Memory Management in the Cloud

Pierre Louis Aublin
IIJ Research Laboratory
2023年12月19日

Meet Tanaka san

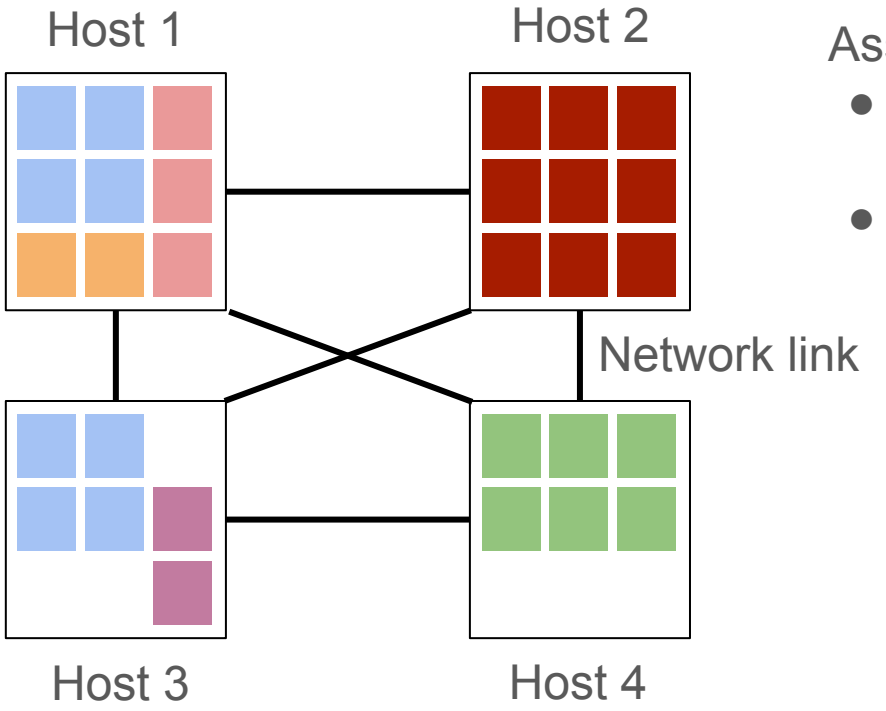
- Software engineer at **Super Infinity Cloud Provider (IaaS)**

SUPER INFINITY CLOUD
PROVIDER

Tanaka san in charge of VM infrastructure

- VM allocation
- VM migration
- Efficient resource utilization
- Fault-tolerance
- Isolation and security

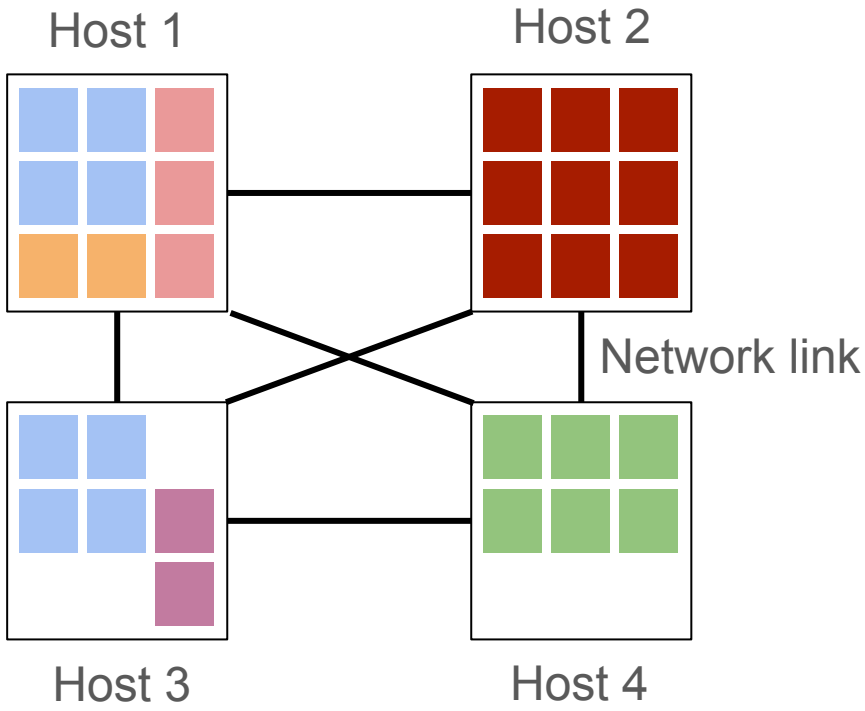
VM Memory usage example



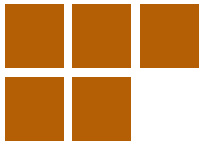
Assumptions:

- Each host has 3 CPU cores
- At most 3 VMs machine per host

A new VM needs to be allocated

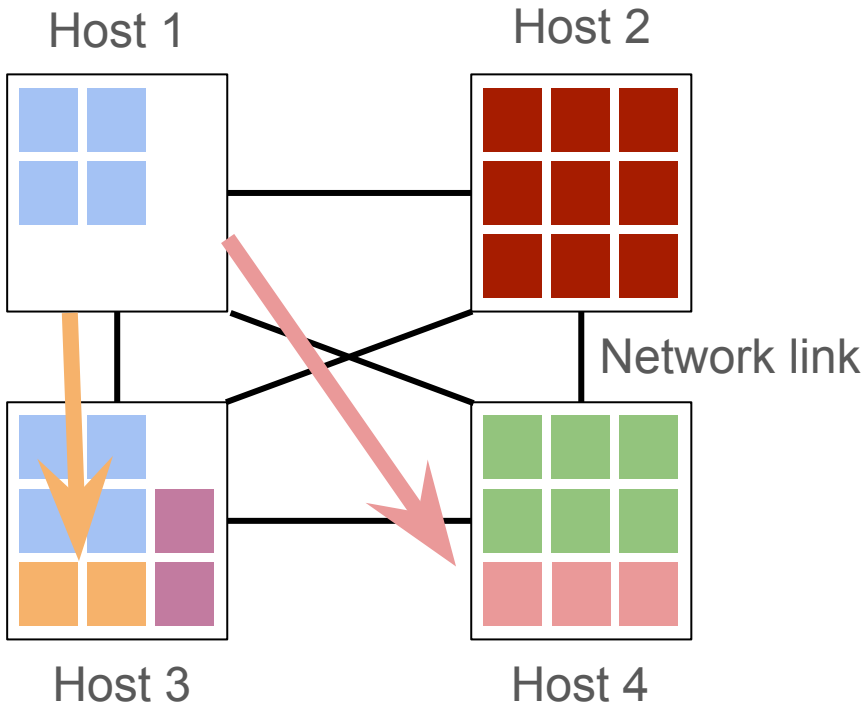


New VM!

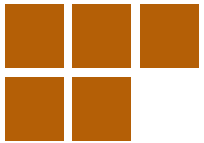


Where to allocate?

Migrate VMs

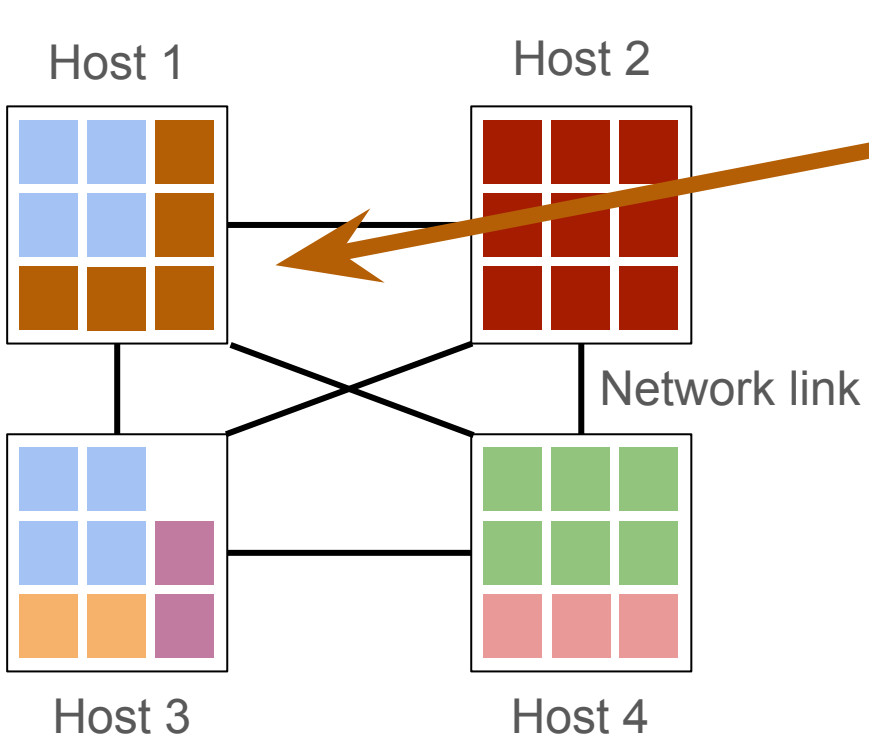


New VM!



Where to allocate?

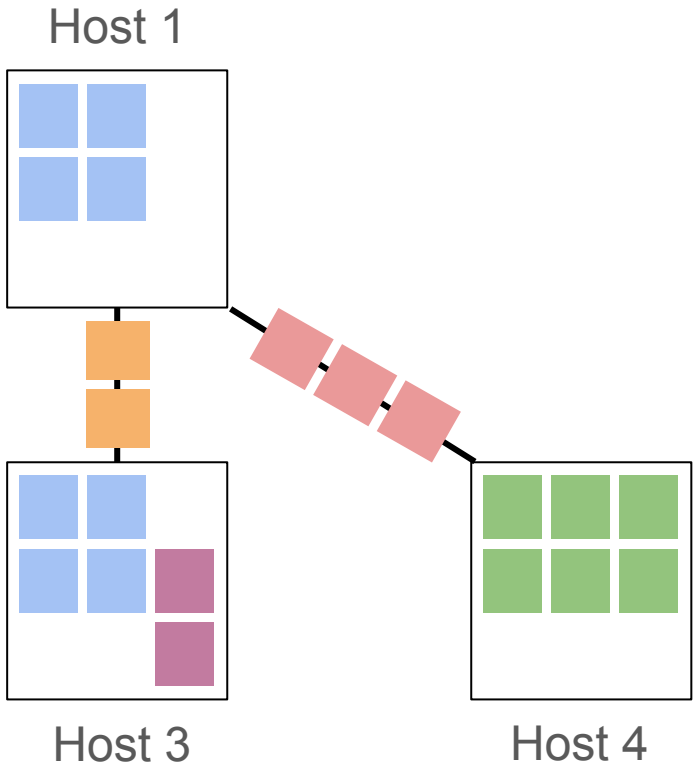
Allocate new VM



New VM!

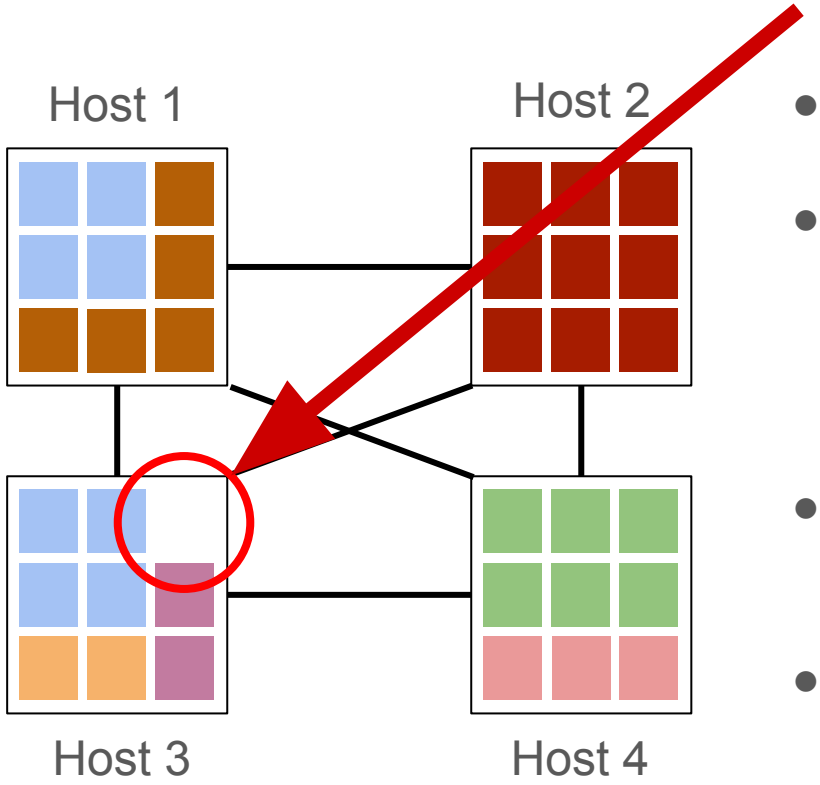
Success!

Is migration the solution?



- Need to move data/computation across the cluster
- Interruption of service for the migrated VMs

Cluster is full; Memory stranding appears…



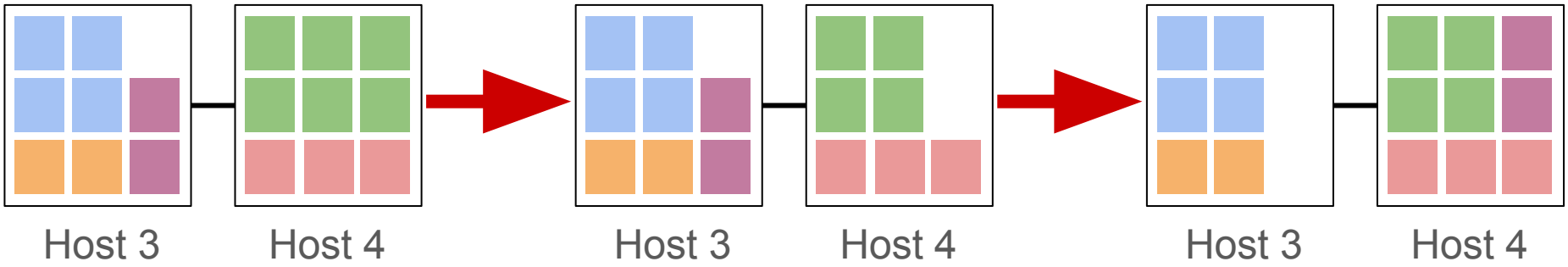
- Unused memory
- Cannot be used to allocate a new VM
  - "All cores have been rented, but there is still free memory on the server"
- Microsoft Azure reports **>25%** of stranded memory

  Li et al., "Pond: CXL-Based Memory Pooling Systems for Cloud Platforms", ASPLOS 2023
- Studies at Microsoft, Google, Alibaba show **50%** of memory is not utilized

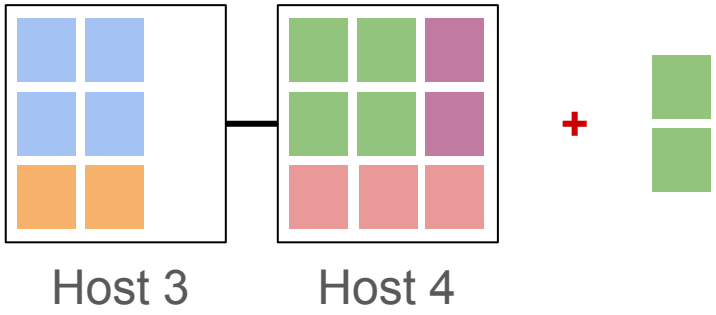
  Zhang et al., "Redy: Remote Dynamic Memory Cache", 2021

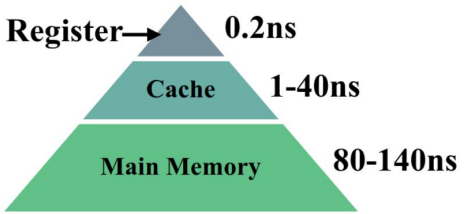
Let's solve memory stranding via memory pooling!

- Memory pooling
  - Dynamic memory allocation scheme
  - Divides system memory into blocks
  - Each block can be allocated to / reclaimed from a VM to follow memory usage



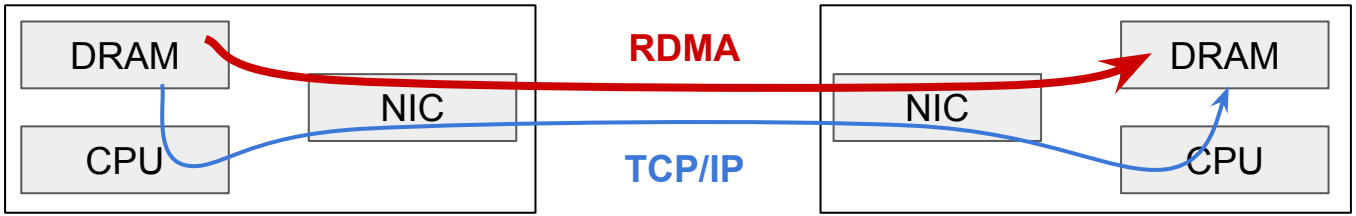
Where to store the reclaimed memory?



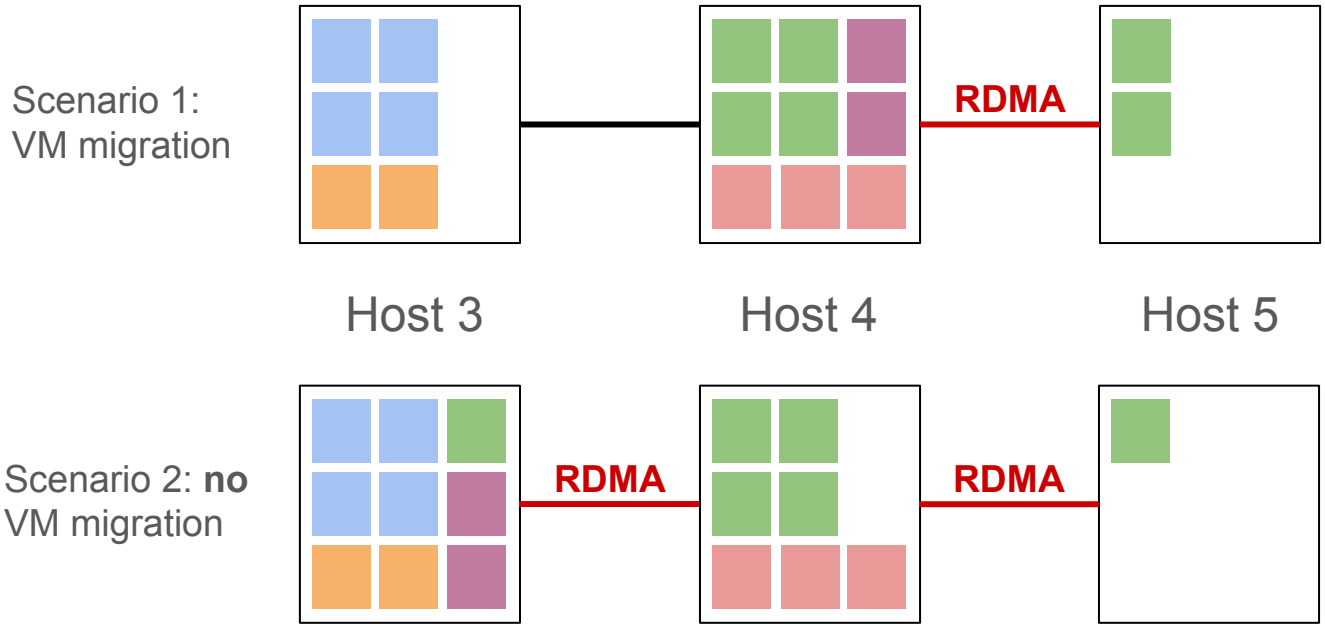


Network-attached memory

- Remote Direct Memory Access
  - Access a computer's memory from another computer
  - Operating System/CPU not involved
  - Low-latency operation



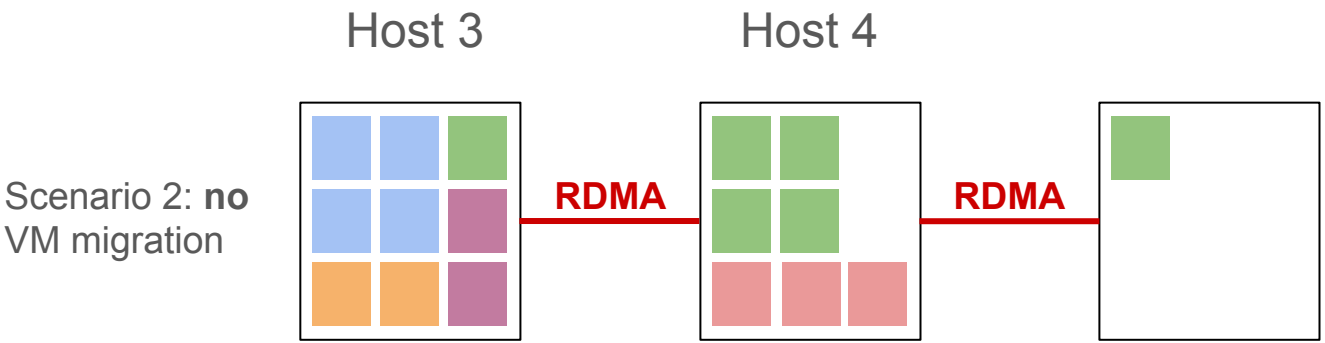
VM memory access via RDMA



- Access to  VM reclaimed memory ("cold memory") incurs latency penalty compared to DRAM access

Memory disaggregation

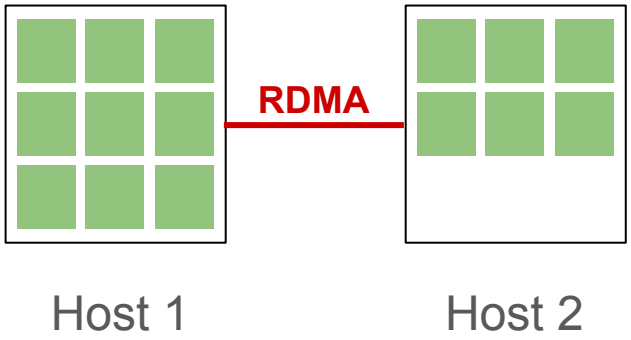
- Separate memory from the compute resources
- Memory of a single machine can be shared across a network of servers
- Improves memory utilization and scalability



Leap: a solution to manage remote memory

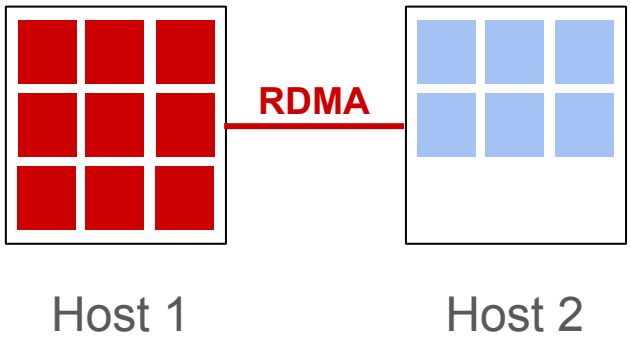
- "Effectively Prefetching Remote Memory with Leap"
  - Al Maruf and Chowdhury. Michigan university
  - USENIX ATC 2020

- Problem
  - Remote memory access is slow (µs compared to ns)
  - This slows down memory intensive applications
  - How do we choose which data to allocate remotely?



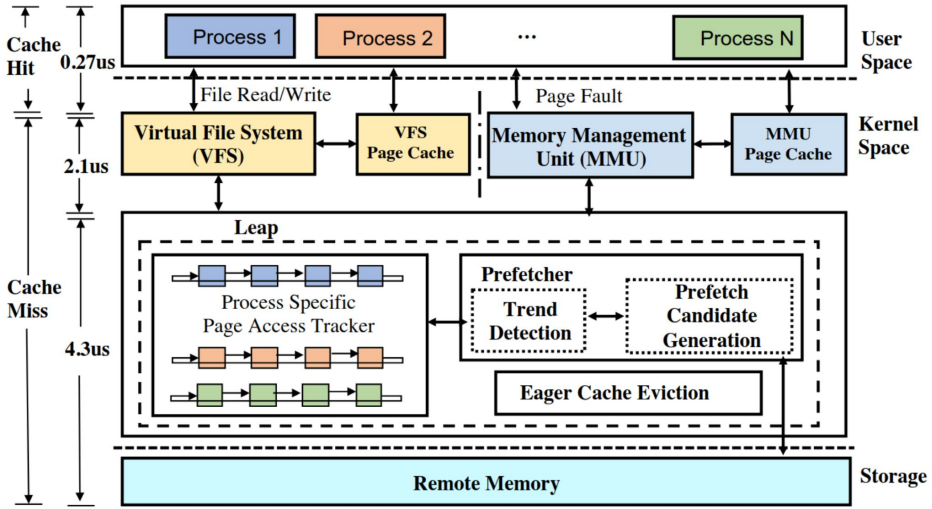
Hot vs Cold memory pages

- Hot memory page
  - Frequent access
  - Move to faster, local memory for better performance
- Cold memory page
  - Infrequent access
  - Can be moved to slower, farther memory



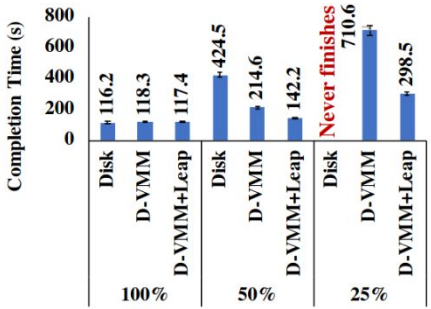
Leap architecture

- Implemented in Linux kernel
  - Applications not modified
- Online prefetcher
  - Identify remote memory accesses patterns
- Local cache
  - Avoid pollution
  - Increase cache hit rate
- https://github.com/SymbioticLab/Leap

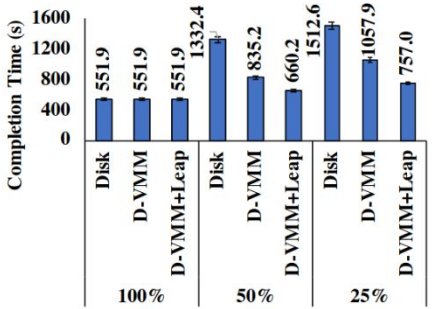


Leap performance evaluation

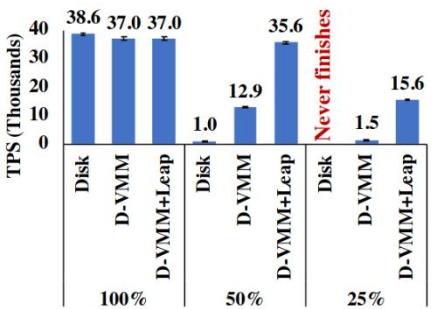
- System setup
  - 56 Gbps Infiniband cluster
  - Each machine: 64 GB RAM, 2x Intel Xeon E5-2650 v2 (16 cores)
- Real-world benchmarks
  - PowerGraph, NumPy, VoltDB, Memcached
- Performance gain: up to 10x compared to Infiniswap, a state-of-the-art solution



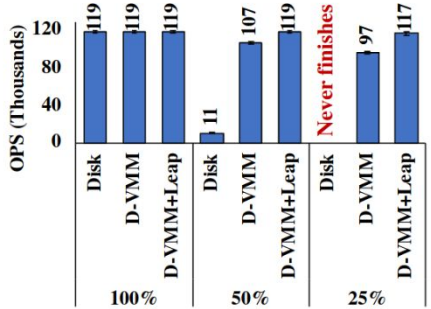
(a) PowerGraph Completion Time



(b) NumPy Completion Time

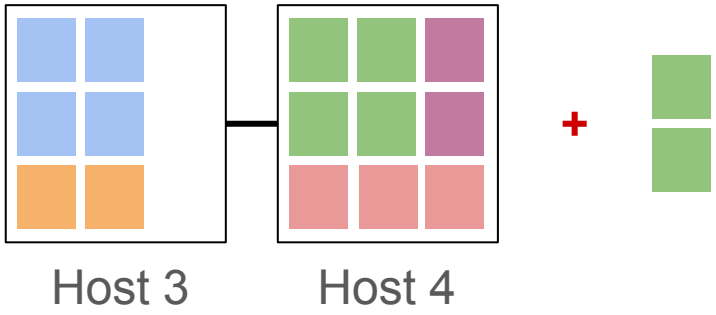


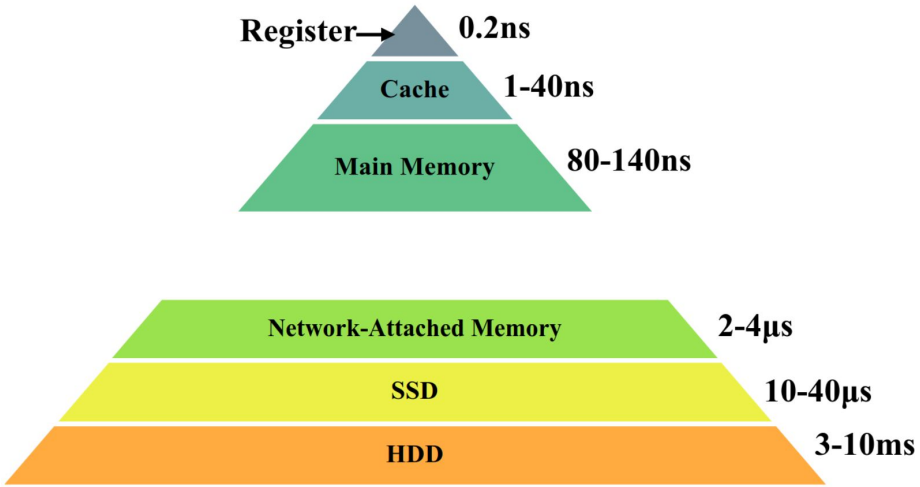
(c) VoltDB Throughput



(d) Memcached Throughput

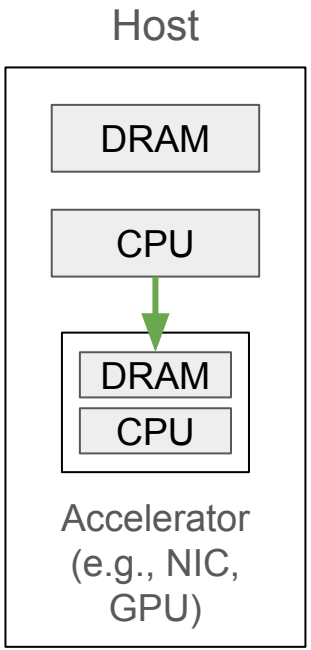
Where to store the reclaimed memory? (Revisited)

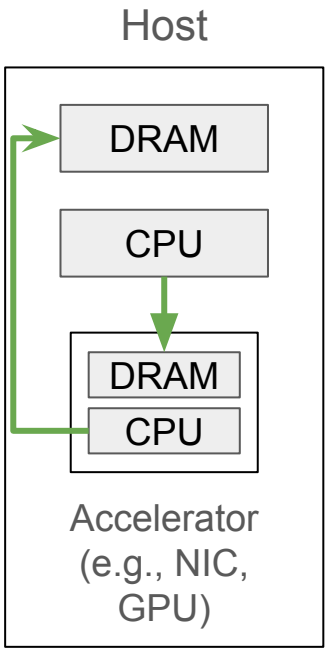


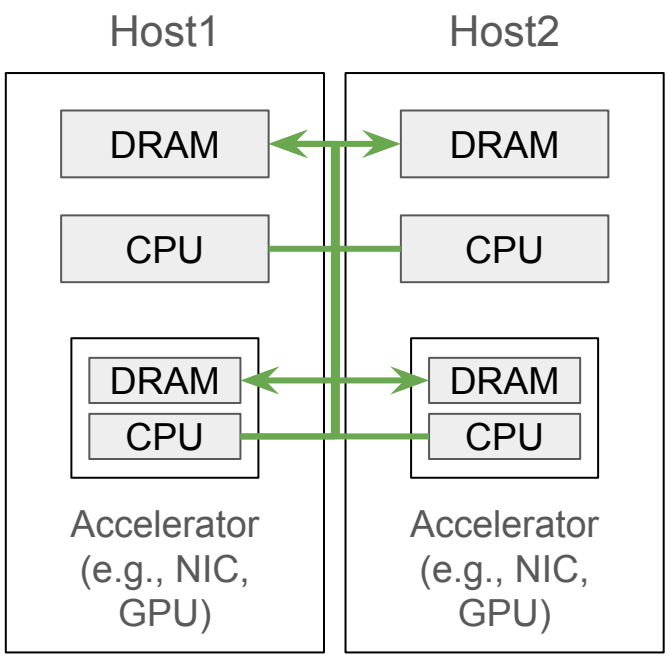


- Compute-eXpress Link
  - Industry-backed open standard
  - https://www.computeexpresslink.org/
- High-speed **cache-coherent** interconnect
  - Built on top of PCIe
- Protocols
  - CXL.io: provides configuration, discovery, etc.
  - CXL.cache: allow devices to coherently access and cache host CPU memory
  - CXL.mem: allow host CPU to coherently access cached device memory
- Device types
  - Type 1: CXL.io and CXL.cache
    - e.g., smartNIC
  - Type 2: CXL.io, CXL.cache and CXL.mem
    - e.g., GPU or FPGA
  - Type 3: CXL.io, and CXL.mem
    - e.g., memory expansion board

Three CXL Specifications







1.1: CPU -> accelerator
access cache-coherent

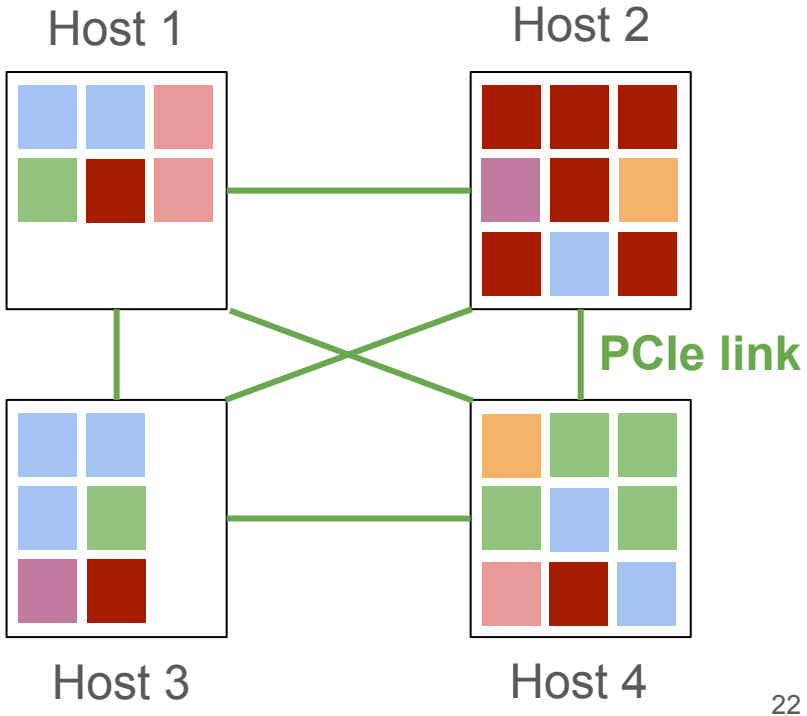
2.0: CPU <-> accelerator
access cache-coherent

3.0: P2P access
cache-coherent

What it means for Tanaka san's company

SUPER INFINITY CLOUD PROVIDER

- VM resources spread across entire rack/cluster
  - Remote memory access with very low performance penalty compared to local DRAM
- Decoupling between compute nodes (CPUs) and resource nodes (memory)
  - Memory disaggregation
- Better scalability and resource utilization



Pond: a CXL-Based Memory Pooling System

- Pond: CXL-Based Memory Pooling Systems for Cloud Platforms
  - Li et al., Virginia tech & Microsoft Azure
  - ASPLOS 2023

- Research question: How can VMs efficiently use DRAM?
  - DRAM is expensive (50% of hardware cost at Azure)
  - Local memory node accesses faster than remote accesses
  - Cloud provider should not inspect what is running inside VMs

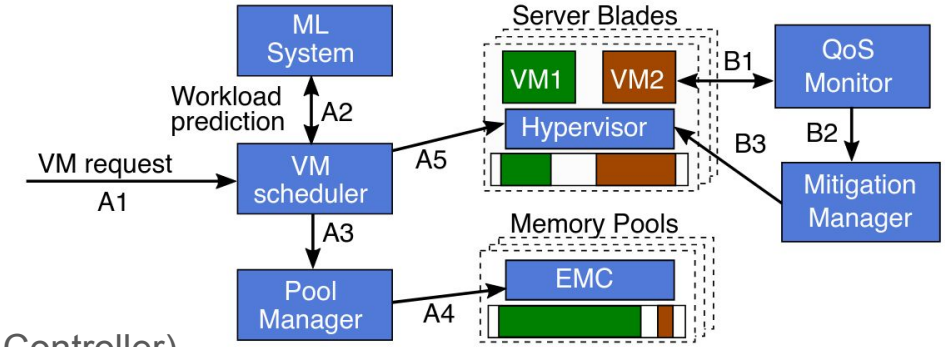
- Existing solutions
  - Add substantial latency (~µs)
  - Require changes to the VM

Memory usage at Azure

- Analysis of traces from 100 production clusters and 158 workloads
- Consider CXL access to be similar to remote NUMA node access
- Grouping memory of 16 CPUs together in a single pool achieves "sufficient" DRAM saving while adding <100ns latency
  - 7% DRAM saving, which corresponds to ~100M$ of savings
- Overhead of pooling compared to same-NUMA node memory access
  - Within 5% for 40% of the workloads
  - >25% for 21% of the workloads
- ~50% of all VMs touch less than 50% of their rented memory
  - We can allocate the remaining rented memory on a remote node ("zNUMA") with no performance penalty

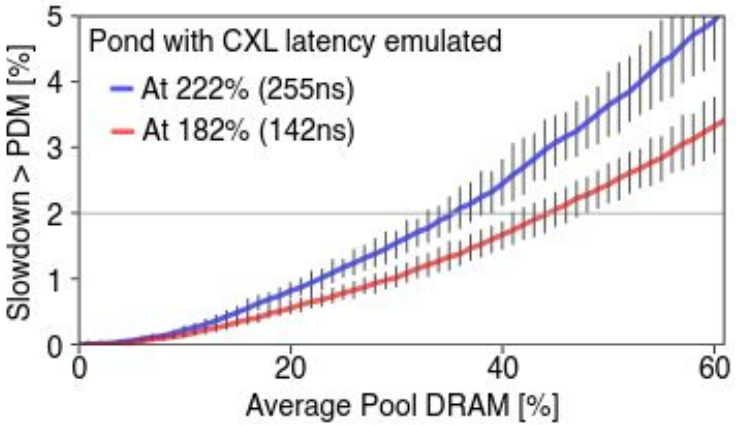
The Pond System

- Rely on CXL
  - Pooled (remote) memory access with ns latency
- Predict VM memory allocation behaviour
  - Is it ok to allocate memory on a remote CXL node?
  - ML model
- Monitor memory access
  - To fix wrong predictions
  - Based on hardware counters
- Hardware changes
  - Implement a new EMC (External Memory Controller)
- Open-source: https://github.com/vtess/Pond



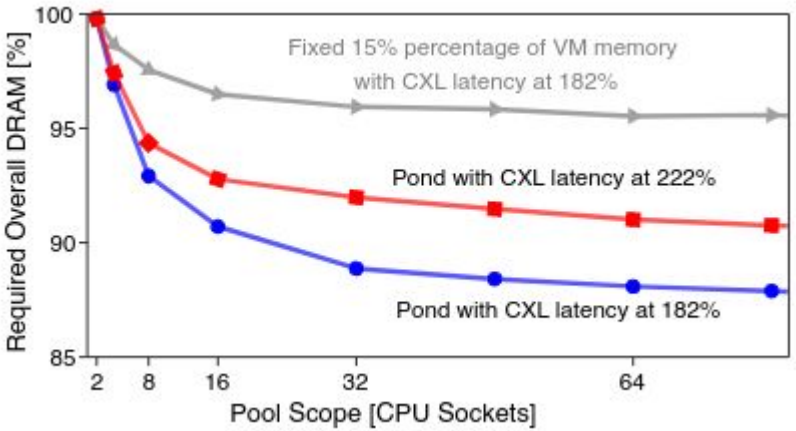
Performance evaluation

- Simulate CXL on a 2 CPUs machine
  - One CPU acts as a zNUMA node: cores offline, only memory is used



Model false positive rate

Performance Degradation Margin (PDM): max acceptable slowdown



Memory savings

CXL hardware

- CPUs
  - Intel Sapphire Rapids
  - AMD Zen 4 Epyc ("Genoa" and "Bergamo")
  - Arm Neoverse V2
  - AmpereOne (https://amperecomputing.com/)

CXL 1.1

- FPGA
  - Intel Agilex
  - Xilinx Adaptive Compute Acceleration Platforms Versal Premium lineup

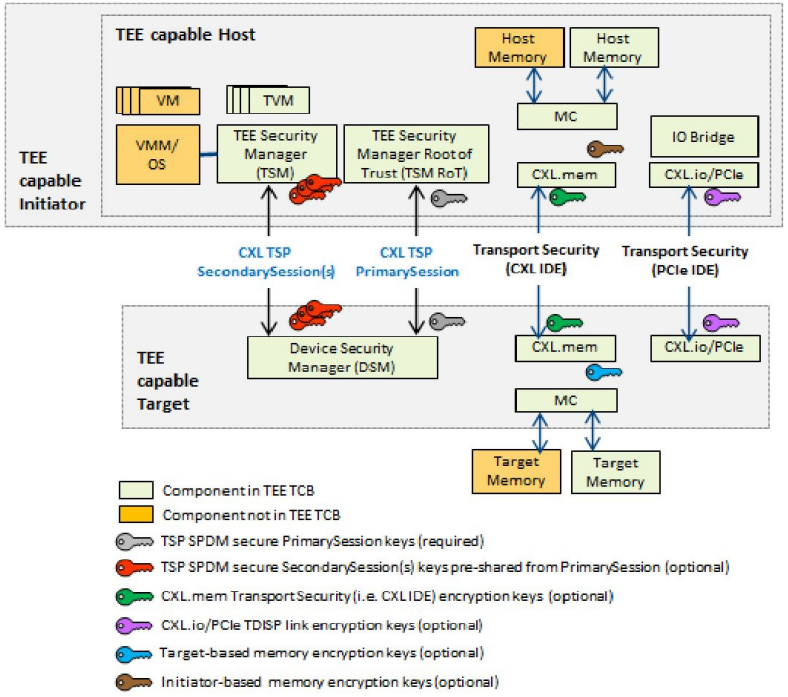
CXL 2.0

- Memory module
  - Samsung Memory-Semantic SSD
- Complete Memory system
  - Panmnesia (https://panmnesia.com/)
  - Unifabrix (https://www.unifabrix.com/)

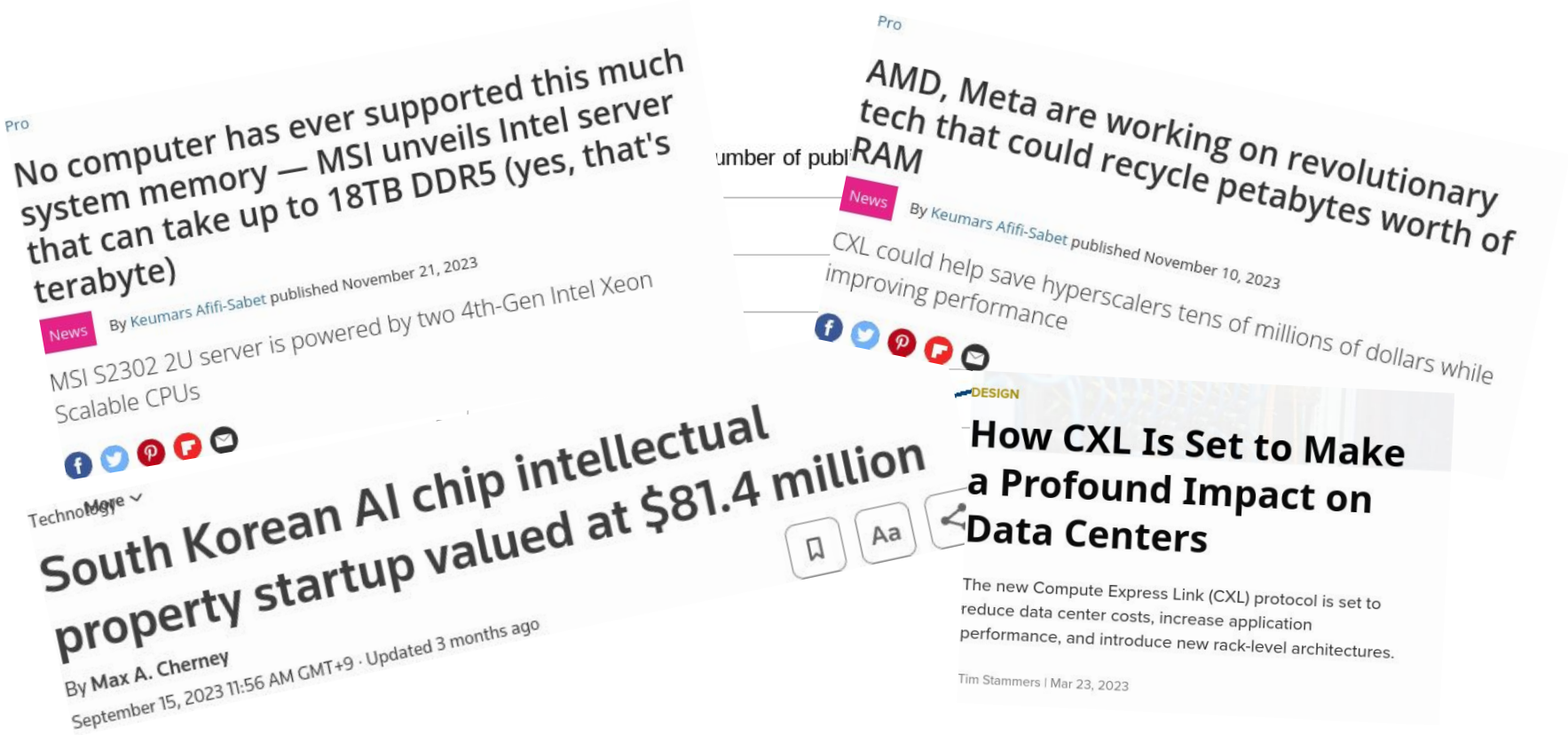
CXL 3.0

Latest update!!! CXL 3.1 brings Trusted Execution support

- Enables Confidential Computing
  - Securely run workloads without exposing data to untrusted party (OS, other VM, etc.)
  - Targets Trusted VMs (TVMs)
- TSP protocol
  - Extends CXL specification to include CXL devices into the TVM trust boundary



CXL and disaggregated memory are hot topics

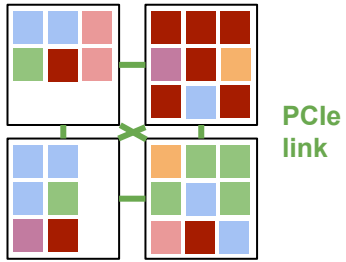


Concluding words

- Cloud providers need to understand their memory usages to offer the best performance to their customers

- Memory disaggregation offers several advantages
  - Better utilization
  - Better scalability
  - Reduces costs

- CXL provides foundation for memory disaggregation at high speed (ns)



pierrelouis@iij.ad.jp