# Infiniswap

*Efficient Memory Disaggregation*

Mosharaf Chowdhury

with Juncheng Gu, Youngmoon Lee, Yiwen Zhang, and Kang G. Shin

University of Michigan

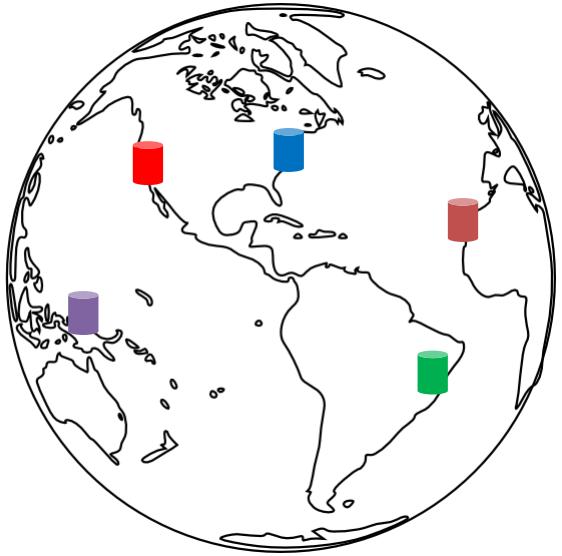
## Rack-Scale Computing

Proactive Analytics
Before You Think!

## Datacenter-Scale Computing

| | |
|---|---|
| Coflow Networking | *Open Source* |
| Apache Spark | *Open Source* |
| Cluster File System | *Facebook* |
| Resource Allocation | *Microsoft* |
| DAG Scheduling | *Apache YARN* |
| Cluster Caching | *Alluxio* |

## Geo-Distributed Computing



Fast Analytics
Over the WAN

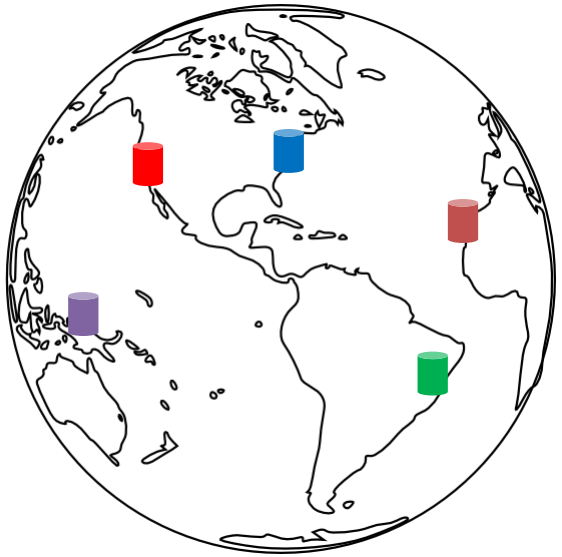
## Rack-Scale Computing

< 0.01 ms

## Datacenter-Scale Computing

~ 1 ms

## Geo-Distributed Computing



> 100 ms

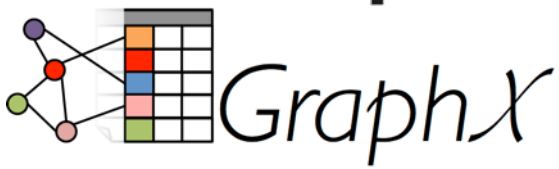
# Memory-Intensive Applications

**The volume of data we want to *make sense of* is increasing**

Memory is getting bigger and cheaper

- Many workloads fit in memory

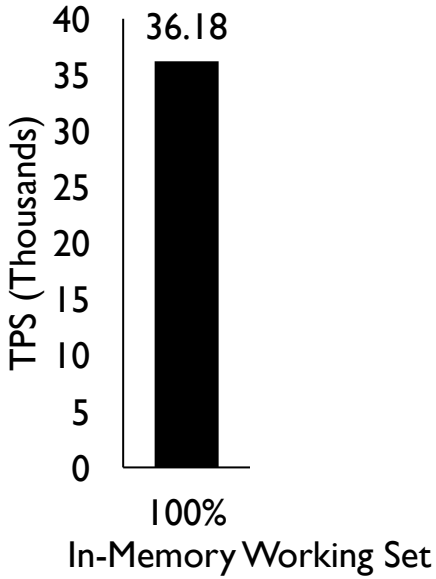
In-memory * is all the rage!



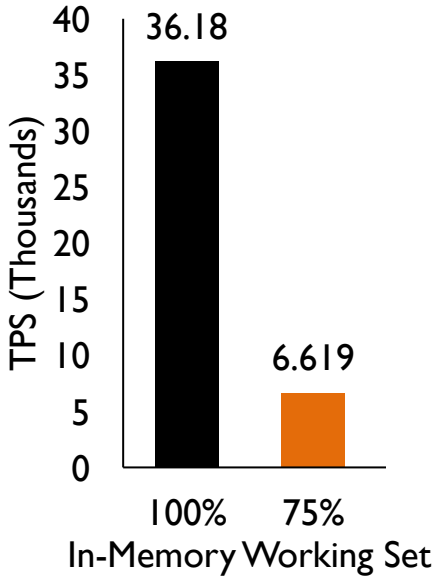


# Perform Great!



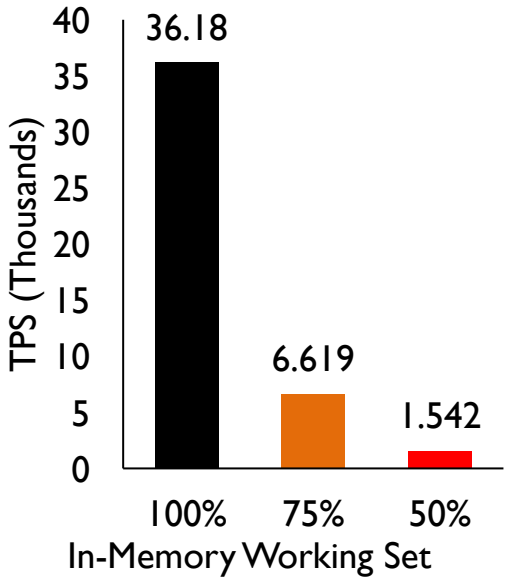

**TPC-C on VoltDB**

# Perform Great Until Memory Runs Out

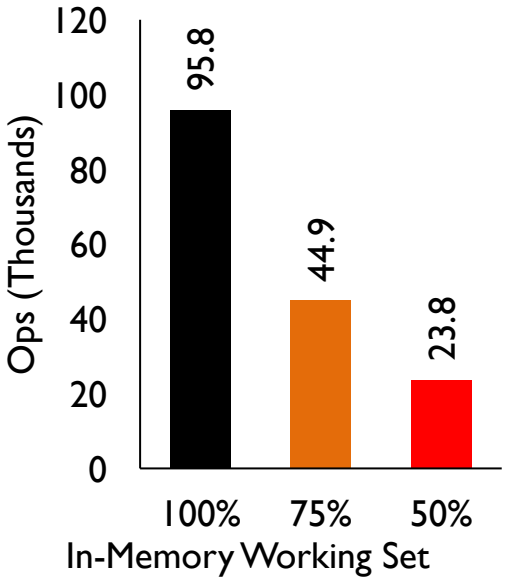



**TPC-C on VoltDB**

# Perform Great Until Memory Runs Out

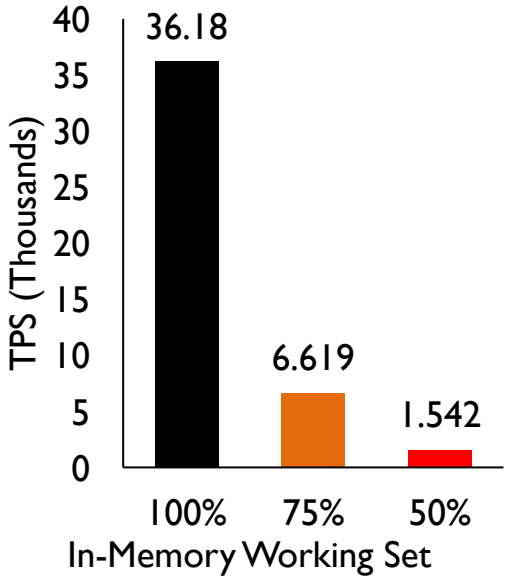



**TPC-C on VoltDB**

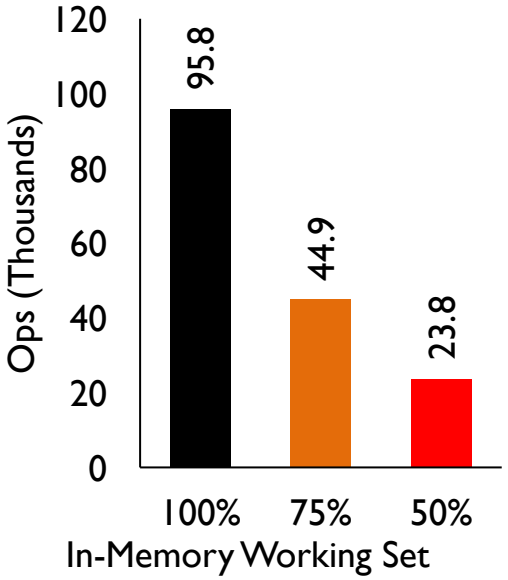



**FB Workload on Memcached**

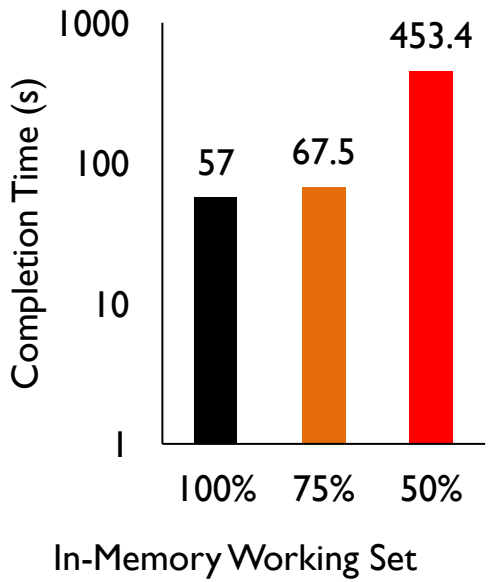
# Perform Great Until Memory Runs Out




**TPC-C on VoltDB**

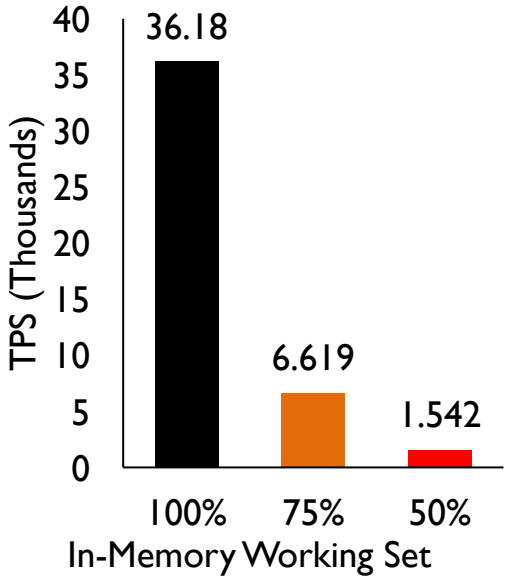



**FB Workload on Memcached**



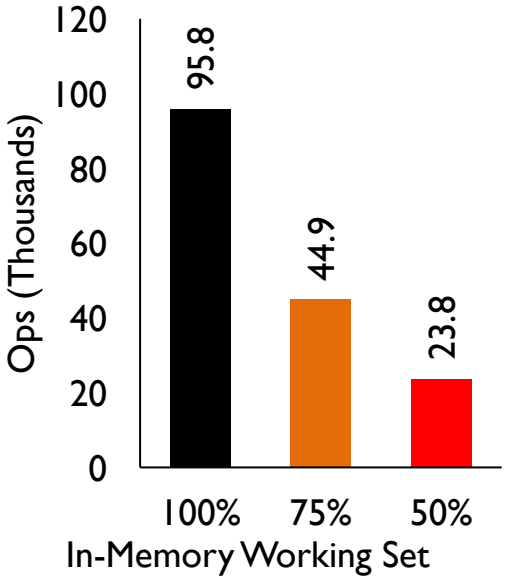

**PageRank on PowerGraph**

# 50% Less Memory Causes Slowdown of …



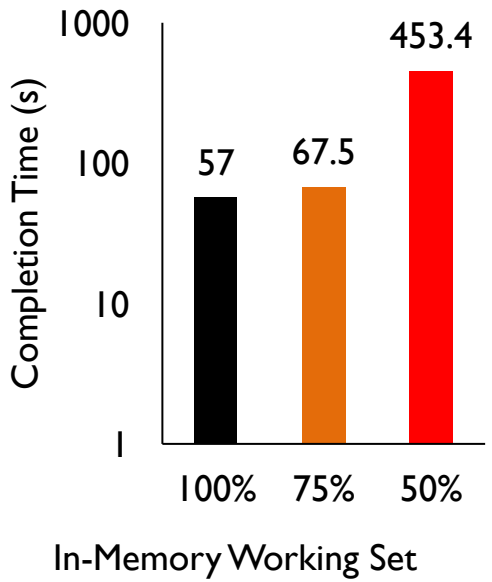

**TPC-C on VoltDB**

**24X**




**FB Workload on Memcached**

**4X**




**PageRank on PowerGraph**

**8X**

# Between a Rock and a Hard Place

**Underallocation**

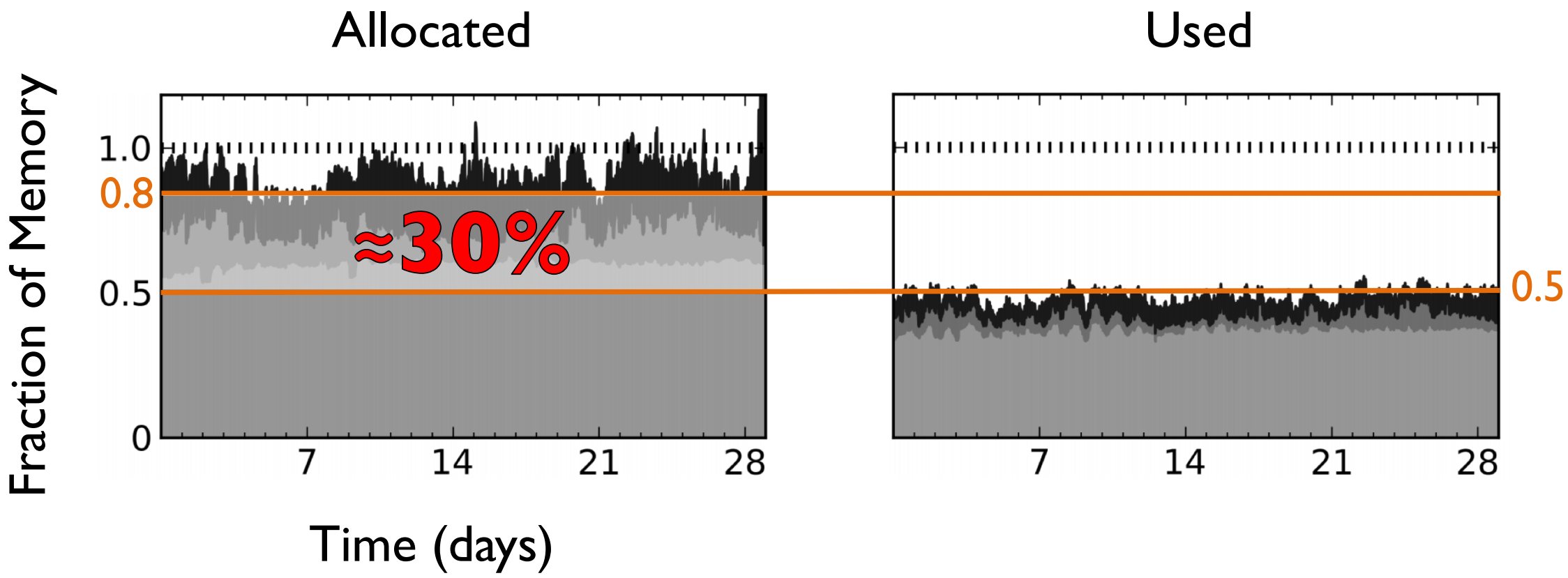
Leads to severe performance loss

**vs.**

**Overallocation**

Leads to underutilization

# Memory Underutilization at Google[1]




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

# Memory Load Imbalance

Measured as the 99th percentile to median memory utilization ratio

| **Perfect** Balance | **Google** Cluster | **Facebook** Cluster |
|---|---|---|
| ≈1 | 3.35 | 2.4 |

# How Can We Recover This Memory?

# Infiniswap

*Disaggregates Memory*

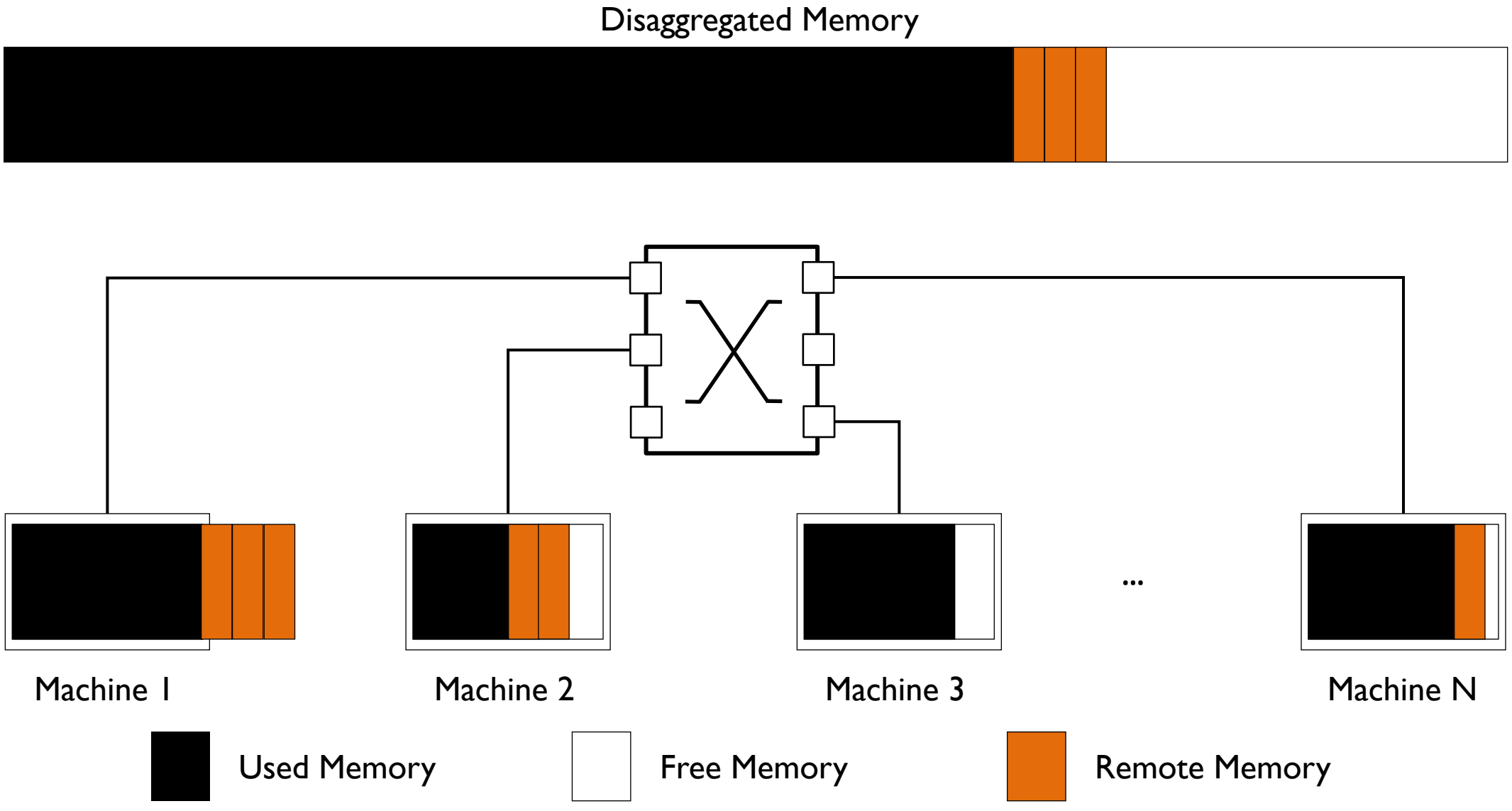
Exposes memory across server boundaries in a

- scalable,
- fault-tolerant, and
- efficient manner

without modifying any

- applications,
- operating systems, or
- hardware

# Memory Disaggregation



# Design Goals

**Improve application performance and cluster efficiency**

**Minimize deployment overhead**

- No new hardware
- No software modification

**Tolerate failures**

- Machine crash, network disconnection

**Manage remote memory at scale**

# Selected Prior Efforts

| | No H/W Design | No App Modification | Fault-Tolerant | Scalable |
|---|---|---|---|---|
| Memory Blade[ISCA'09] | ✗ | ✓ | ✓ | ✓ |
| HPBD[CLUSTER'05] / nbdX[1] | ✓ | ✓ | ✗ | ✗ |
| RDMA key-value service (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

# Infiniswap

*Exposes free remote memory as swap devices in a decentralized manner w/o affecting remote processes*

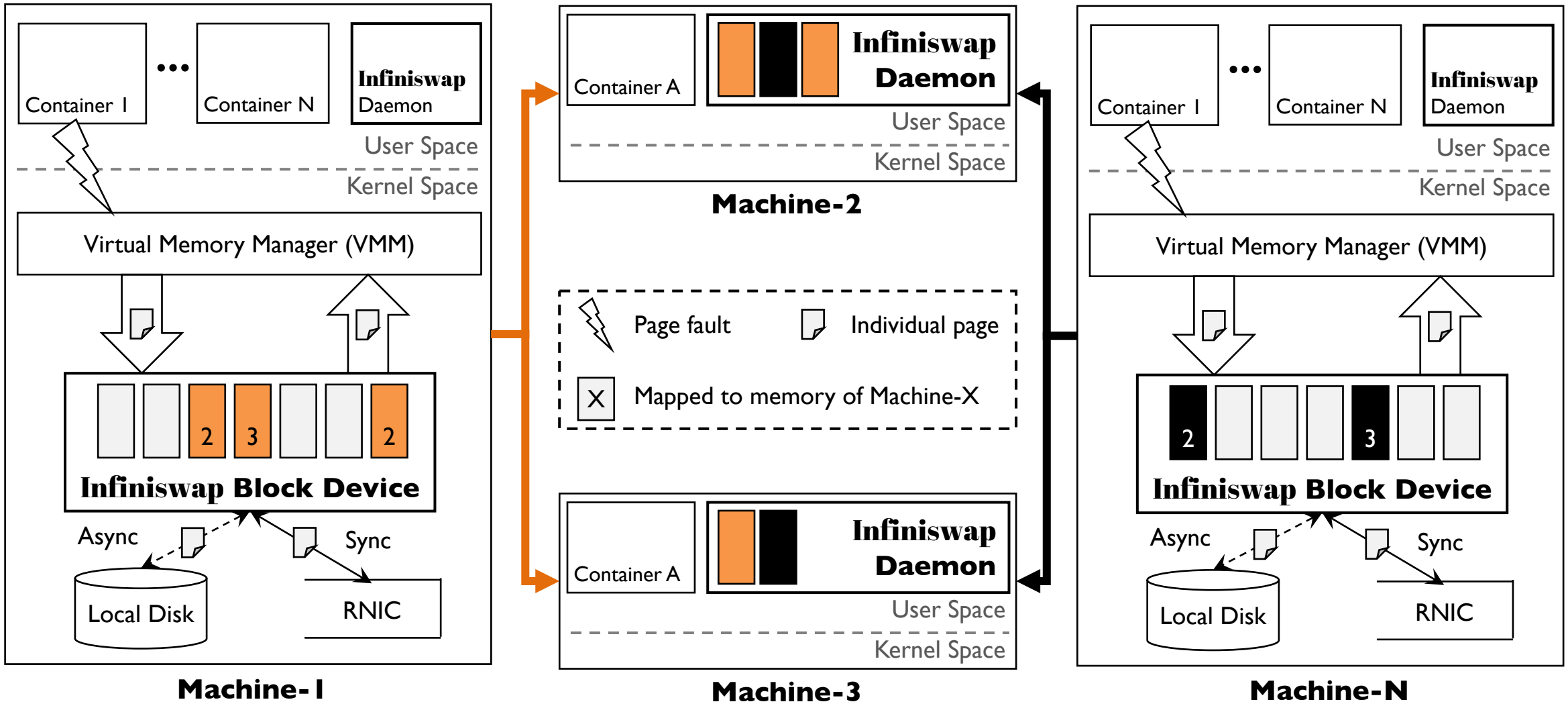
1. **Infiniswap** Block Device

   *Finds free remote memory, maps pages, and provides fault tolerance without any central coordination*

2. **Infiniswap** Daemon

   *Proactively evicts remote pages to ensure transparent, best-effort service*

# Infiniswap in One Slide



# Are We There Yet?

**Improve application performance and cluster efficiency**

**Minimize deployment overhead** → Remote memory paging over RDMA

- No new hardware
- No software modification

**Tolerate failures** → Async. backup to disk

- Machine crash, network disconnection

**Manage remote memory at scale** → ?

# Scalability Challenges

**How to find remote memory in the cluster?**

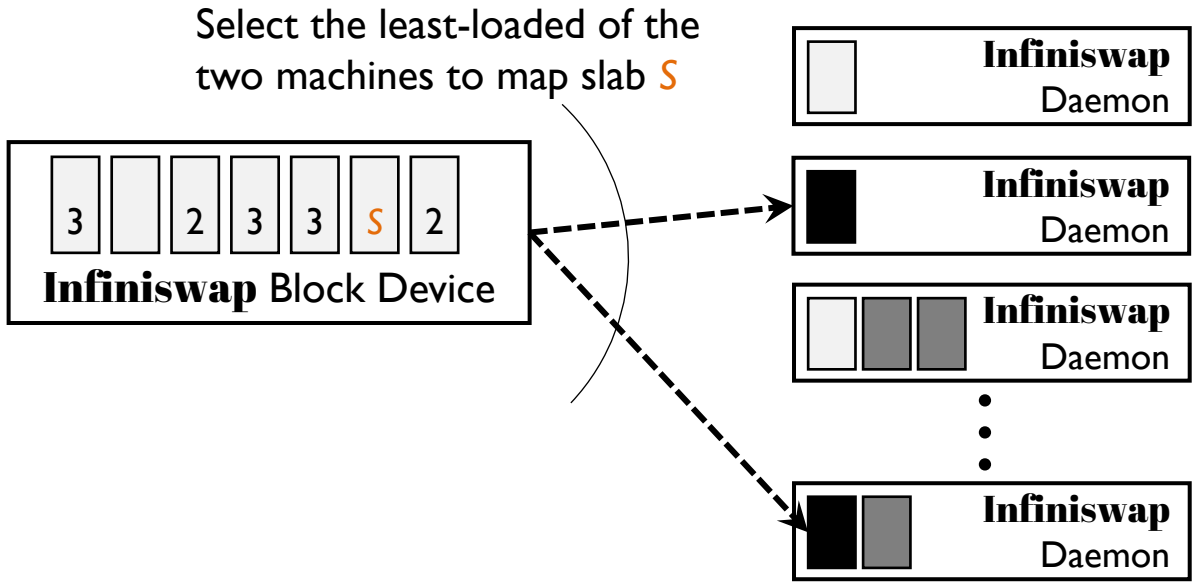
- Too many pages lead to too much management overhead
- Centralized solution can be slow and expensive

# Decentralized Mapping

Use large slab instead of page for memory management

Power of two choices

- Select from new machines
- After activity crosses a threshold



# Scalability Challenges

**How to find remote memory in the cluster?**

- Too many pages lead to too much management overhead
- Centralized solution can be slow and expensive

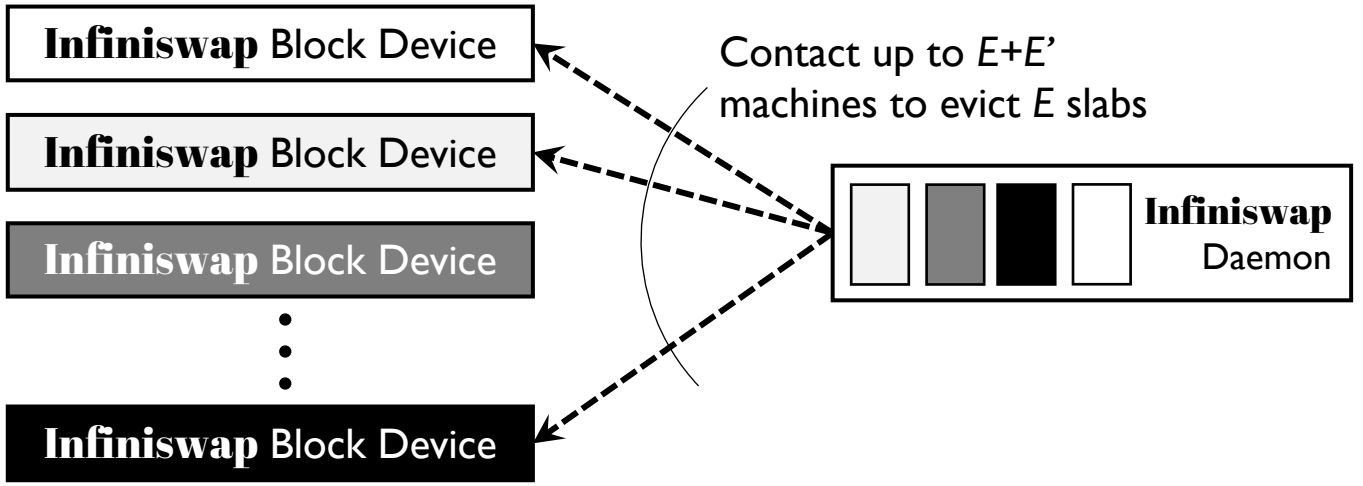
**Which remote mapping should we evict?**

- Should be performed to avoid affecting remote applications' performance
- **Problem:** Paging estimation is hard because one-sided RDMA do not involve CPU

# Batch Eviction

**Power of many choices**

- Approximate LFU
- Without contacting all slabs
- When free memory falls below a threshold



# Infiniswap Design Choices

Improve application performance and cluster efficiency

**Minimize deployment overhead**
- No new hardware
- No software modification

→ Remote memory paging over RDMA

**Tolerate failures**
- Machine crash, network disconnection

→ Async. backup to disk

**Manage remote memory at scale**

→ Decentralized mapping and eviction

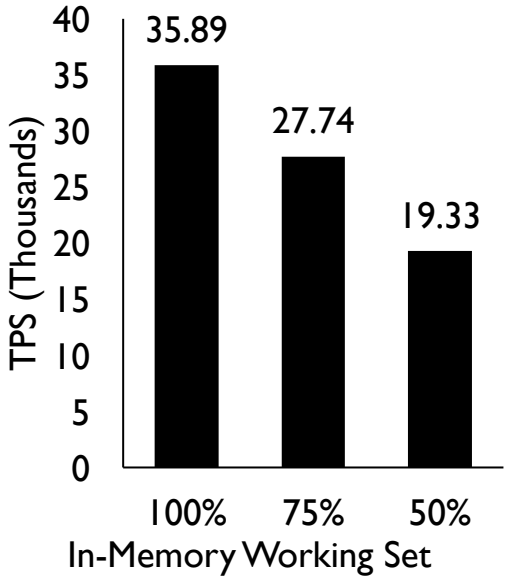
# Evaluation

*Deployment and evaluation on a 32-node 56-Gbps InfiniBand network on CloudLab using memory-intensive applications*

1. Does it improve performance?
2. Does it improve utilization?
3. Does it scale?
4. Can it handle failure?
5. …

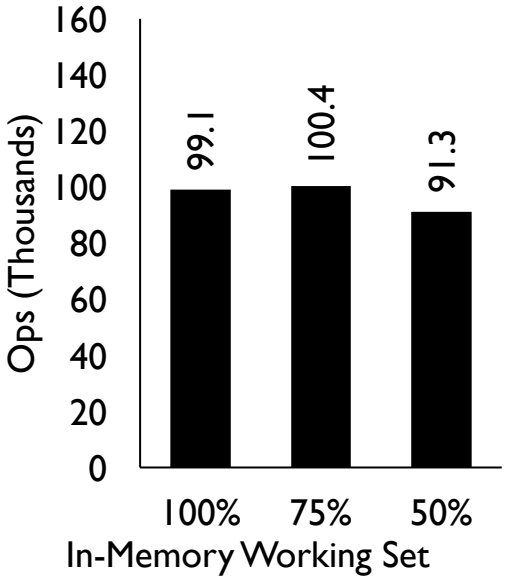
**YES**

# Even on 50% Memory, Slowdown is



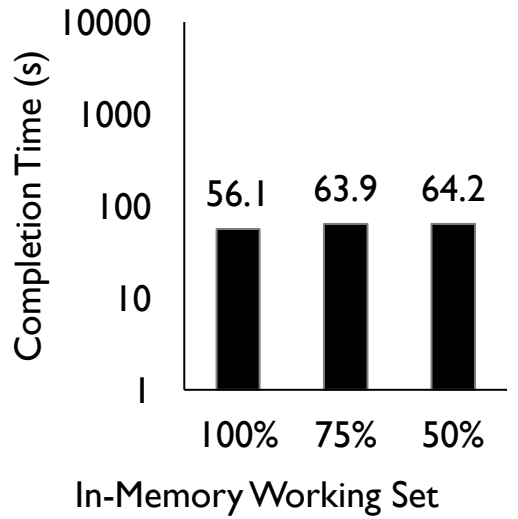

**TPC-C on VoltDB**

**<2X**




**FB Workload on Memcached**

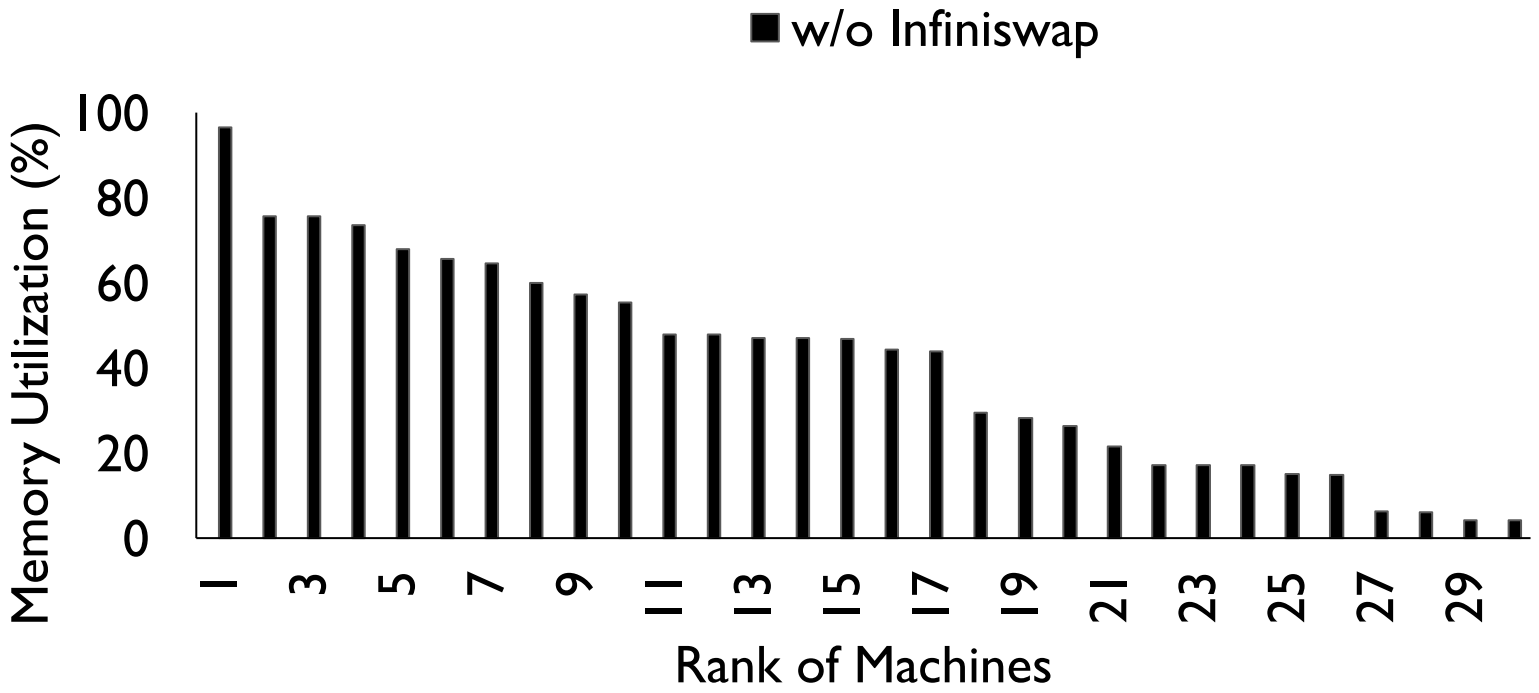
**≈1X**



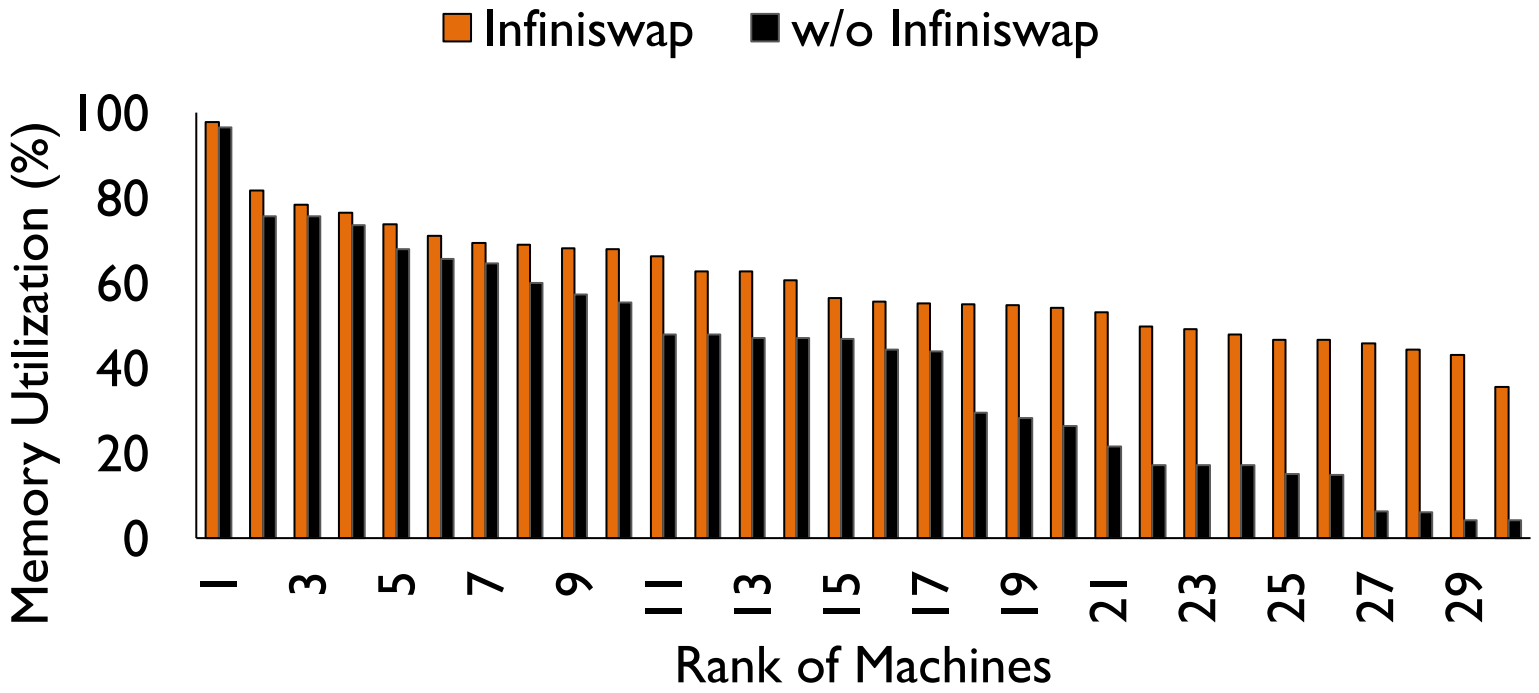

**PageRank on PowerGraph**

**≈1X**

# Higher & More Balanced Memory Utilization



# Higher & More Balanced Memory Utilization




**47% Higher Utilization**

# #1

## Performance Isolation

Between multiple tenants
In VMM and RDMA API

# #2

## Avoid Disk Backups

Performance during failures
Handle large paging bursts

# #3

## Rethink Paging Subsystem

For high-speed block devices
**Infiniswap** & NVMe devices

# Infiniswap

*Disaggregates Memory*

Exposes memory across server boundaries in a

- scalable,
- fault-tolerant, and
- efficient manner

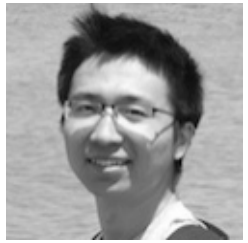
without modifying any

- applications,
- operating systems, or
- hardware

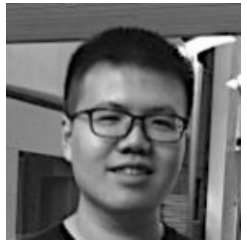
# Infiniswap

*Disaggregates Memory*

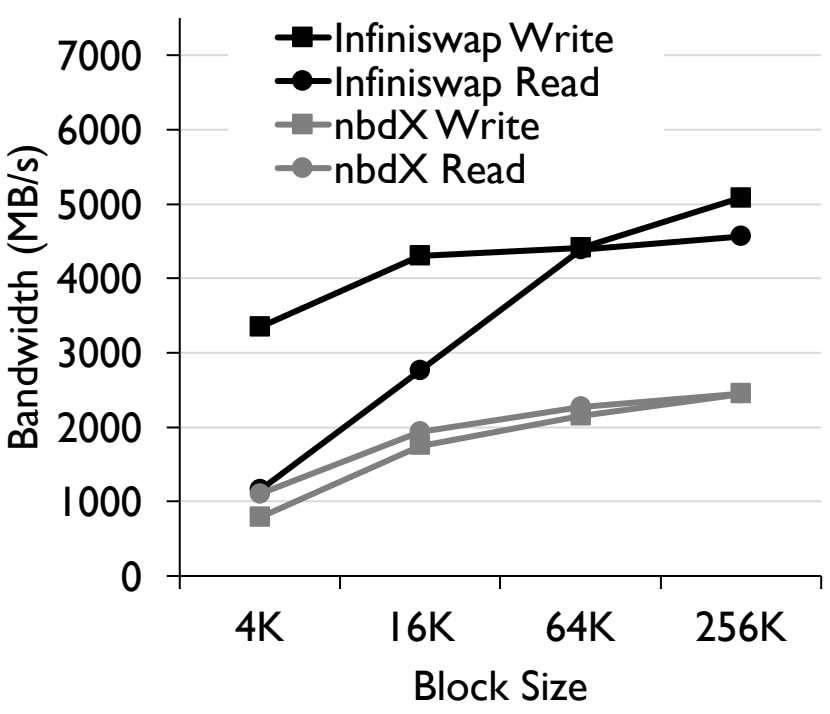
- Learn more in our NSDI'17 paper
- Try it from https://github.com/infiniswap
- Contact us at infiniswap@umich.edu



Juncheng Gu

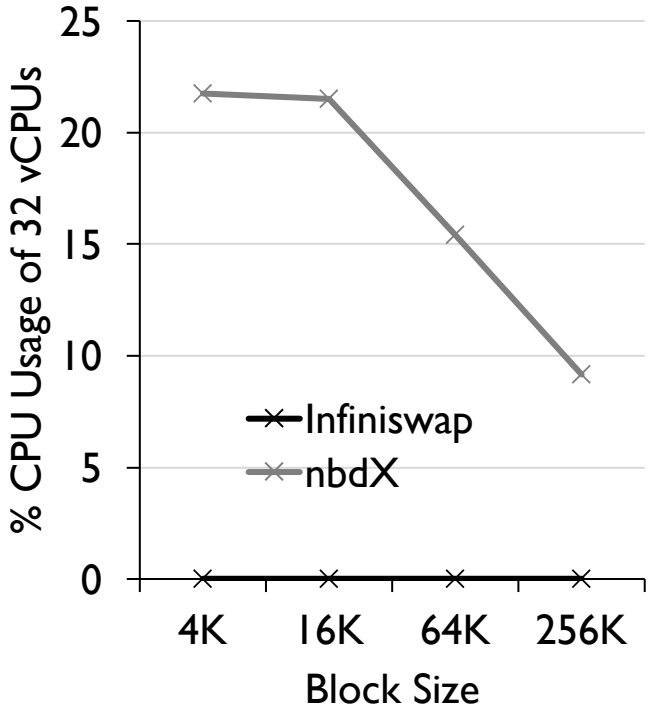

Youngmoon Lee


Yiwen Zhang

# Infiniswap Microbenchmarks

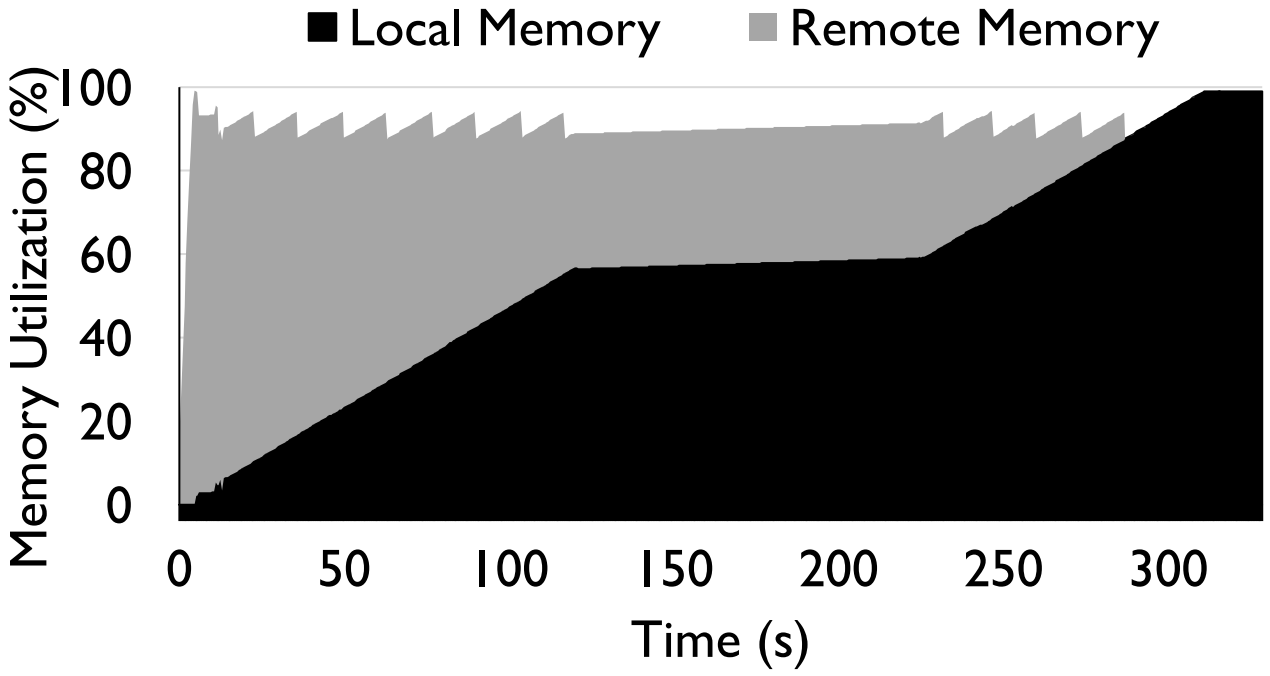



**Higher I/O Bandwidth**

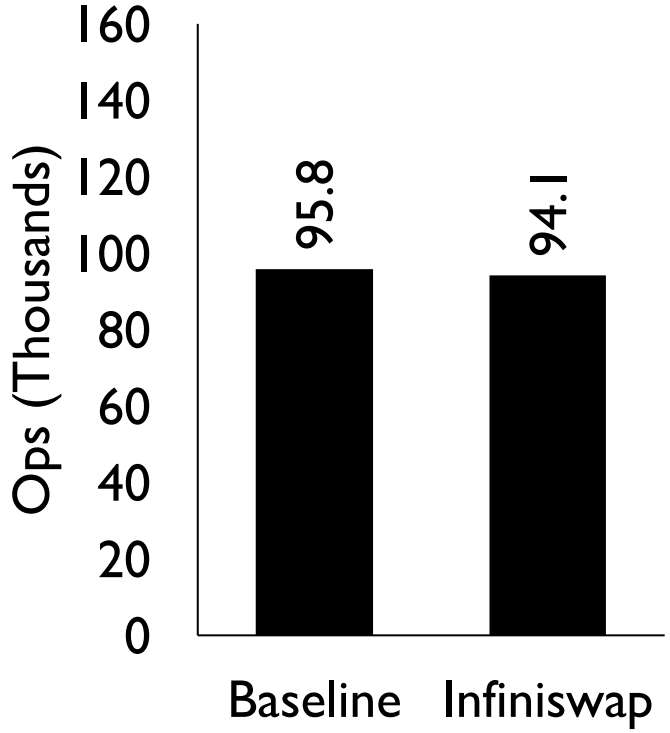



**NO Remote CPU Usage**

# Host Performance Unaffected




**Proactive Eviction**




**NO Impact on Performance**