Efficient Memory Disaggregation with Infiniswap

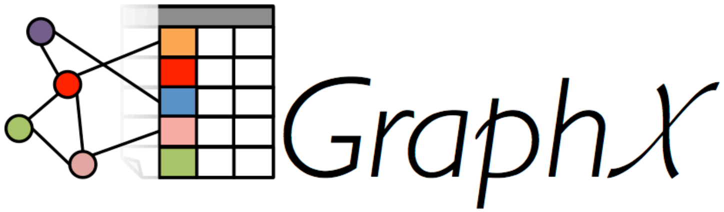
Juncheng Gu, Youngmoon Lee, Yiwen Zhang,
Mosharaf Chowdhury, Kang G. Shin

UNIVERSITY OF
MICHIGAN

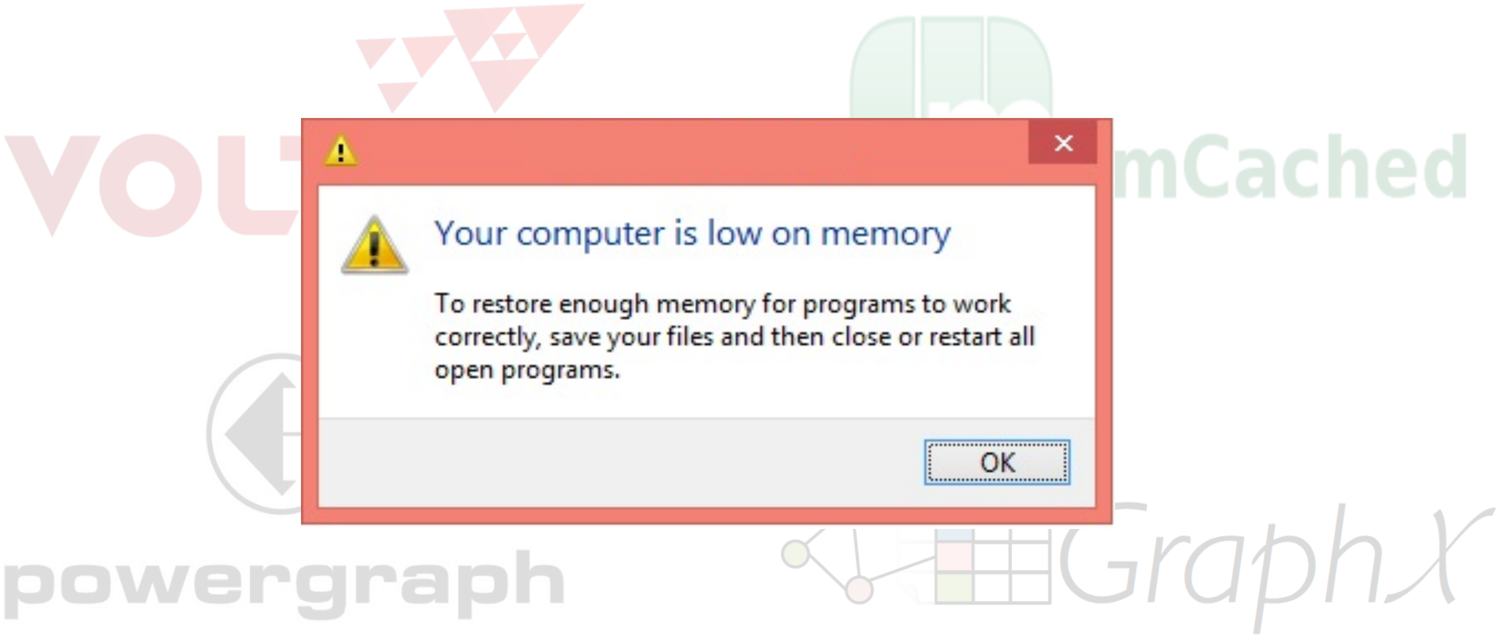
Agenda

- **Motivation and related work**
- Design and system overview
- Implementation and evaluation
- Future work and conclusion

Memory-intensive applications



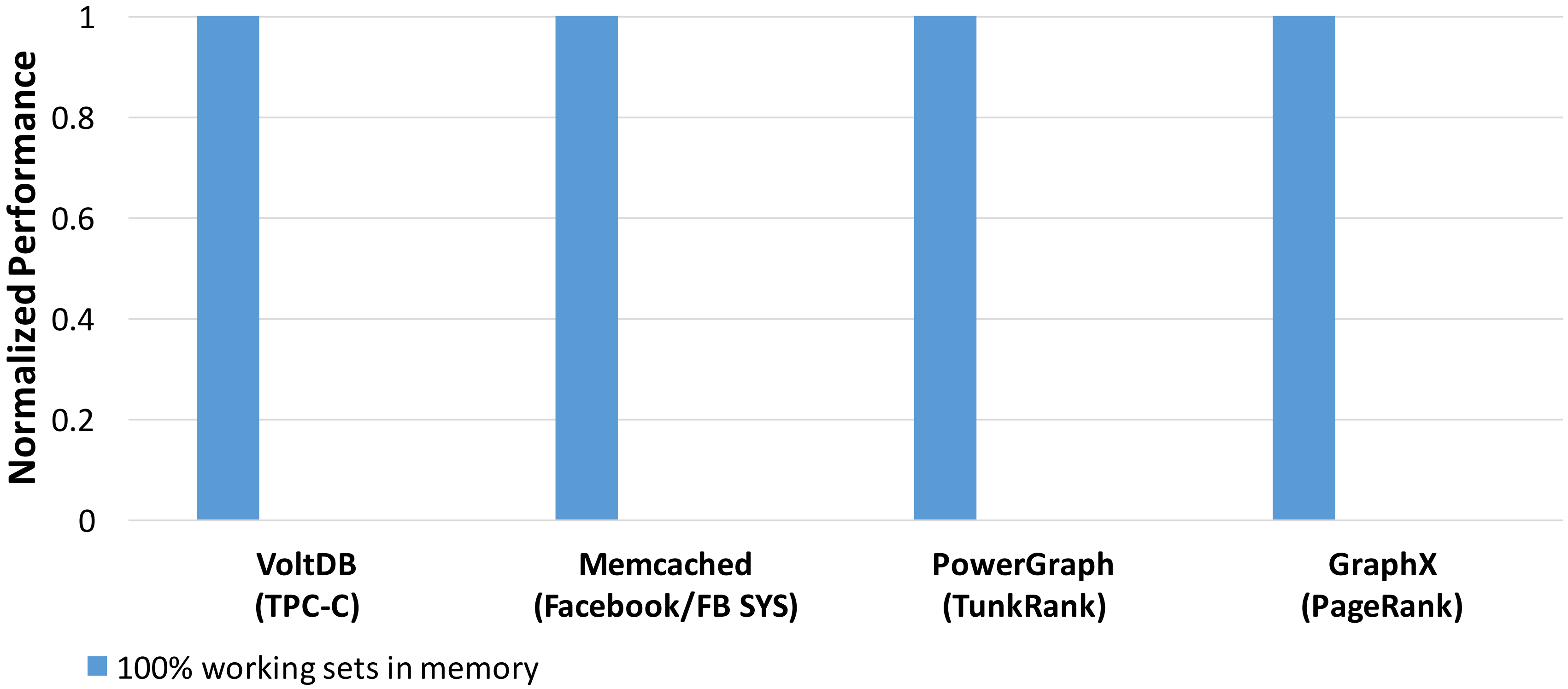
Memory-intensive applications



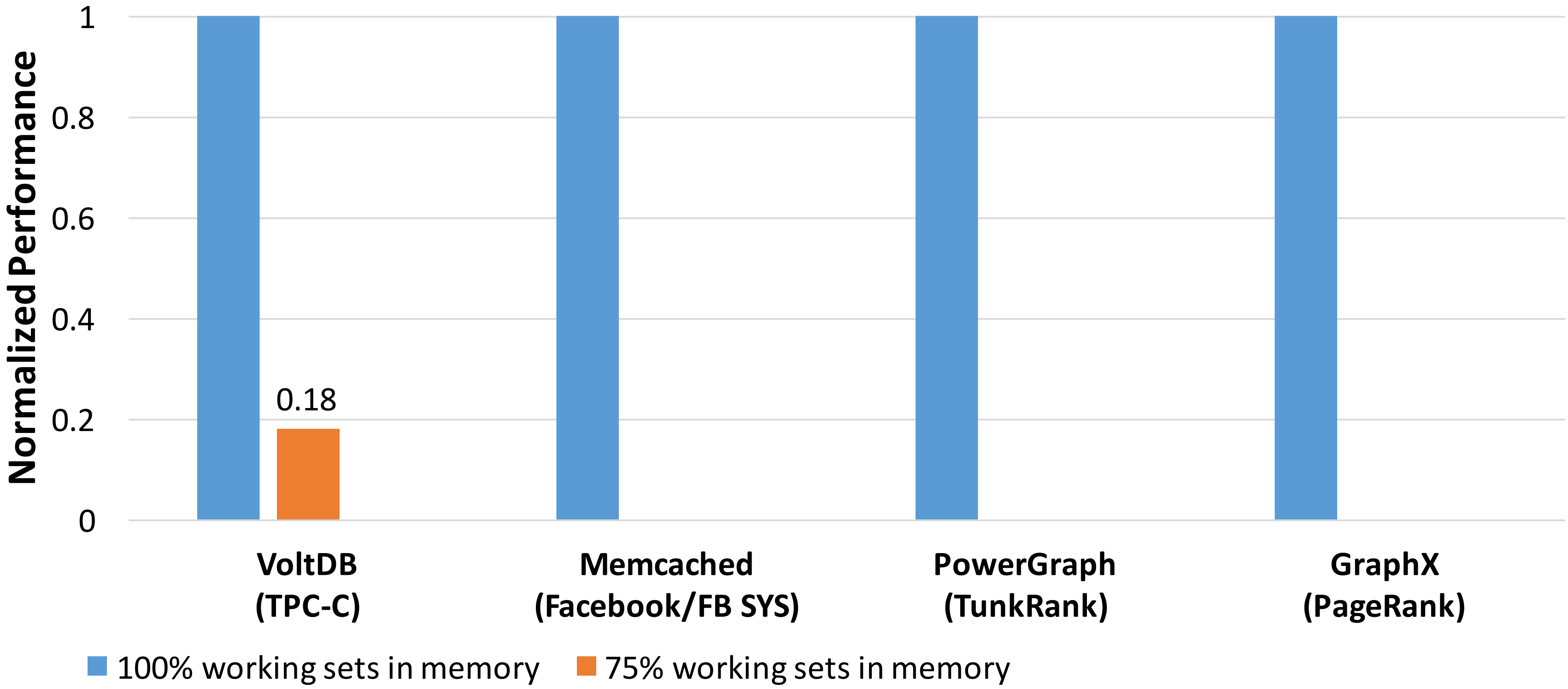
Your computer is low on memory

To restore enough memory for programs to work correctly, save your files and then close or restart all open programs.

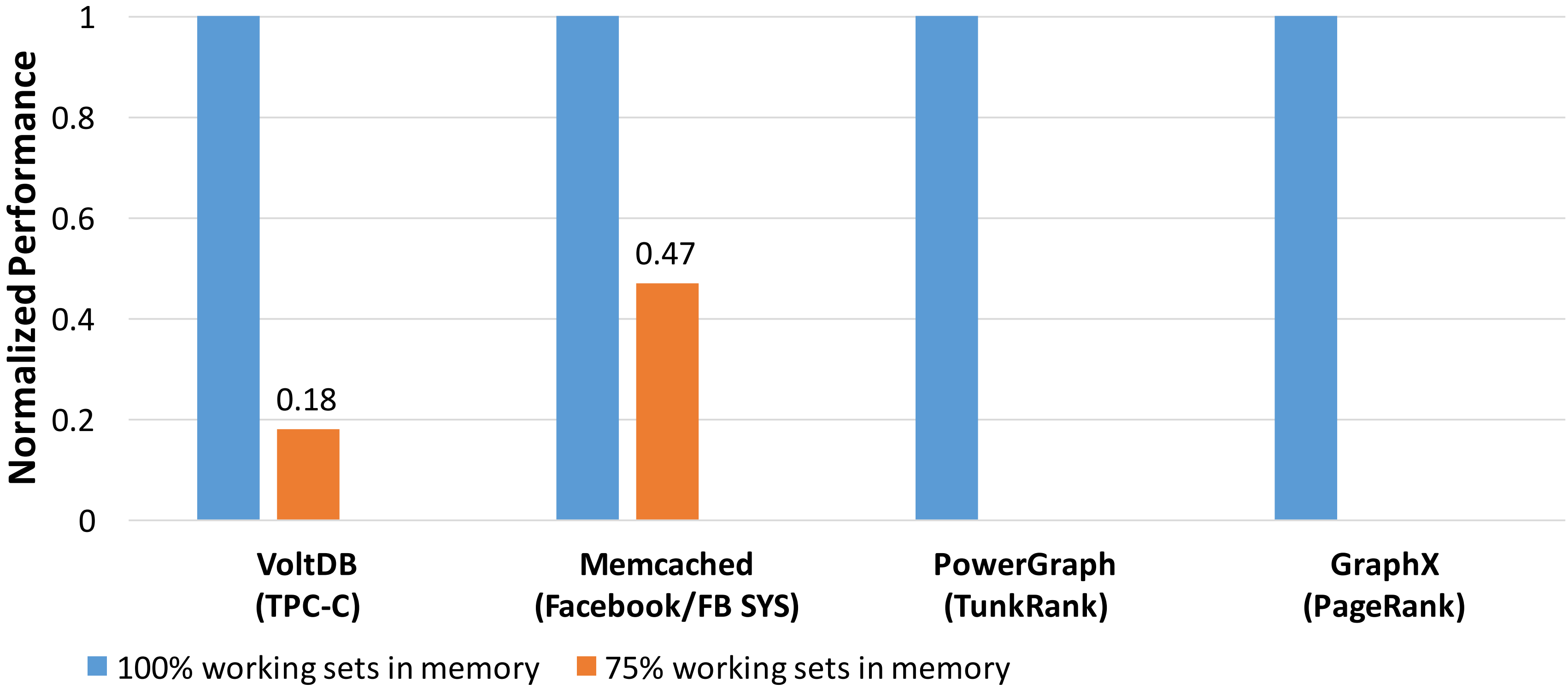
Performance degradation



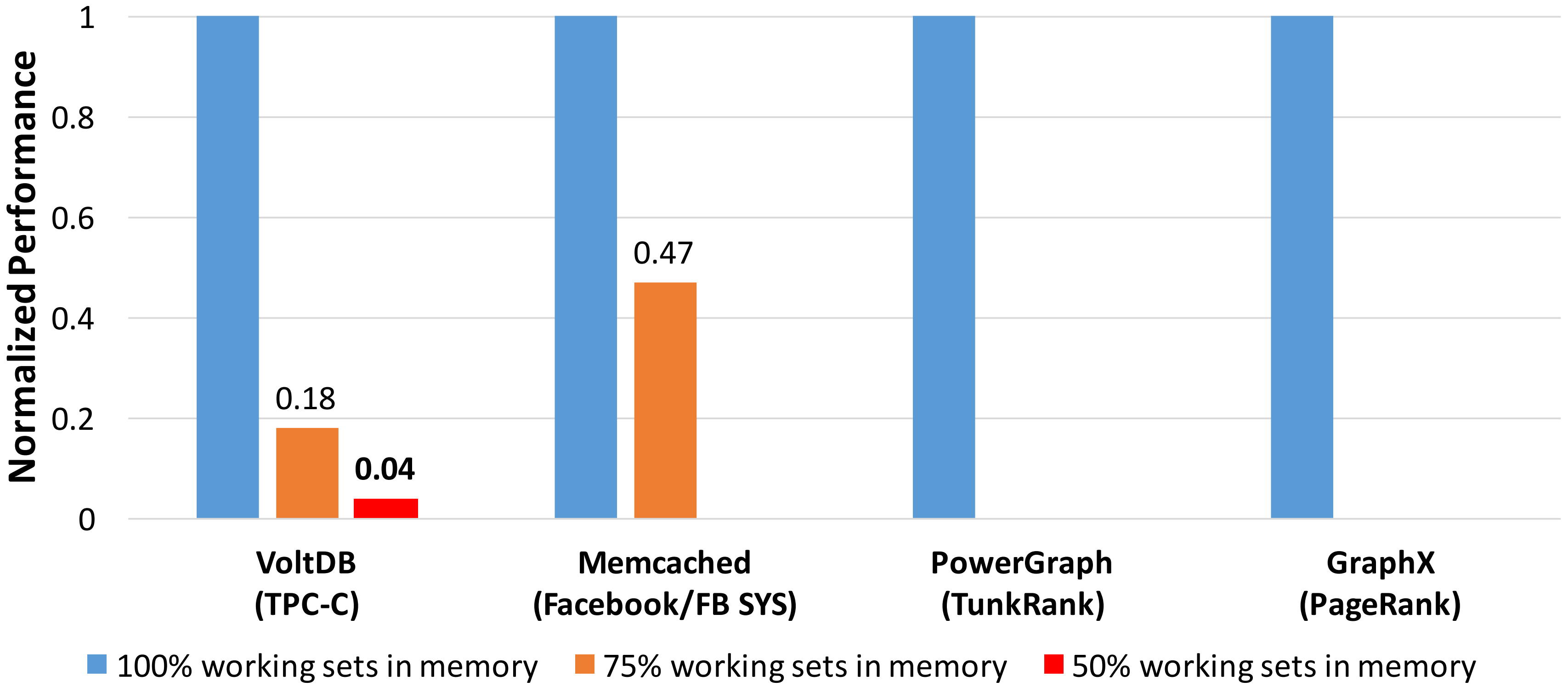
Performance degradation



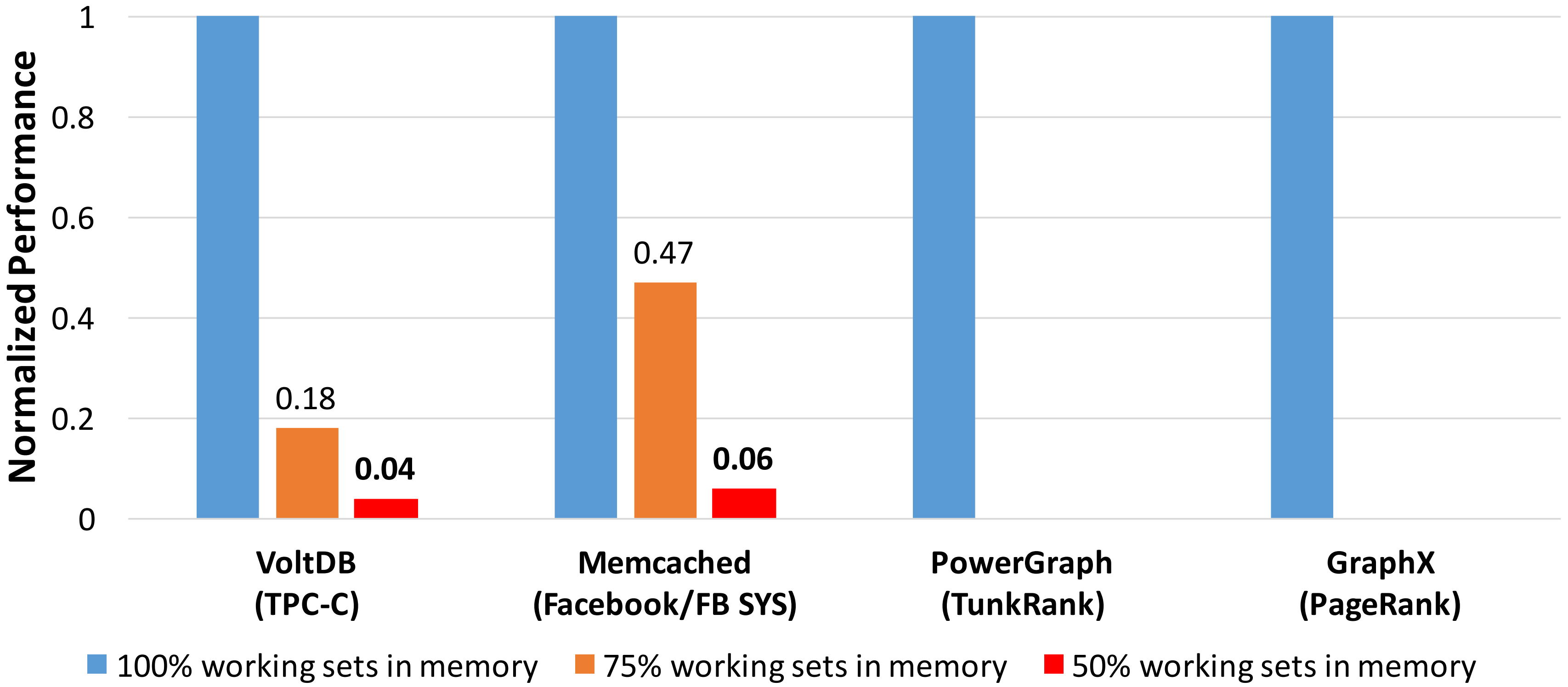
Performance degradation



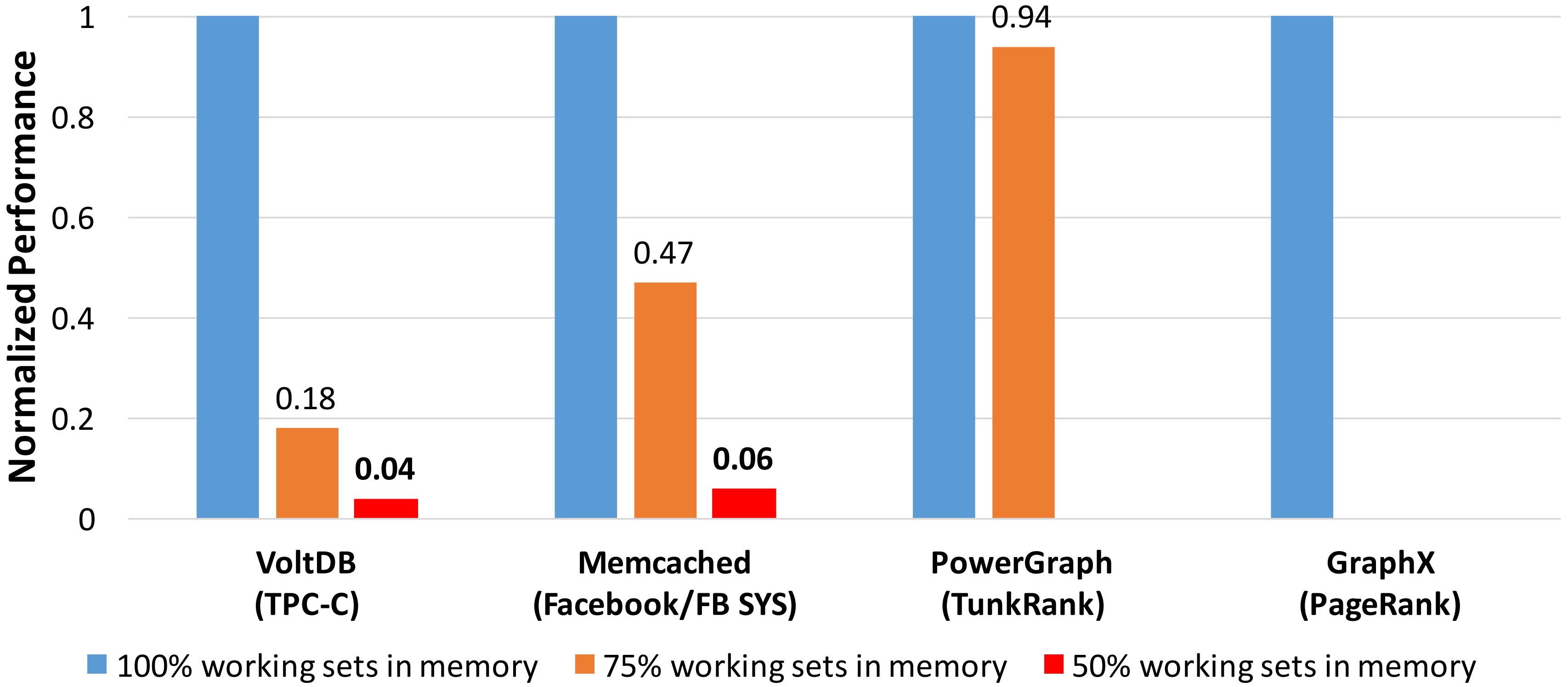
Performance degradation



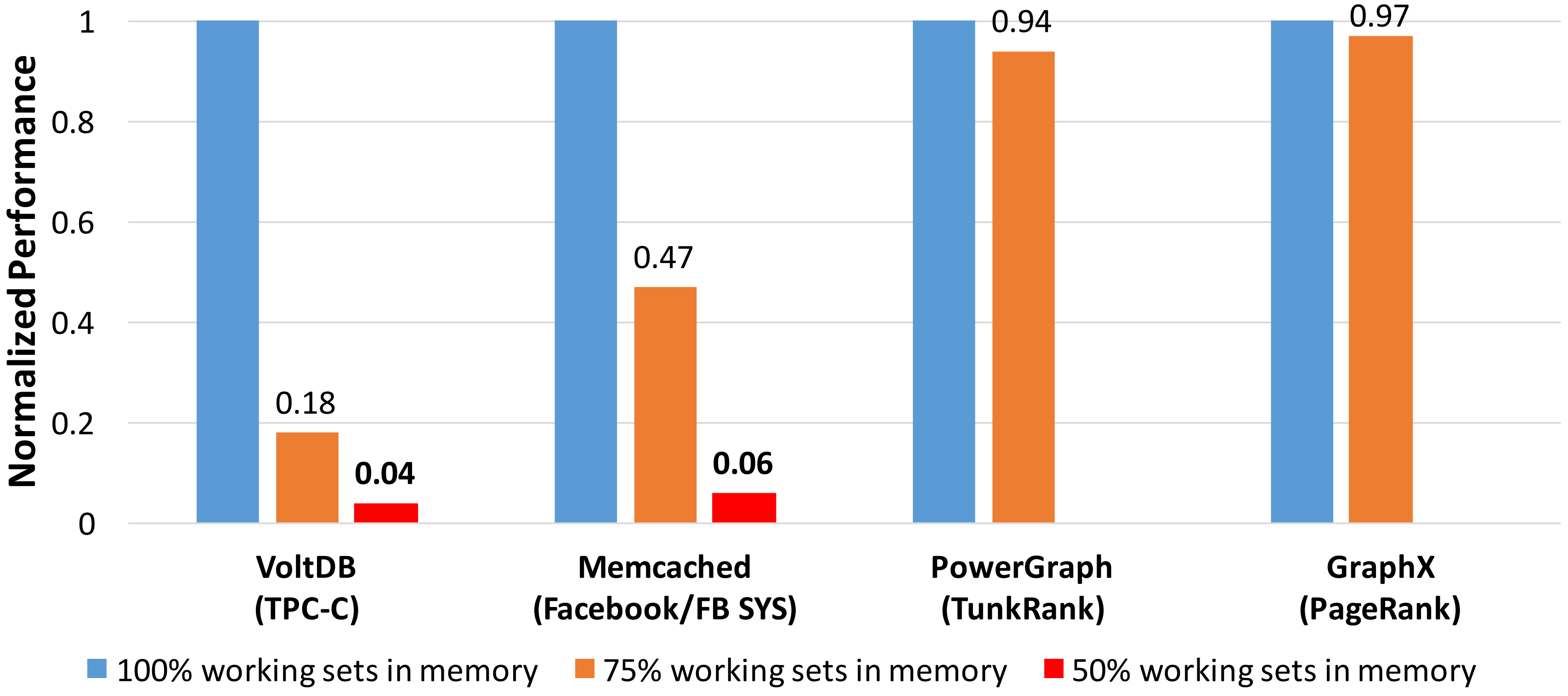
Performance degradation



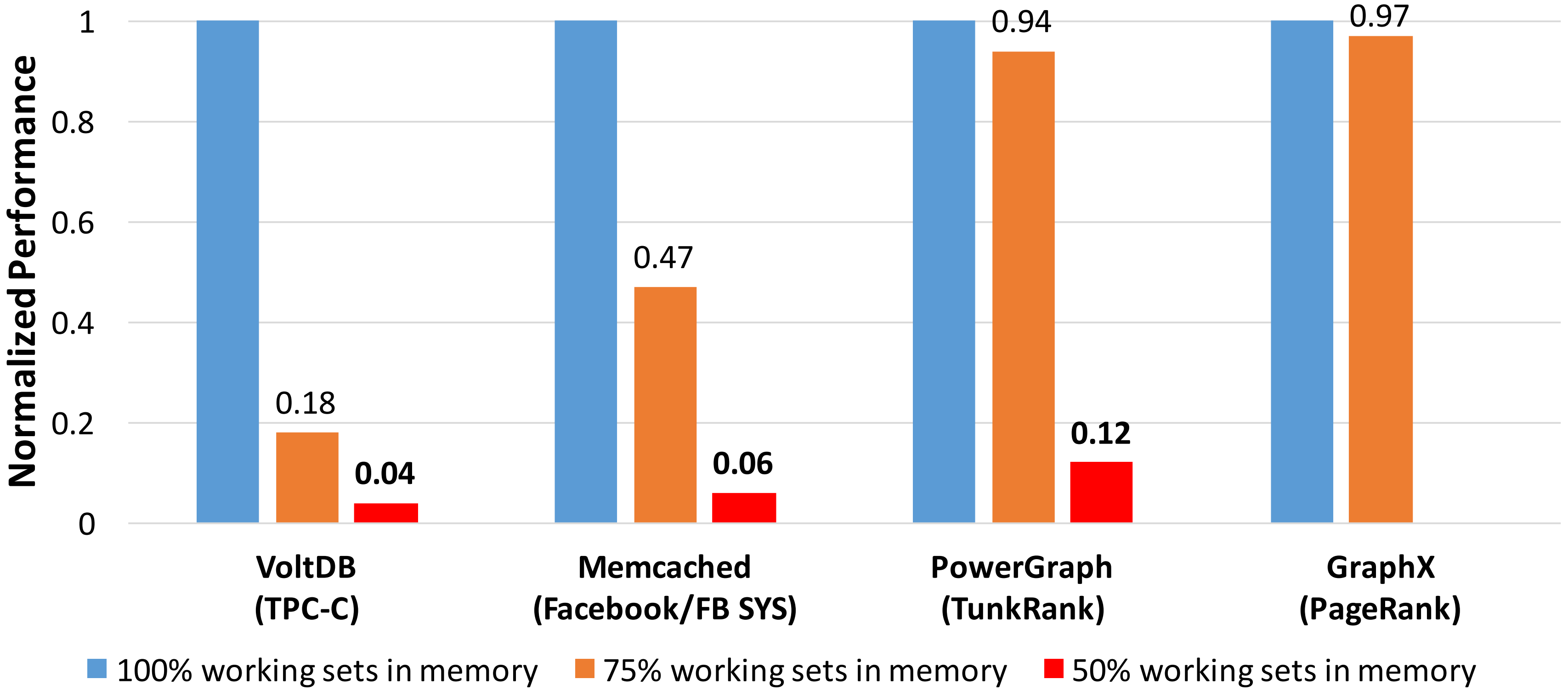
Performance degradation



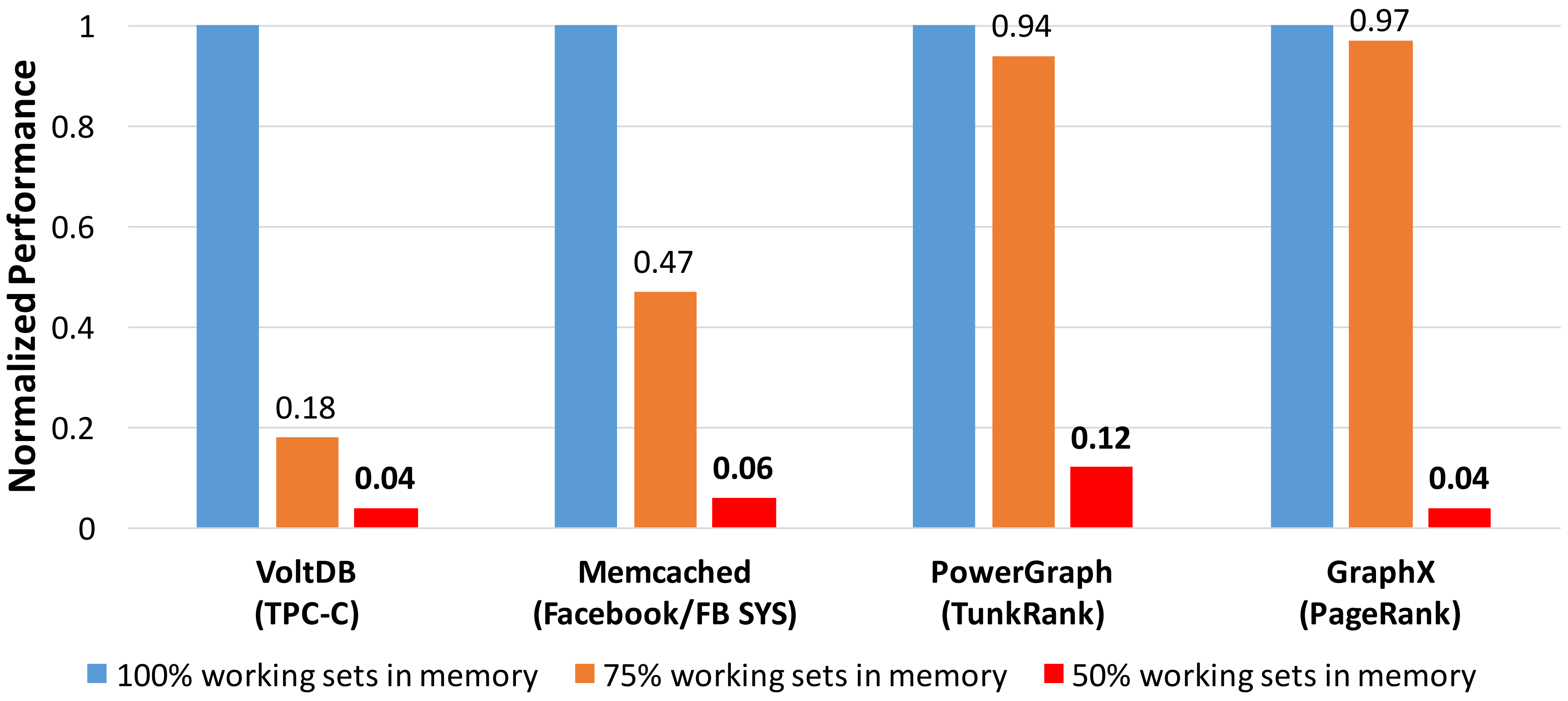
Performance degradation



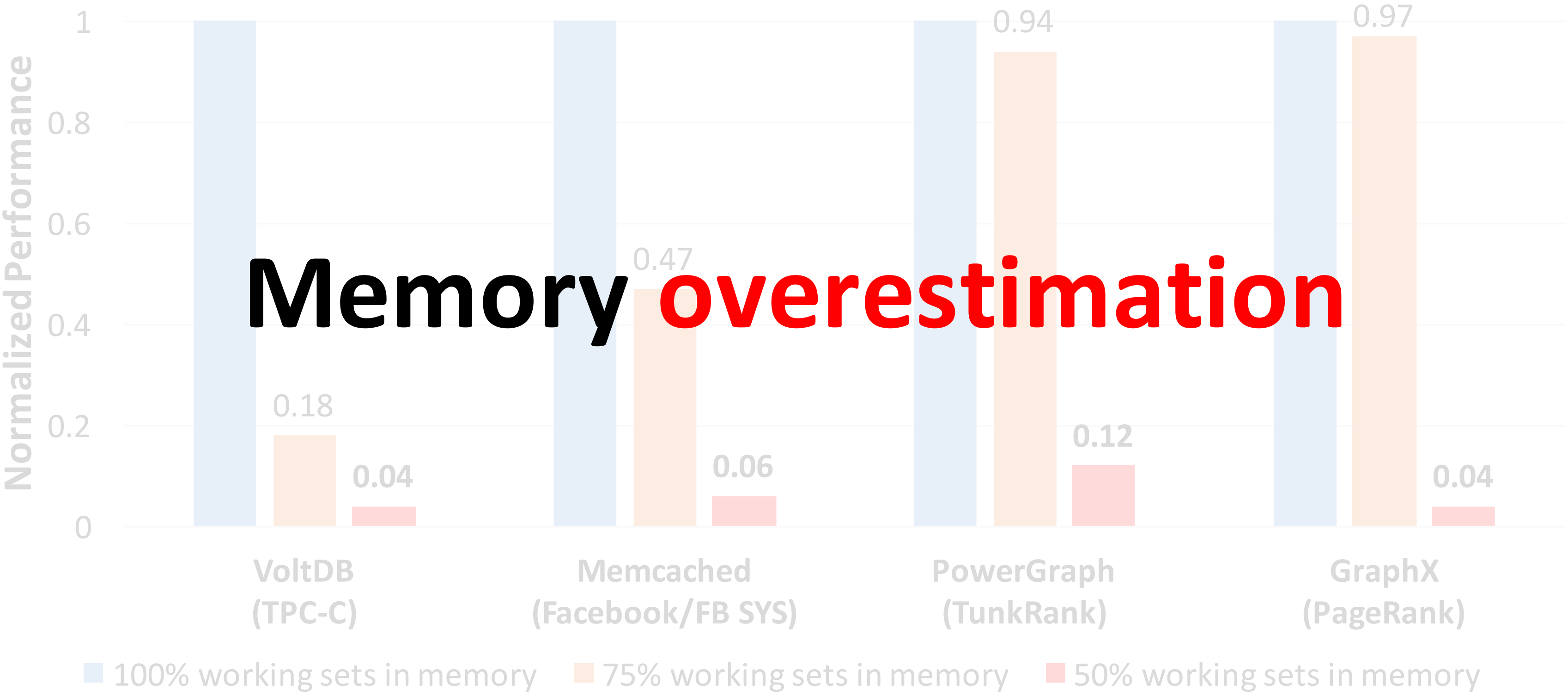
Performance degradation



Performance degradation



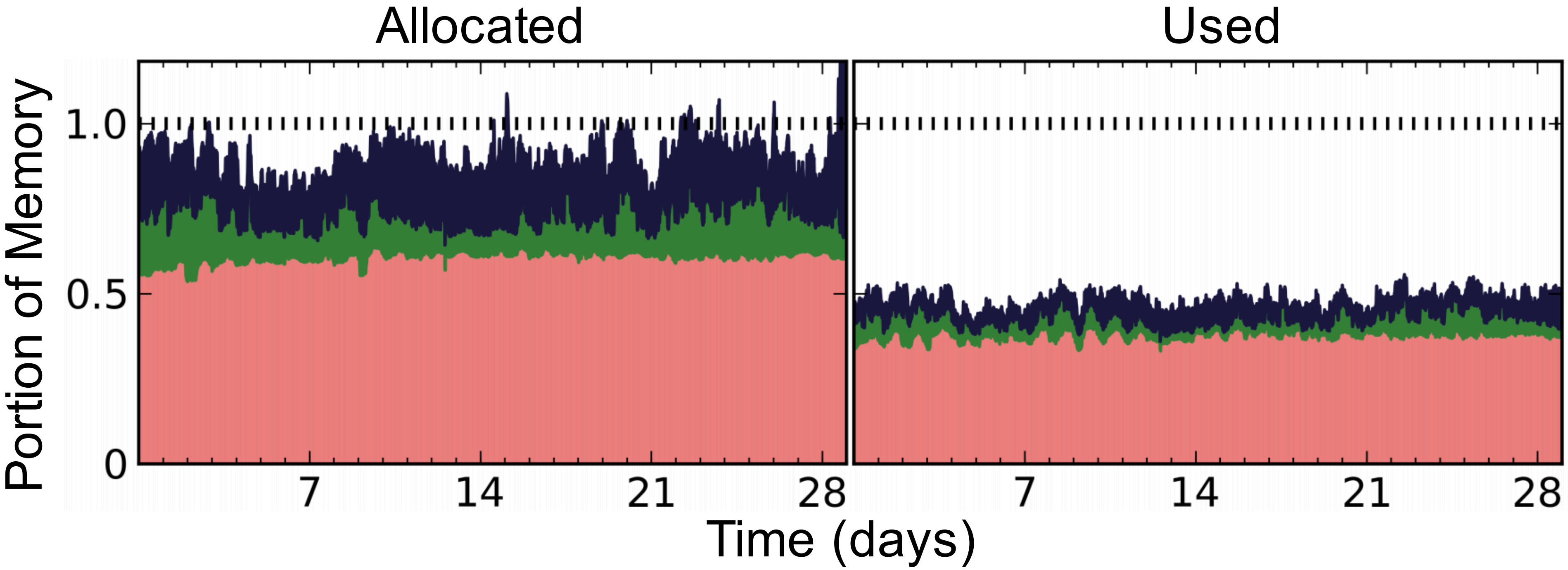
Performance degradation



Memory overestimation

Memory underutilization

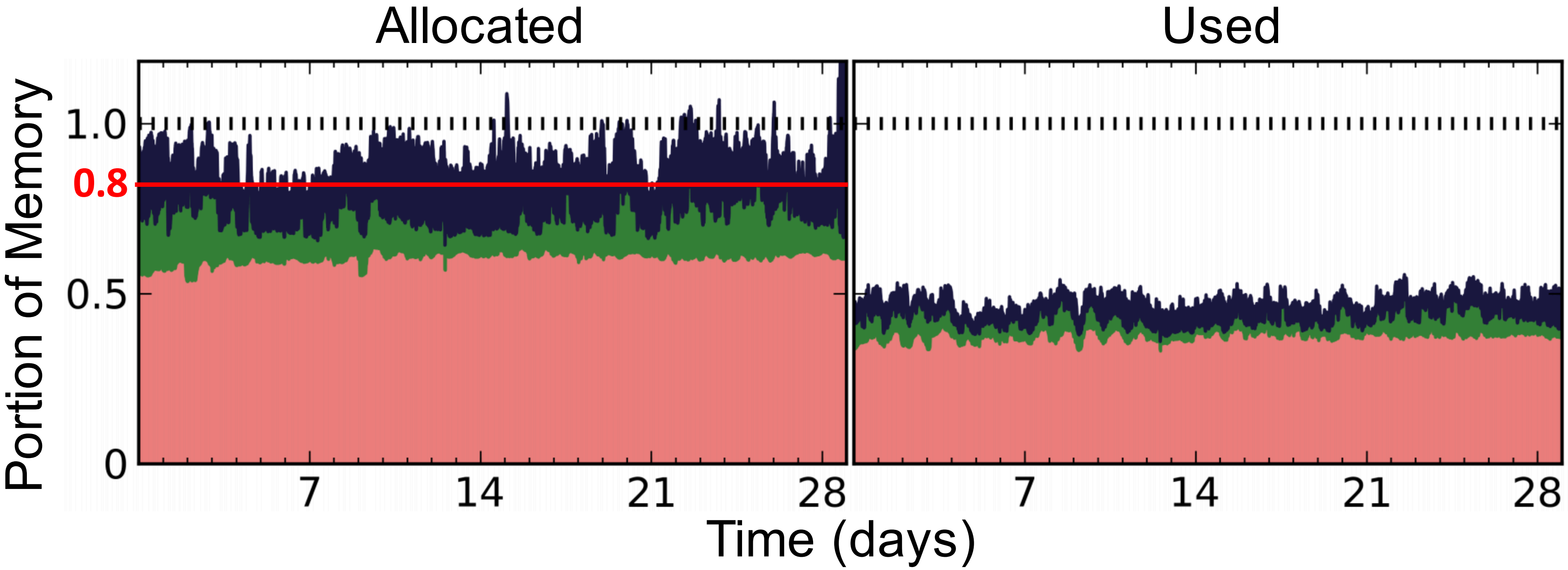
- Google Cluster Analysis[1]



[1] Reiss, Charles, et al. "Heterogeneity and dynamicity of clouds at scale: Google trace analysis." *SoCC'12*.

Memory underutilization

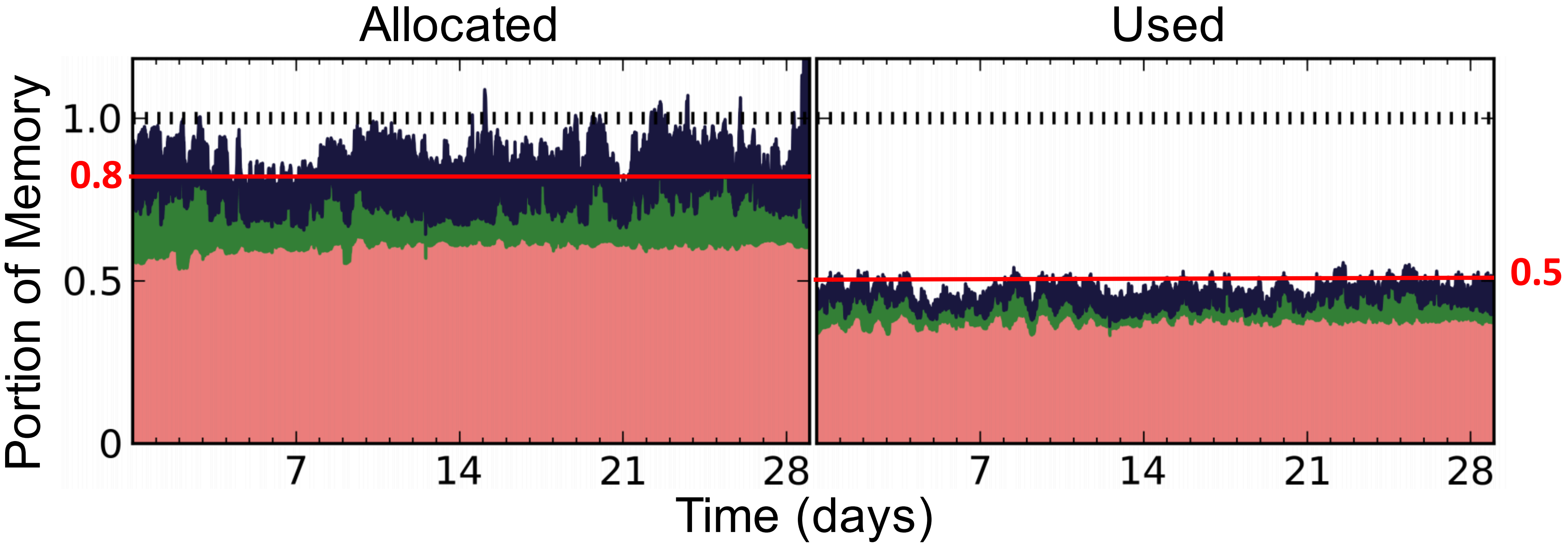
- Google Cluster Analysis[1]



[1] Reiss, Charles, et al. "Heterogeneity and dynamicity of clouds at scale: Google trace analysis." *SoCC'12*.

Memory underutilization

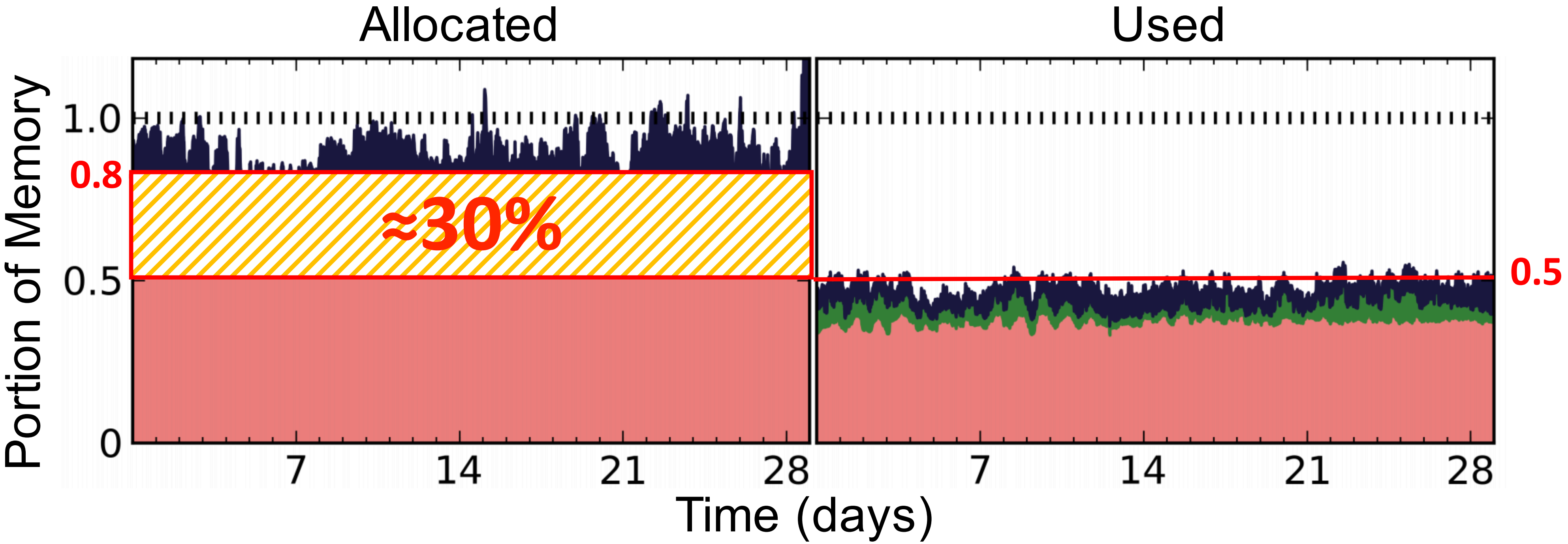
- Google Cluster Analysis[1]



[1] Reiss, Charles, et al. "Heterogeneity and dynamicity of clouds at scale: Google trace analysis." *SoCC'12*.

Memory underutilization

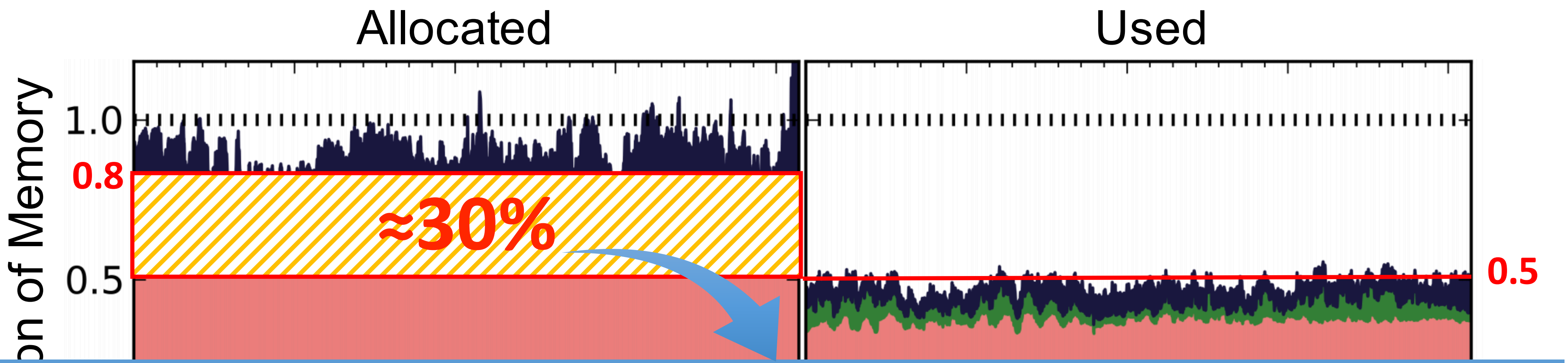
- Google Cluster Analysis[1]



[1] Reiss, Charles, et al. "Heterogeneity and dynamicity of clouds at scale: Google trace analysis." *SoCC'12*.

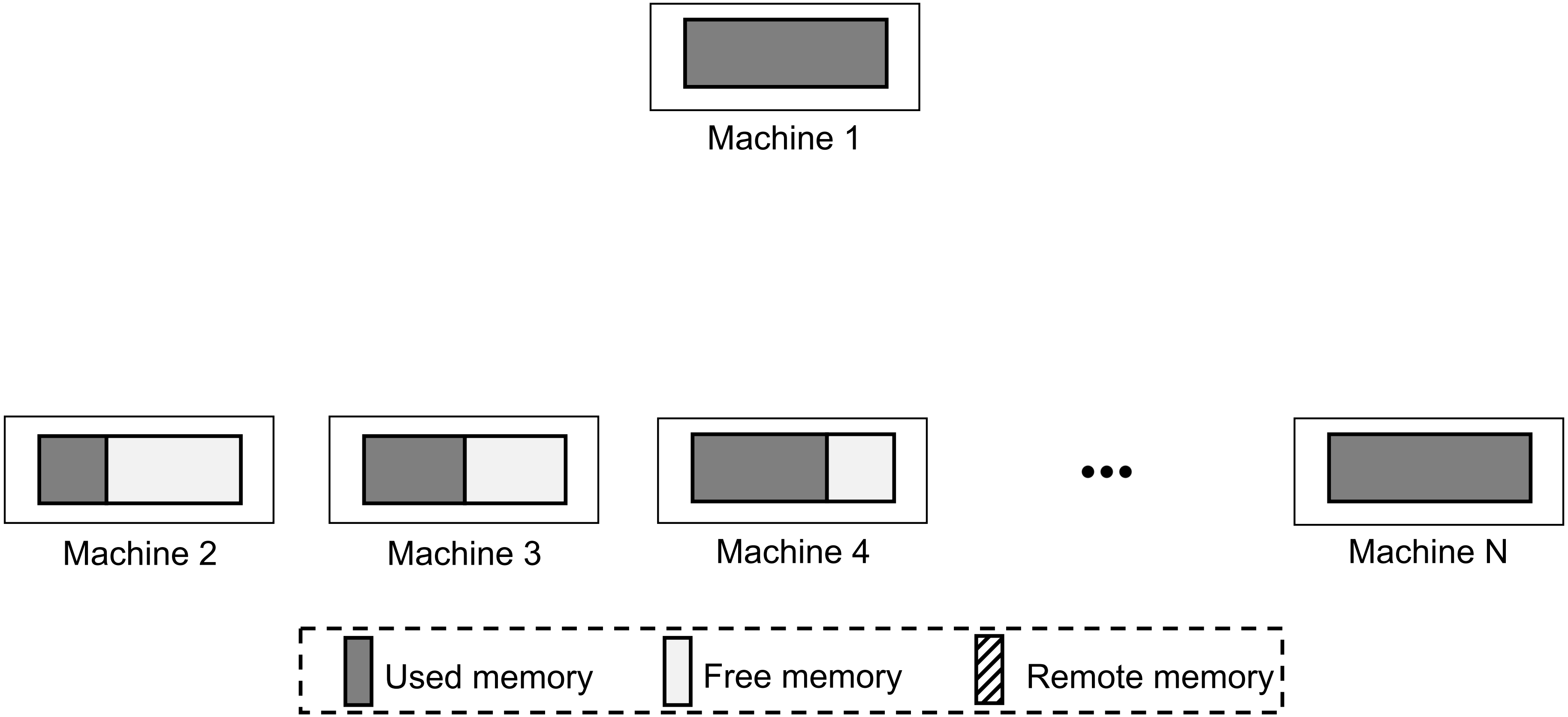
Memory underutilization

- Google Cluster Analysis[1]



Can we utilize this memory?

[1] Reiss, Charles, et al. "Heterogeneity and dynamicity of clouds at scale: Google trace analysis." *SoCC'12*.



Machine 1

Machine 2

Machine 3

Machine 4

•••

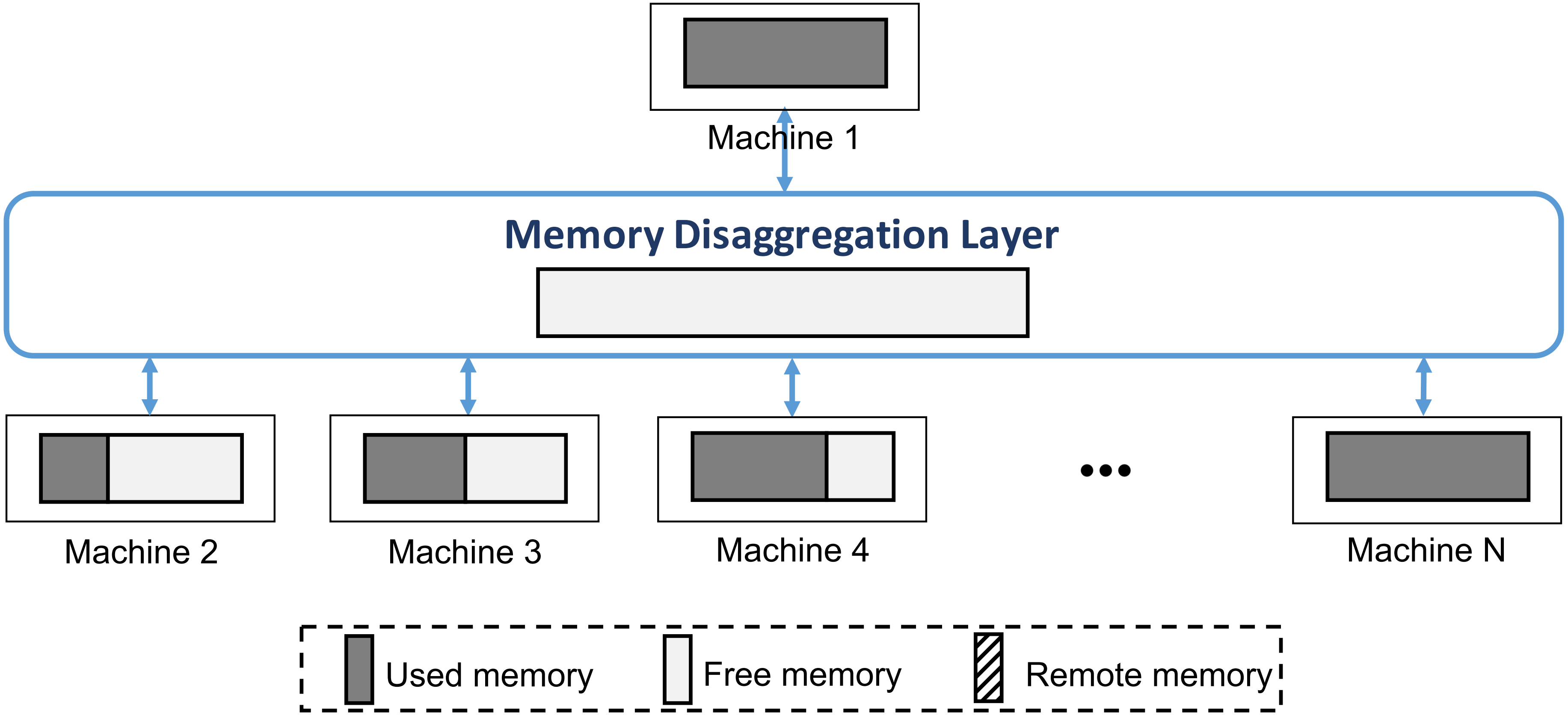
Machine N

Used memory

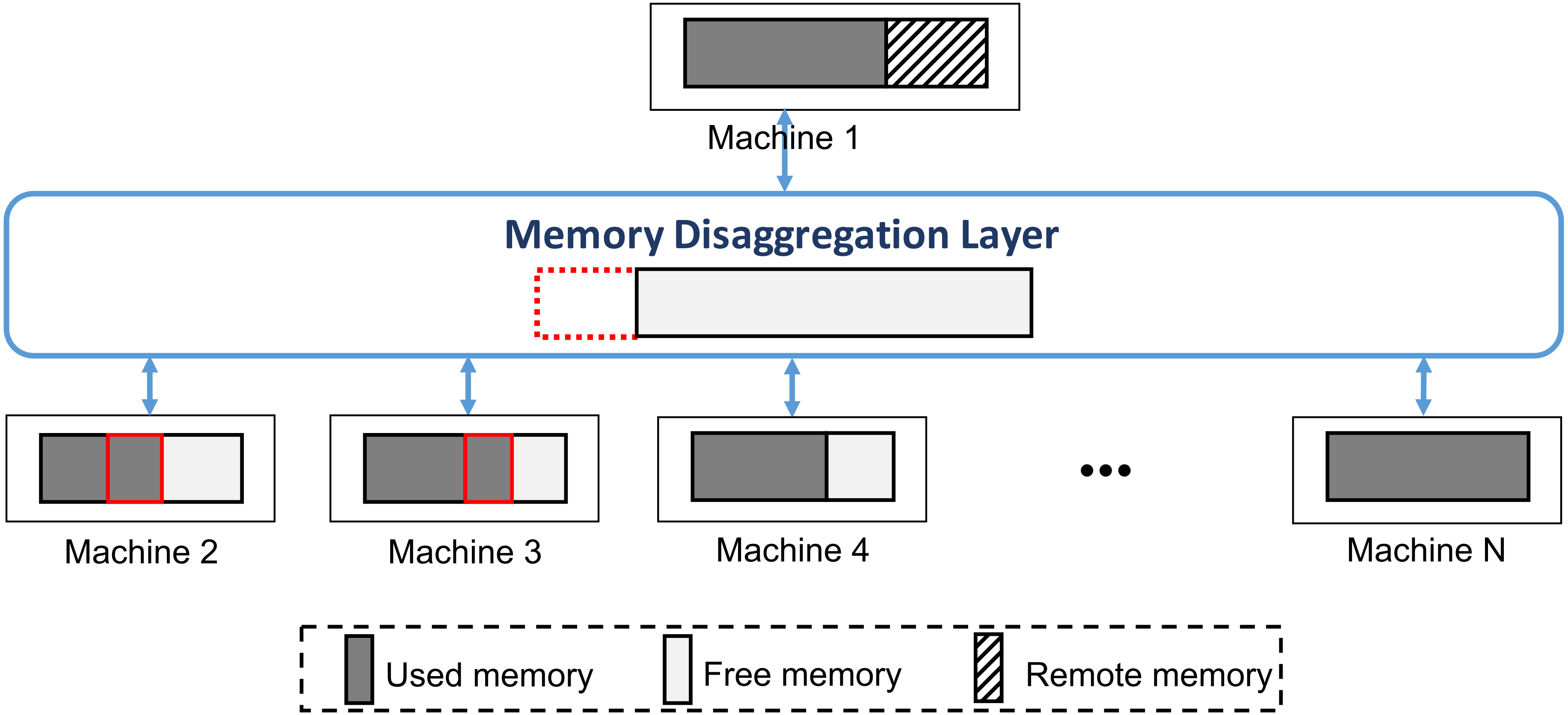
Free memory

Remote memory

Disaggregate free memory



Disaggregate free memory



What are the challenges?

- Minimize deployment overhead
  - No hardware design
  - No application modification
- Tolerate failures
  - e.g. network disconnection, machine crash
- Manage remote memory at scale

Recent work on memory disaggregation

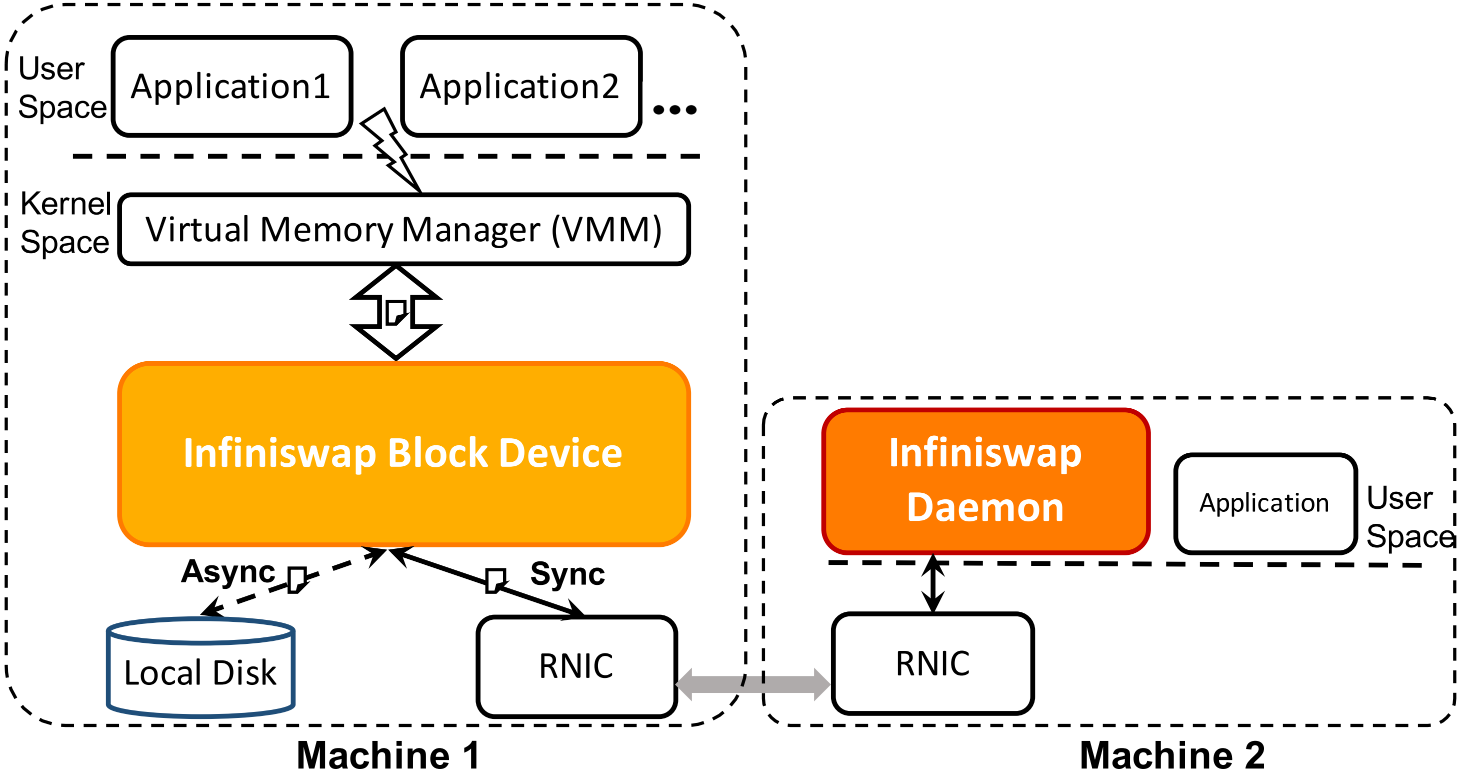
	No HW design	No app modification	Fault-tolerance	Scalability
Memory Blade[ISCA'09]	✗	✓	✓	✓
HPBD[CLUSTER'05] / **NBDX**[1]	✓	✓	✗	✗
RDMA key-value service (e.g. HERD[SIGCOMM'14], FaRM[NSDI'14])	✓	✗	✓	✓
Intel Rack Scale Architecture (RSA)[2]	✗	✓	✓	✓
Infiniswap	✓	✓	✓	✓

[1] https://github.com/accelio/NBDX
[2] http://www.intel.com/content/www/us/en/architecture-and-technology/rack-scale-design-overview.html

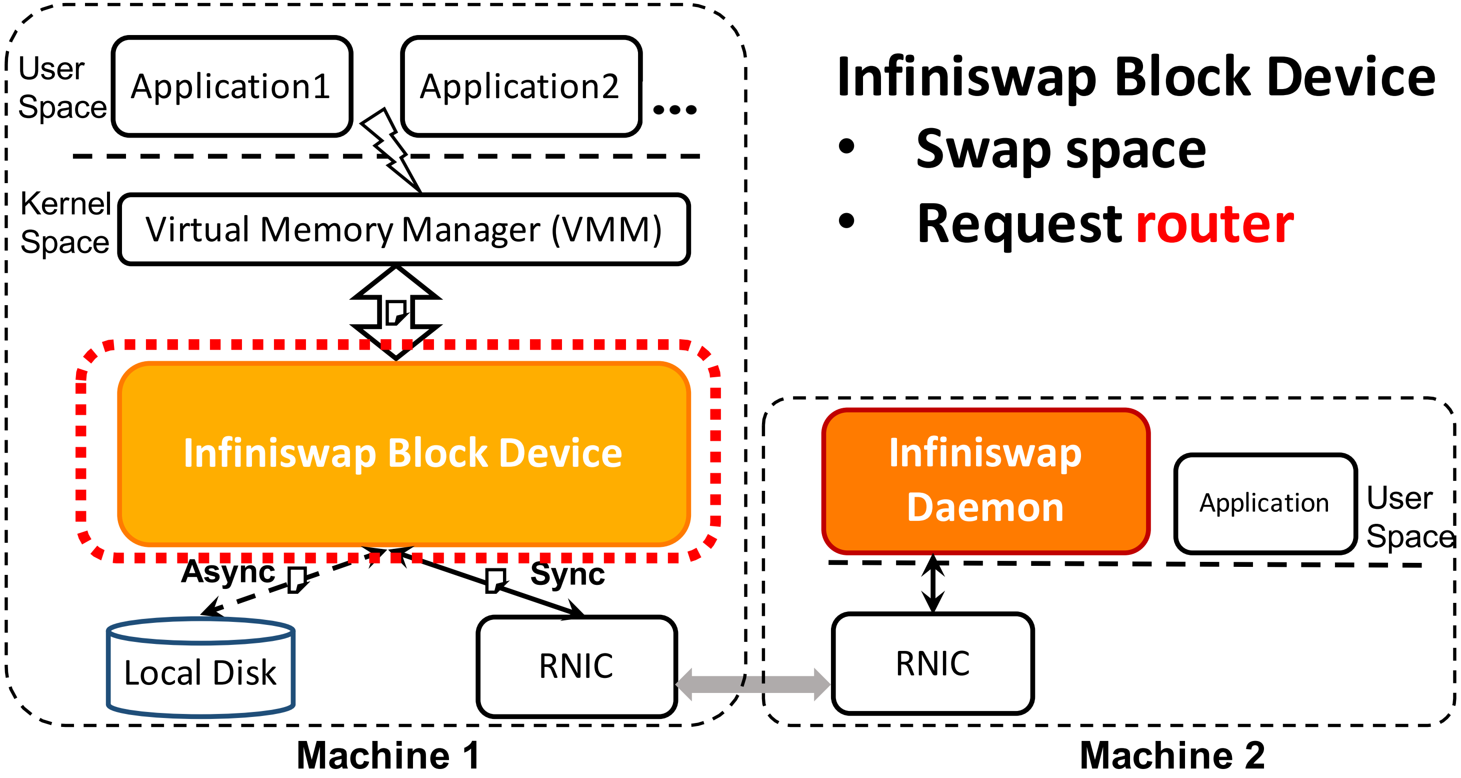
Agenda

- Motivation and related work
- **Design and system overview**
- Implementation and evaluation
- Future work and conclusion

System Overview



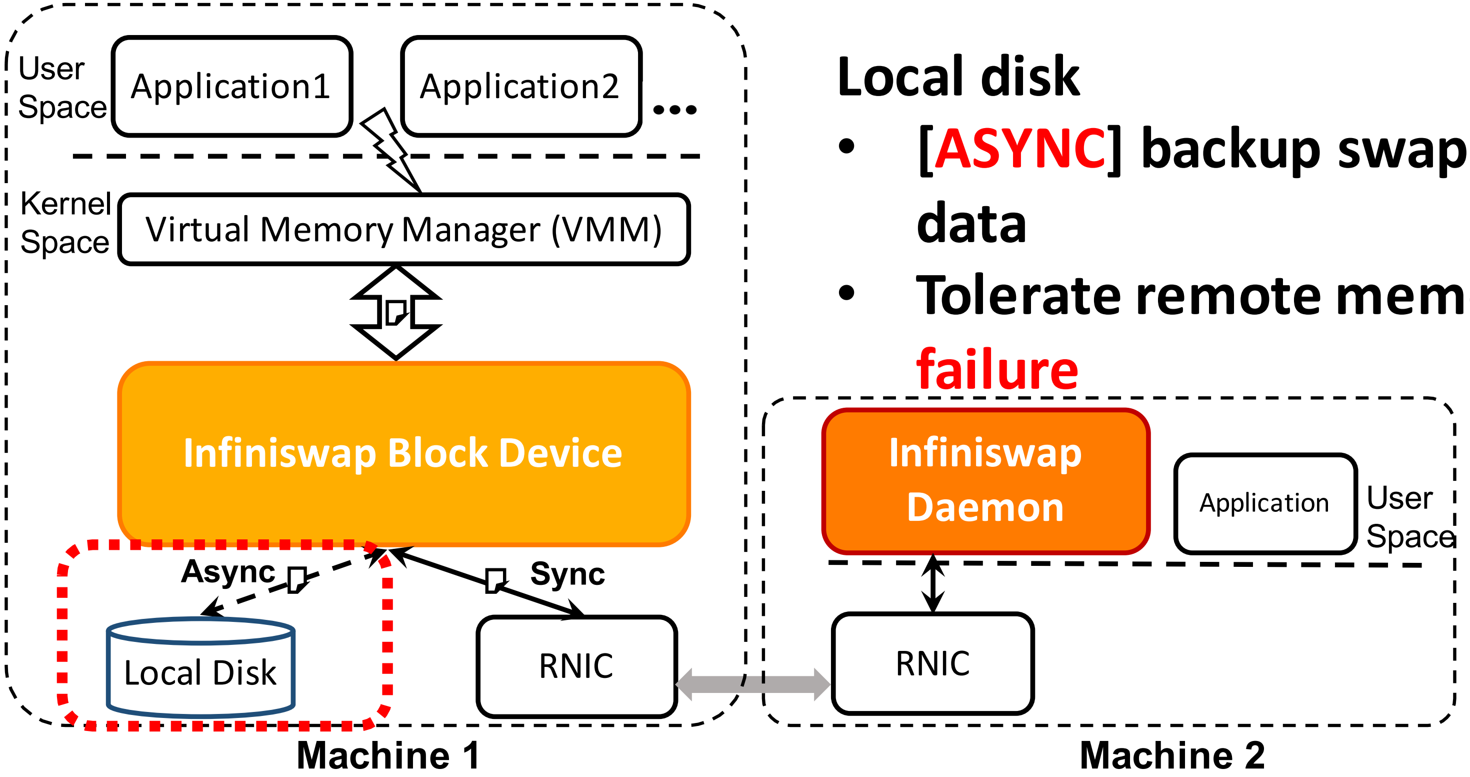
System Overview



Infiniswap Block Device

- **Swap space**
- **Request router**

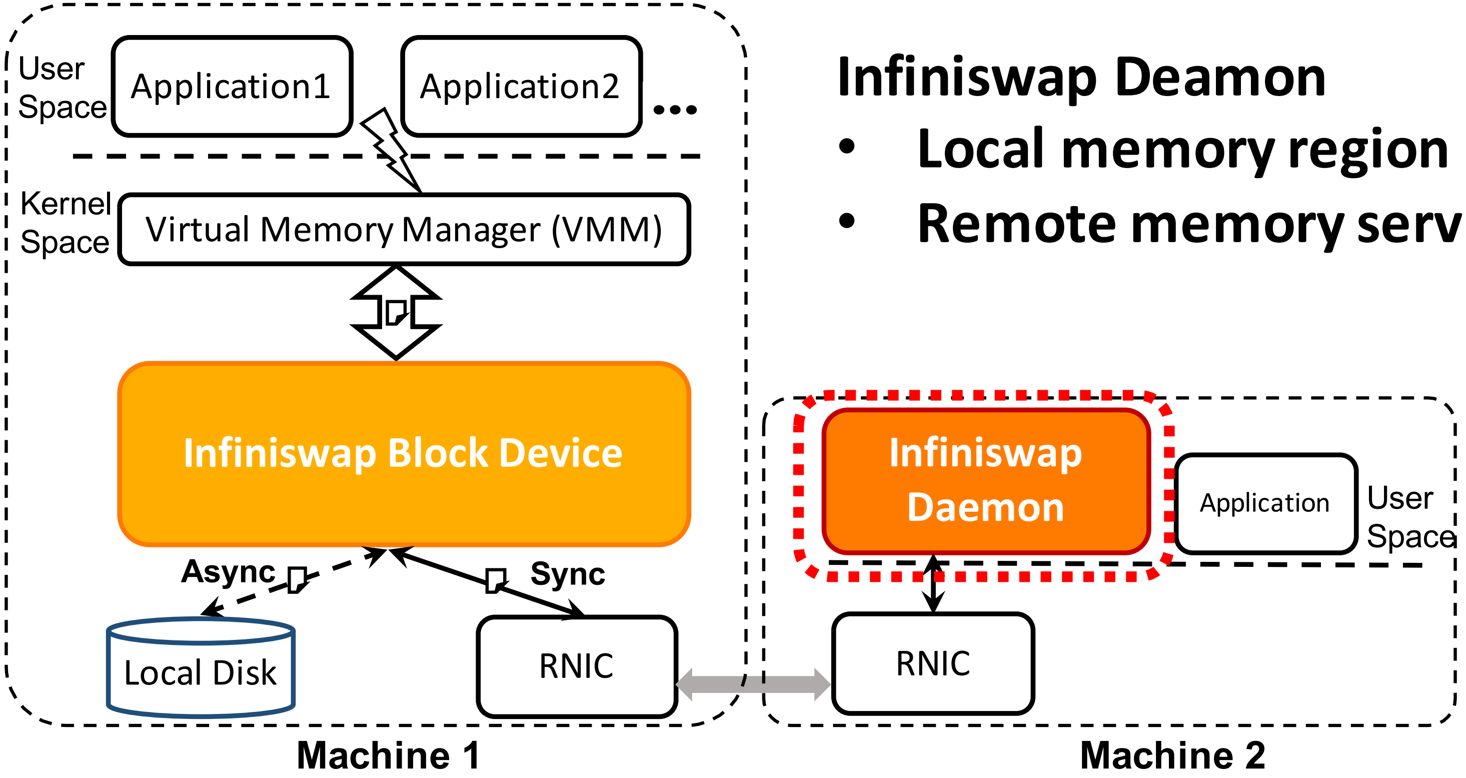
System Overview



Local disk

- **[ASYNC] backup swapped-out data**
- **Tolerate remote memory failure**

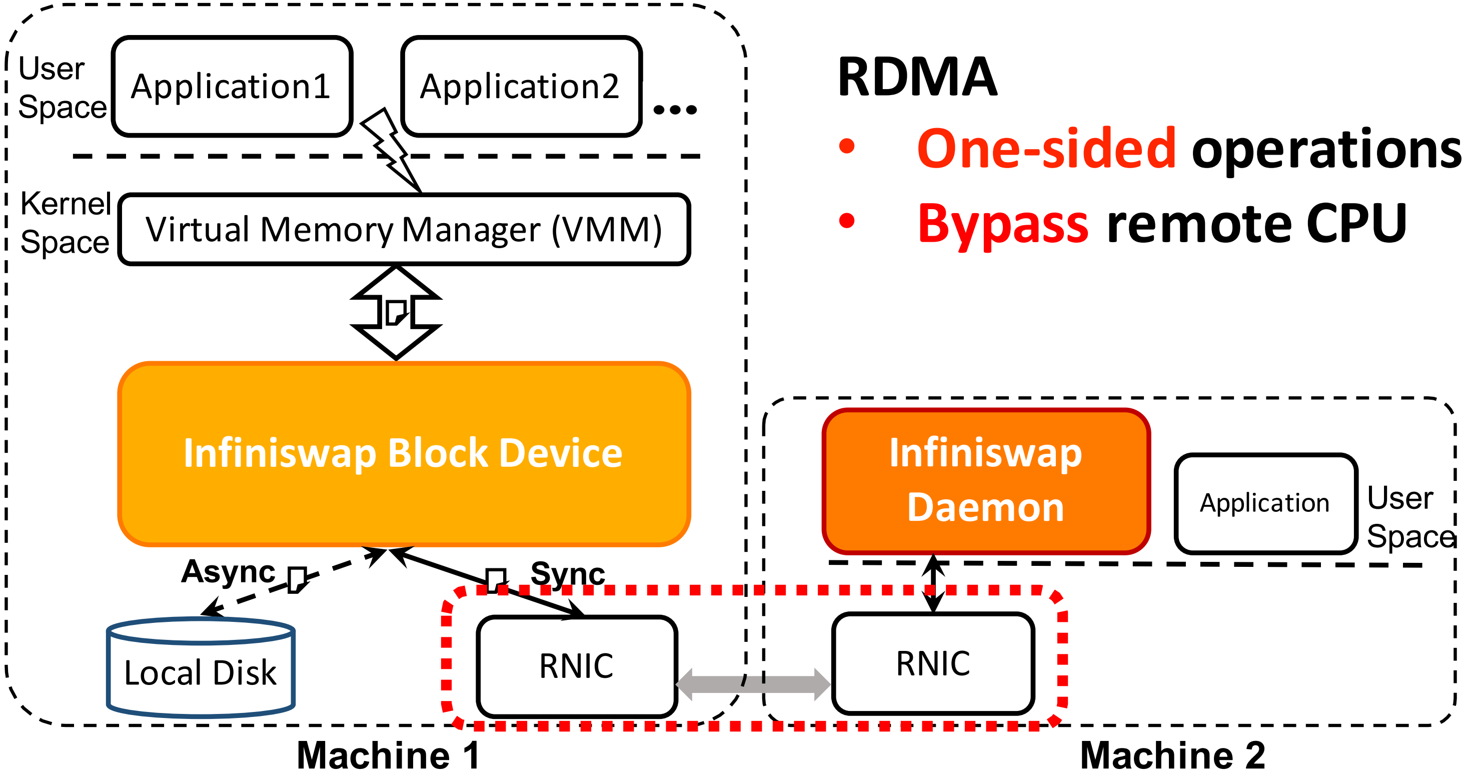
System Overview



Infiniswap Deamon

- **Local memory region**
- **Remote memory service**

System Overview



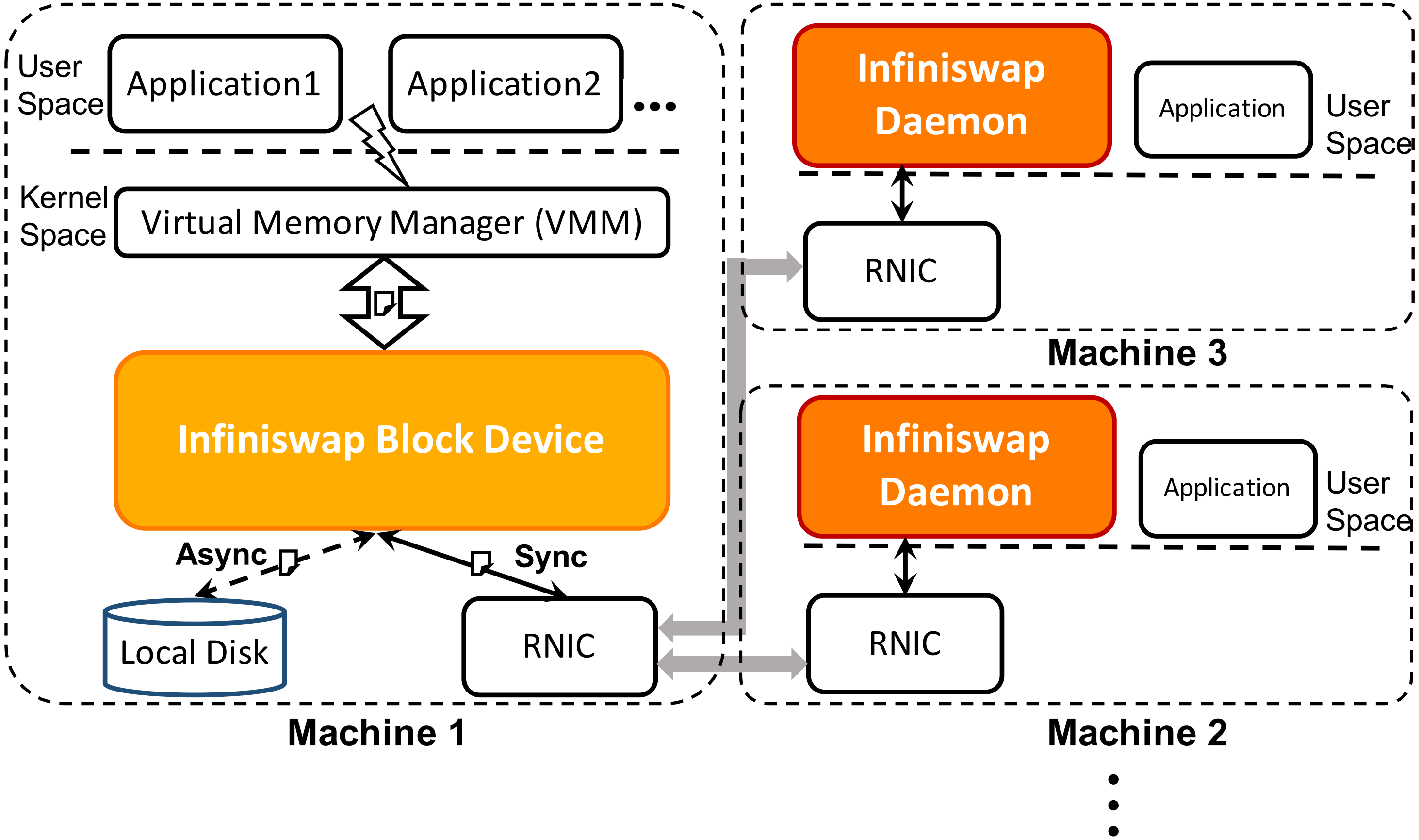
RDMA

- **One-sided operations**
- **Bypass remote CPU**

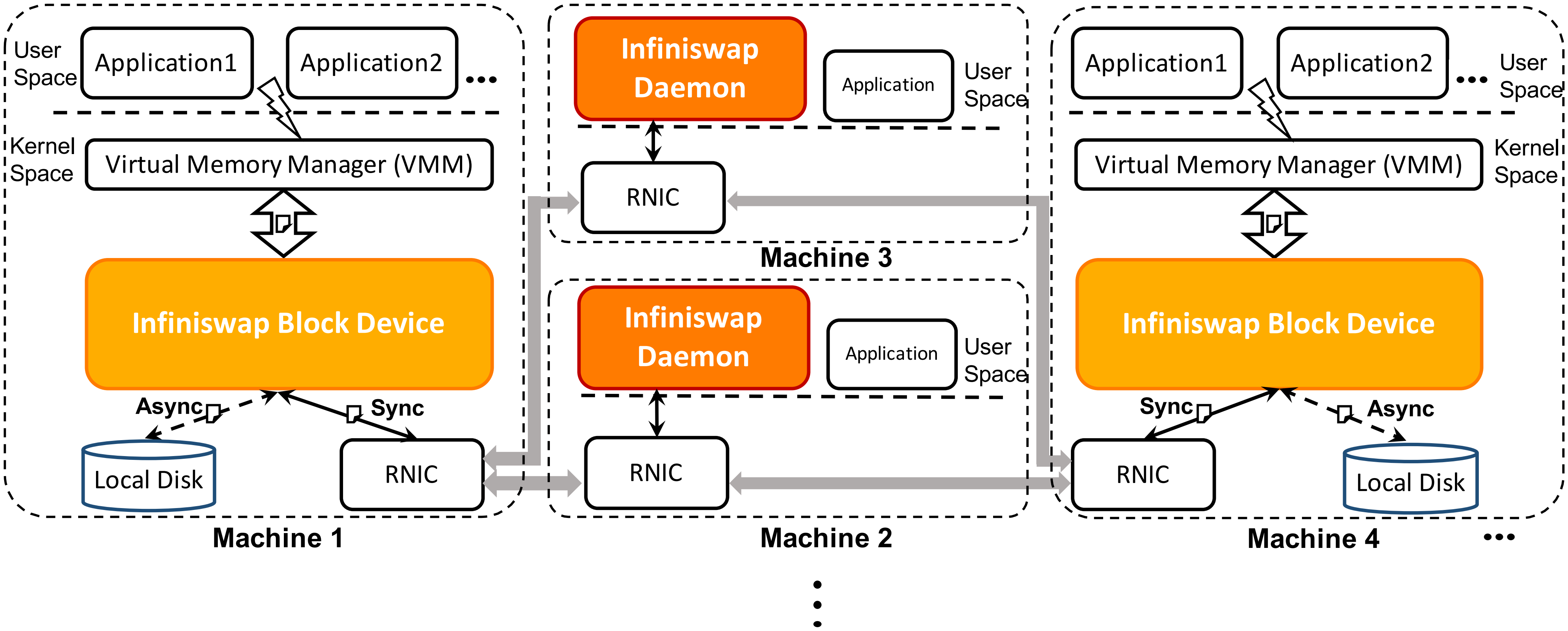
How to meet the design objectives?

Objectives	Ideas
No hardware design	Remote paging
No application modification	
Fault-tolerance	Local backup disk

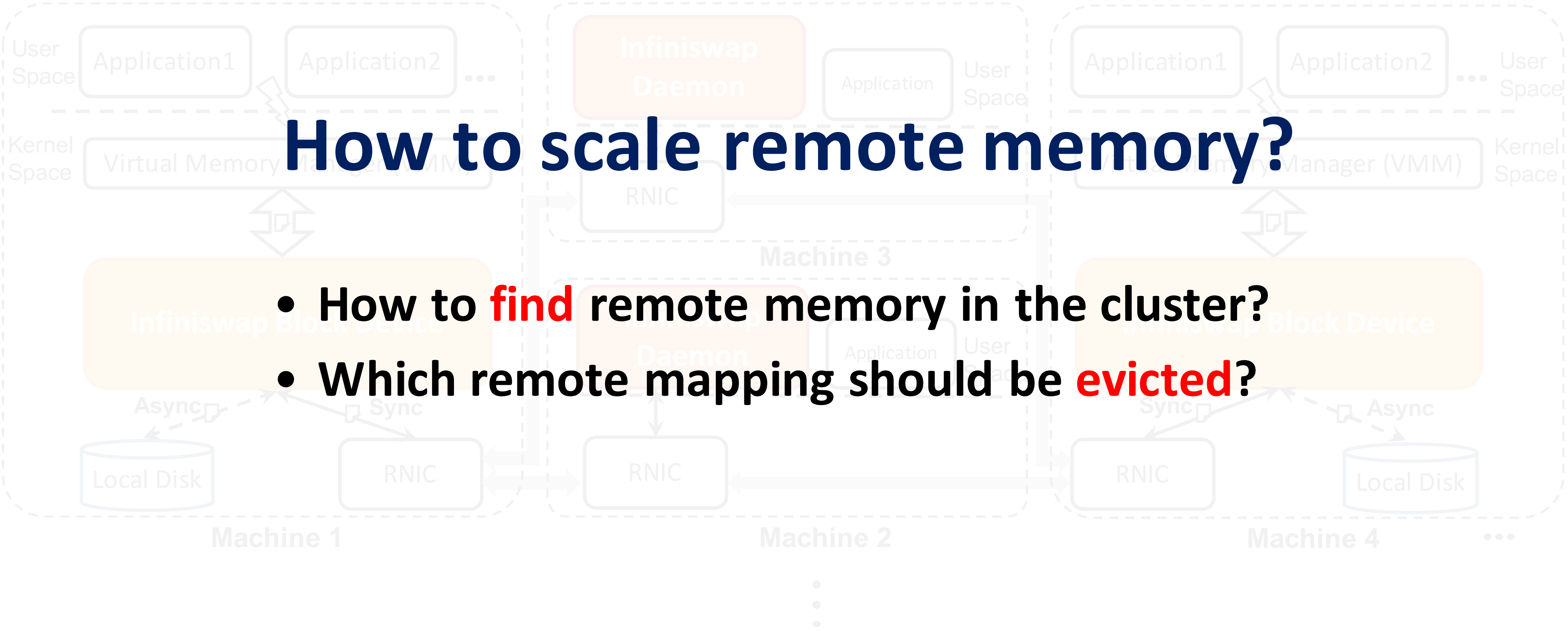
One-to-many


User Space
Application1
Application2
...
Kernel Space
Virtual Memory Manager (VMM)
Infiniswap Block Device
Async
Sync
Local Disk
RNIC
Machine 1
Infiniswap Daemon
Application
User Space
RNIC
Machine 3
Infiniswap Daemon
Application
User Space
RNIC
Machine 2

Many-to-many


User Space
Application1
Application2
...
Kernel Space
Virtual Memory Manager (VMM)
Infiniswap Block Device
Async
Sync
Local Disk
RNIC
Machine 1
Infiniswap Daemon
Application
User Space
RNIC
Machine 3
Infiniswap Daemon
Application
User Space
RNIC
Machine 2
Application1
Application2
...
User Space
Kernel Space
Virtual Memory Manager (VMM)
Infiniswap Block Device
Sync
Async
RNIC
Local Disk
Machine 4
...

Many-to-many



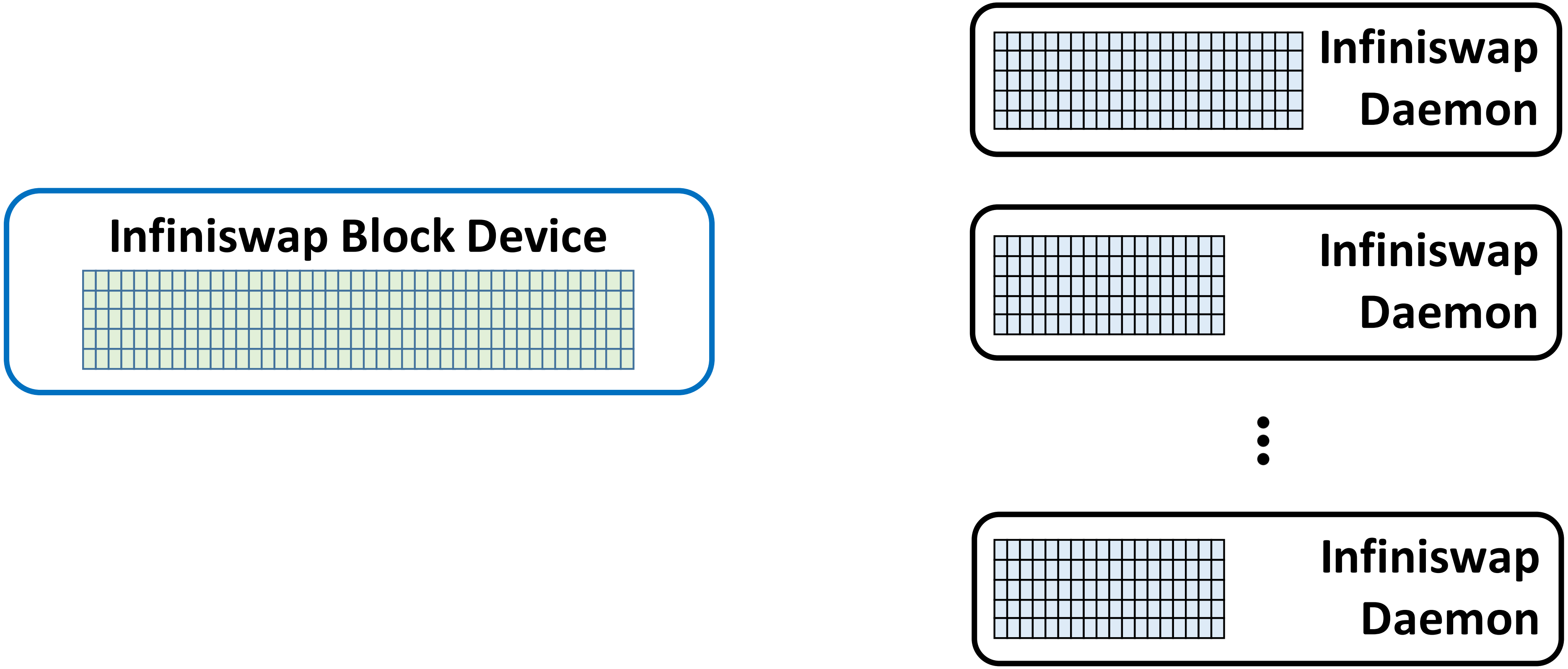
How to scale remote memory?

- How to **find** remote memory in the cluster?
- Which remote mapping should be **evicted**?

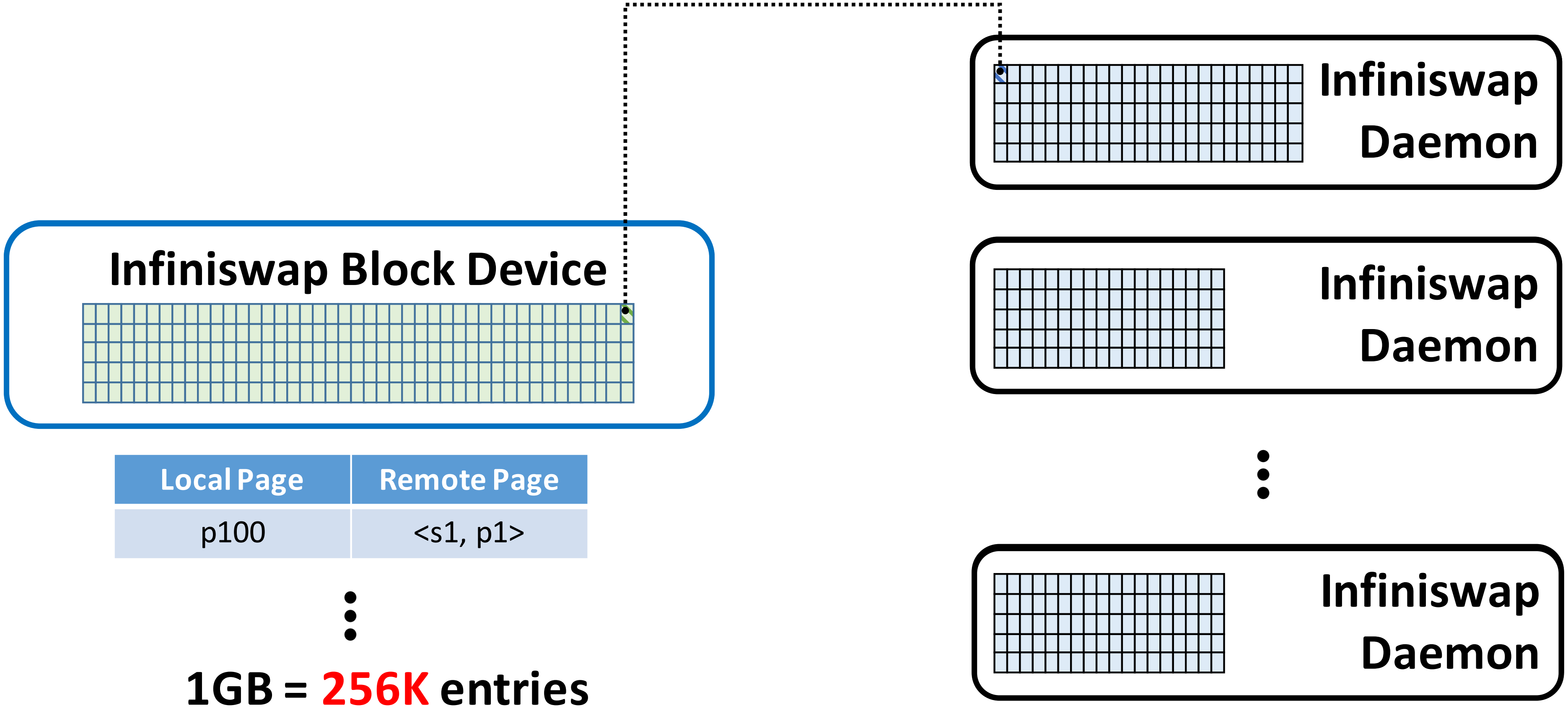
How to meet the design objectives?

Objectives	Ideas
No hardware design	Remote paging
No application modification	
Fault-tolerance	Local backup disk
Scalability	**Decentralized** remote memory management

Management unit: memory page?



Management unit: memory page?



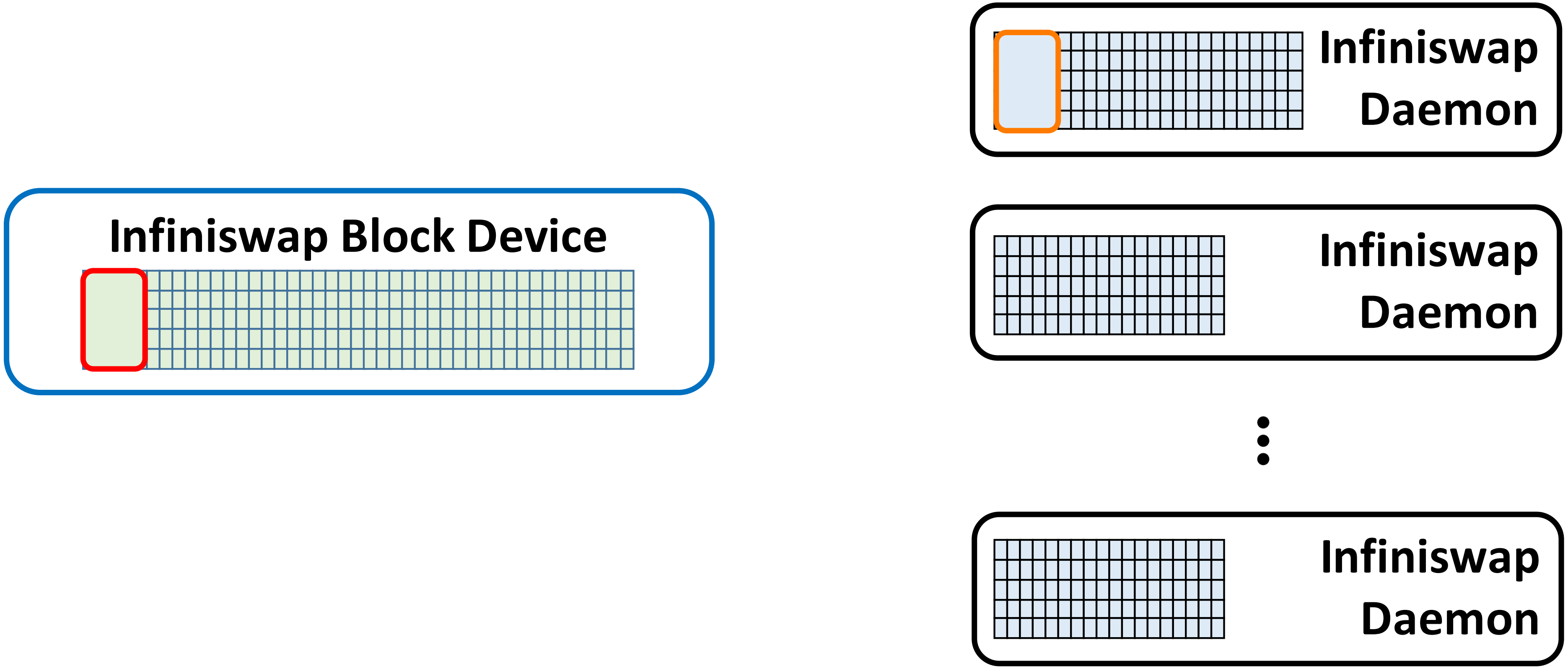
Local Page	Remote Page
p100	<s1, p1>

⋮

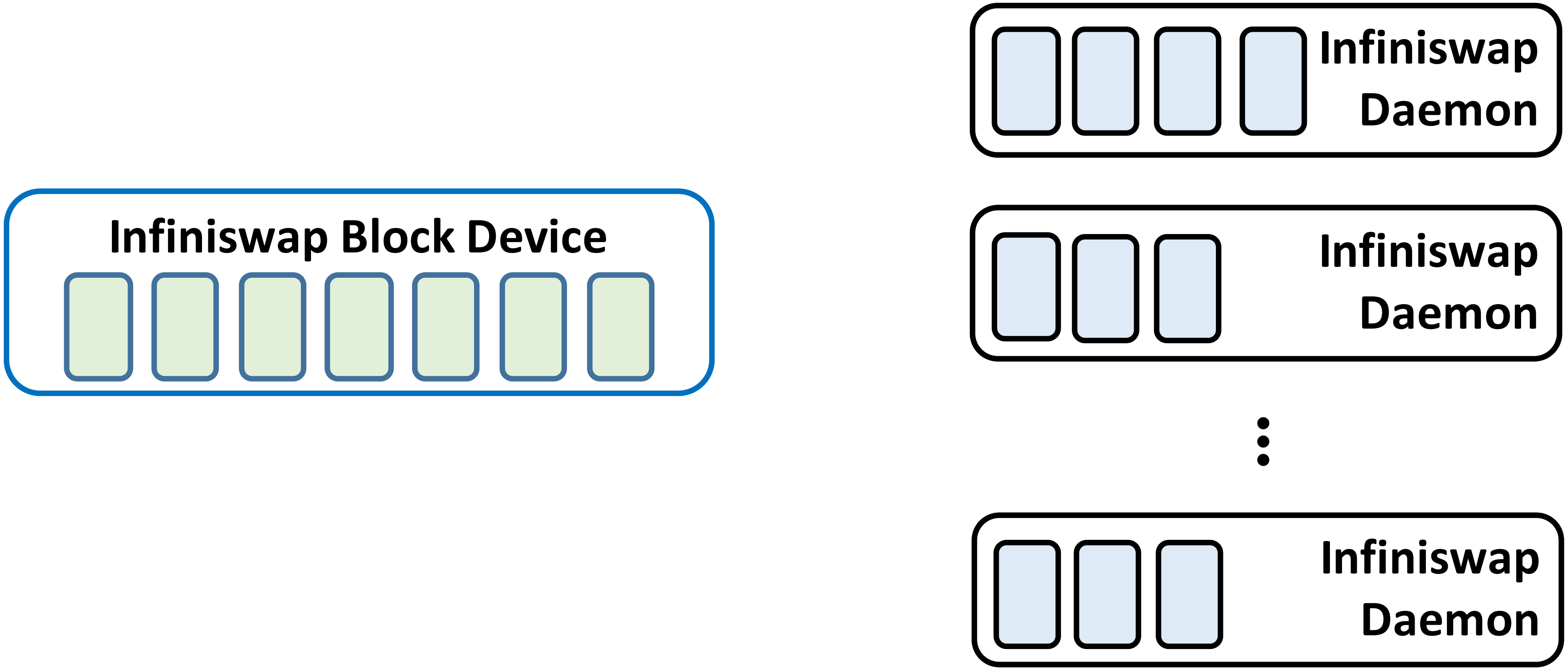
1GB = 256K entries

1GB = 256K RTTs

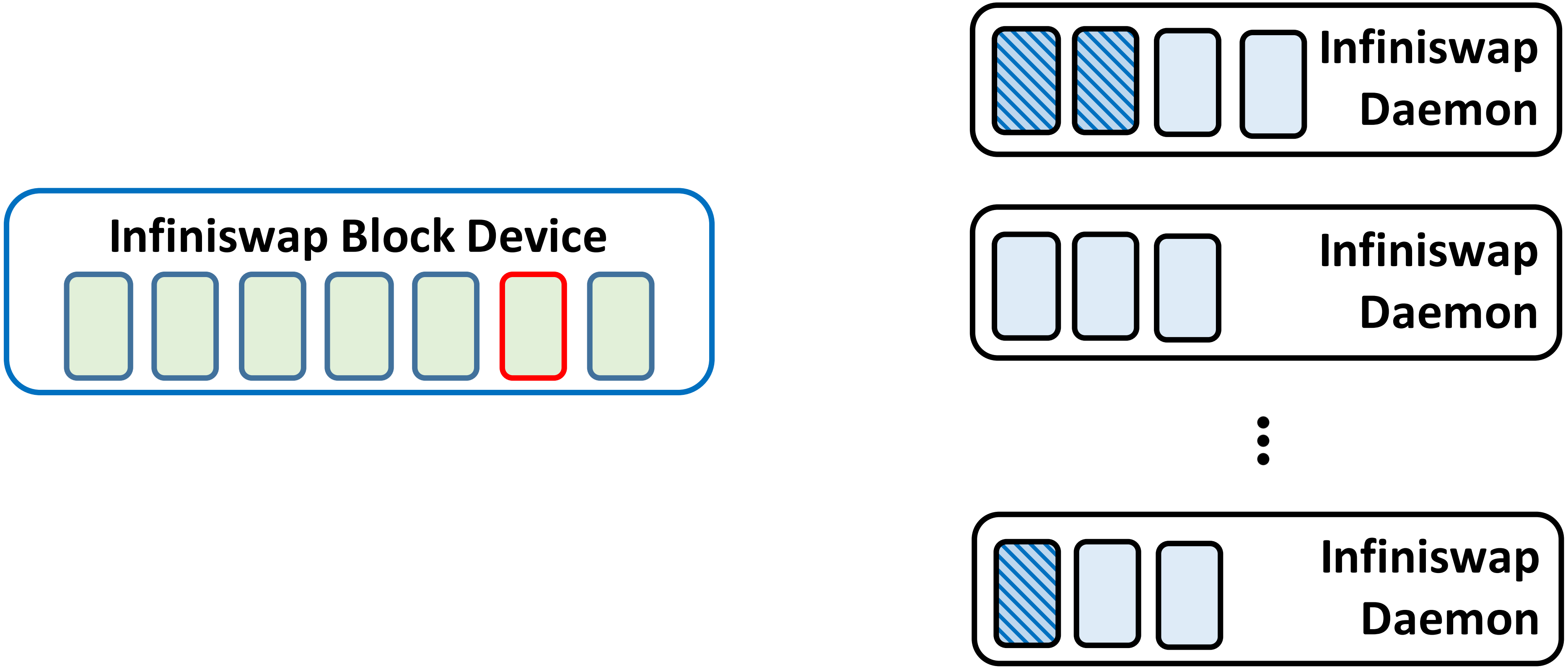
Management unit: memory slab!



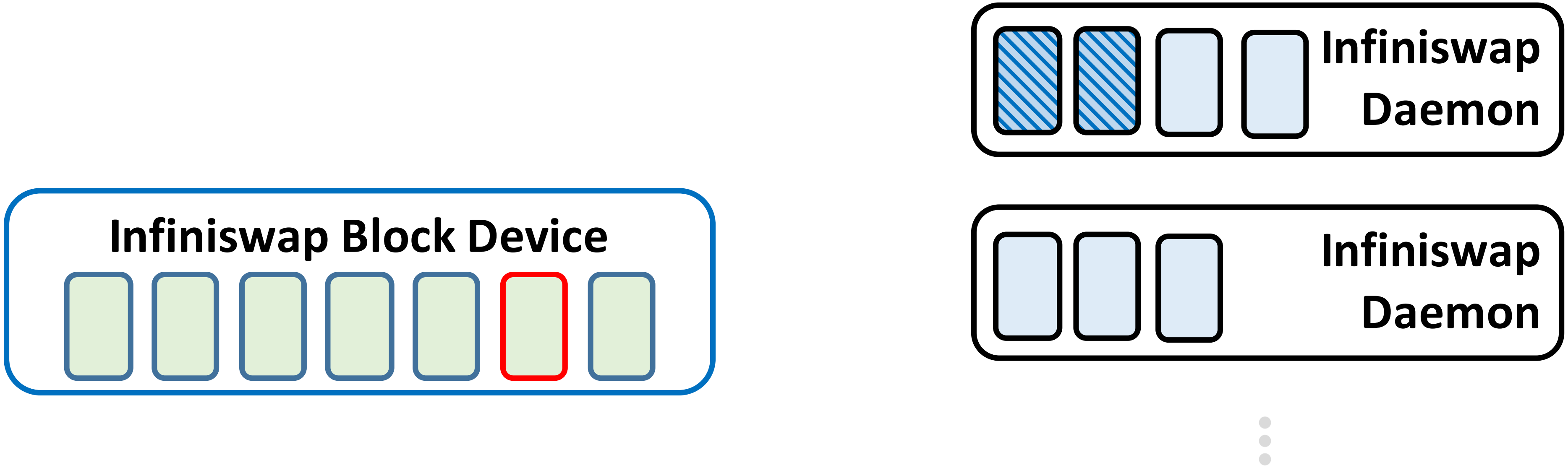
Management unit: memory slab!



Which remote machine should be selected?

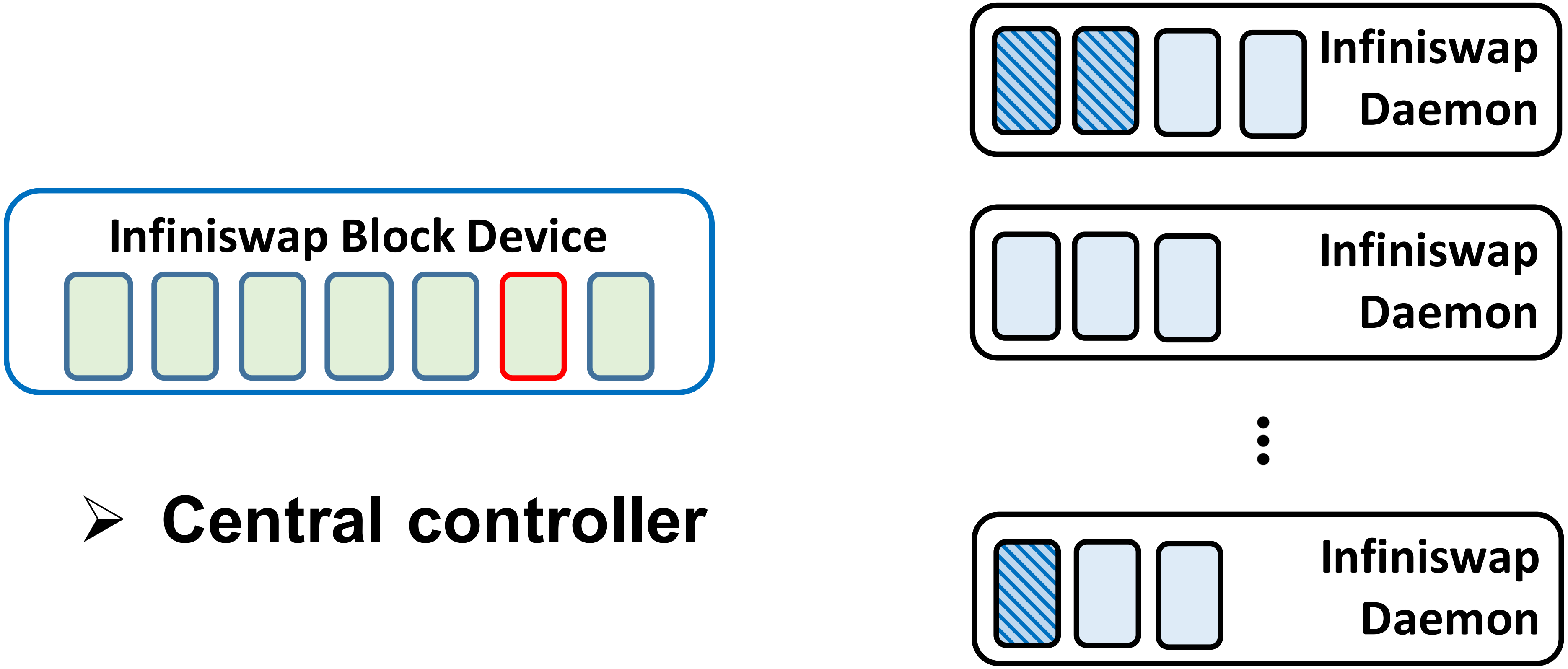


Which remote machine should be selected?



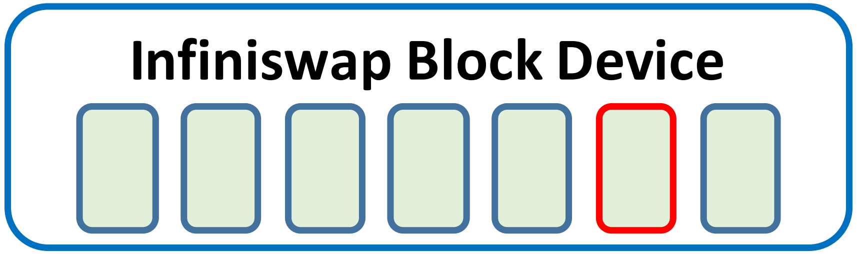
Goal: balance memory utilization

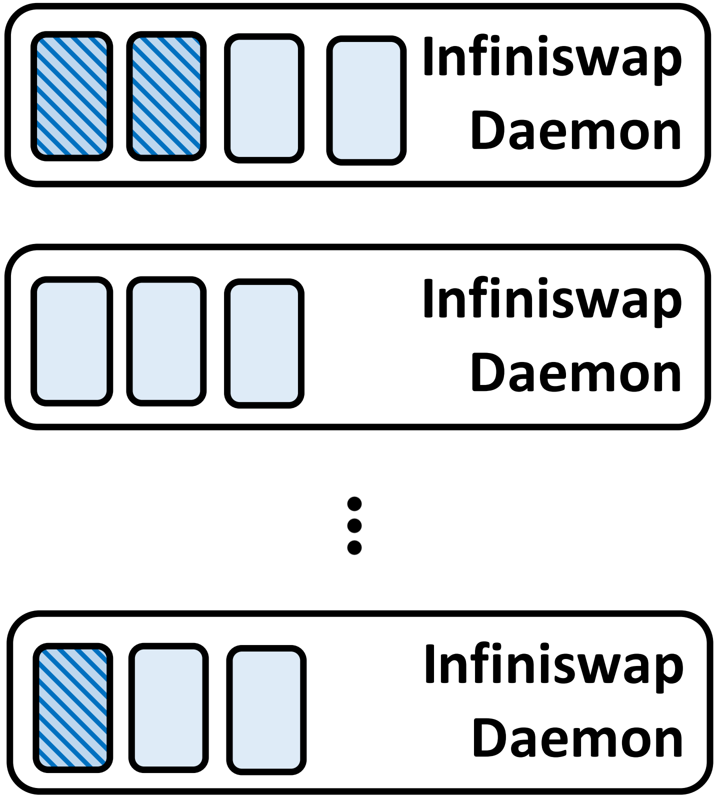
Which remote machine should be selected?



- Central controller

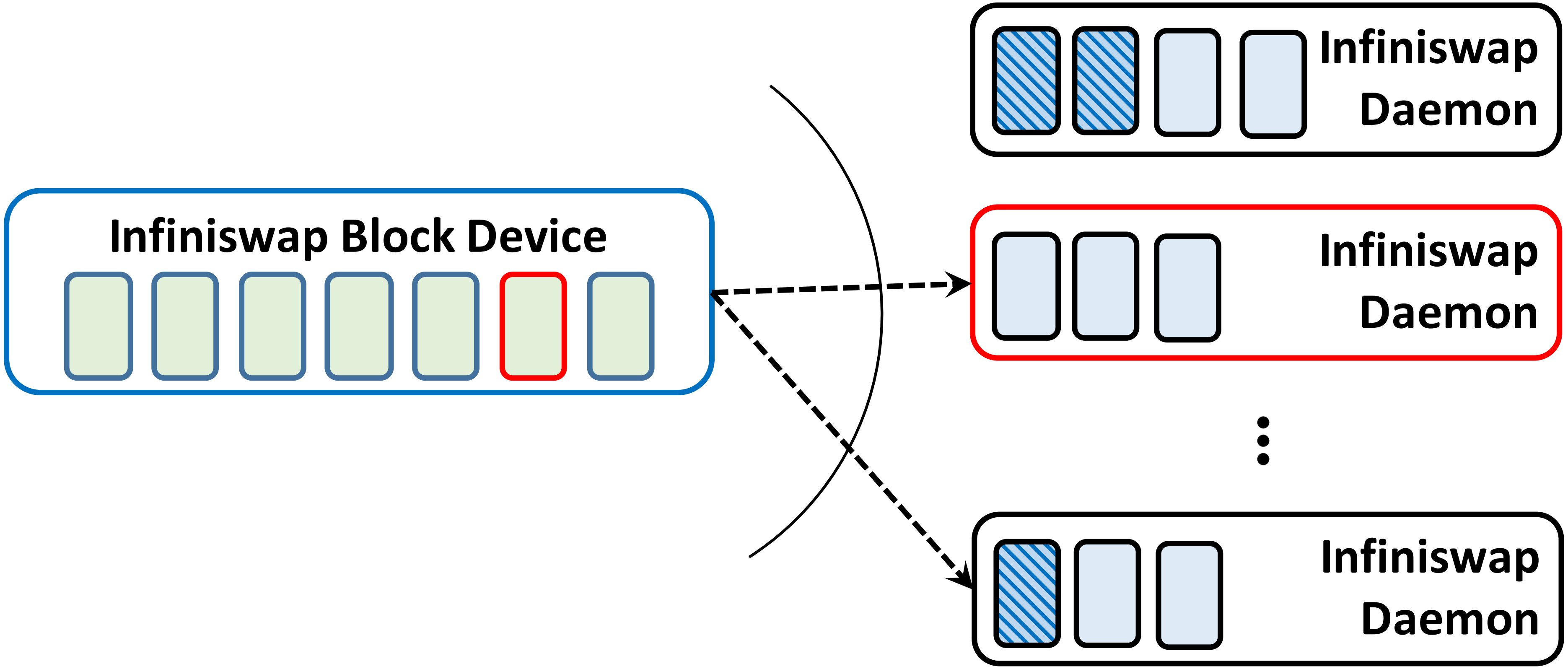
Which remote machine should be selected?





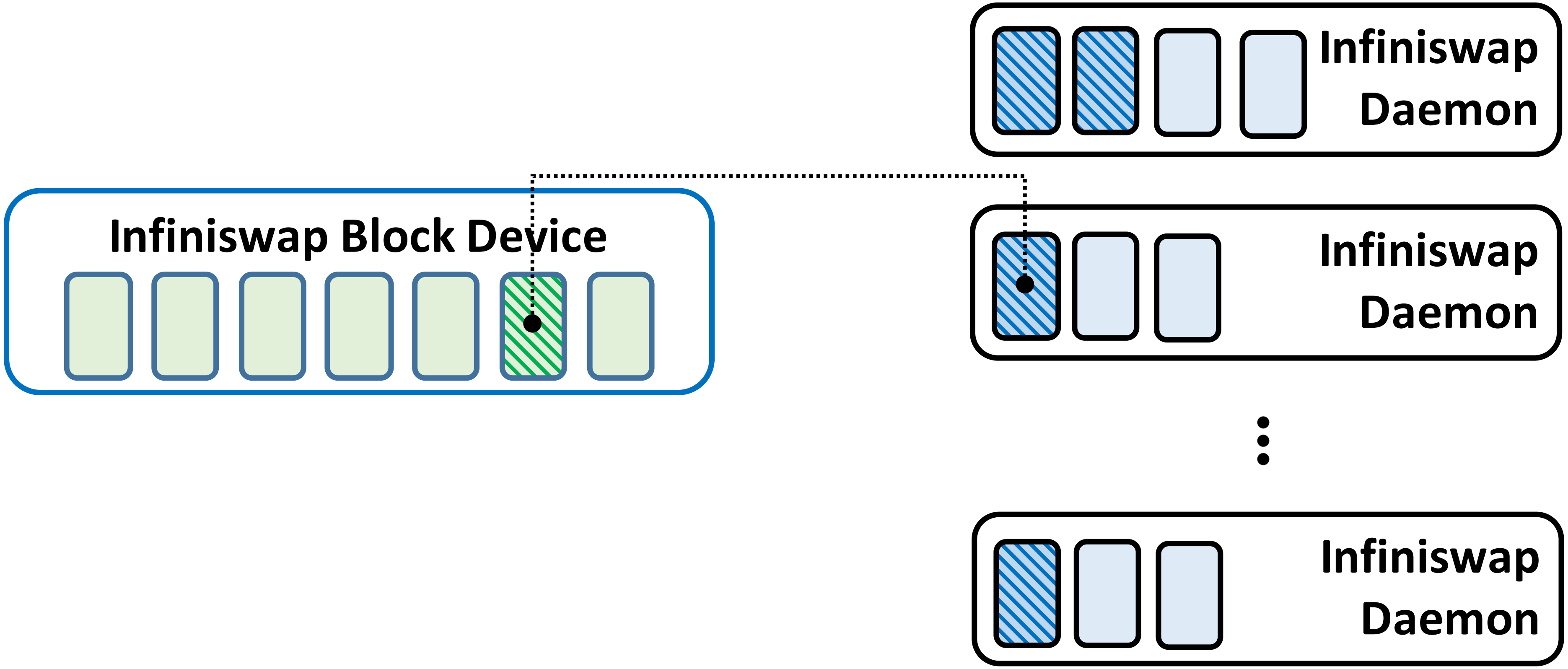
- ~~Central controller~~
- Decentralized approach

Power of two choices[1]



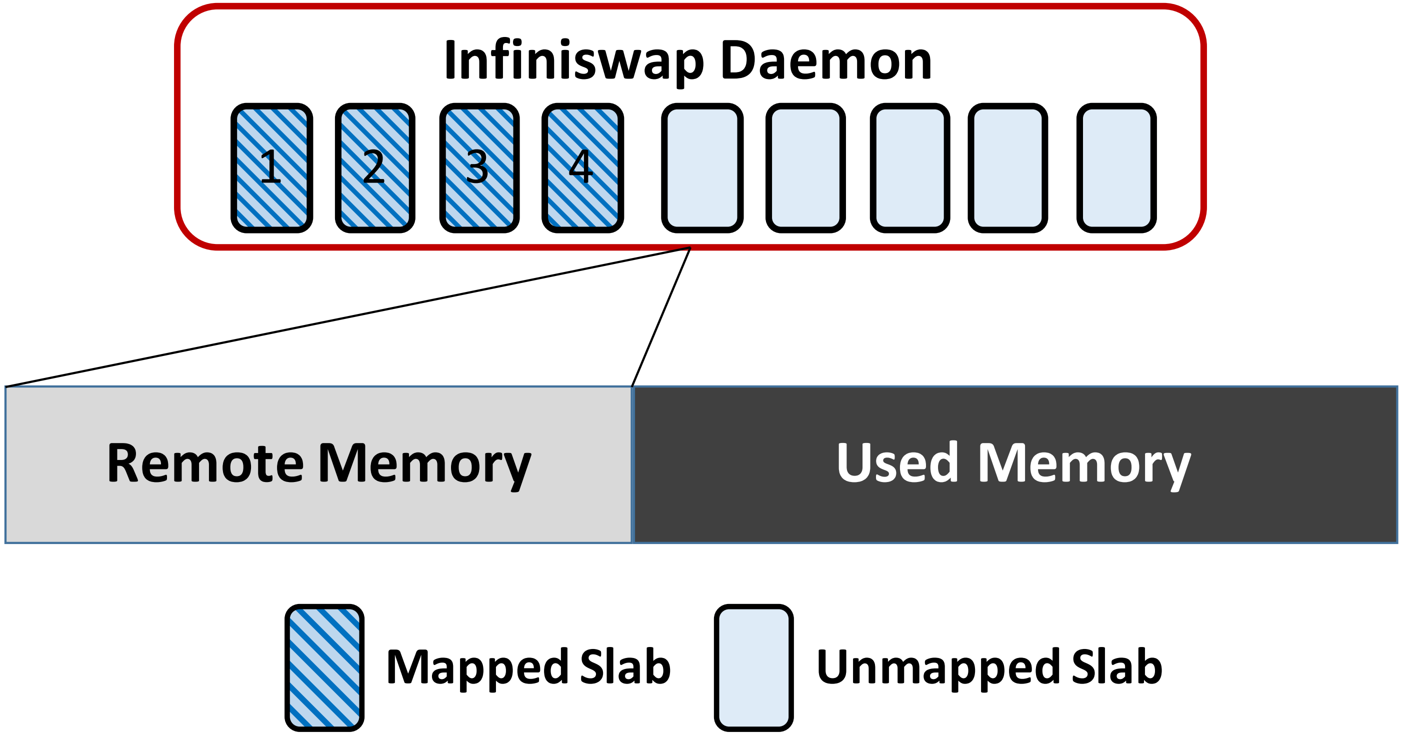
[1] Mitzenmacher, Michael. "The power of two choices in randomized load balancing.", Ph.D. thesis, U.C. Berkeley, 1996

Power of two choices[1]

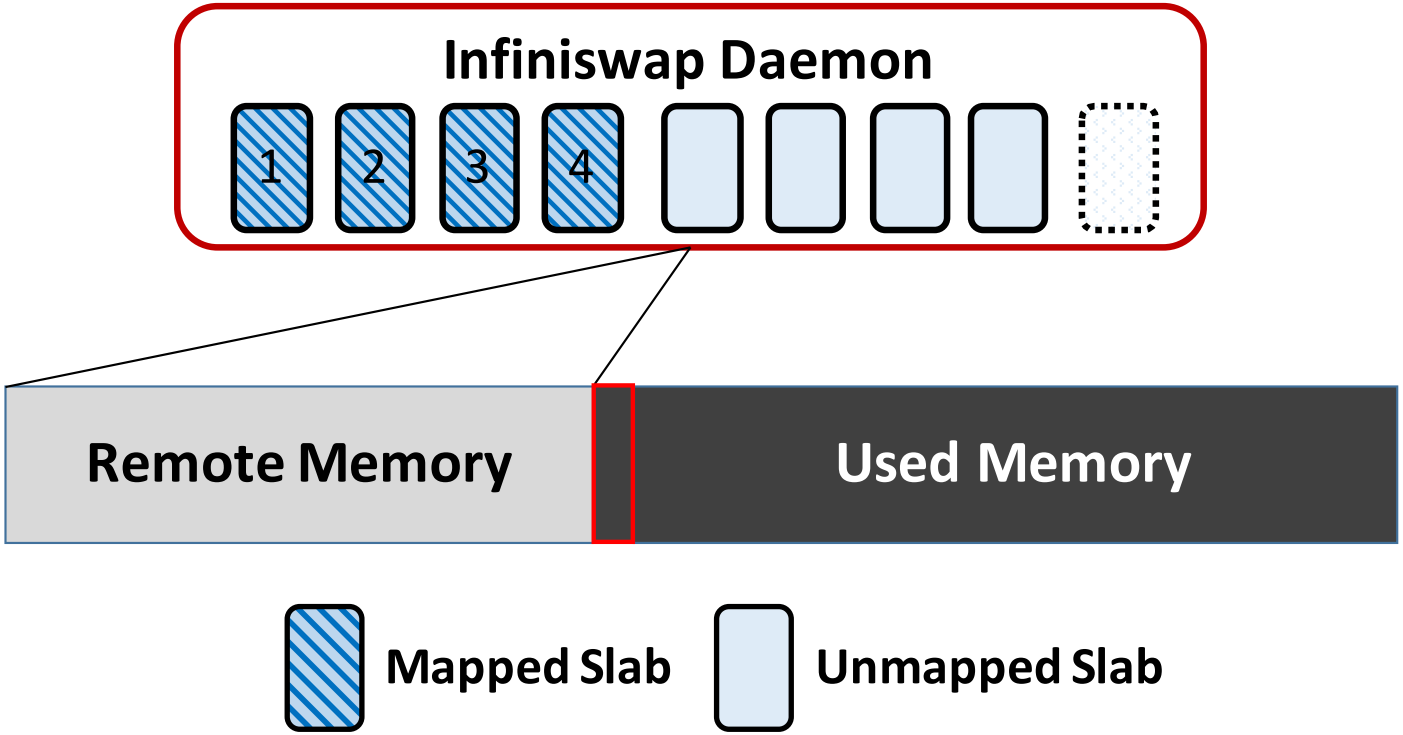


[1] Mitzenmacher, Michael. "The power of two choices in randomized load balancing.", Ph.D. thesis, U.C. Berkeley, 1996

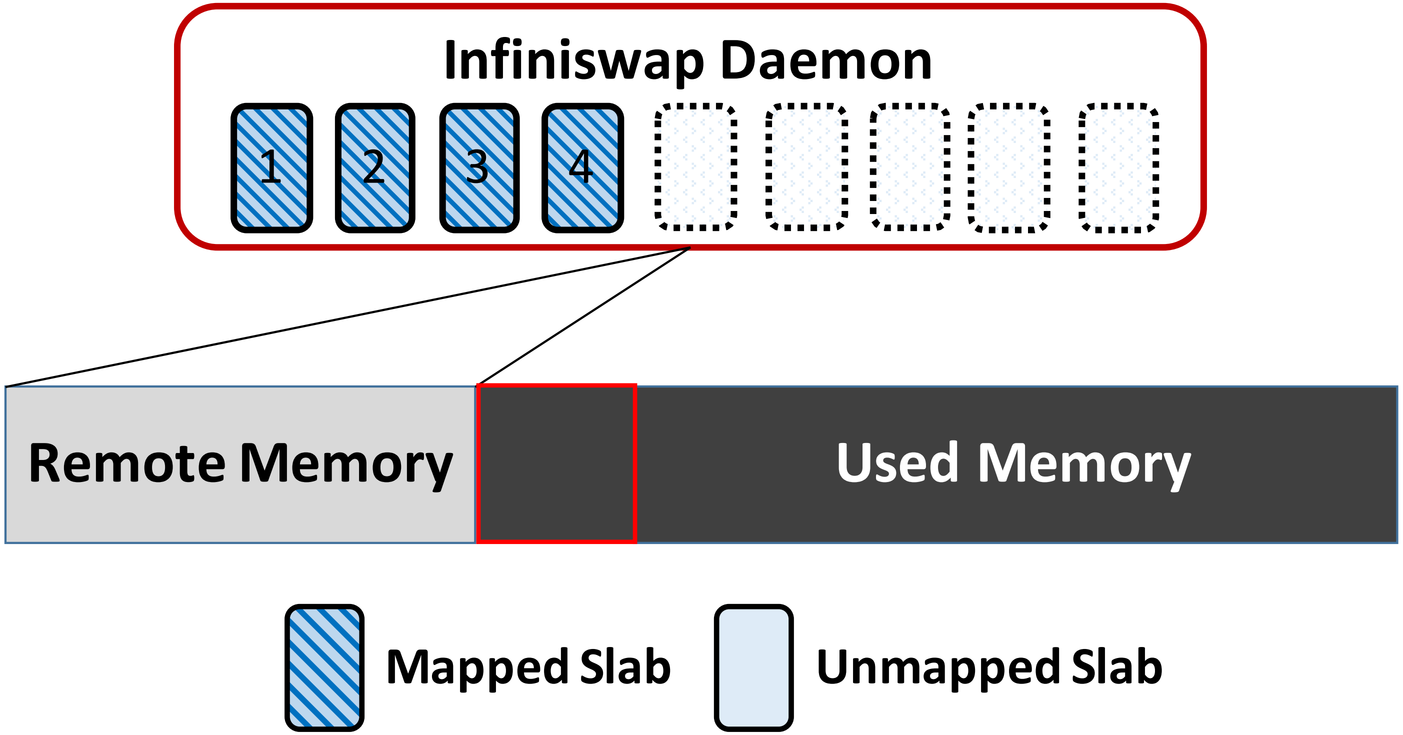
Slab eviction



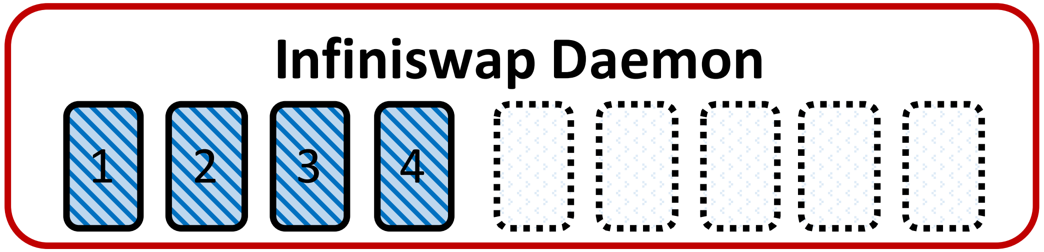
Slab eviction



Slab eviction

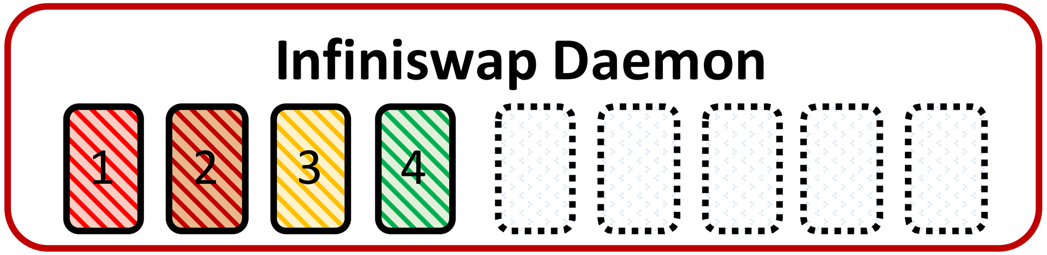


Which slab should be evicted?



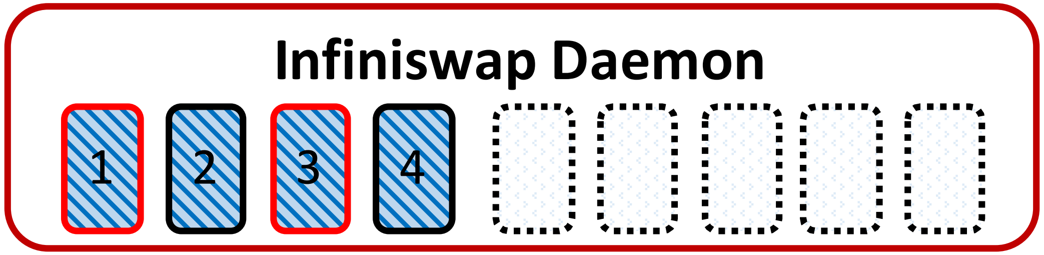
Daemon: Does not know the swap activities

Which slab should be evicted?



Daemon: Too expensive to query all the slabs

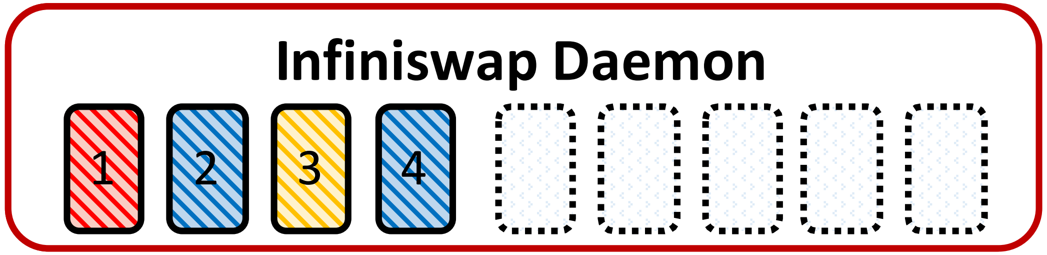
Power of multiple choices[1]



Select E least-active slabs from E+E’ random slabs

[1] Park, Gahyun. "A generalization of multiple choice balls-into-bins.” PODC’11

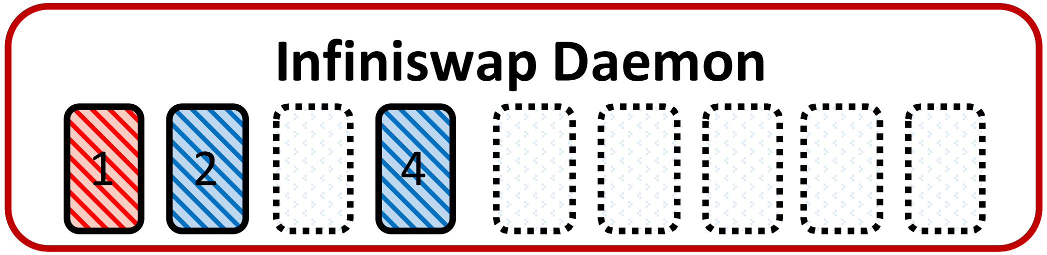
Power of multiple choices[1]



Select E least-active slabs from E+E' random slabs

[1] Park, Gahyun. "A generalization of multiple choice balls-into-bins." PODC'11

Power of multiple choices[1]



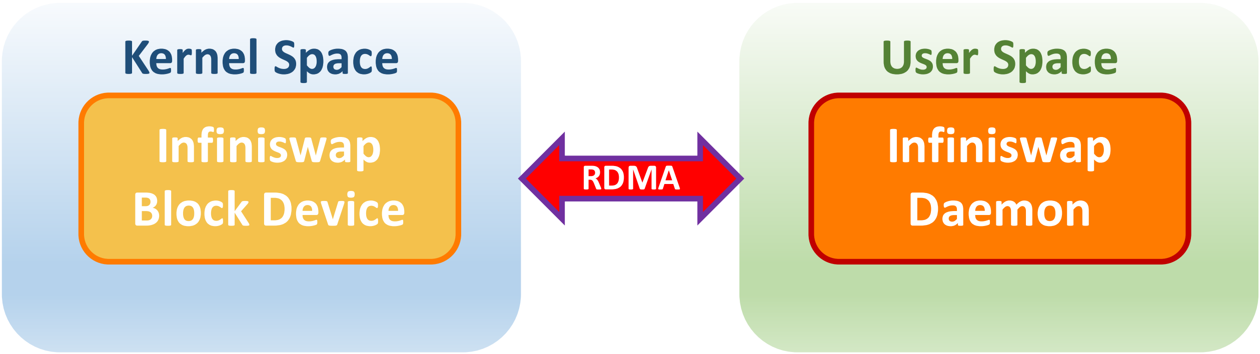
Select E least-active slabs from E+E' random slabs

[1] Park, Gahyun. "A generalization of multiple choice balls-into-bins." PODC'11

Agenda

- Motivation and related work
- Design and system overview
- **Implementation and evaluation**
- Future work and conclusion

Implementation



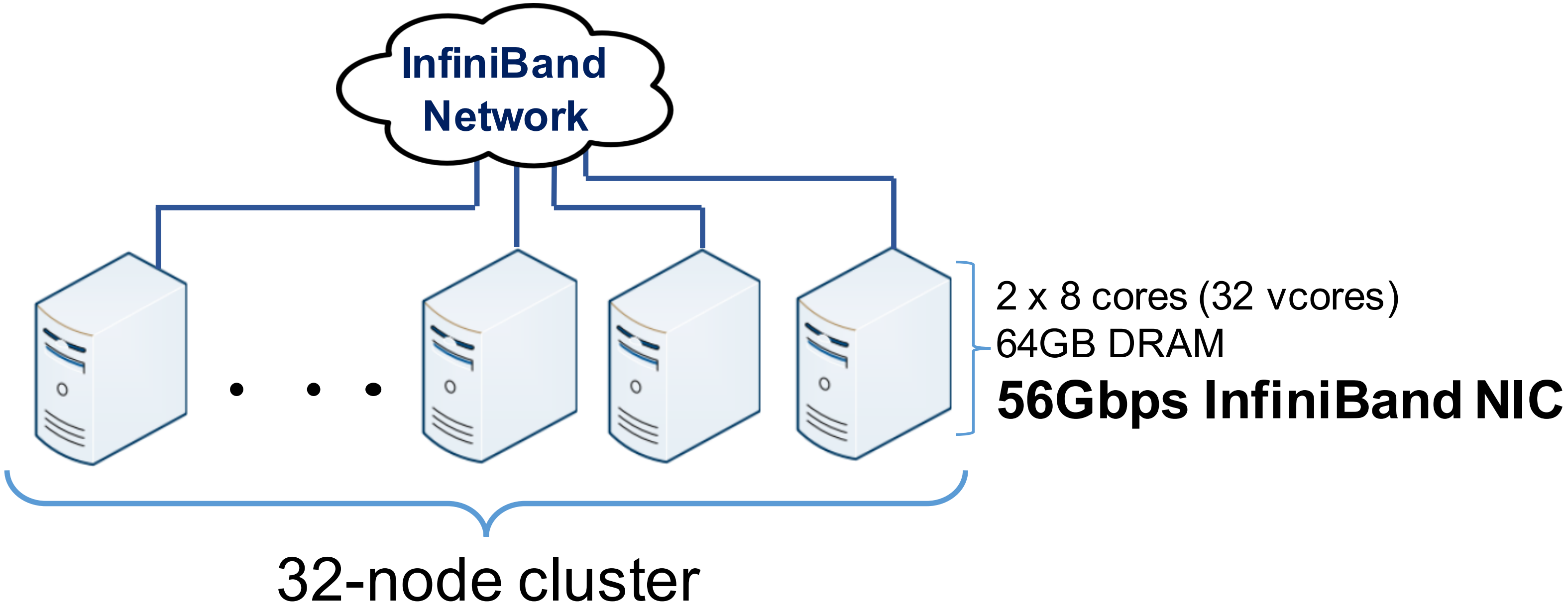
- Connection Management
  - **One** RDMA connection per active block device - daemon pair
- Control Plane
  - **SEND, RECV**
- Data Plane
  - **One-sided RDMA READ, WRITE**

What are we expecting from Infiniswap?

- **Application performance**
- **Cluster memory utilization**
- Network usage
- Eviction overhead
- Fault-tolerance overhead
- Performance as a block device

⋮

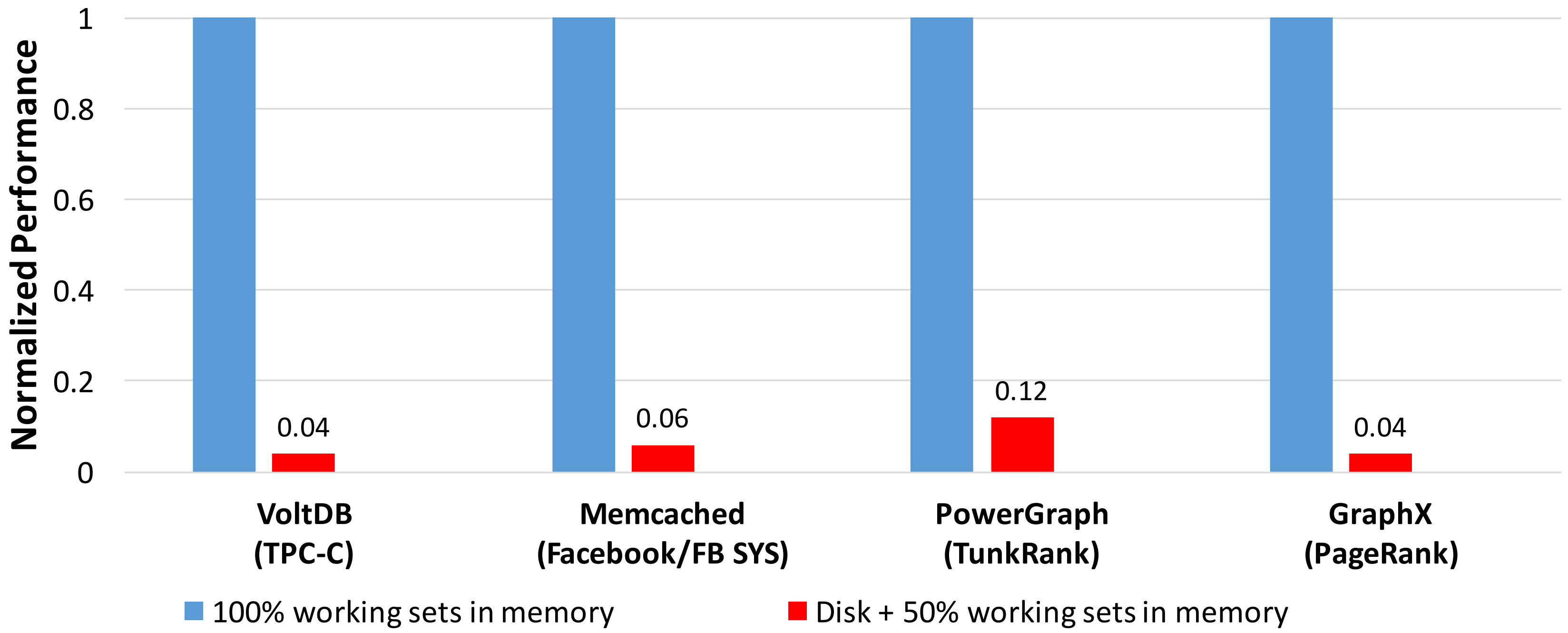
Evaluation



VOLTDB memCached powergraph GraphX

Application performance

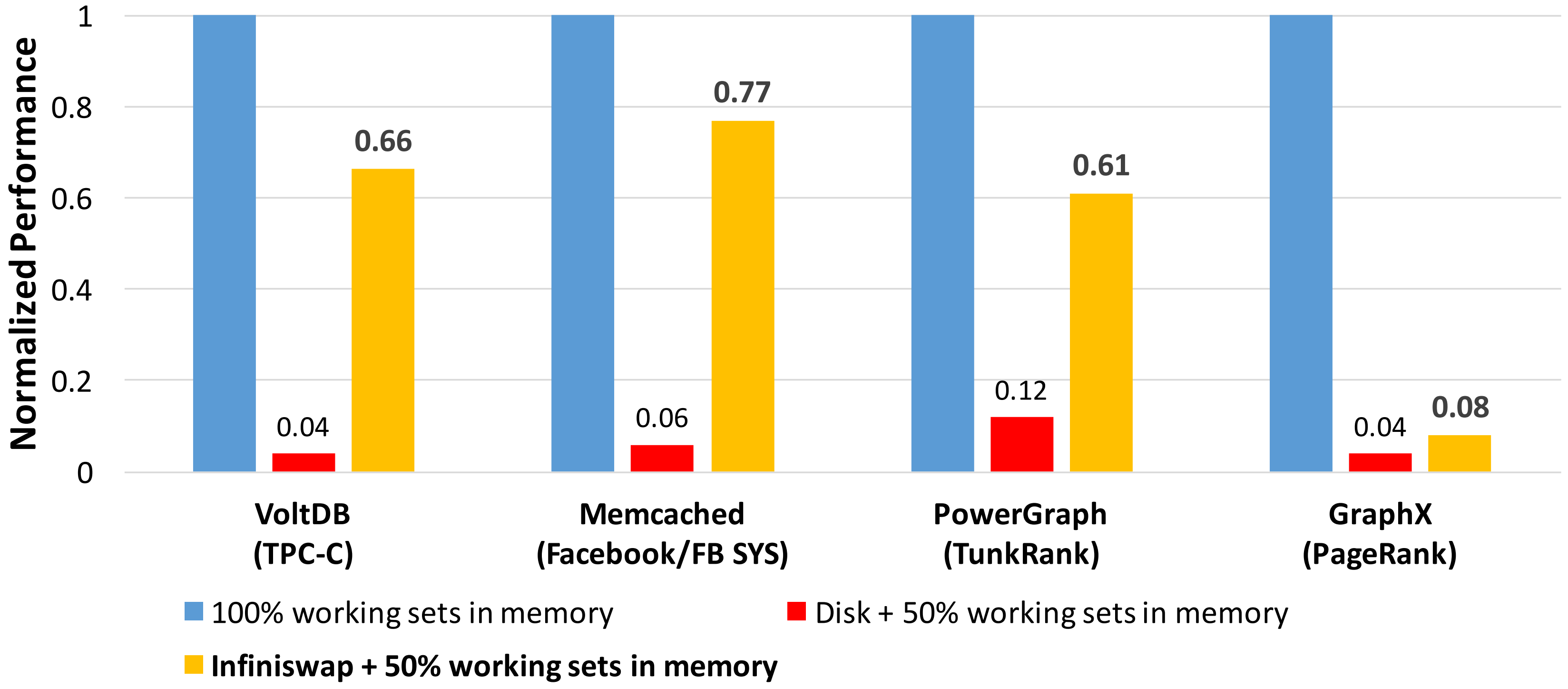
- 50% working sets in memory



- Application performance is improved by 2-16x

Application performance

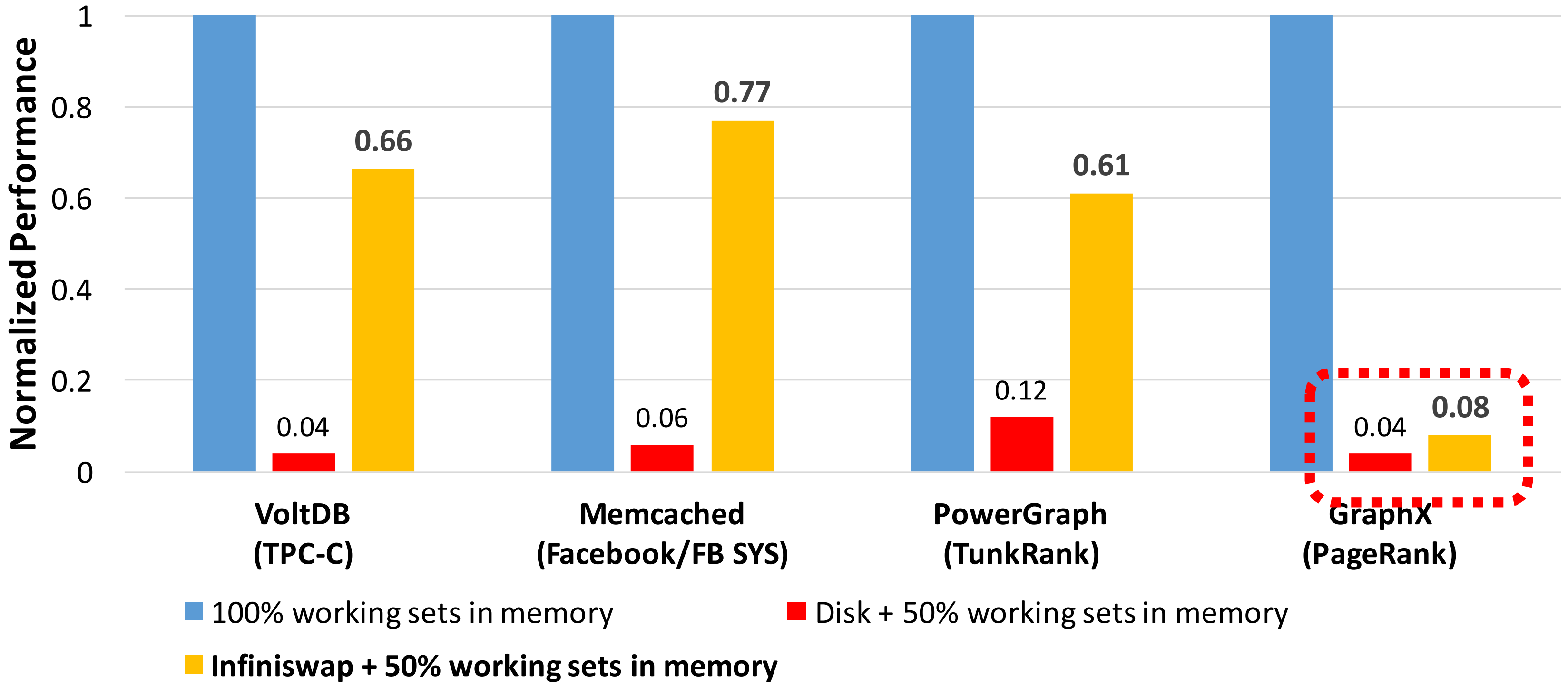
- 50% working sets in memory



- Application performance is improved by 2-16x

Application performance

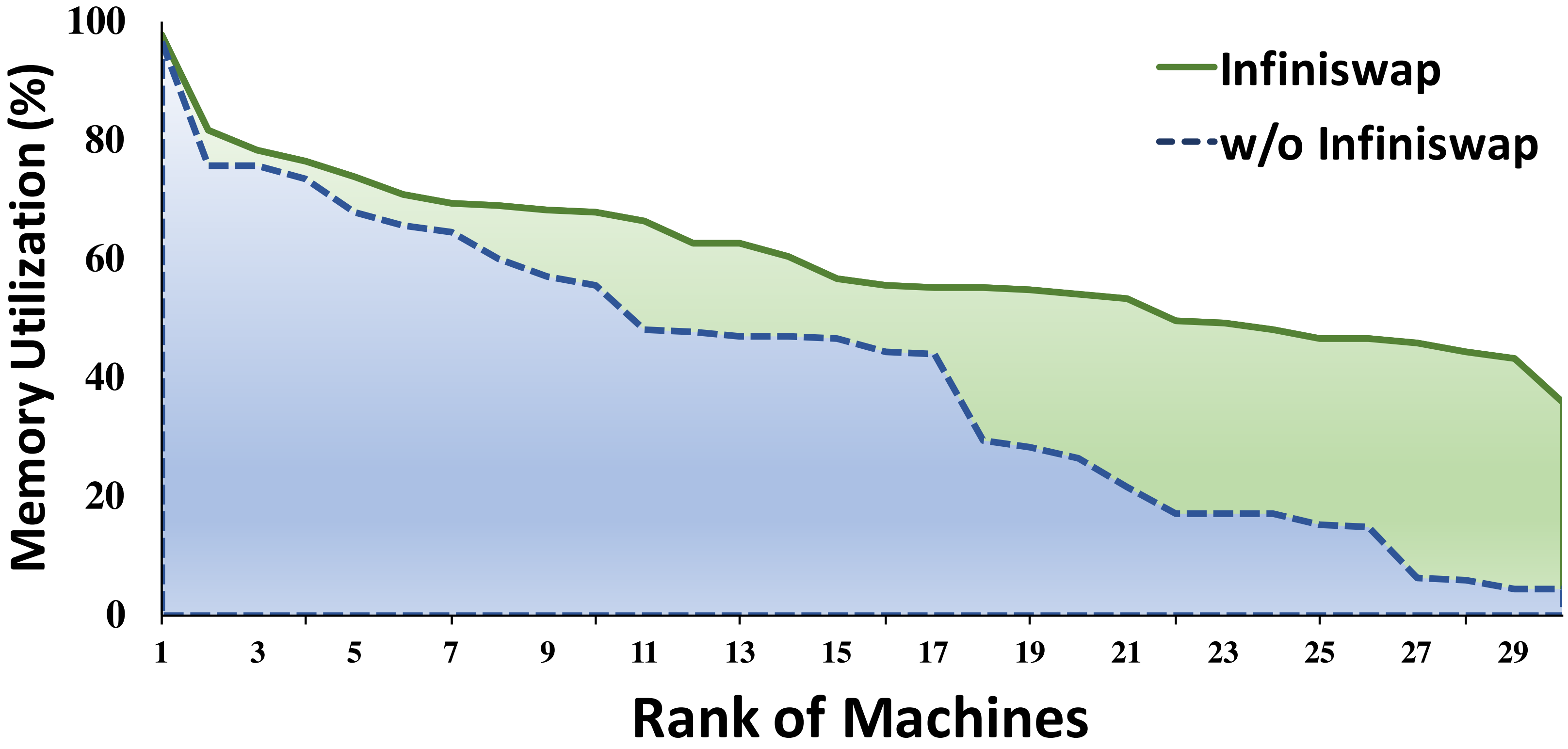
- 50% working sets in memory



- Application performance is improved by 2-16x

Cluster memory utilization

- 90 containers (applications), mixing all applications and memory constraints.



- Cluster memory utilization is improved from **40.8%** to **60%** (1.47x)

Agenda

- Motivation and related work
- Design and system overview
- Implementation and evaluation
- **Future work and conclusion**

Limitations and future work

- **Trade-off in fault-tolerance**
  - Local disk is the bottleneck
  - Multiple remote replicas
    - Fault-tolerance vs. space-efficiency
- **Performance isolation among applications**
  - W/o limitation on each application's usage
  - W/o mapping between remote memory and applications

Conclusion

- **Infiniswap: remote paging over RDMA**
  - Application performance
  - Cluster memory utilization
- **Efficient, practical memory disaggregation**
  - No hardware design
  - No application modification
  - **Fault-tolerance**
  - **Scalability**

Source code is coming soon!

https://github.com/Infiniswap/infiniswap.git

Thank You !