Why Memory Is Your Next Bottleneck And How To Overcome It

Marcos K. Aguilera
Principal Researcher
VMware Research Group

Cloud bottlenecks

What blocks you from
using rest of the cloud effectively

The memory problem

Need	is growing
Capacity	is limited
Upgrade	is hard
Cost	is growing

The memory problem

Need	is growing
Capacity	is limited
Upgrade	is hard
Cost	is growing

The memory problem

Need	is growing
Capacity	is limited
Upgrade	is hard
Cost	is growing

Big Data moving to memory

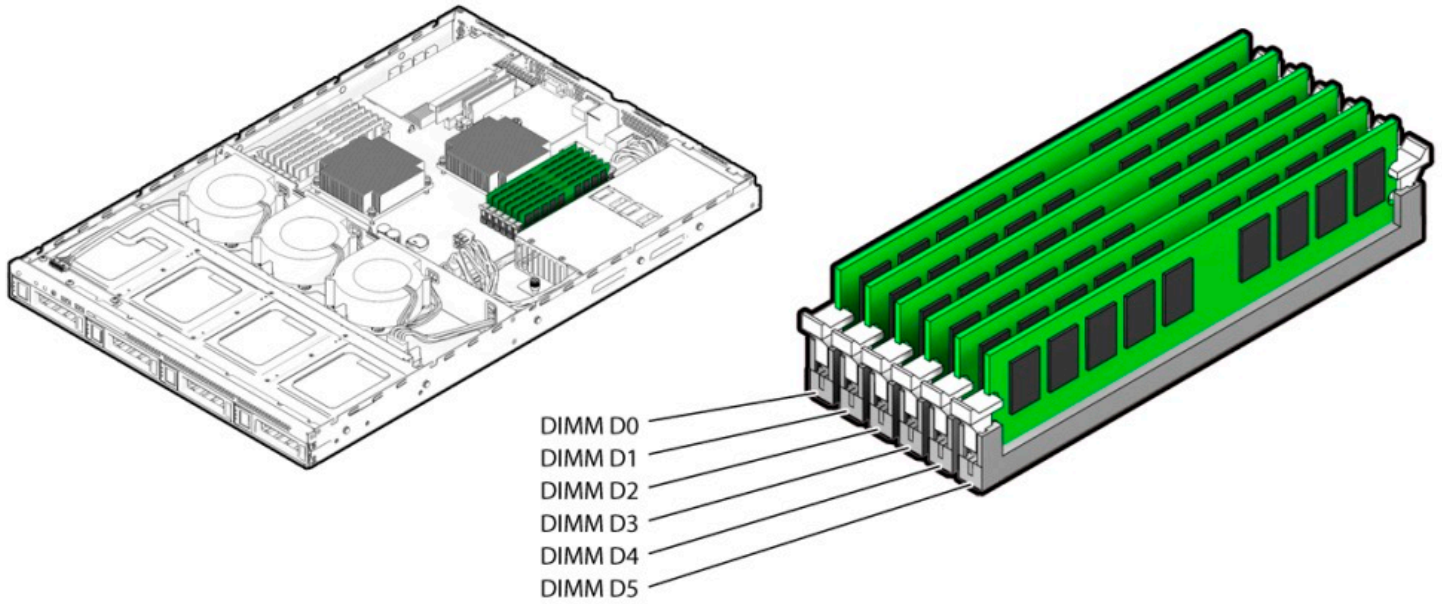
Many memory-hungry use cases

Hardware changes

Cores per CPU getting to hundreds

The memory problem

Need	is growing
Capacity	is limited
Upgrade	is hard
Cost	is growing



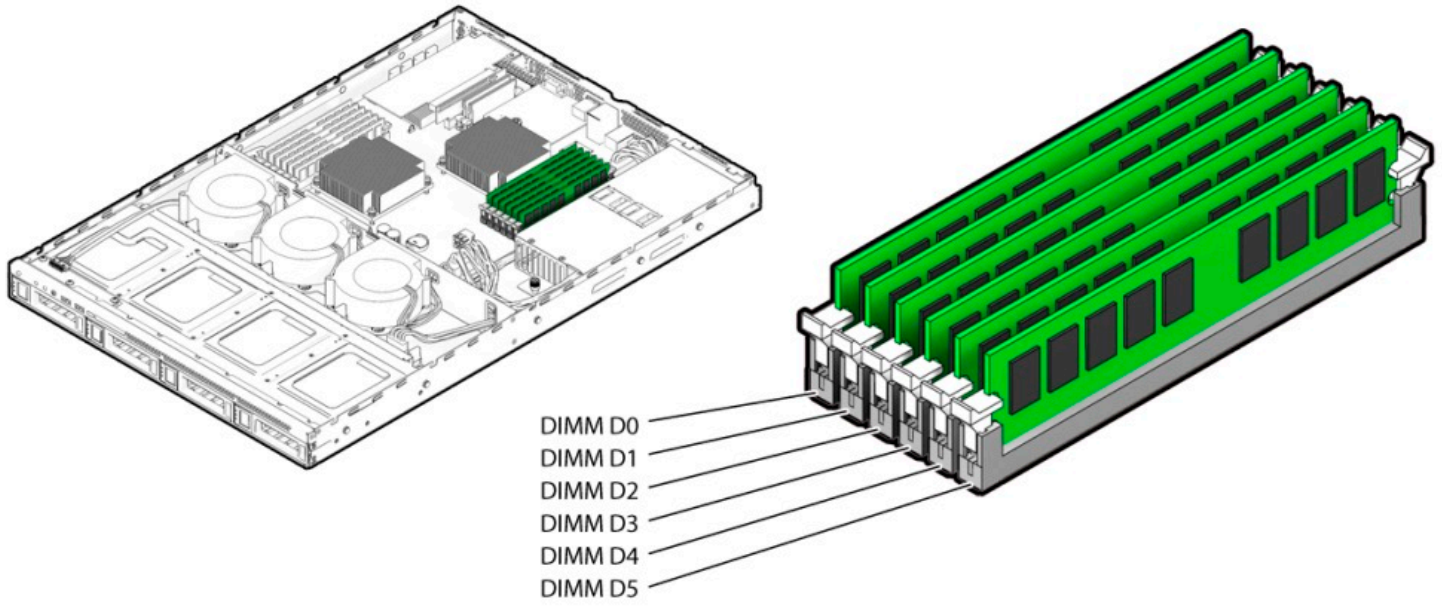
The memory problem

Need is growing

Capacity is limited

Upgrade is hard

Cost is growing



DIMM capacity	16 GB	32 GB	64 GB	128 GB	256 GB
$/GB	6.50	5.50	5.00	8.00	13.50

The memory problem

Need	is growing
Capacity	is limited
Upgrade	is hard
Cost	is growing

Conjunctural reasons

New memory-hungry uses

Structural reasons

Industry organization

Hardware to the rescue

Hardware to the rescue

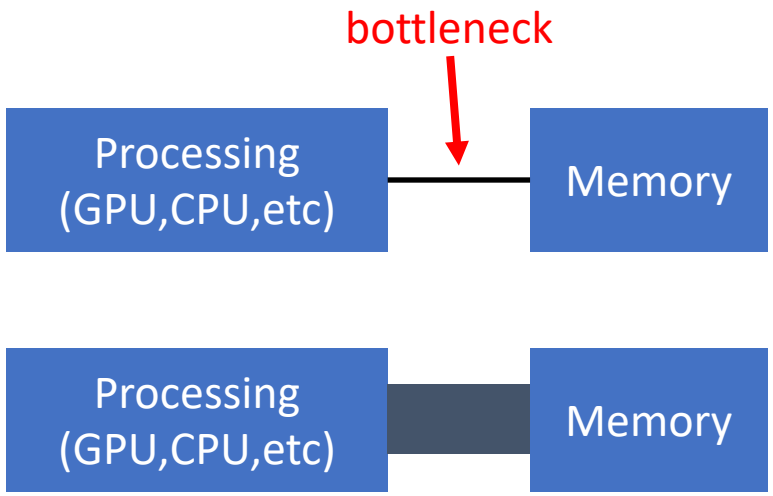
- New memory technologies
- High-bandwidth memory
- New memory interconnects
- Memory disaggregation

Hardware to the rescue

- New memory technologies — 3D Xpoint, MRAM, FeRAM
- High-bandwidth memory
- New memory interconnects
- Memory disaggregation

Hardware to the rescue

- New memory technologies
- High-bandwidth memory
- New memory interconnects
- Memory disaggregation



Hardware to the rescue

- New memory technologies
- High-bandwidth memory
- New memory interconnects
- Memory disaggregation

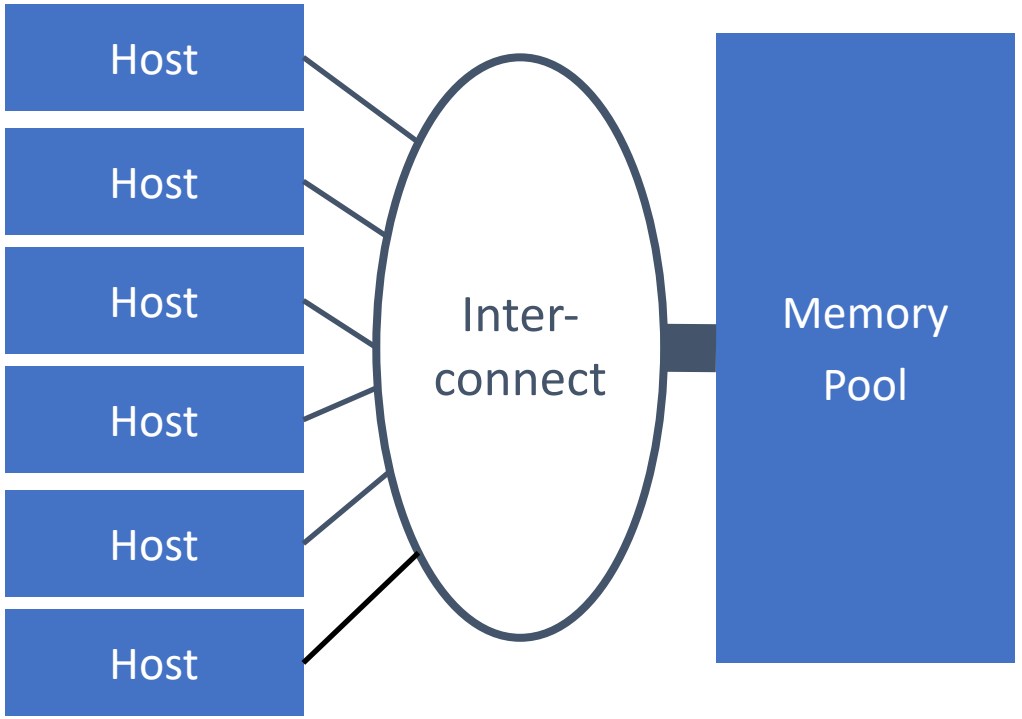
Compute Express Link (CXL)

Extra memory

Extra bandwidth

Hardware to the rescue

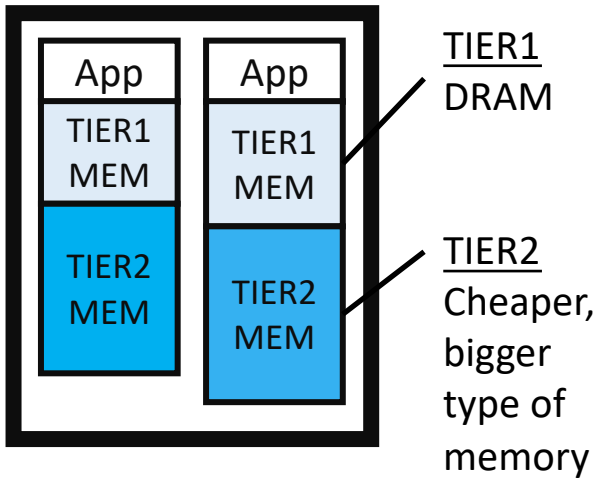
- New memory technologies
- High-bandwidth memory
- New memory interconnects
- Memory disaggregation



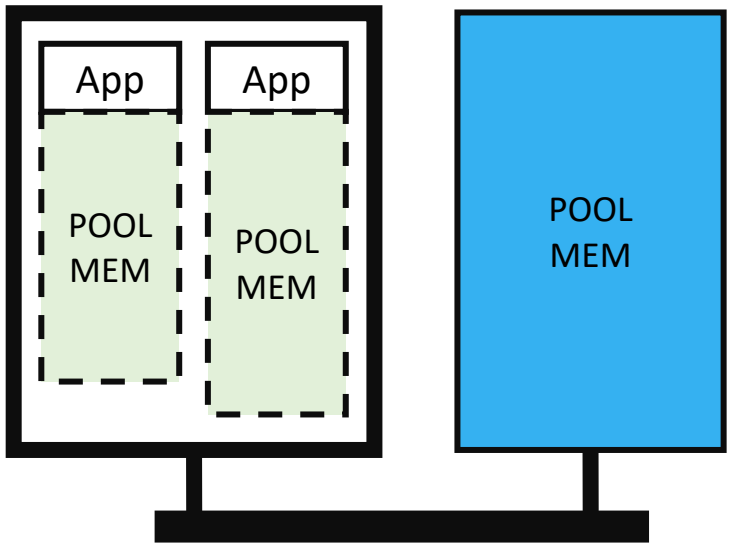
Systems research to the rescue

Two high-level ideas

- **Memory Tiering**

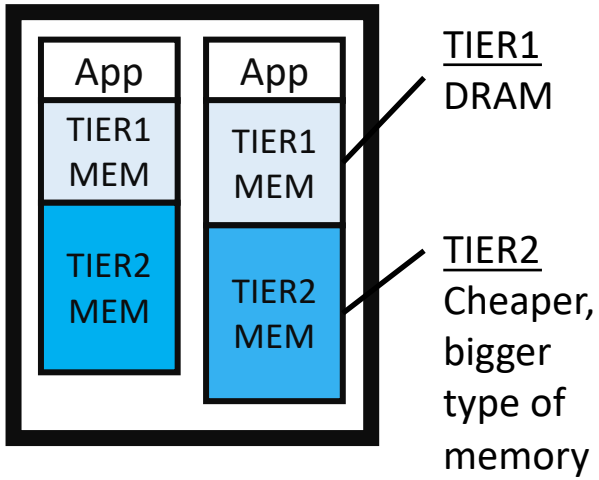


- Memory Pooling

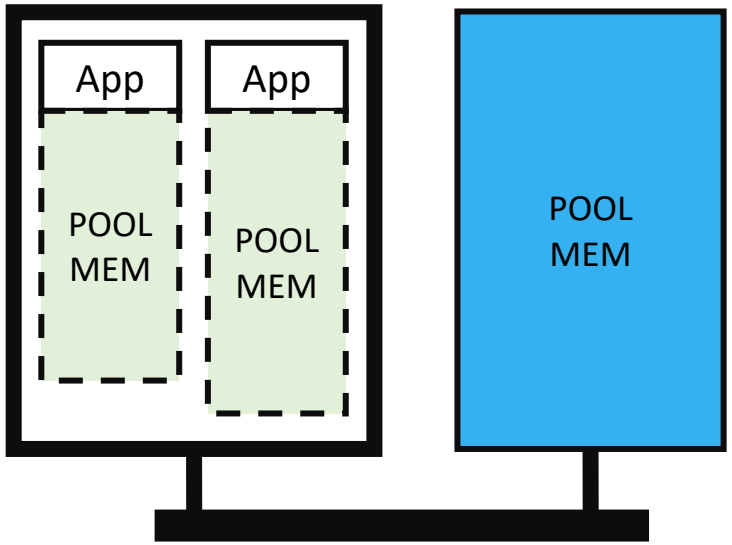


Two high-level ideas

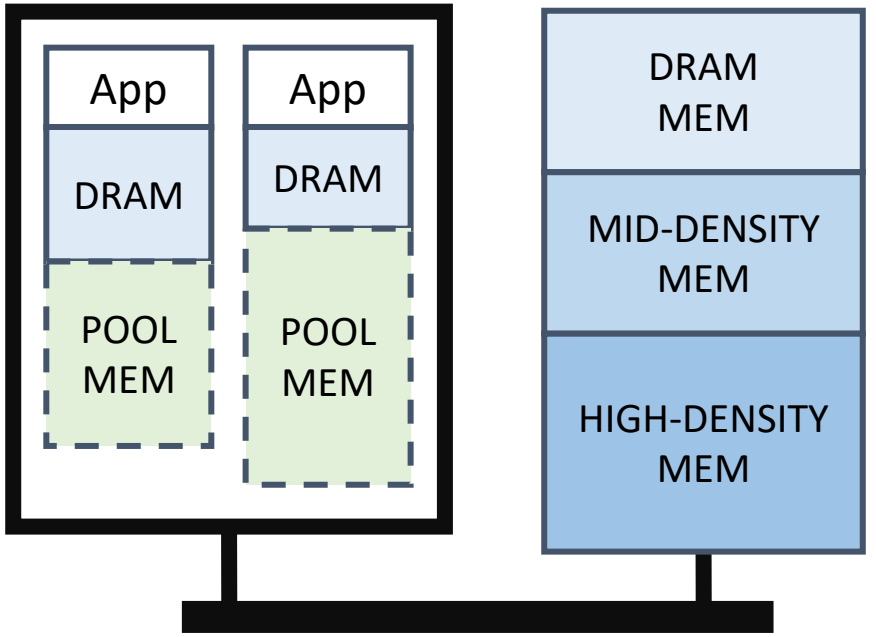
- Memory Tiering



- **Memory Pooling**



Tiering and Pooling



Work in this space

- **System-level Implications of Disaggregated Memory.**
  Kevin Lim, Yoshio Turner, Jose Renato Santos, Alvin AuYoung, Jichuan Chang, Parthasarathy Ranganathan, Thomas F. Wenisch.
  HPCA 2012

- **Network Requirements for Resource Disaggregation.**
  Peter X. Gao, Akshay Narayan, Sagar Karandikar, Joao Carreira, Sangjin Han, Rachit Agarwal, Sylvia Ratnasamy, Scott Shenker.
  OSDI 2016

- **Remote Memory in the Age of Fast Networks.**
  Marcos K. Aguilera, Nadav Amit, Irina Calciu, Xavier Deguillard, Jayneel Gandhi, Pratap Subrahmanyam, Lalith Suresh, Kiran Tati, Rajesh Venkatasubramanian, Michael Wei.
  SoCC 2017

- **Thermostat: Application-transparent page management for two-tiered main memory.**
  Neha Agarwal ,Thomas F. Wenisch
  ASPLOS 2017

- **Efficient Memory Disaggregation with Infiniswap.**
  Juncheng Gu, Youngmoon Lee, Yiwen Zhang, Mosharaf Chowdhury, Kang Shin
  NSDI 2017

LegoOS: A Disseminated, Distributed OS for Hardware Resource Disaggregation.
Yizhou Shan, Yutong Huang, Yilun Chen, Yiying Zhang.
OSDI 2018

Remote Regions: a Simple Abstraction for Remote Memory.
Marcos K. Aguilera, Nadav Amit, Irina Calciu, Xavier Deguillard, Jayneel Gandhi, Stanko Novakovic, Arun Ramanathan, Pratap Subrahmanyam, Lalith Suresh, Kiran Tati, Rajesh Venkatasubramanian, Michael Wei.
ATC 2018

Software-Defined Far Memory in Warehouse-Scale Computers.
Andres Lagar-Cavilla, Junwhan Ahn, Suleiman Souhlal, Neha Agarwal, Radoslaw Burny, Shakeel Butt, Jichuan Chang, Ashwin Chaugule, Nan Deng, Junaid Shahid, Greg Thelen, Kamil Adam Yurtsever, Yu Zhao, Parthasarathy Ranganathan
ASPLOS 2019

Can far memory improve job throughput?
Emmanuel Amaro, Christopher Branner-Augmon, Zhihong Luo, Amy Ousterhout, Marcos K. Aguilera, Aurojit Panda, Sylvia Ratnasamy, Scott Shenker.
EuroSys 2020

AIFM: High-Performance, Application-Integrated Far Memory.
Zhenyuan Ruan, Malte Schwarzkopf, Marcos K. Aguilera, Adam Belay.
OSDI 2020

Memory Tiering

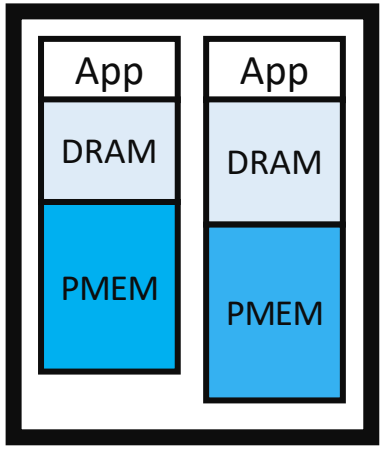
- What Tiers to Use
- How Tiers are Exposed

Memory Tiering: What Tiers to Use

Type:	Physical vs Functional
Cost:	$3 to $17/GB
Latency:	100ns to 4us
BW:	10s to 100s GBps
Access:	Mapped, Paged, Custom

Memory Tiering: What Tiers to Use

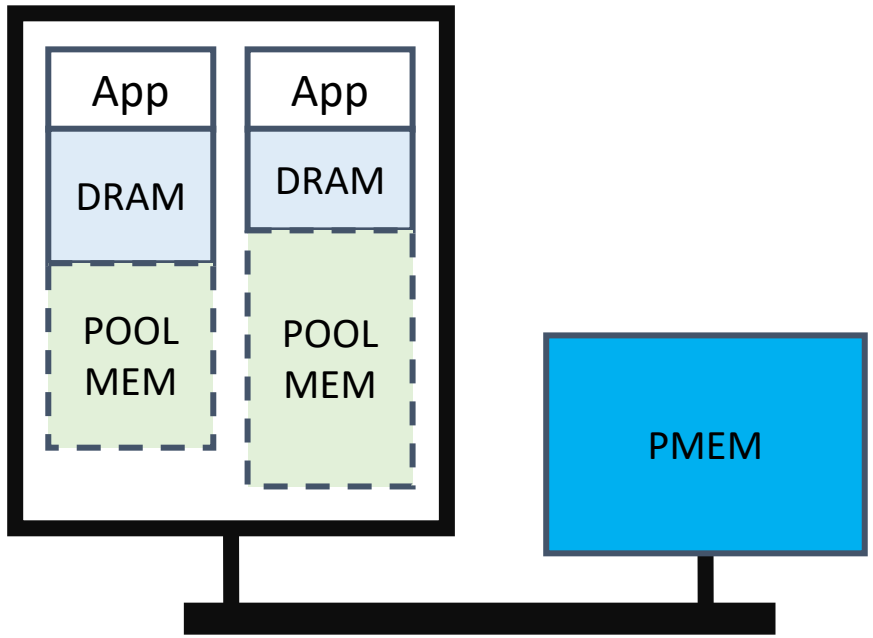
Example 1



DIMM capacity	16 GB	32 GB	64 GB	128 GB	256 GB	512 GB
RAM $/GB	6.50	5.50	5.00	8.00	13.50	—
PMEM $/GB	—	—	—	4.50	7.50	16.50

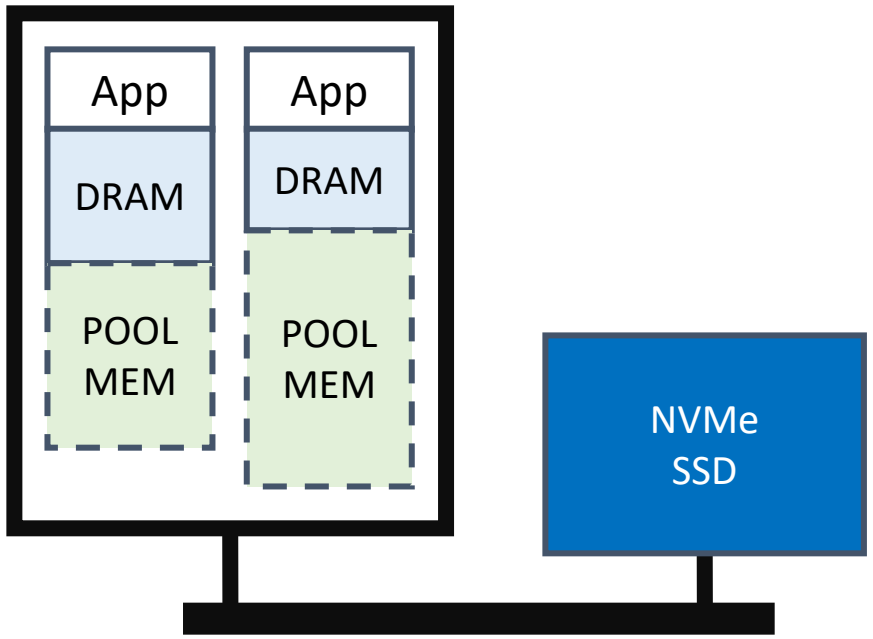
Memory Tiering: What Tiers to Use

Example 2



Memory Tiering: What Tiers to Use

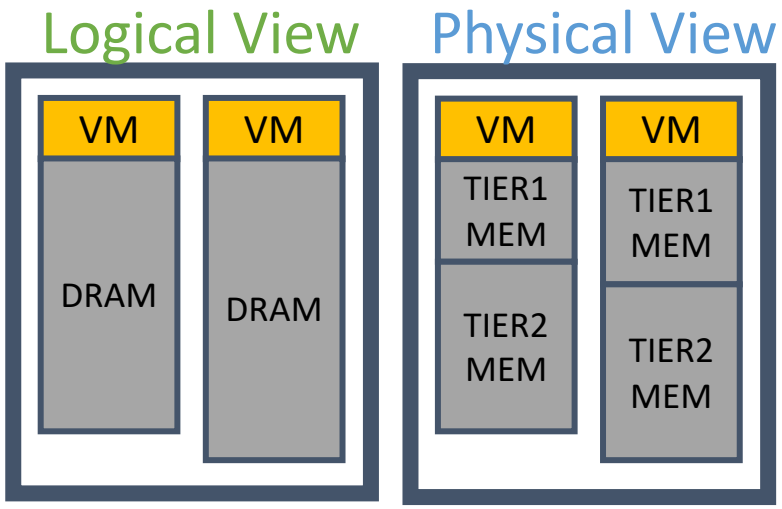
Example 3



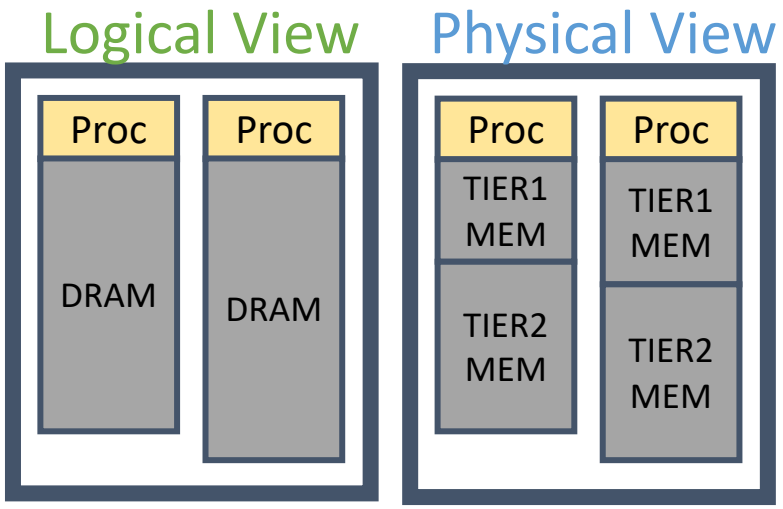
DIMM capacity	16 GB	32 GB	64 GB	128 GB	256 GB
$/GB	6.50	5.50	5.00	8.00	13.50

SSD capacity	375 GB	750 GB	1.5 TB
$/GB	3.30	3.60	3.80

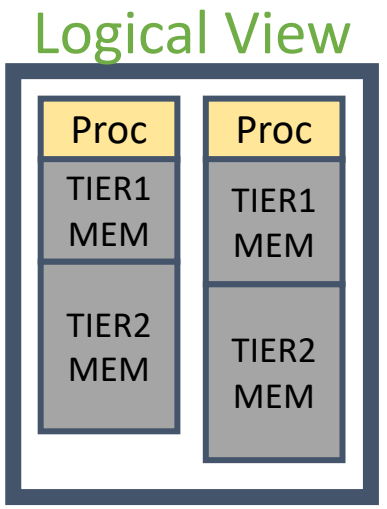
Memory Tiering: How Tiers are Exposed



Hypervisor handles tiering
Transparent to OS and process

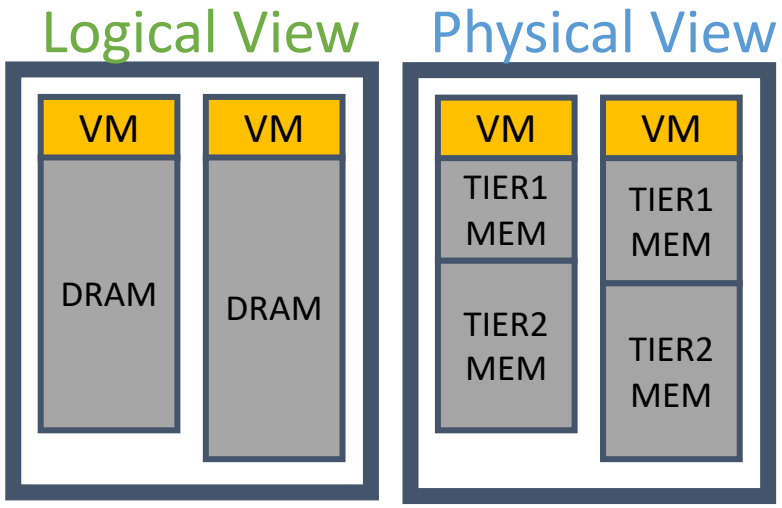


OS handles tiering
Transparent to process

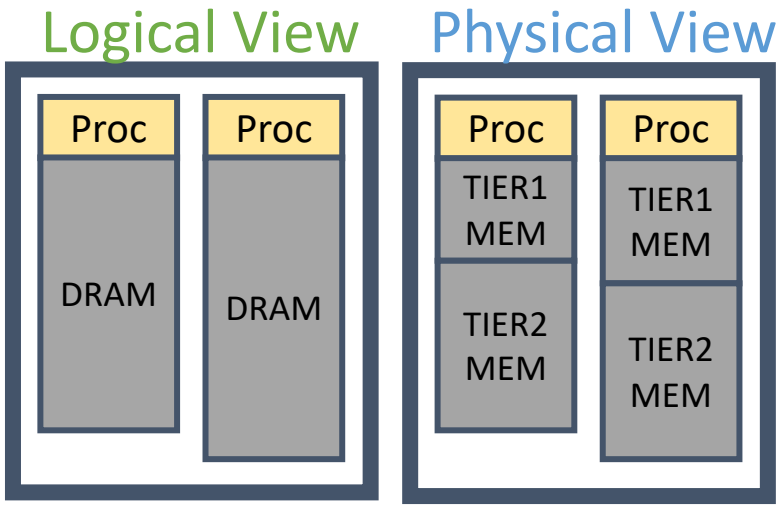


Process handles tiering
➔ new app interfaces
Remote Regions
AIFM

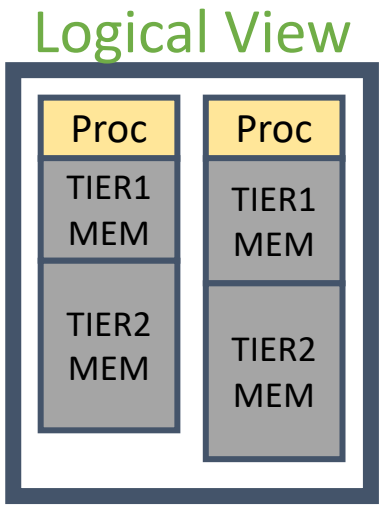
Memory Tiering: How Tiers are Exposed



Hypervisor handles tiering
Transparent to OS and process

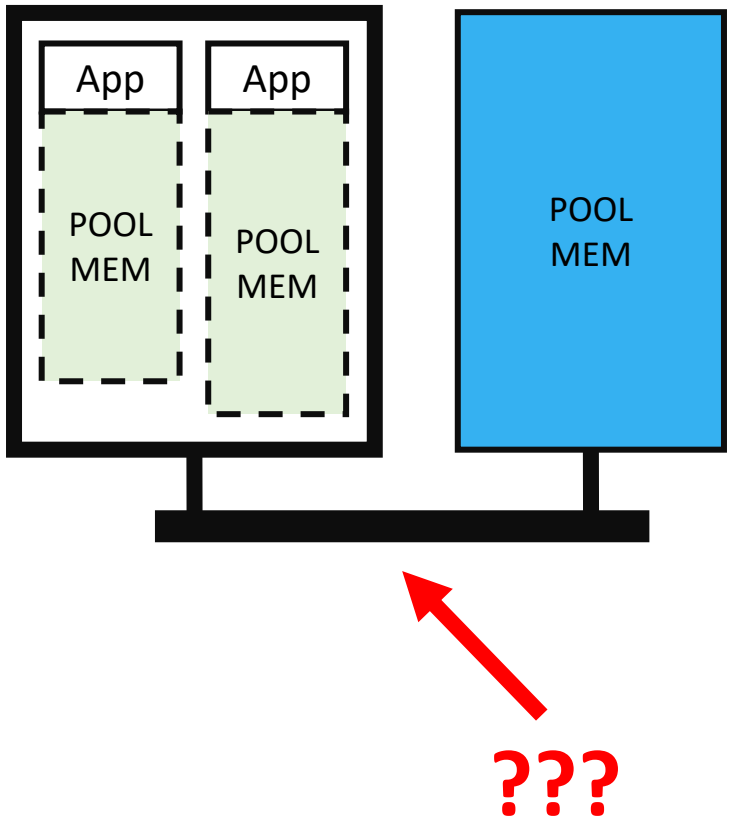


OS handles tiering
Transparent to process
LATER IN TALK



Process handles tiering
➔ new app interfaces
Remote Regions
AIFM
LATER IN TALK

Memory Pooling: Interconnect



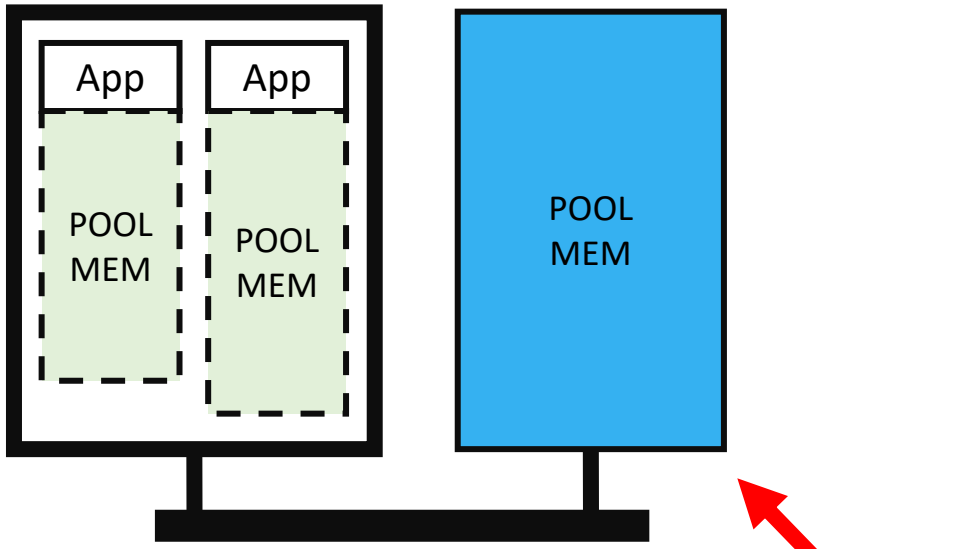
Choices

RDMA

Gen-Z

CXL 2.0

Memory Pooling: The Pool



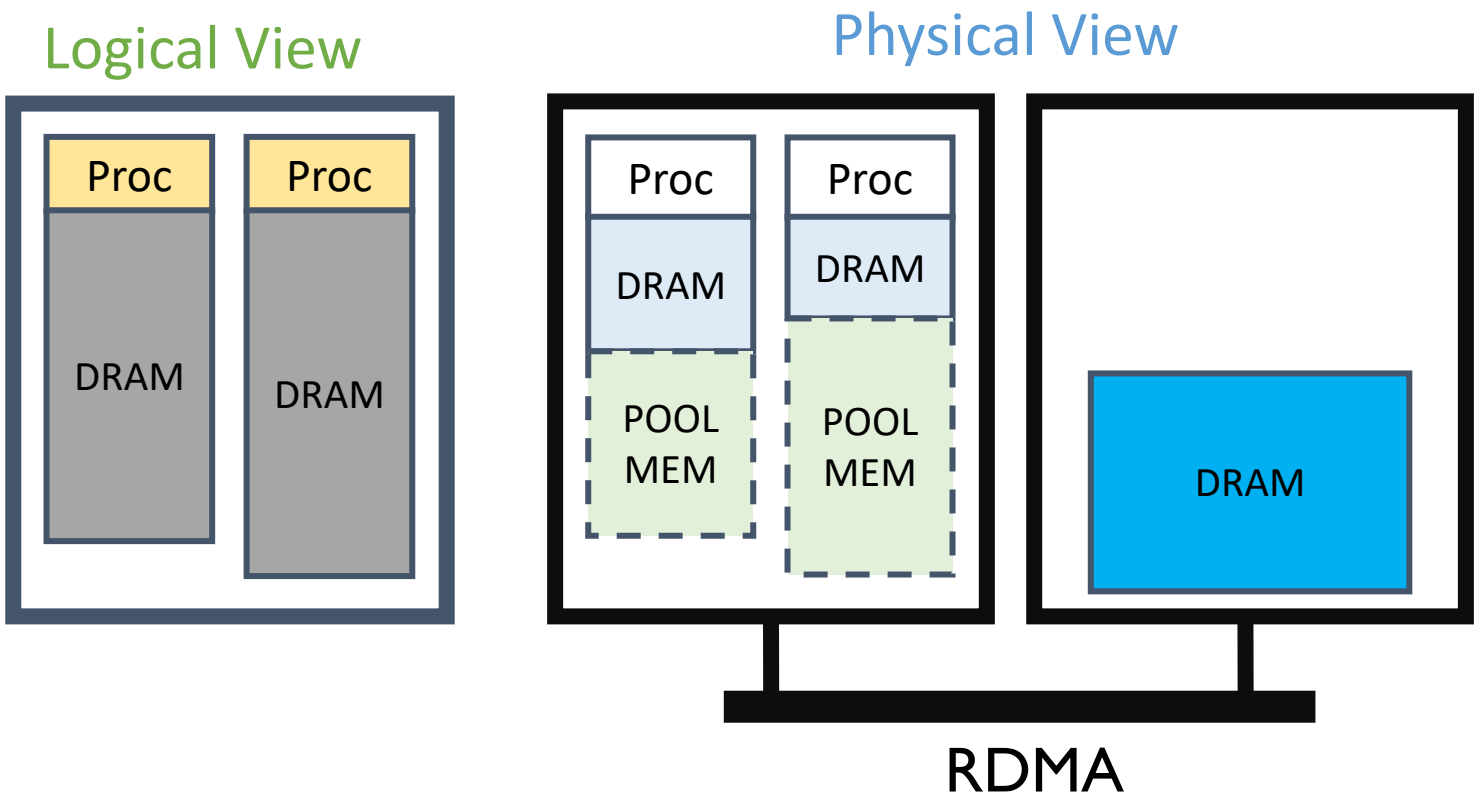
Choices

Another host

Memory server

JBOM

Fastswap EUROSYS'20

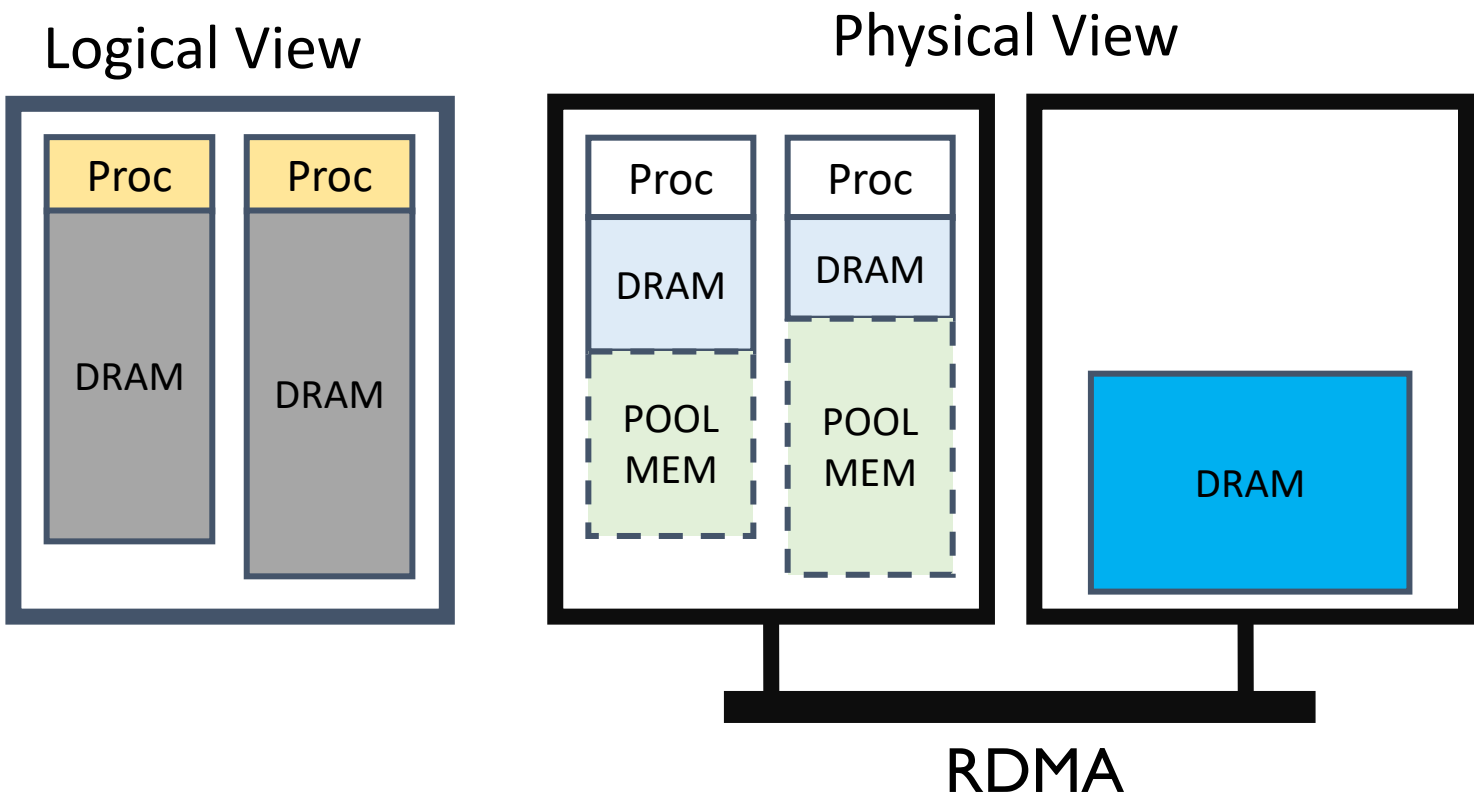


Transparent tiering with OS paging

Traditional paging slow for RDMA

1. Head-of-line blocking
2. Asynchronous page reads
3. Reclamation during faults

Fastswap



Transparent tiering with OS paging

Traditional paging slow for RDMA

1. Head-of-line blocking
2. Asynchronous page reads
3. Reclamation during faults

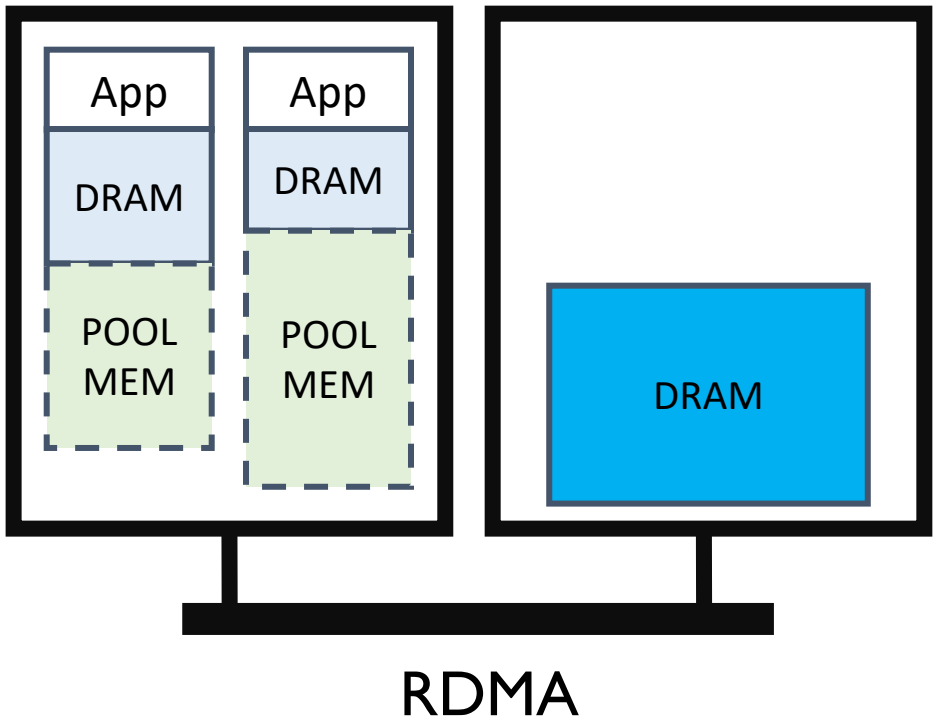
Optimize paging in Linux

1. Multiple RDMA queues
2. Frontswap for sync faults
3. Dedicated core to reclaim

New App Interfaces for Tiering

- Expose tiering to apps
- Aimed at new apps
- What should they see?

Remote Regions Interface ATC'18



Better interface to access memory over RDMA

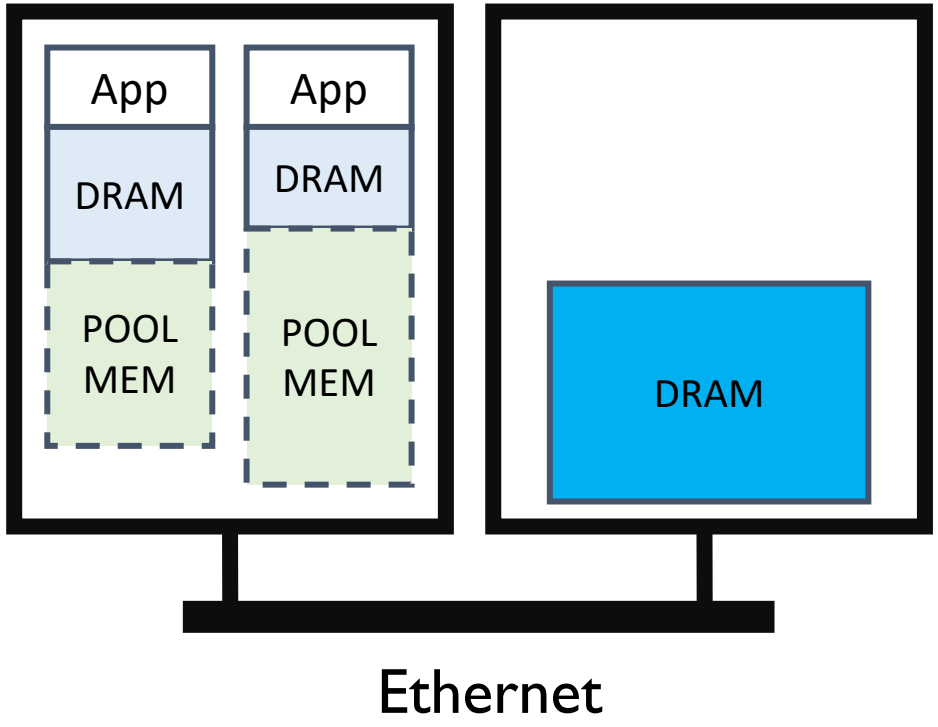
RDMA too hard to use

Introduce new interface to replace RDMA

1. Everything is a file
2. RegionFS

```
fd=open("/regions/ez", O_RDWR);
…
ptr=rmalloc(fd, 1024)
sprintf(ptr, "hello world");
```

Application Integrated Far Memory OSDI'20



Move tier gating to app library

Page faults are too slow

Remoteable pointers

Dereference scopes

Pauseless memory evacuator

Throughput (accesses/s)

	64B object	4KB object
OS Paging	582K	582K
AIFM	3975K	1059K

Key ideas in AIFM

Remoteable Pointer

```
RemUniquePtr<T> rptr;
T* ptr;
...
ptr = rptr.deref();
```

Dereference Scope

```
{
  DerefScope scope;
  ptr = rptr.deref(scope);
  ... use ptr ...
}
// ptr now invalid
```

Pauseless mem evac

When local memory is low
And object out of scope
Pick object to swap out

Conclusion

- Memory is becoming major pain in data centers
- Solution is tiering and pooling
- Tiering leverages cheaper bigger memory types
- Pooling disaggregates memory
- Transparent approaches have a cost
- Better performance if willing to change apps