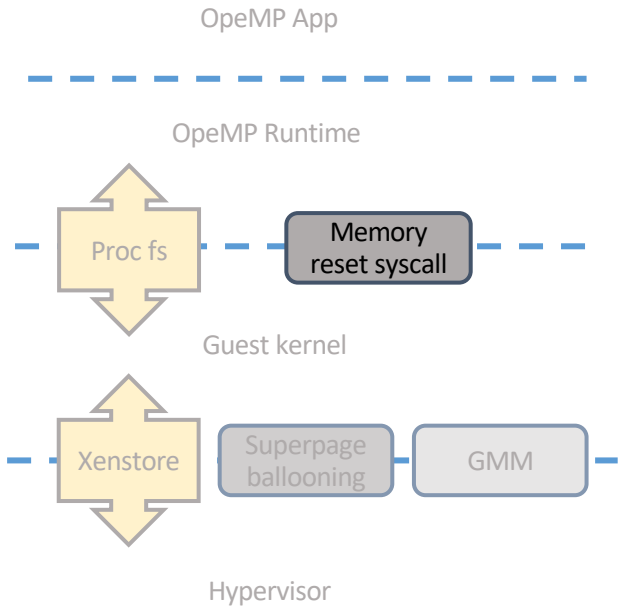


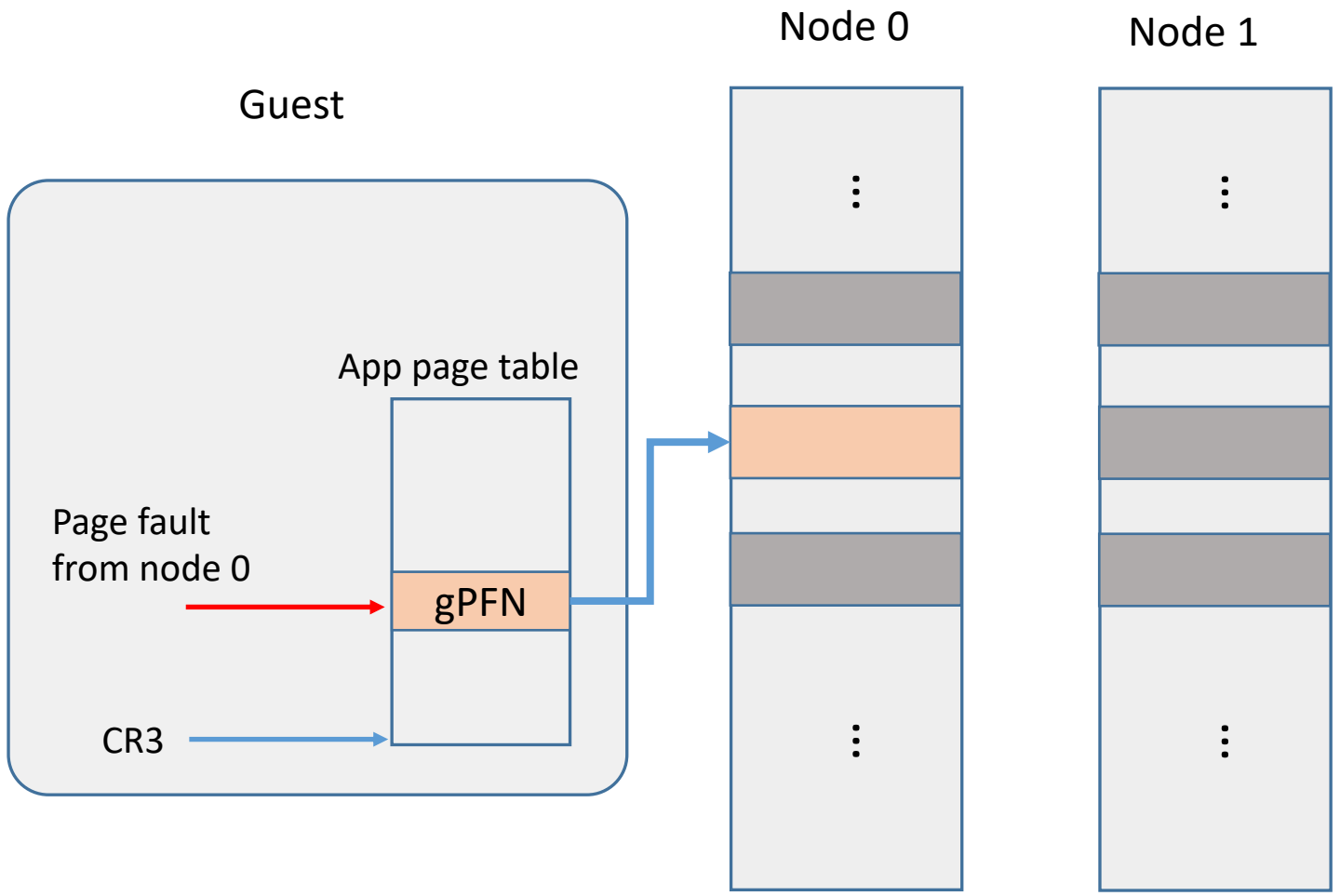
Application adapt to topology change (Next-touch)



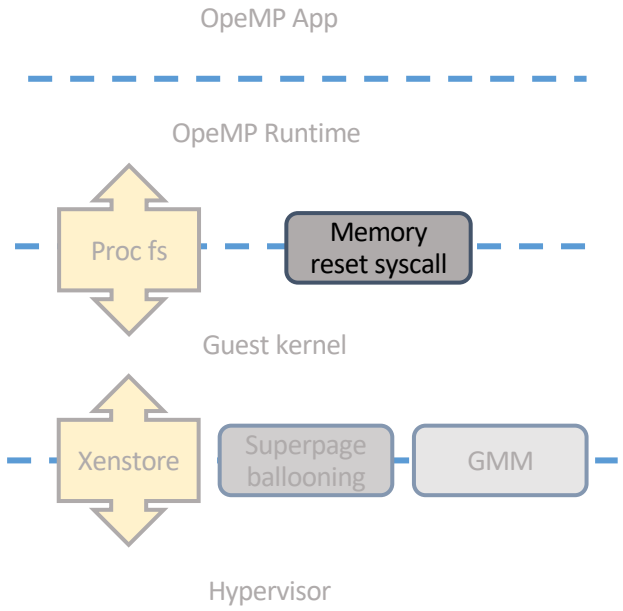


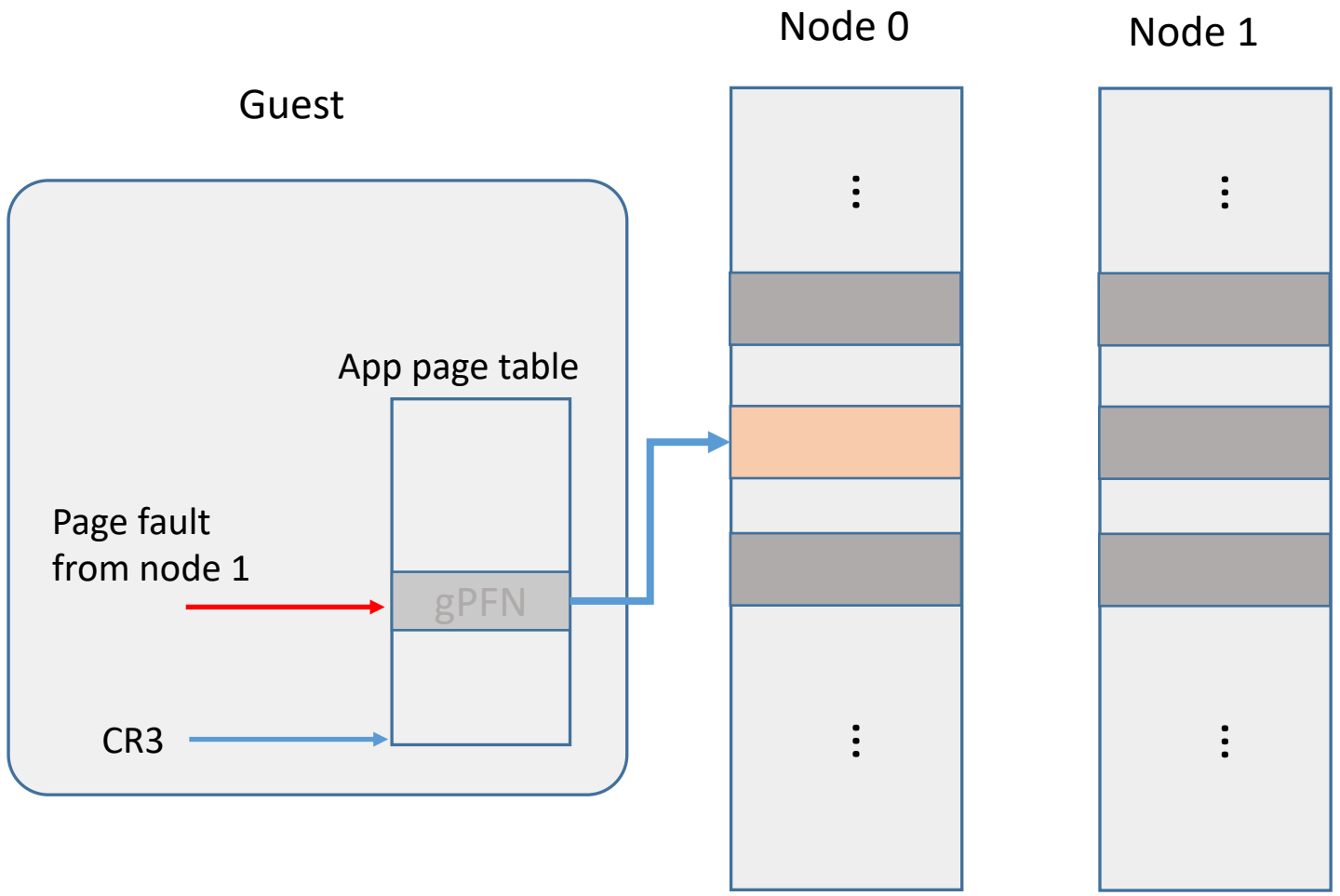
Application adapt to topology change (Next-touch)



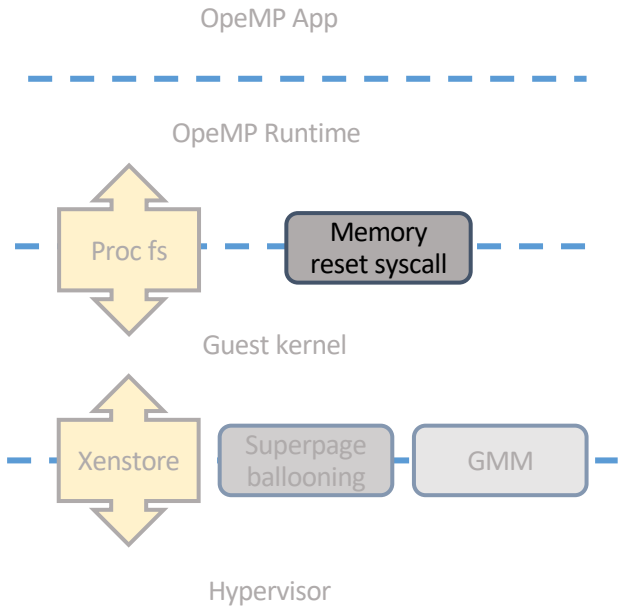


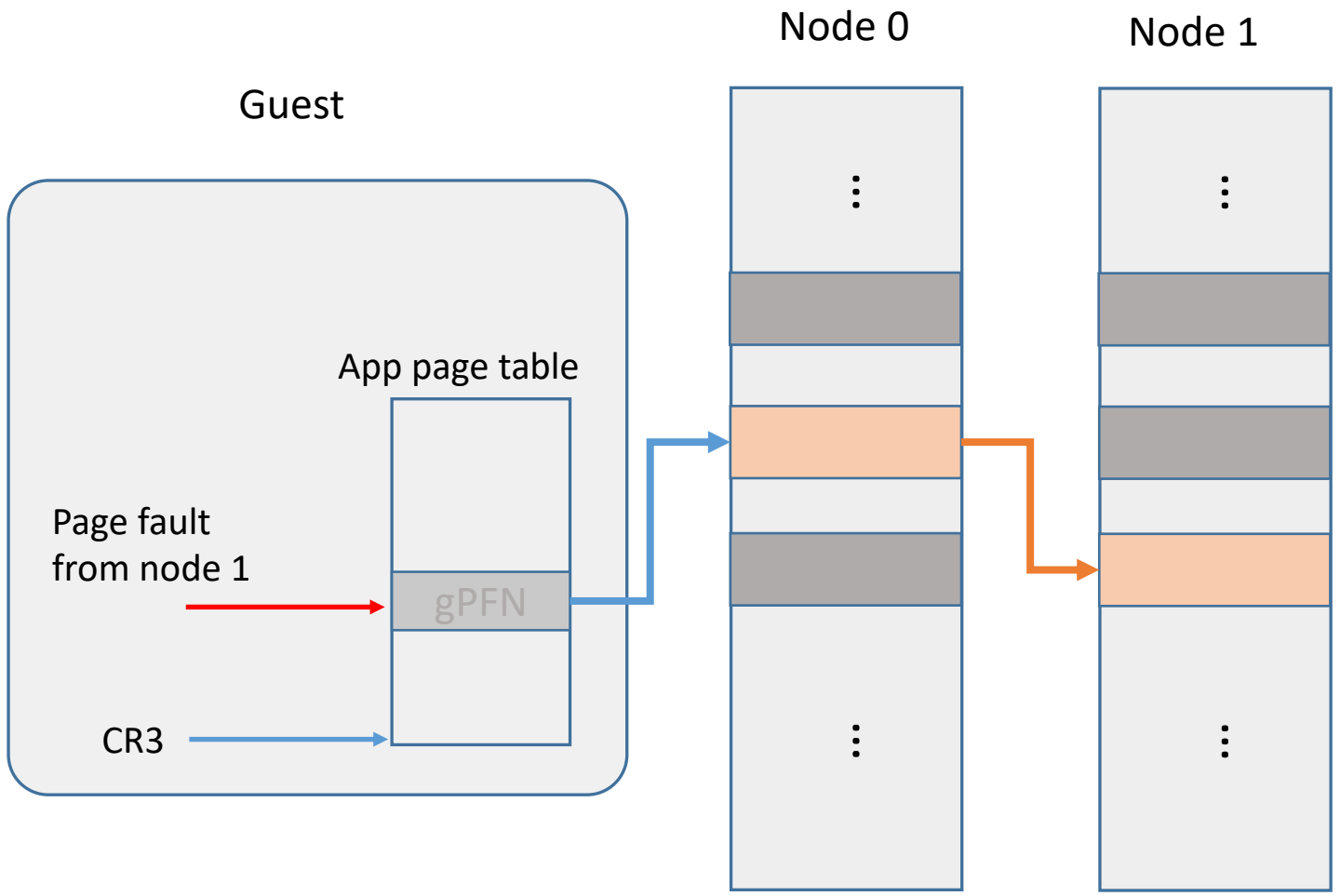
Application adapt to topology change (Next-touch)



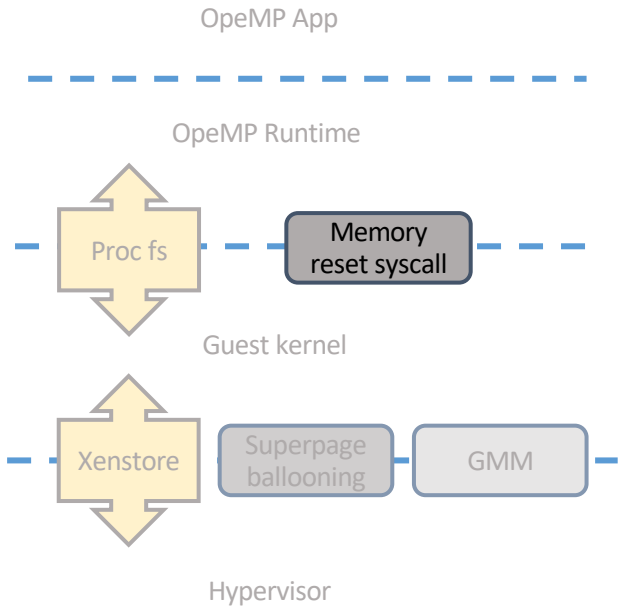


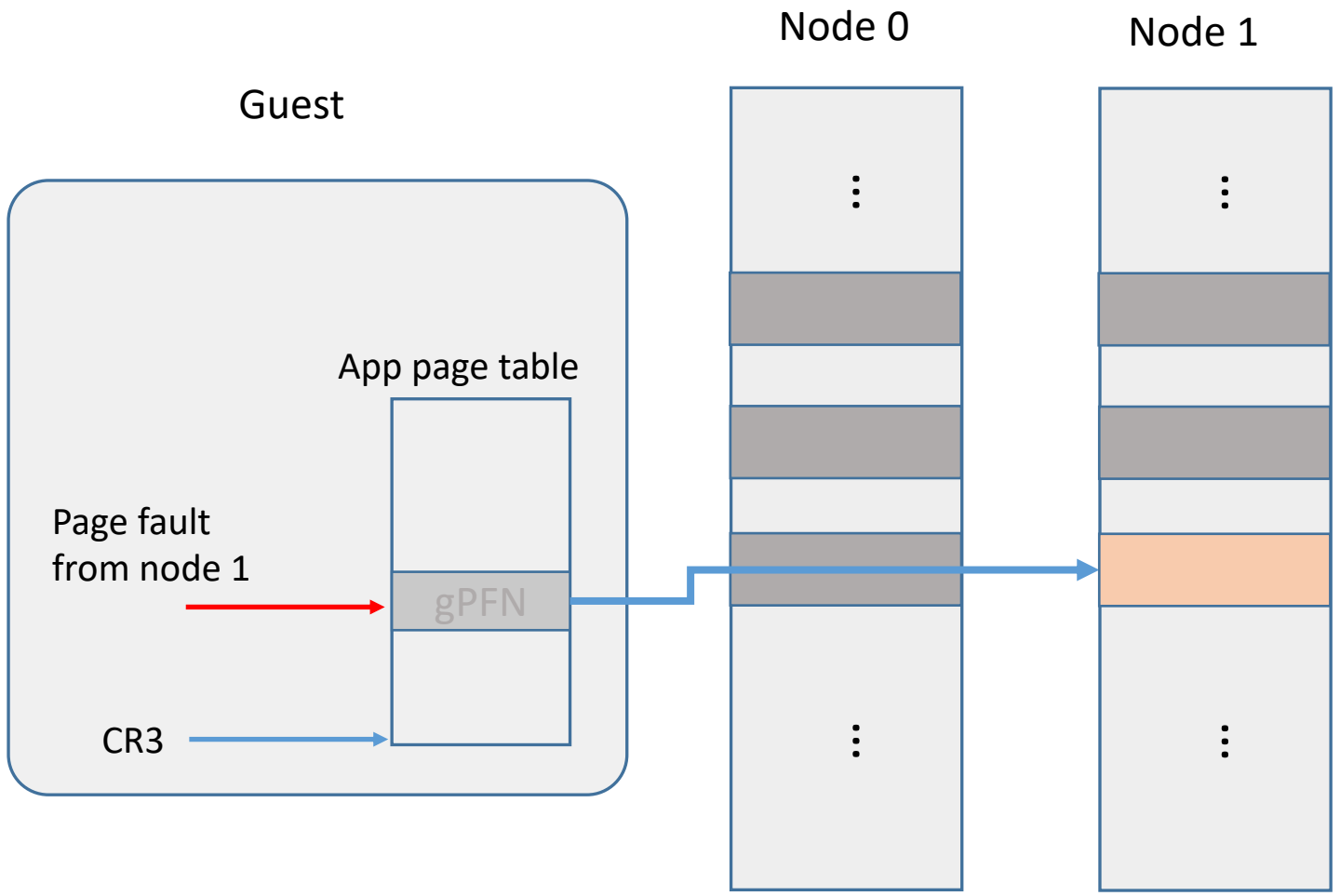
Application adapt to topology change (Next-touch)



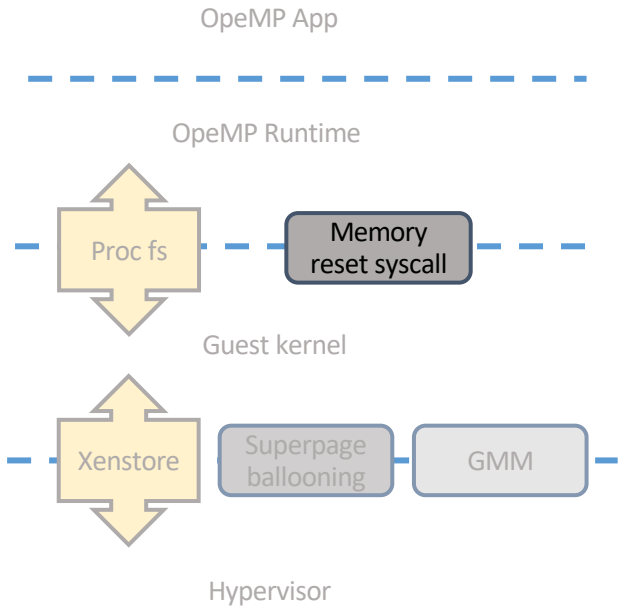


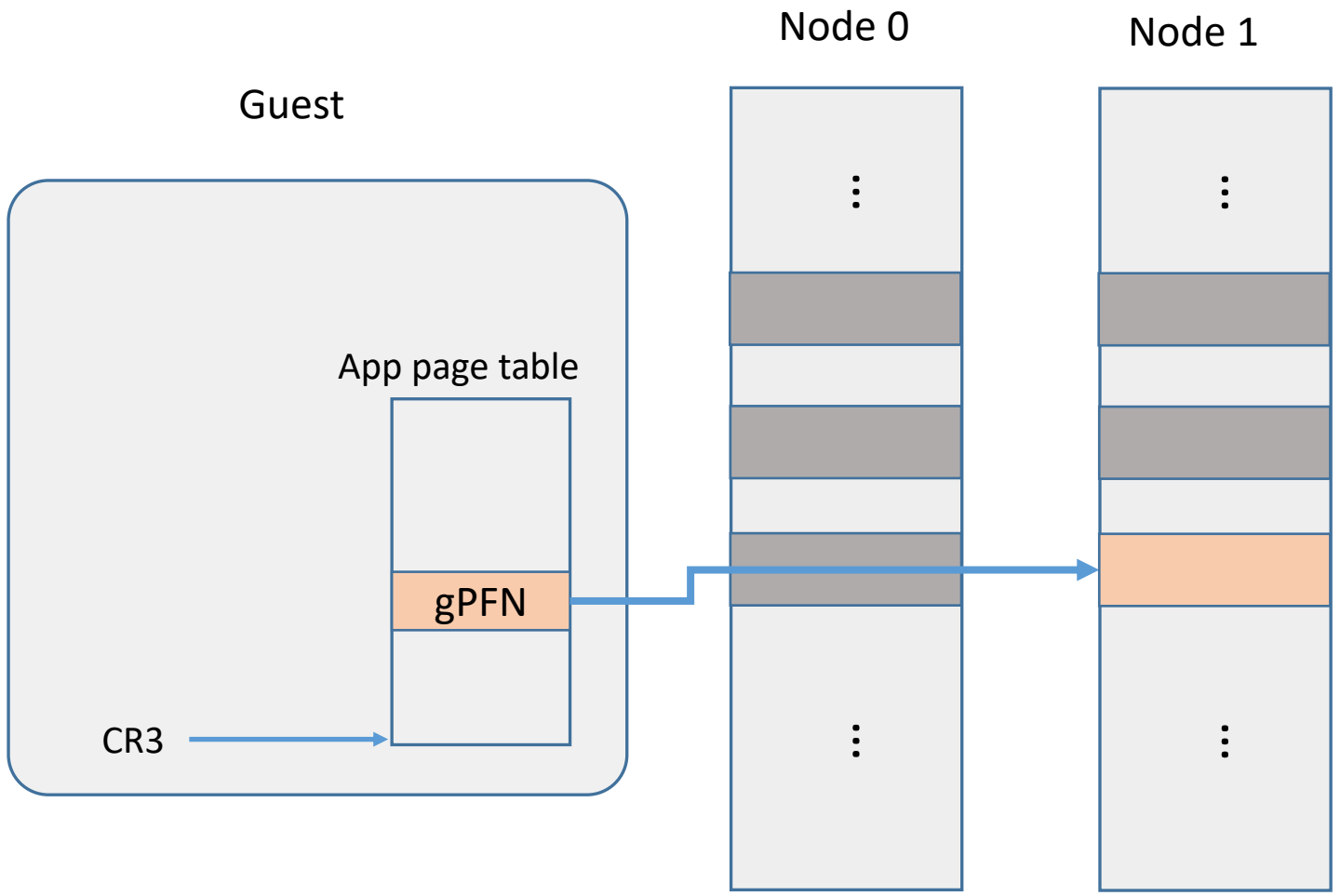
Application adapt to topology change (Next-touch)





Application adapt to topology change (Next-touch)





Re-interleave

- Relocate interleaved memory pages to the new set of available nodes after topology change in a round-robin way.
- Does not requires page fault to initiate the migration
- Two versions available, serial version and parallel version

Evaluation

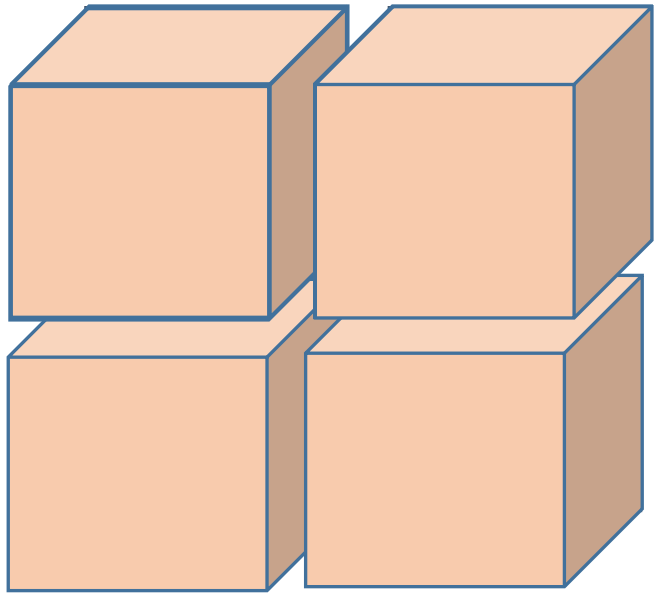
Experiment setup

- Environment:
  - AMD EPYC 7551p, 32 cores, 64 hardware threads
  - 4 NUMA nodes
  - Each node has 2666 MHz DDR4 channels with 16GB of memory
  - AMD’s Infinity Fabric max bandwidth of 21.325 GB/s
  - Xen 4.11 and Linux 4.18 with GCC 7.3

Experiment setup

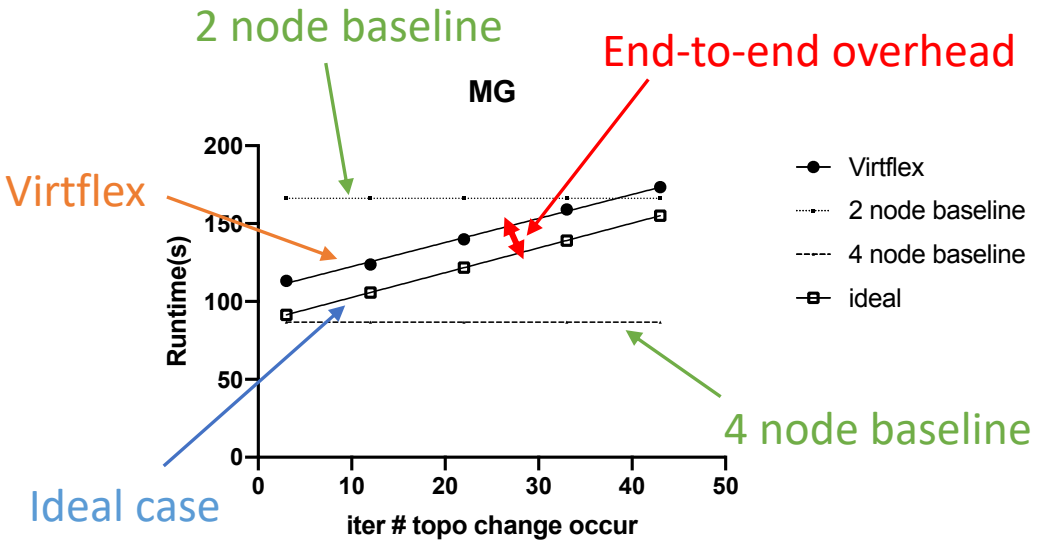
- Benchmarks:
  - NPB 3.3.1, Parsec 3.0, HPC Challenge's RandomAccess ("GUPS")
- Two scenarios
  - Adding node
  - Removing node

End-to-end experiment (topology expansion)



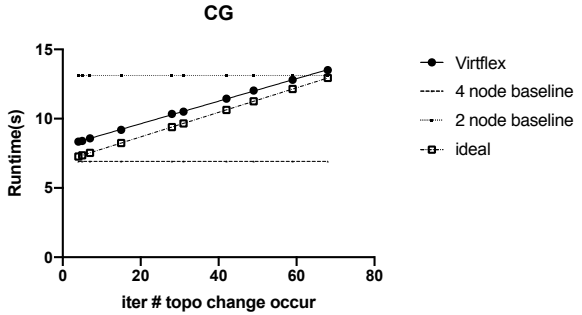
- VM booted up 2 NUMA nodes
- Application starts
- Expand topology, 2 other nodes are populated

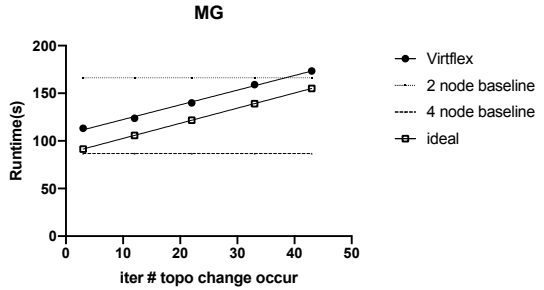
Evaluation (adding nodes)

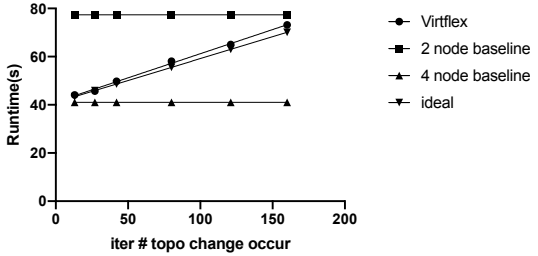


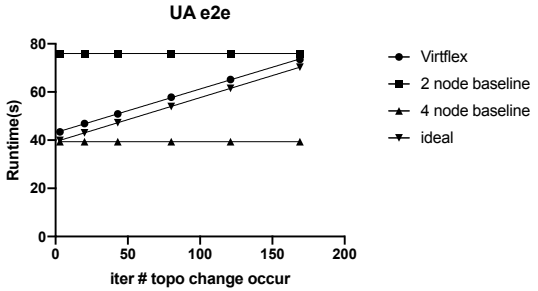
- End-to-end topology change adaptation overhead is on average 7.27%

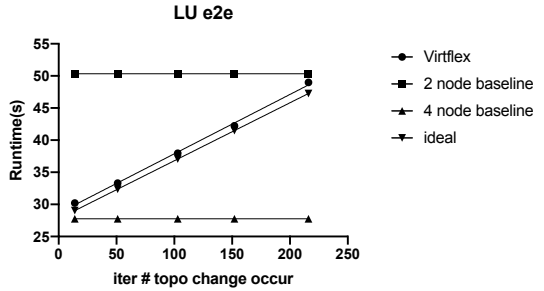
More results

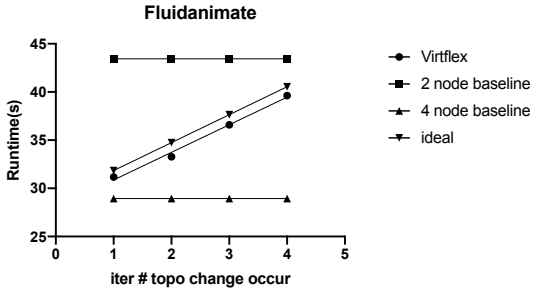




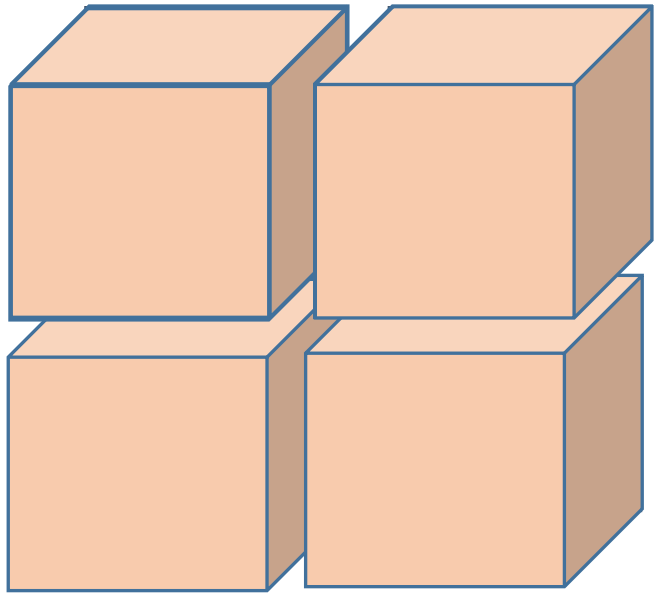








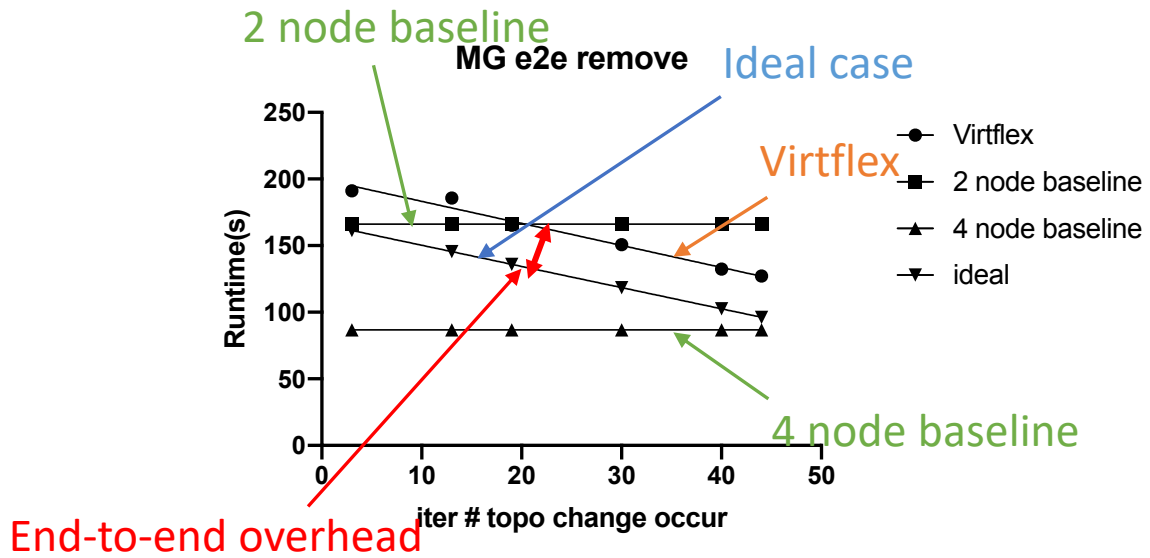
End to end experiment (topology shrinking)



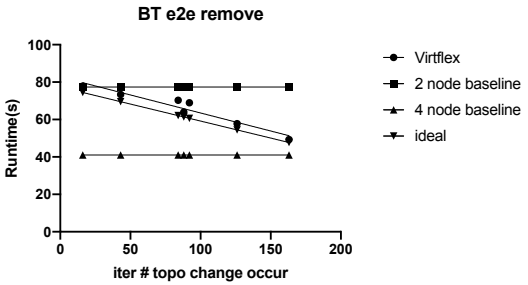
- VM booted up with 4 fully-blown nodes
- Application starts
- Shrinking topology, 2 nodes are de-populated

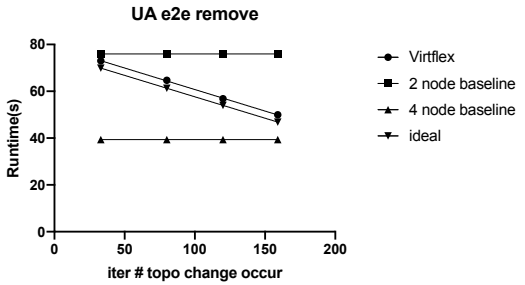
Evaluation (removing nodes)

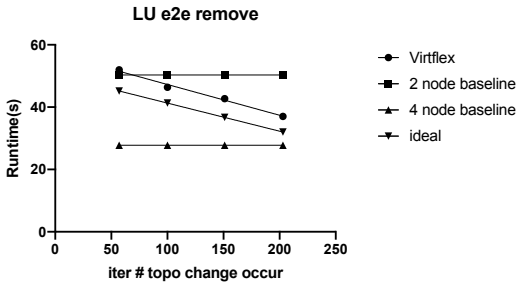
- The average overhead for the remove case is 19.39%.
  - Inflating balloon takes longer
  - There are less CPU cores doing the migration in the removing case

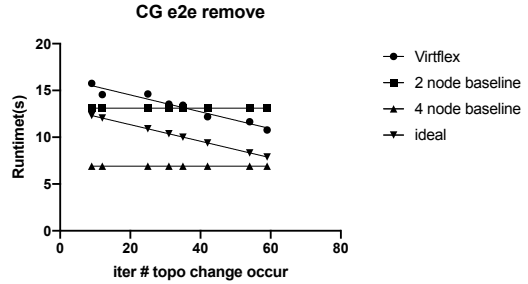


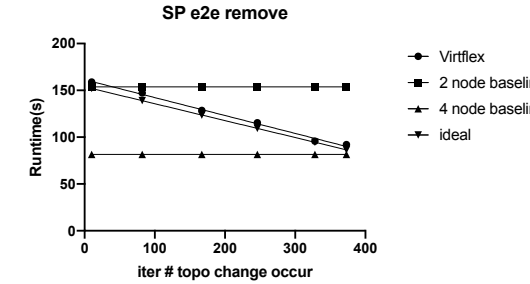
More results

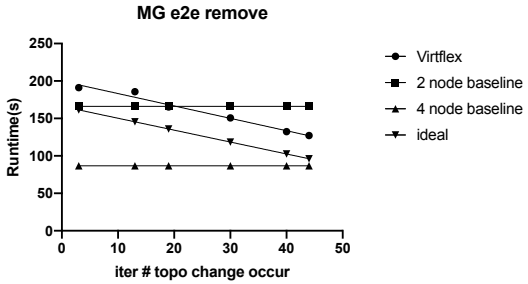












Conclusion

- Virtflex allows un-modified OpenMP applications to adapt automatically to NUMA topology changes with low overhead.
  - NUMA-aware Superpage ballooning changes topology fast
  - Guest memory migration changes topology completely
  - Memory reset syscall allow application to adapt with ease

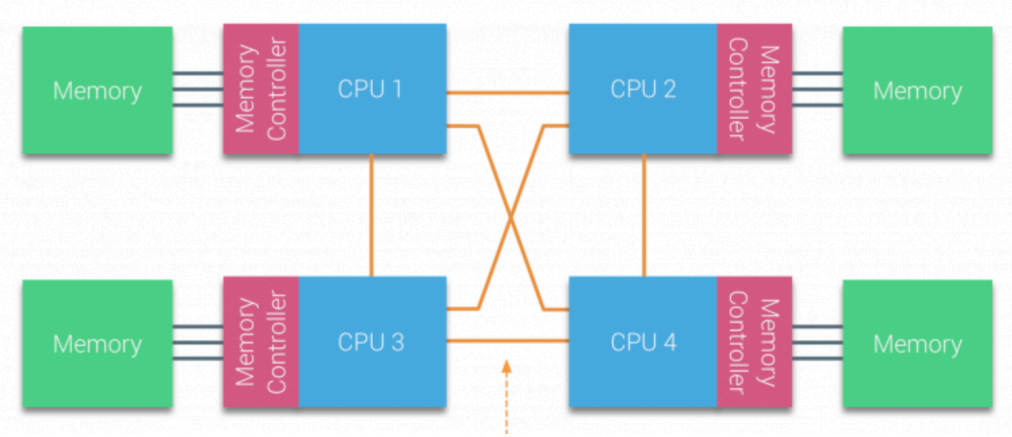
Selected References

- [1] S. L. Olivier, A. K. Porterfield, K. B. Wheeler, M. Spiegel, and J. F. Prins, “Openmp task scheduling strategies for multicore numa systems,” The Inter- national Journal of High Performance Computing Applications, vol. 26, no. 2, pp. 110–124, 2012.
- [2] M. Durand, F. Broquedis, T. Gautier, and B. Raffin, “An efficient openmp loop scheduler for irregular applications on large-scale numa machines,” in Interna- tional Workshop on OpenMP, pp. 141–155, Springer, 2013.
- [3] Muddukrishna, Ananya, Peter A. Jonsson, Vladimir Vlassov, and Mats Brorsson. "Locality-aware task scheduling and data distribution on NUMA systems." In *International Workshop on OpenMP*, pp. 156-170. Springer, Berlin, Heidelberg, 2013.
- [9] J. Rao, K. Wang, X. Zhou, and C.-Z. Xu, “Optimizing virtual machine scheduling in numa multicore systems,” in 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA), pp. 306–317, IEEE, 2013.
- [10] D. S. Rao and K. Schwan, “vnuma-mgr: Managing vm memory on numa plat- forms,” in 2010 International Conference on High Performance Computing, pp. 1–10, IEEE, 2010.
- [11] M. Liu and T. Li, “Optimizing virtual machine consolidation performance on numa server architecture for cloud workloads,” in 2014 ACM/IEEE 41st In- ternational Symposium on Computer Architecture (ISCA), pp. 325–336, IEEE, 2014.
- [12] S. Wu, H. Sun, L. Zhou, Q. Gan, and H. Jin, “vprobe: Scheduling virtual ma- chines on numa systems,” in 2016 IEEE International Conference on Cluster Computing (CLUSTER), pp. 70–79, IEEE, 2016.
- [14] B. Bui, D. Mvondo, B. Teabe, K. Jiokeng, L. Wapet, A. Tchana, G. Thomas, D. Hagimont, G. Muller, and N. DePalma, “When extended para - virtualization (xpv) meets numa,” in Proceedings of the Fourteenth EuroSys Conference 2019, EuroSys ’19, (New York, NY, USA), pp. 7:1–7:15, ACM, 2019.
- [15] O. Agmon Ben-Yehuda, E. Posener, M. Ben-Yehuda, A. Schuster, and A. Mu’alem, “Ginseng: Market-driven memory allocation,” in Proceedings of the 10th ACM SIGPLAN/SIGOPS international conference on Virtual execu- tion environments, pp. 41–52, 2014.
- [16] R. Smith and S. Rixner, “A policy-based system for dynamic scaling of virtual machine memory reservations,” in Proceedings of the 2017 Symposium on Cloud Computing, pp. 282–294, 2017.
- [17] S. Kundu, R. Rangaswami, M. Zhao, A. Gulati, and K. Dutta, “Revenue driven resource allocation for virtualized data centers,” in 2015 IEEE International Conference on Autonomic Computing, pp. 197–206, IEEE, 2015.
- [18] D. Minarolli and B. Freisleben, “Utility-driven allocation of multiple types of resources to virtual machines in clouds,” in 2011 IEEE 13th Conference on Com- merce and Enterprise Computing, pp. 137–144, IEEE, 2011.

Extra slides.

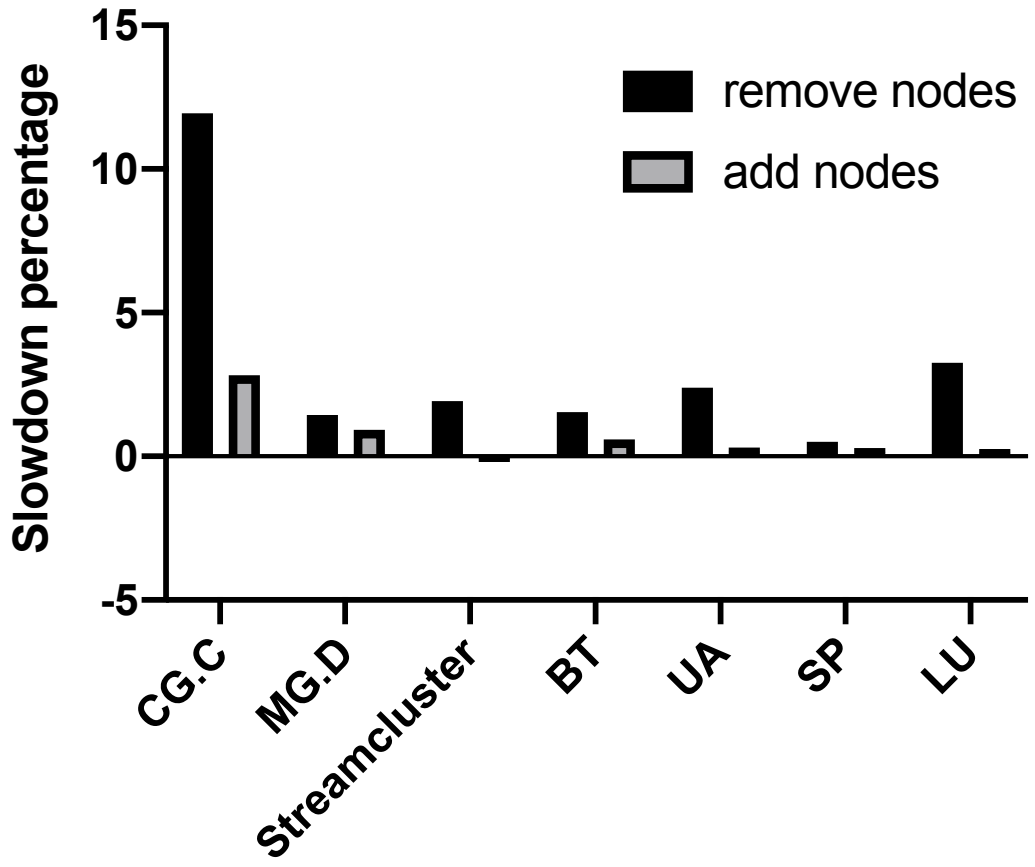
Non-Uniform Memory Access (NUMA) is and will be common

- NUMA is unavoidable
  - Chips become larger
  - Cross-chip/chiplet/racks communication cost becomes high
  - Memory has to be divided into different banks, complex topology
- Future rack-scale computers exhibits NUMA characteristics



https://frankdenneman.nl/2016/07/07/numa-deep-dive-part-1-uma-numa/

Evaluation (background topology change overhead)



- Removing nodes incurrs more overhead than adding nodes for most of the applications.
- The absolute overhead of the topology change on all applications is comparable to standalone ballooning time.