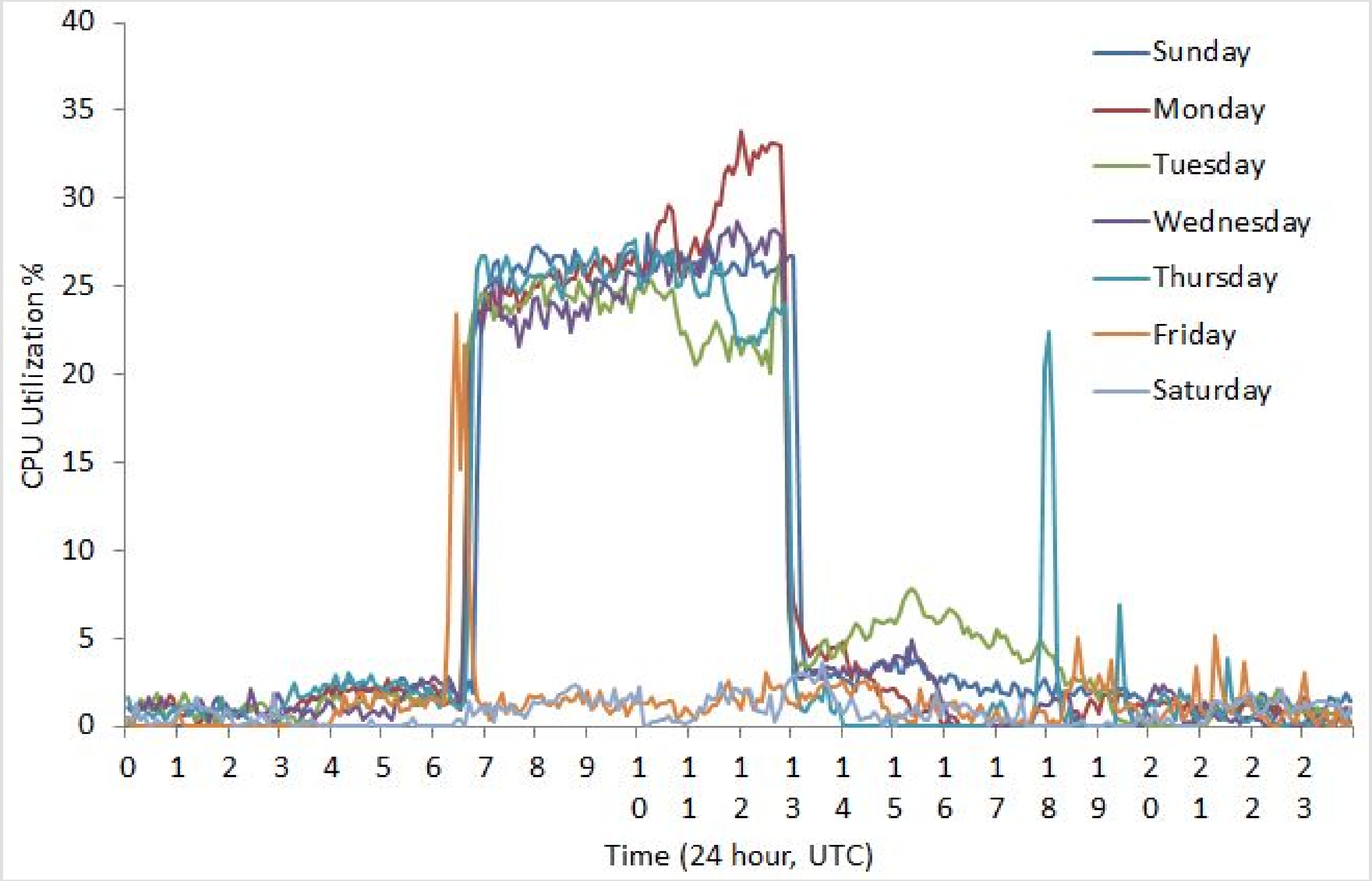
Rack Scale Apps

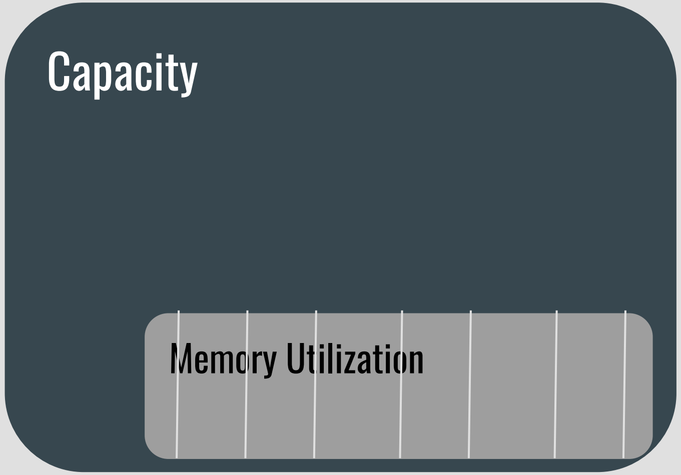
Neel Shah

The Cloud

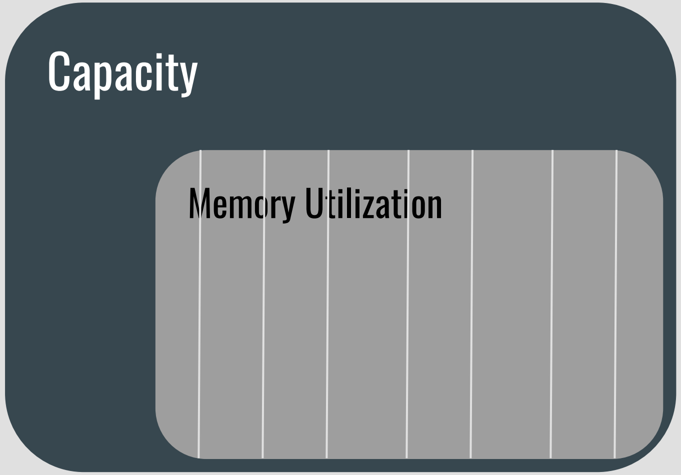
- **Vital to the world!**
- Service providers: Facebook, Twitter, etc.
- Storing/accessing data and programs over the Internet
- Servers living in **DataCenters**
- 2% of energy usage in the US
- **$30 billion** to power *idle servers*



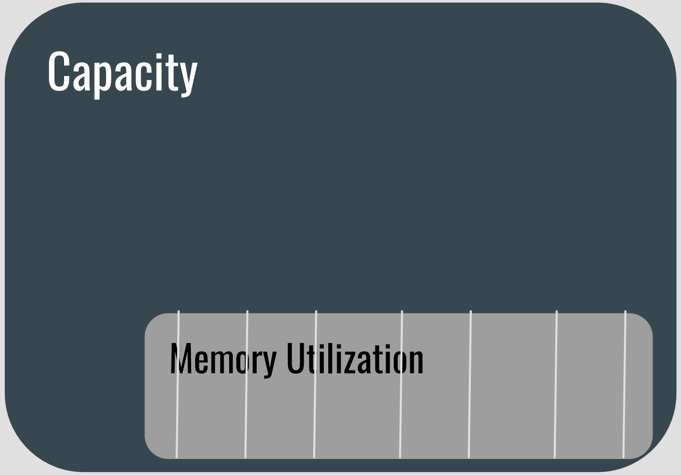
https://huanliu.wordpress.com/2012/02/17/host-server-cpu-utilization-in-amazon-ec2-cloud/



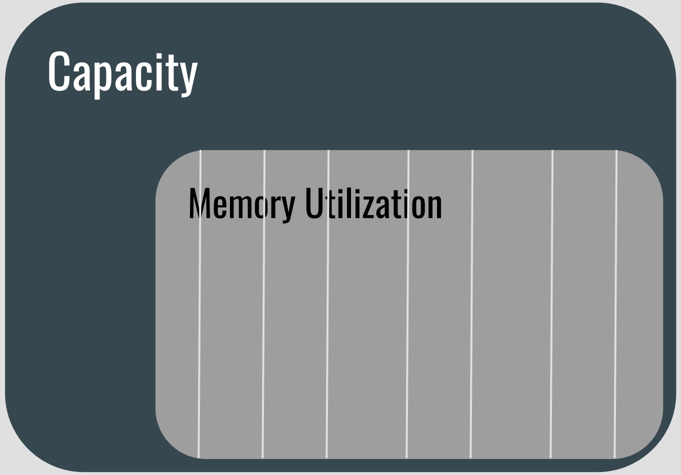
1.) Underutilization



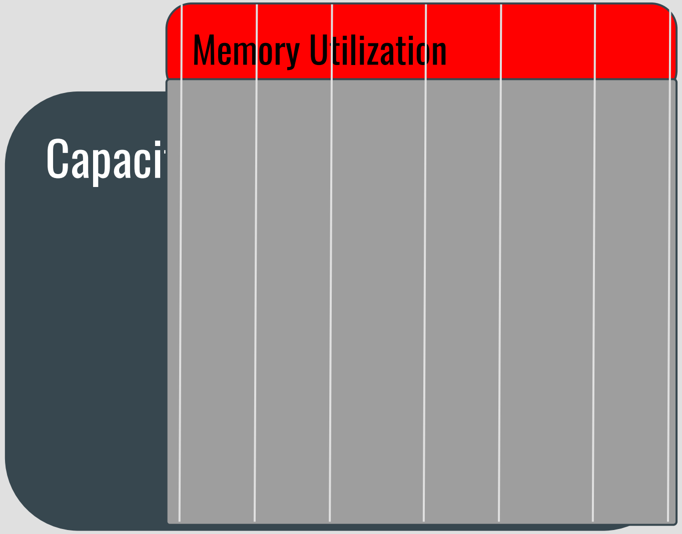
2.) High Utilization



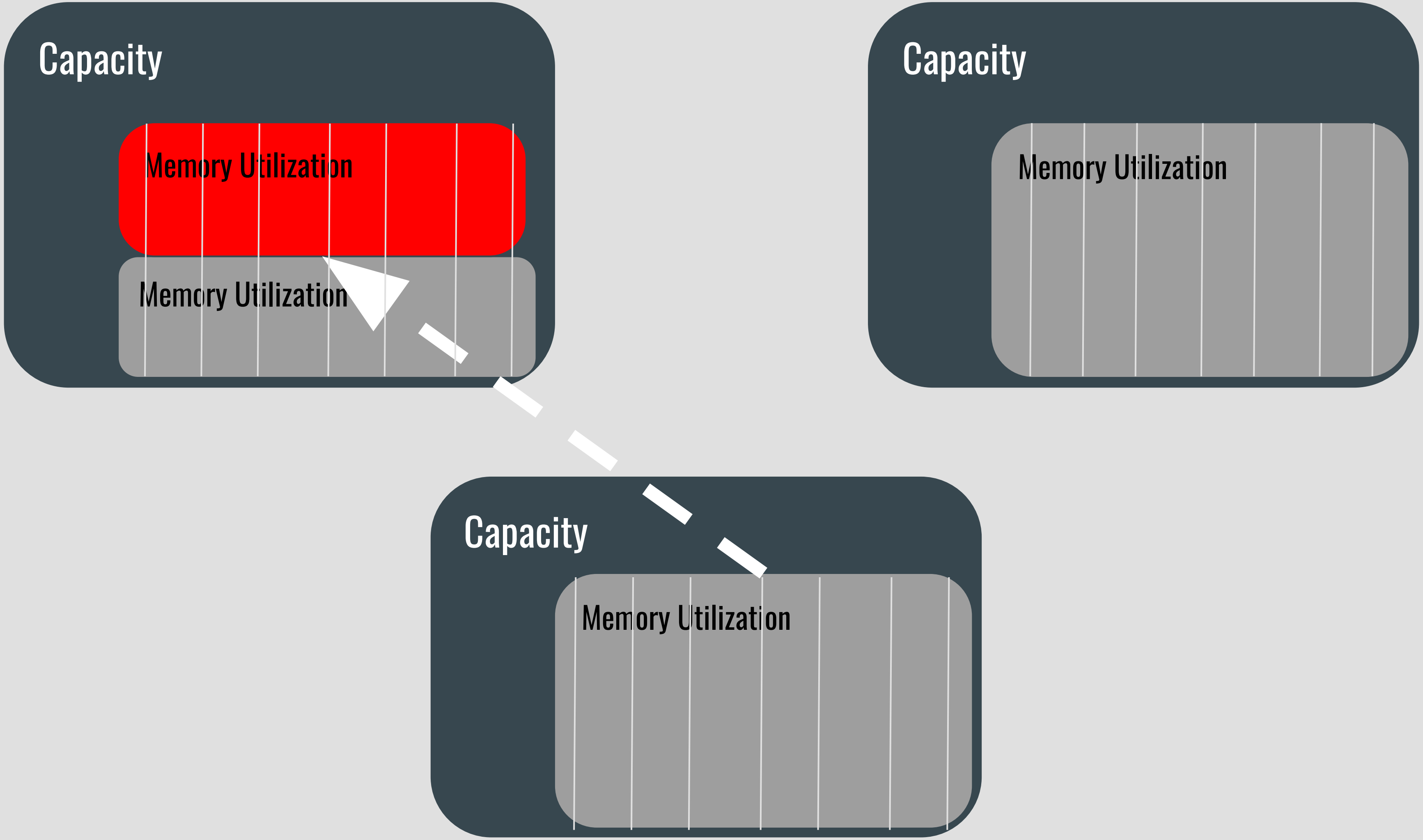
1.) Underutilization

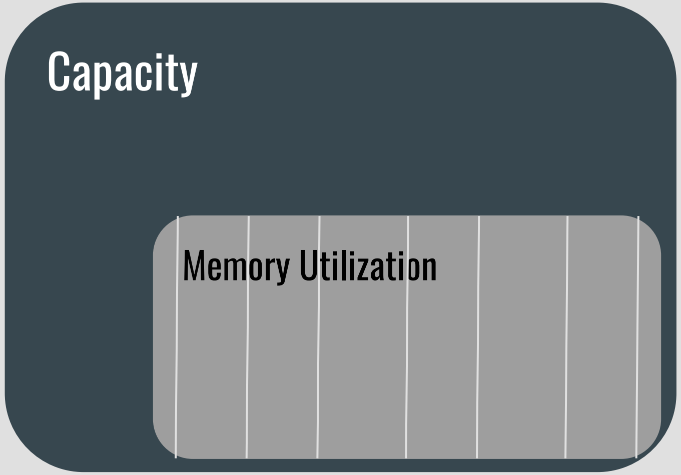


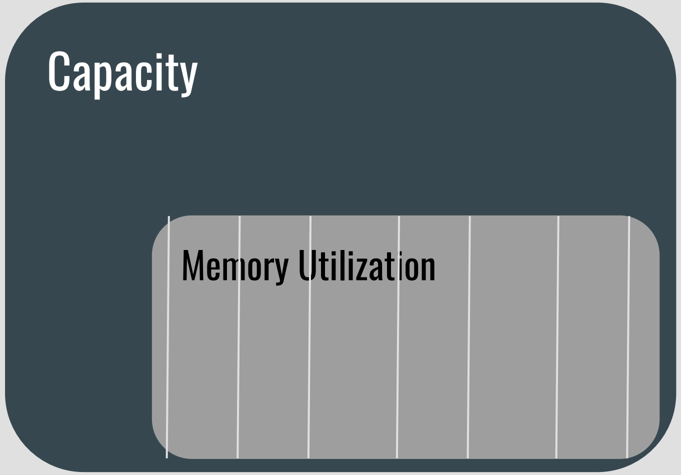
2.) High Utilization

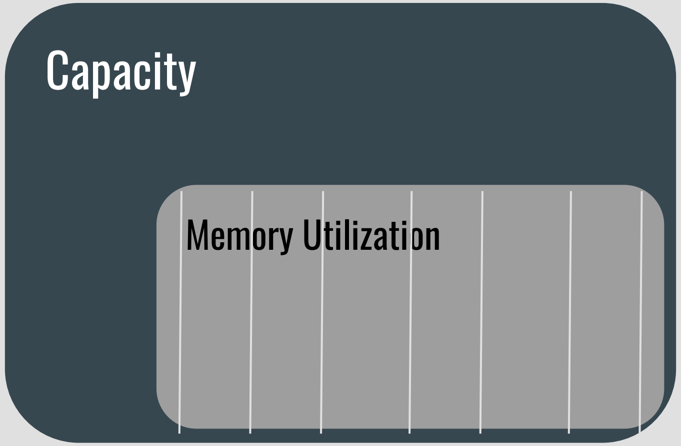


3.) Exceeding Capacity


Capacity
Memory Utilization
Memory Utilization
Capacity
Memory Utilization
Capacity
Memory Utilization

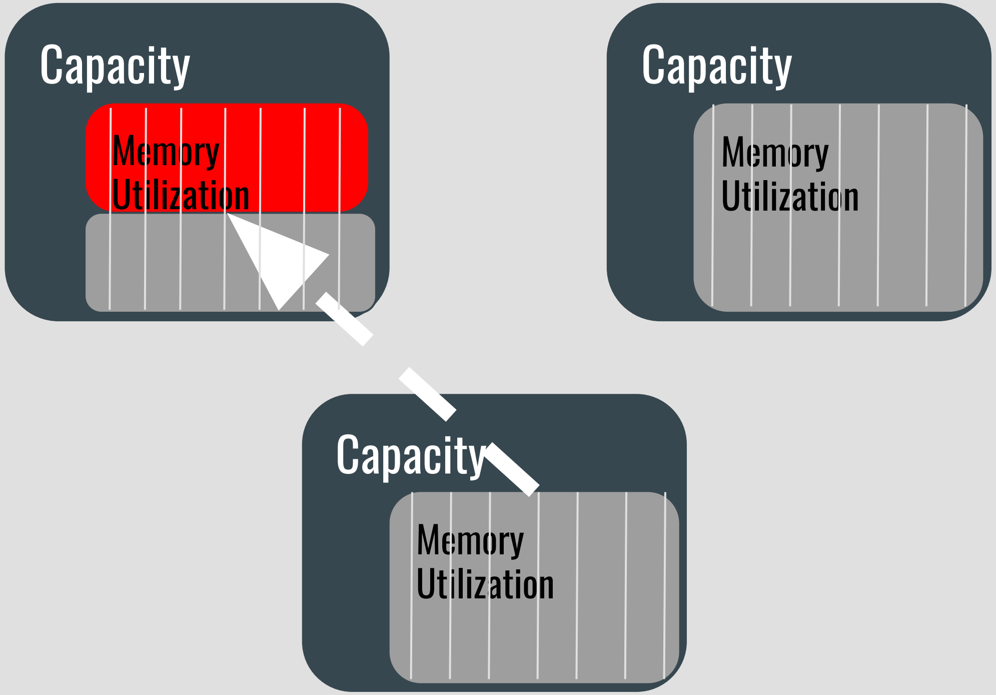

Capacity
Memory Utilization


Capacity
Memory Utilization


Capacity
Memory Utilization

Paging Problem

- Memory organized as "**pages**"
- When do you "**swap**" pages between servers?



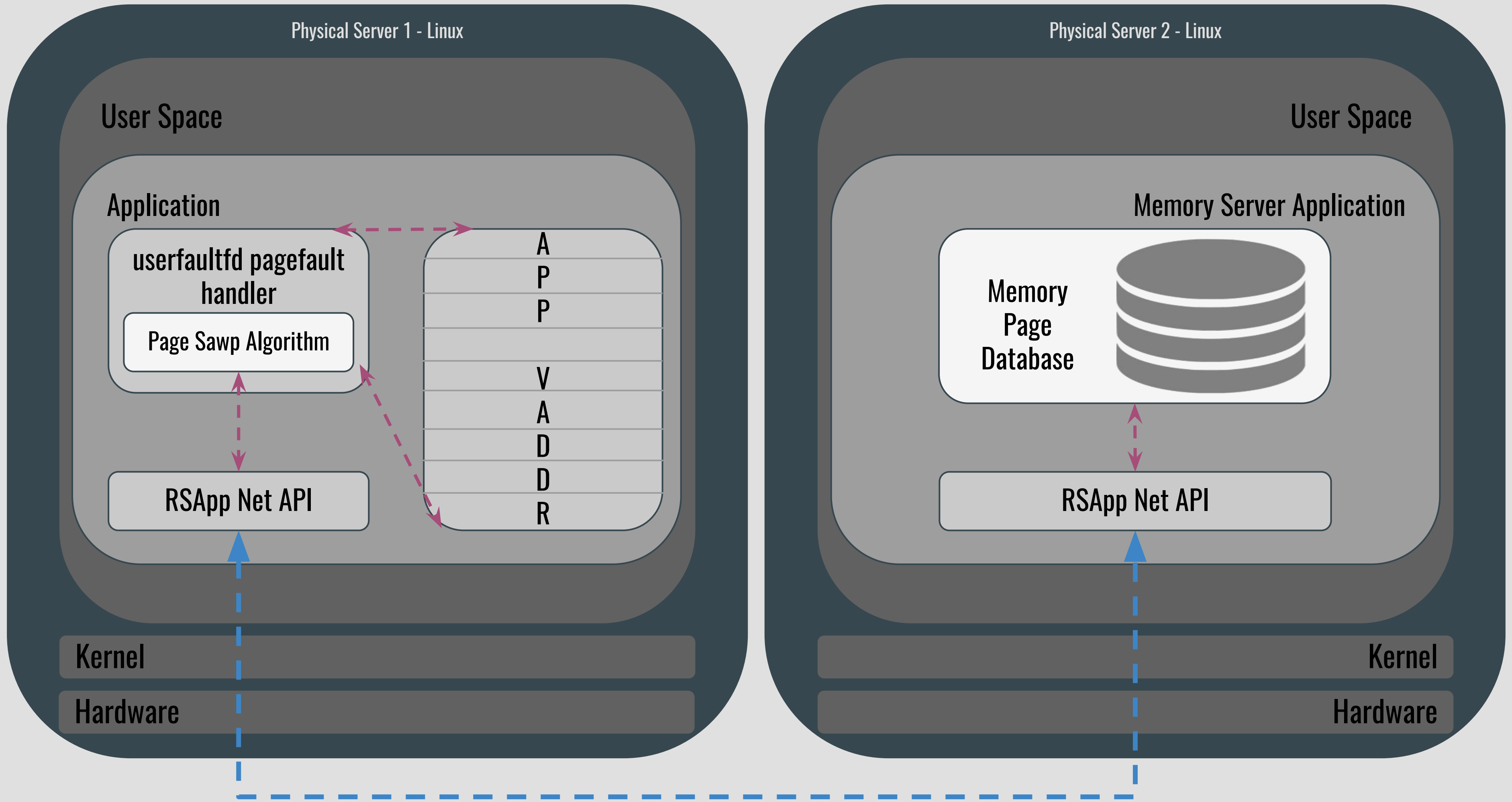
Rack Scale Apps

Rack Scale Apps

- **Pool** server hardware together in a rack
- Focusing only on **memory**
- **Page swapping** algorithm to intelligently manage memory

Rack Scale Apps

- **userfaultfd:** on demand paging and page fault management from userland
- **Memory Server:** remote storage for memory pages
- **RSApp Net API:** connects application to memory server

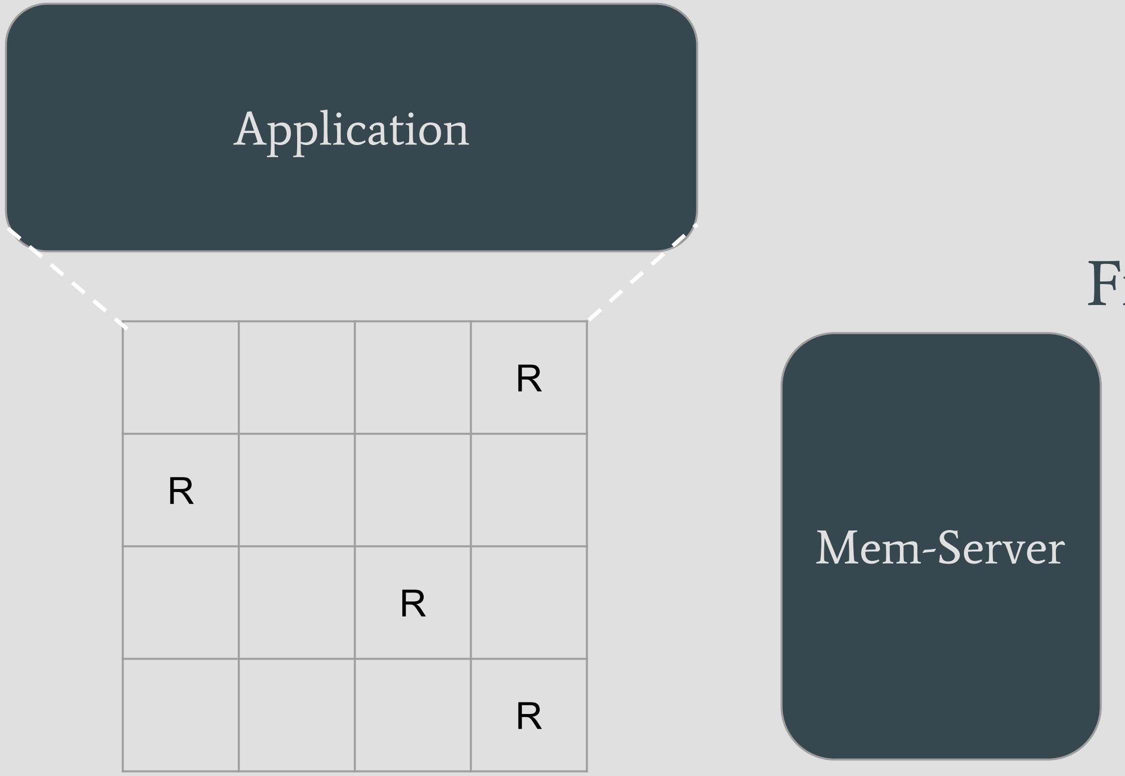

Physical Server 1 - Linux
User Space
Application
userfaultfd pagefault handler
Page Sawp Algorithm
RSApp Net API
A P P V A D D R
Kernel
Hardware
Physical Server 2 - Linux
User Space
Memory Server Application
Memory Page Database
RSApp Net API
Kernel
Hardware

Rack Scale Apps Demo

Page Swapping Algorithm

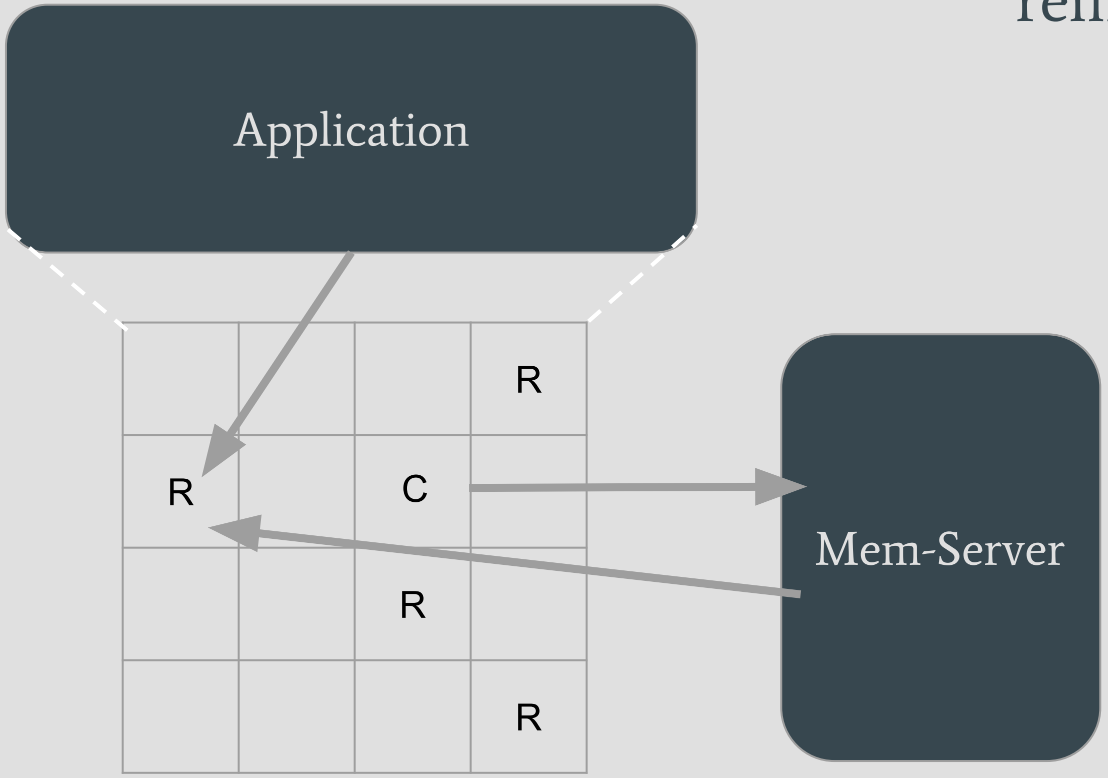
- Fools a process to think it has more memory
- Decides when to keep a page local or remote

Page Swapping Algorithm



First, mark some pages as **"remote"**

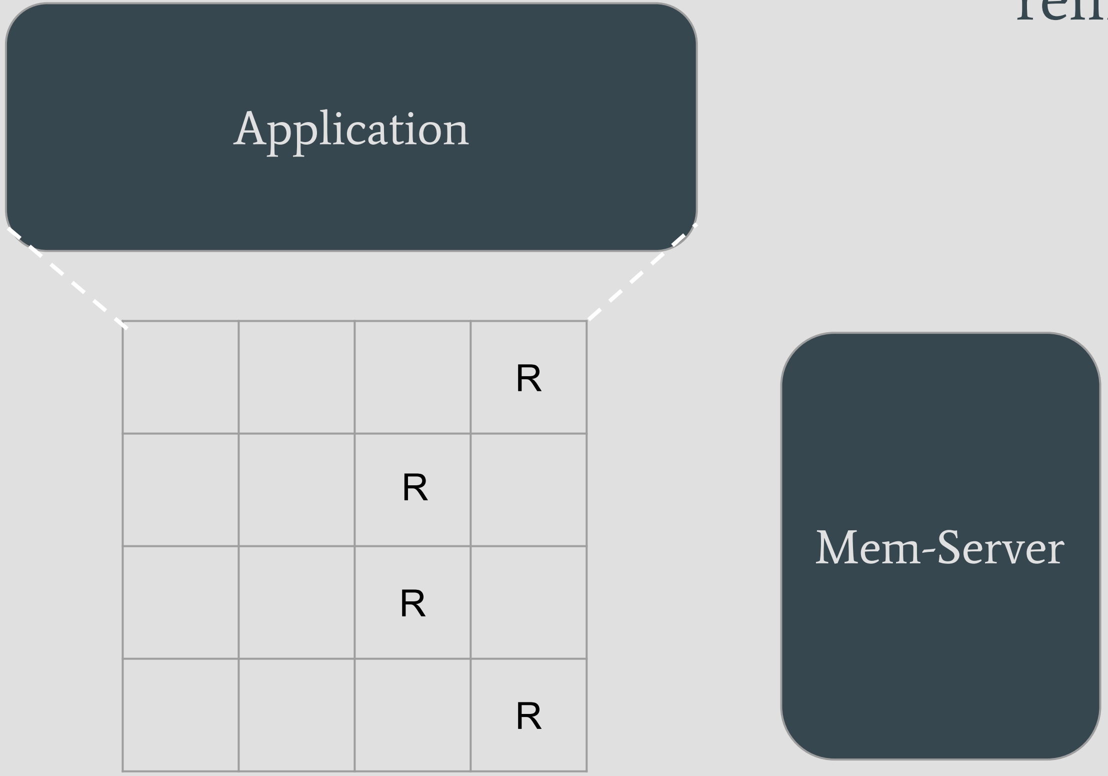
Page Swapping Algorithm



When Application accesses remote page:

1. Find coldest page
2. Swap its data to the mem-server
3. Check mem-server for remote page's data
4. Switch coldest page and remote page

Page Swapping Algorithm



When Application accesses remote page:

1. Find coldest page
2. Swap its data to the mem-server
3. Check mem-server for remote page's data
4. Switch coldest page and remote page

Analysis

- Throttle memory utilization of Memcached
  - **Low memory:** Memcached incurs low performance
  - **High memory:** Memcached incurs better performance
  - **Low memory + RSA:** Memcached performs closer to high memory case

Conclusion

- Efficient use of hardware can save lots of money
- Resource management is challenging
- Rack Scale Architecture saves the day

Rack Scale Apps

•••