Yizhou Shan

Final Defense

Mar 04, 2022

Committee

Professor Yiying Zhang (Chair)
Professor George C. Papen
Professor Alex C. Snoeren
Professor Stefan Savage
Professor Geoffrey M. Voelker

Distributing and Disaggregating Hardware Resources in Data Centers

Yizhou Shan

Mar 2022

Committee

Professor Yiying Zhang (Chair)
Professor George C. Papen
Professor Alex C. Snoeren
Professor Stefan Savage
Professor Geoffrey M. Voelker

Modern Data Centers

- **Data centers are large, complex, consolidated facilities**
  - They host workloads from various industries
  - They run applications affecting billion people's daily life
- **Cloud vendors transform them into a "public computing utility"**

Google Data Center

Top Cloud Vendors

Exciting and Challenging Time to be a Data Center Architect

Applications

Fast-changing, high-demand,
heterogeneous,
emerging industries

hadoop

Flink

APACHE Spark

PyTorch

RAY

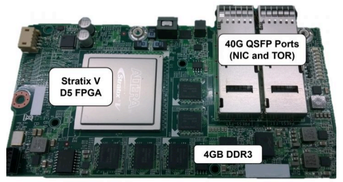
ceph

IoT Applications

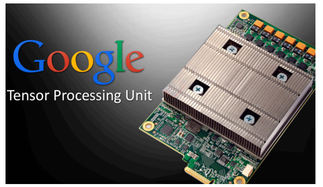
robinhood

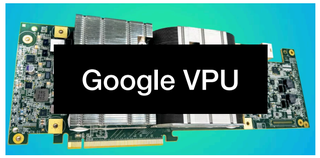
Hardware

Specialized,
faster,
domain-specific



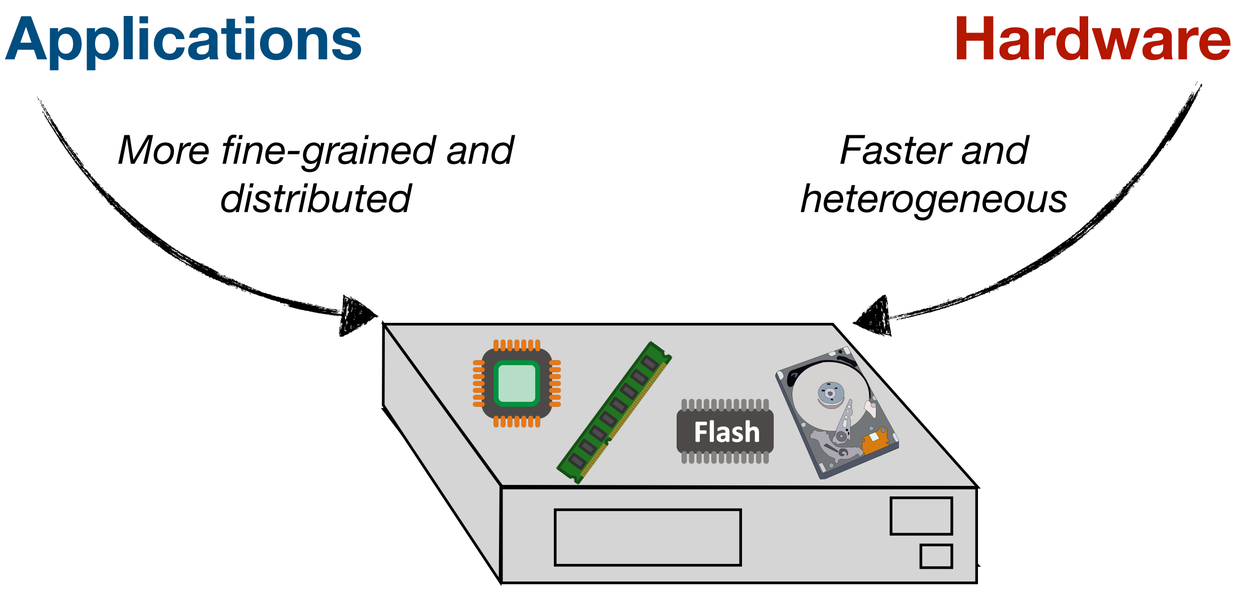
Microsoft FPGA





Nitro Cards

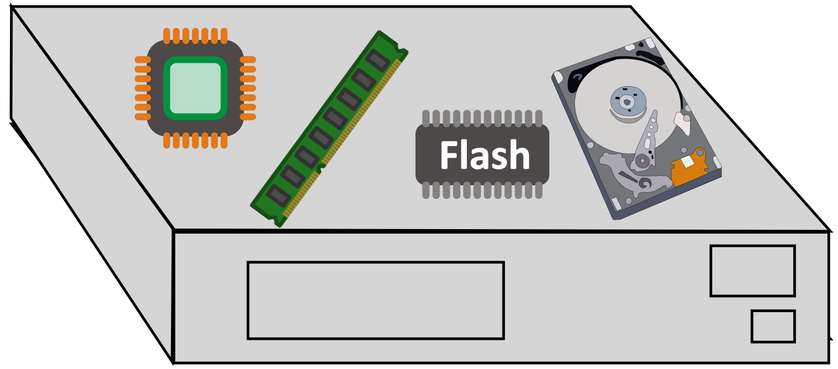
Data Center's Unit of Deployment: Monolithic Server



Unfortunately, it is becoming extremely difficult to fit both onto the monolithic servers!

Root Cause: the Monolithic Server Model

- **The Monolithic Server WALL**
  - Bin-packing issue (***utilization***)
  - Fate-sharing failure domain (***isolation***)
  - No independent resource scaling (***elasticity***)
  - Hard to add extra resources due to *limited slots (**heterogeneity**)*



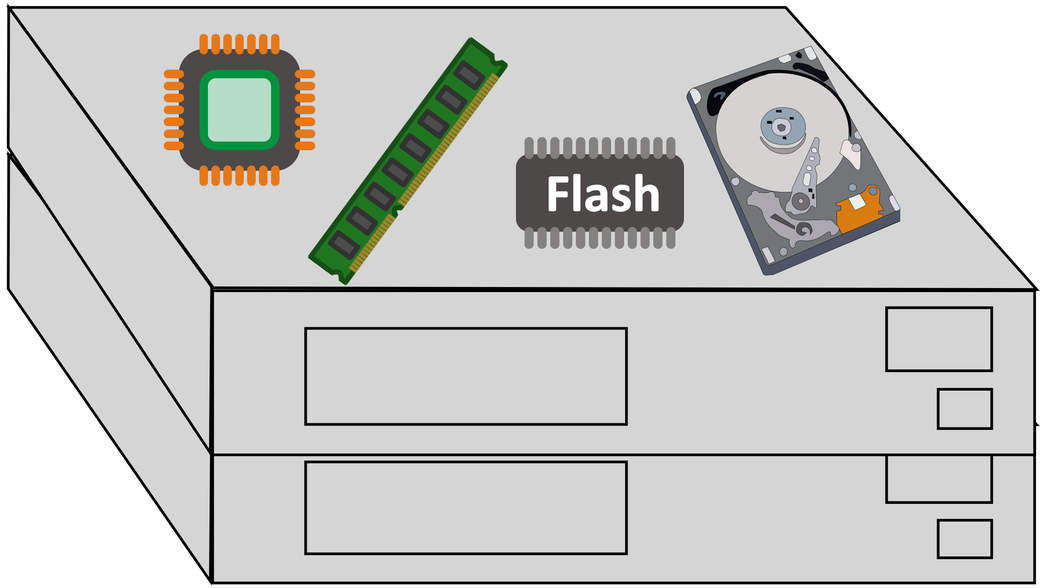
- **It was a blessing for deployment, but hitting limitations now**

How to improve resource utilization, elasticity, heterogeneity, and fault tolerance?

Go beyond physical server boundary!

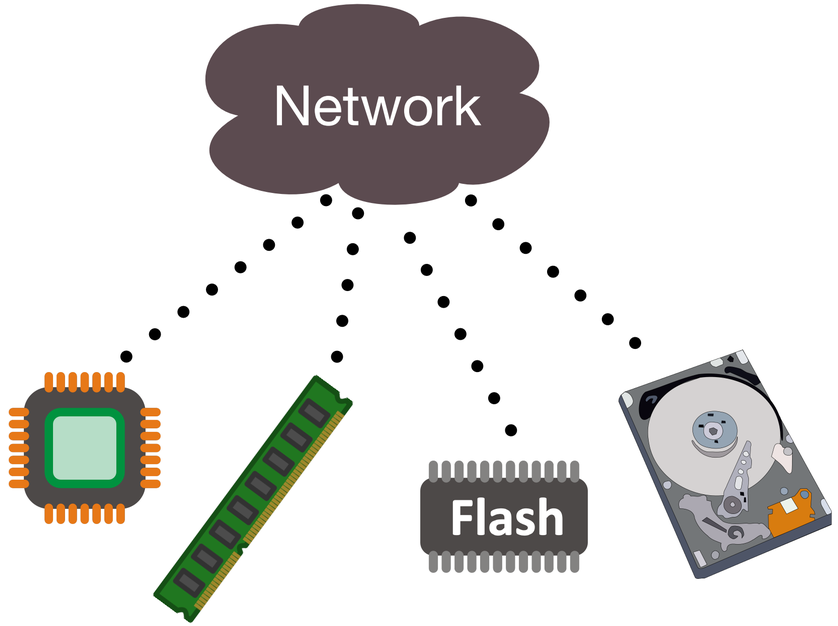
Hardware Resource Disaggregation

Break monolithic servers into ***network-attached*** resource pools



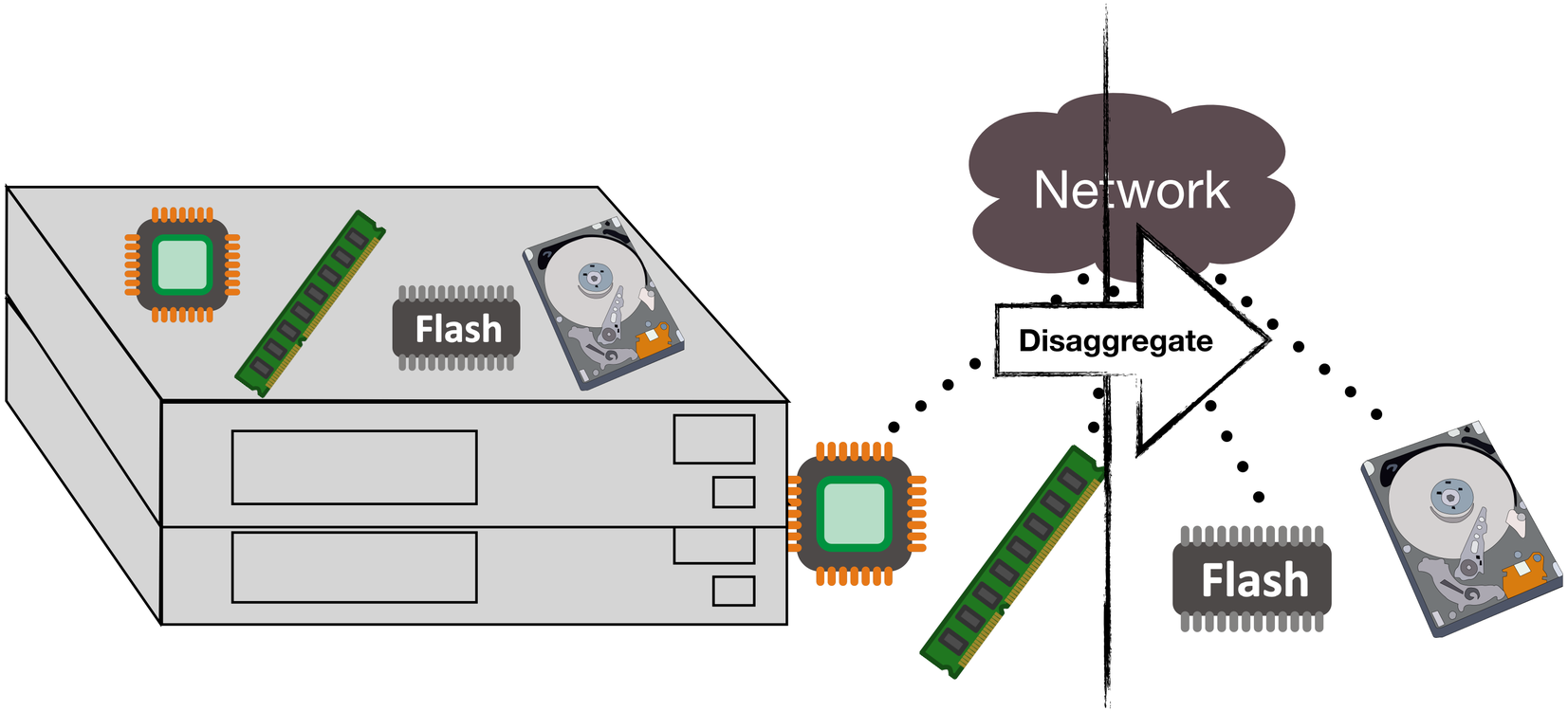
Hardware Resource Disaggregation

Break monolithic servers into ***network-attached*** resource pools



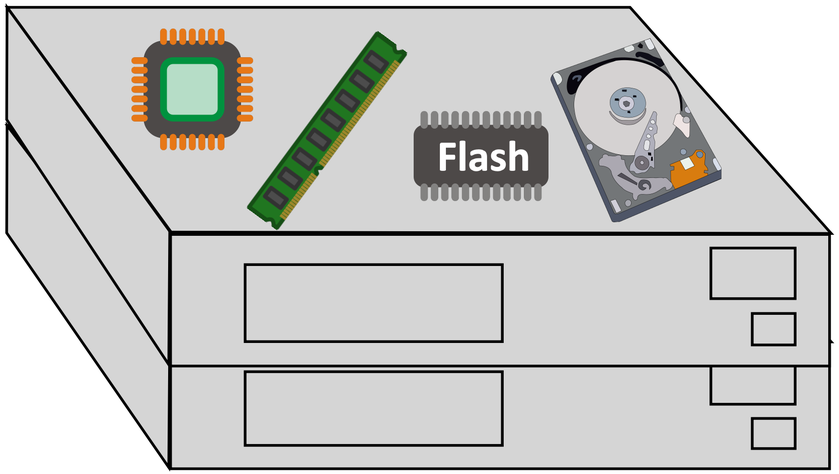
Hardware Resource Disaggregation

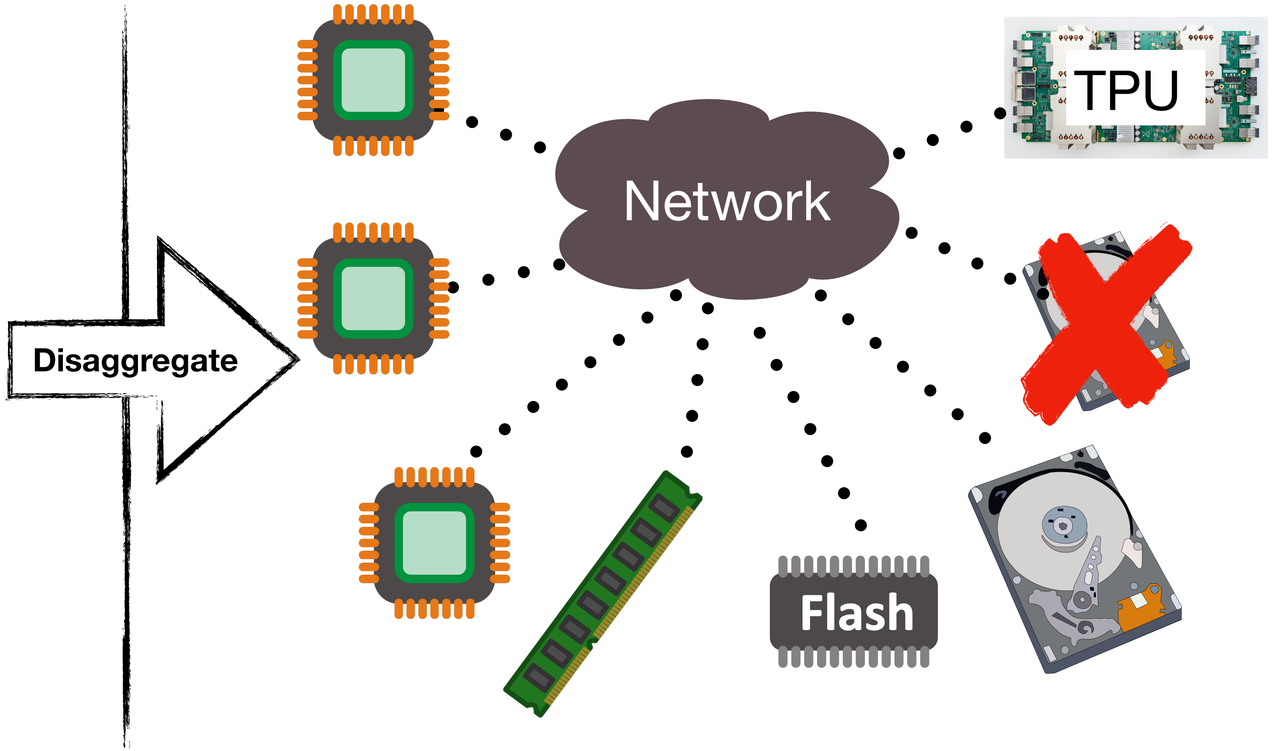
Break monolithic servers into ***network-attached*** resource pools



Hardware Resource Disaggregation

Break monolithic servers into ***network-attached*** resource pools





- **Independent resource scaling**
- **Better support for heterogeneity**
- **Independent failure domain**
- **No bin-packing issue**

Dissertation Statement

Problem

Despite hardware resource disaggregation's great promises,
it is a drastic departure from the traditional computing paradigm.
It was not clear how to deploy it in practical settings.

Statement

This dissertation shows that it is *possible* to overcome the challenge of building and deploying hardware resource disaggregation in real data centers, delivering its promises on better manageability, scalability, and cost.

This dissertation advances the state-of-art of this area,
transforming it from a vague research proposal into one that is tangible,
practical, deployable, and can be approached quantitatively.

Outline

- Intro
- **Background on Resource Disaggregation**
- **Projects Conducted**
  - **Logical Disaggregation** [Hotpot, SoCC'17]
  - **Physical Disaggregation** [LegoOS, OSDI'18]
  - **Hybrid Disaggregation** [Clio, ASPLOS'22]
  - **Network Disaggregation** [SuperNIC, arXiv'21, under submission]
- **Future Work**
- **Conclusion**

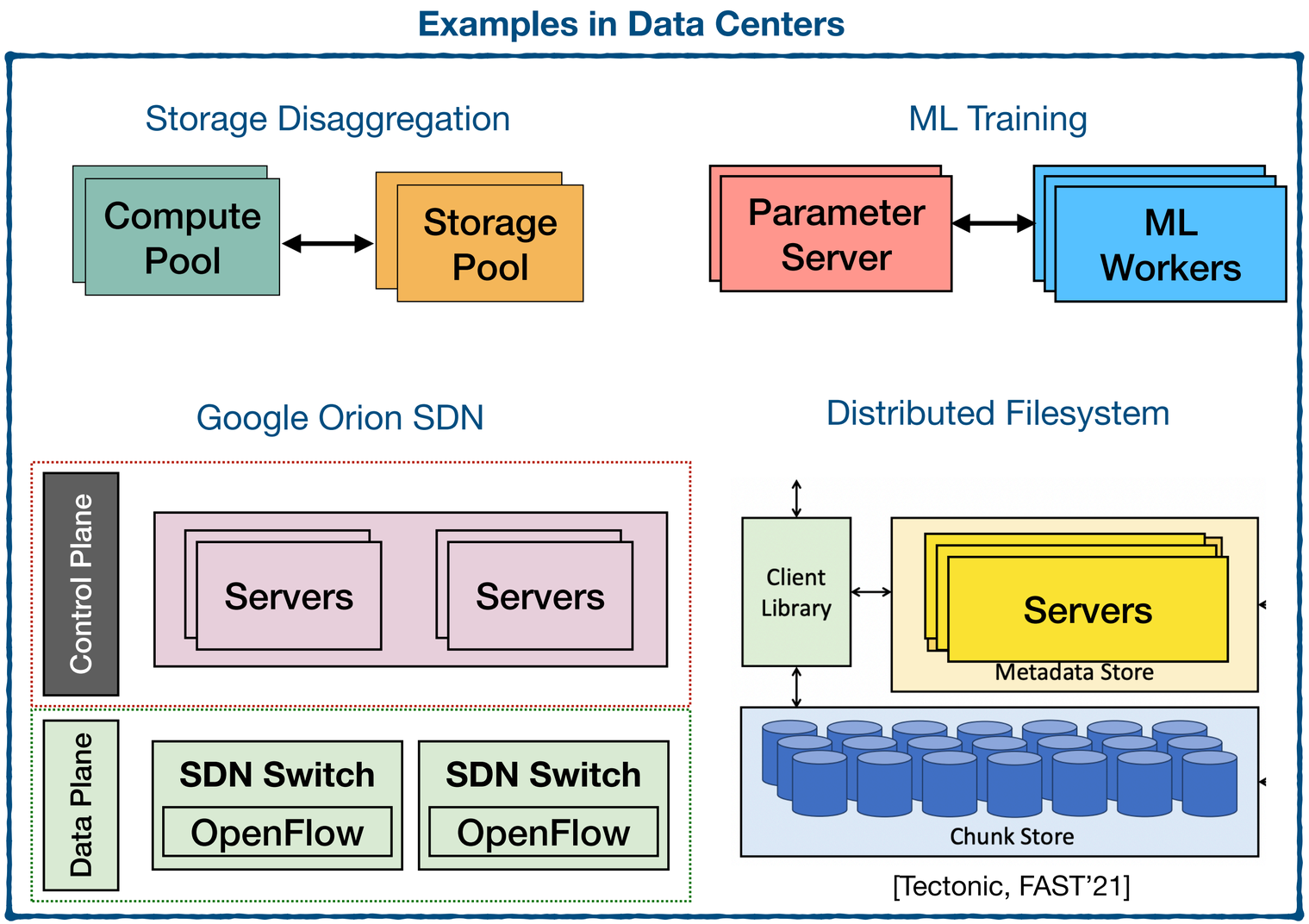
Traditional Resource Disaggregation

Essence

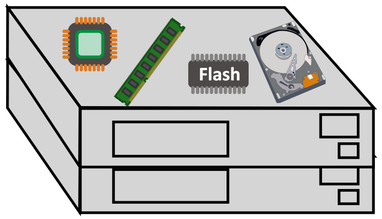
Decoupling
Independent Scaling
Independent Failure

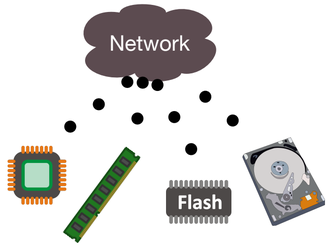
The Resource Disaggregation idea is basically everywhere in data centers!

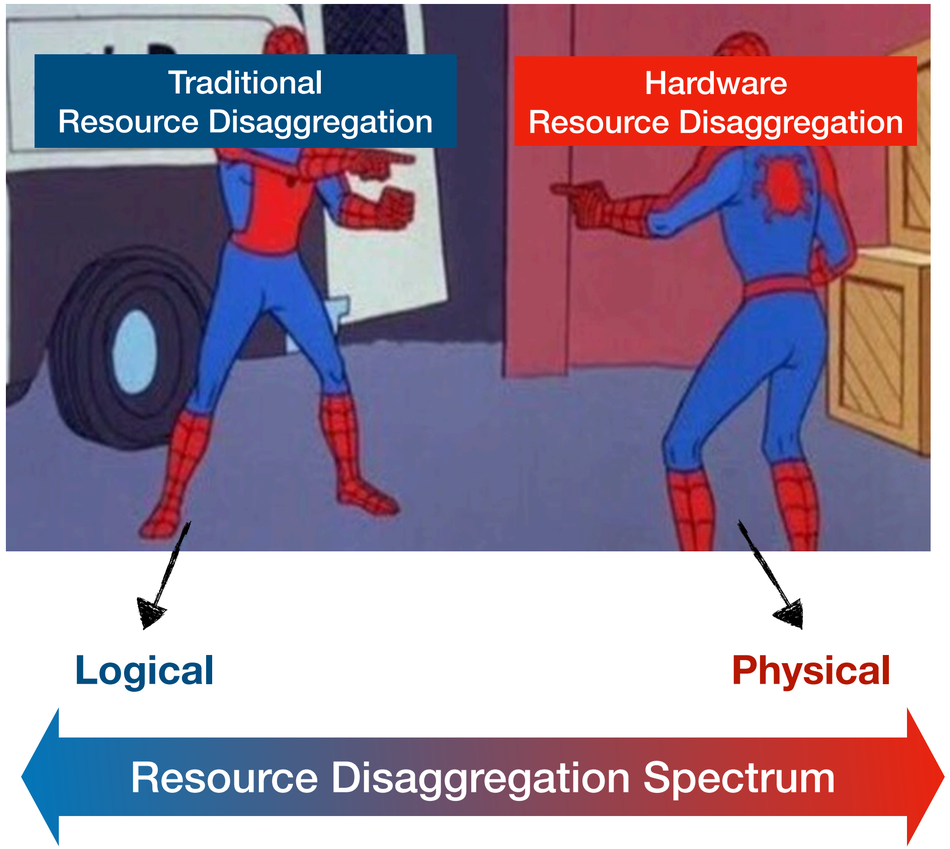
just in different granularities



Hardware resource disaggregation, is it just another buzzword? Another “old wine in a new bottle”?





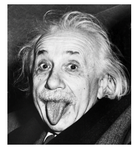


Hold on..
They are actually quite different!
Have you asked yourself these questions?

1. Ok, but how could CPU work w/o memory?
2. Network is slower than the memory bus, the perf must be horrible?
3. Wait, are you telling me Linux no longer works??
4. What about the network? How could it support all these devices?
5. How can you even deploy this thing? Chicken-egg problem, no?

Our Observation

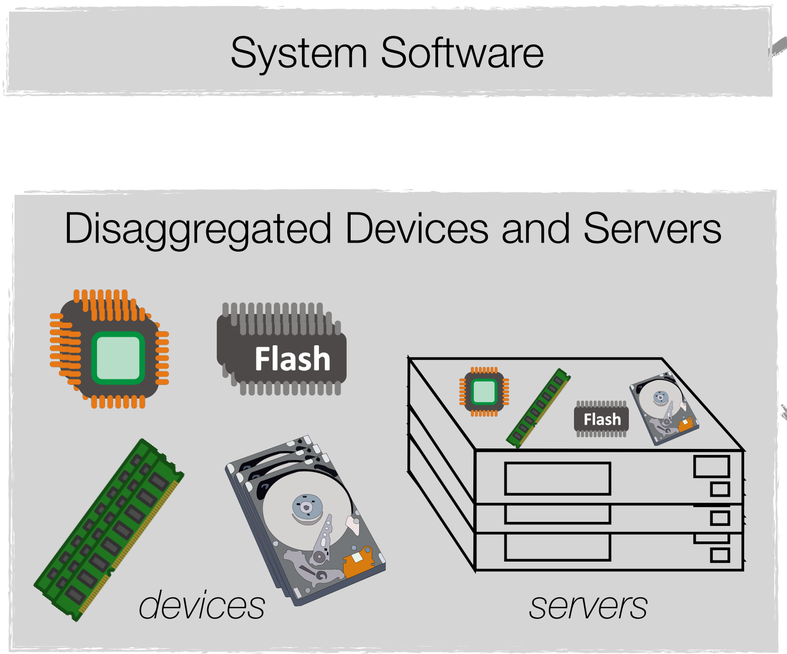
Resource Disaggregation is a general idea with a wide design spectrum that *unifies* everything



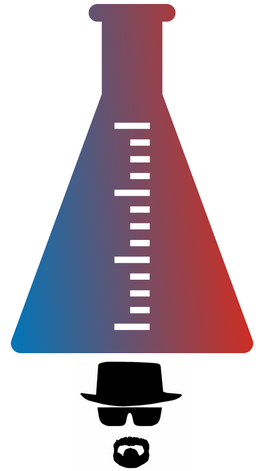
Unified?
Love it!!

Resource Disaggregation's Cooking Formula

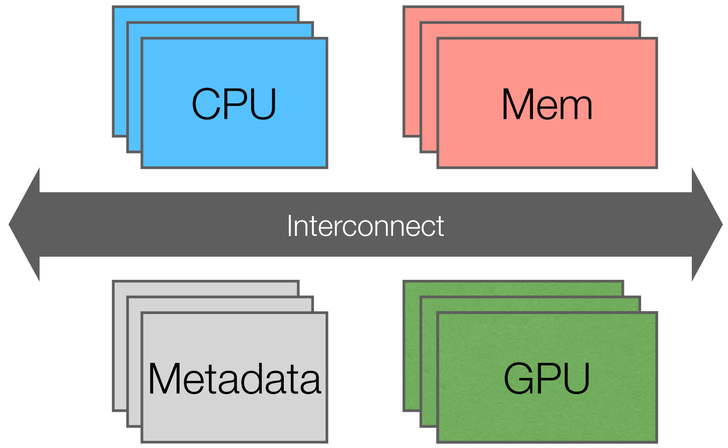
The Ingredients



Resource Disaggregation Cooking Recipes



The Ultimate Conceptual View

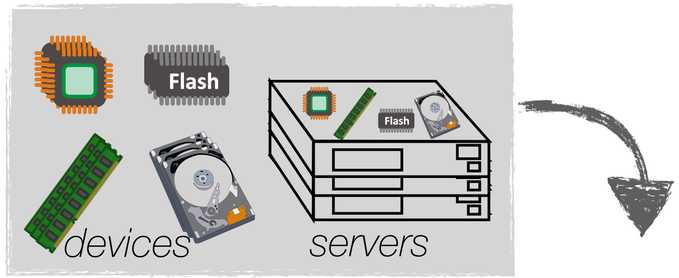


Resource Pools

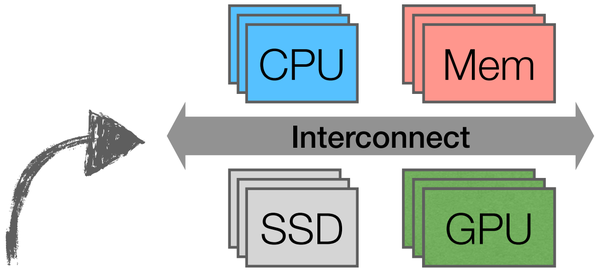
Using various ingredients and recipes, I show that it is possible to achieve disaggregation's great promises.

End of the day, the so-called researchers are just chefs trying to find a right recipe.
- Heisenberg

The Physical Set of Devices and Servers

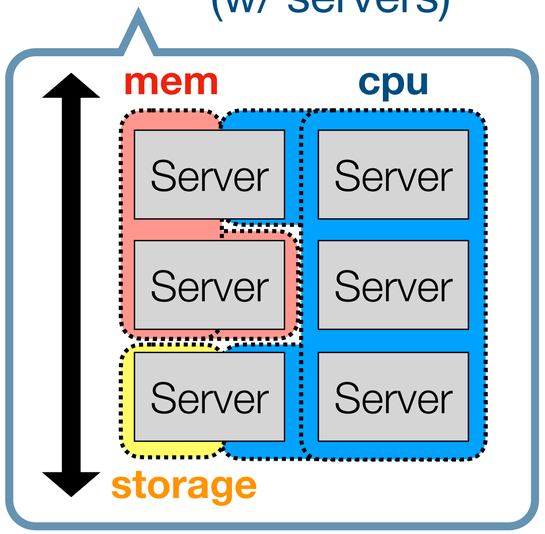


The Resource Conceptual View



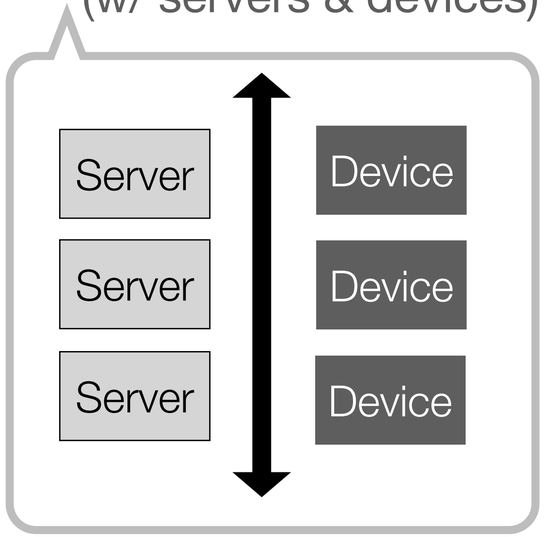
Resource Disaggregation Design (Cooking) Spectrum (Recipes)

Logical (w/ servers)



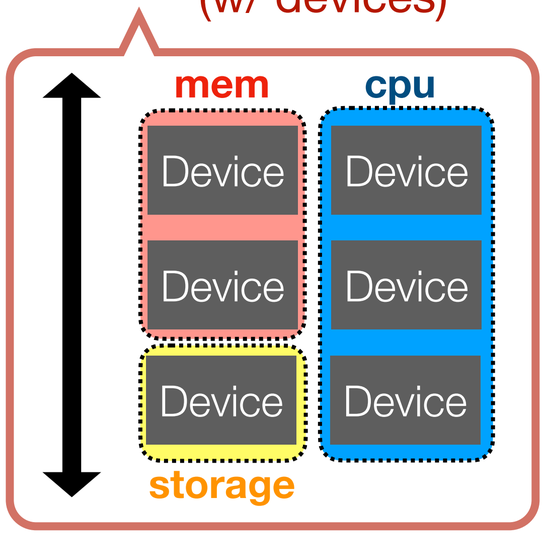
The conceptual view
logically
maps to the servers
(has indirection layer)

Hybrid (w/ servers & devices)



The conceptual view
is a hybrid
of servers and devices

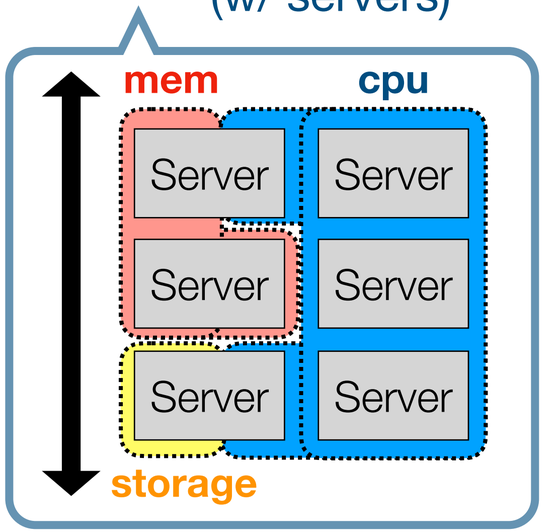
Physical (w/ devices)



The conceptual view
physically
maps to the devices
(no indirection layer)

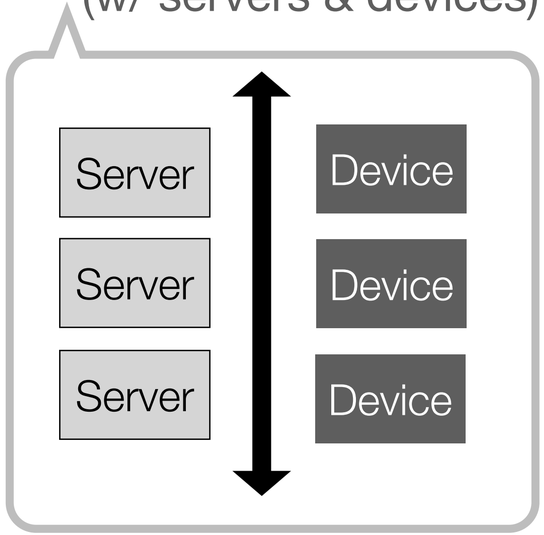
Resource Disaggregation Design (Cooking) Spectrum (Recipes)

Logical
(w/ servers)



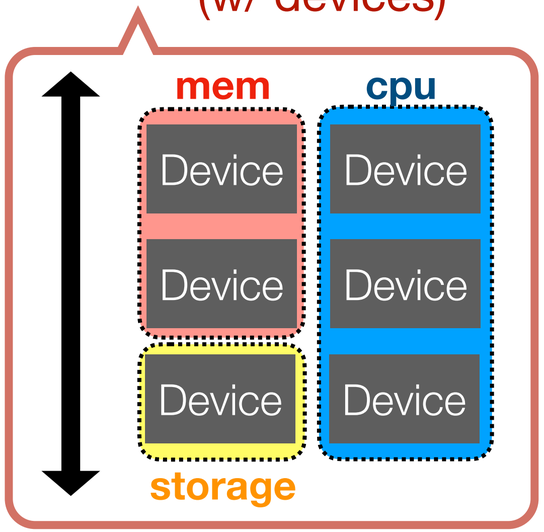
Part 1
Distributed Shared Persistent Memory
[Hotpot, SoCC'17]

Hybrid
(w/ servers & devices)



Part 3
Hardware-based Disaggregated Memory
[Clio, ASPLOS'22]

Physical
(w/ devices)



Part 2
Disaggregated Operating System
[LegoOS, OSDI'18]

Part 4
Disaggregated Networking For the Masses
[SuperNIC, arXiv'21]

Outline

- Background on Resource Disaggregation
- **Projects Conducted**
  - **Logical Disaggregation** [Hotpot, SoCC'17]
  - **Physical Disaggregation** [LegoOS, OSDI'18]
  - **Hybrid Disaggregation** [Clio, ASPLOS'22]
  - **Network Disaggregation** [SuperNIC, arXiv'21, under submission]
- **Future Work**
- **Conclusion**

Hotpot
Distributed Shared Persistent Memory

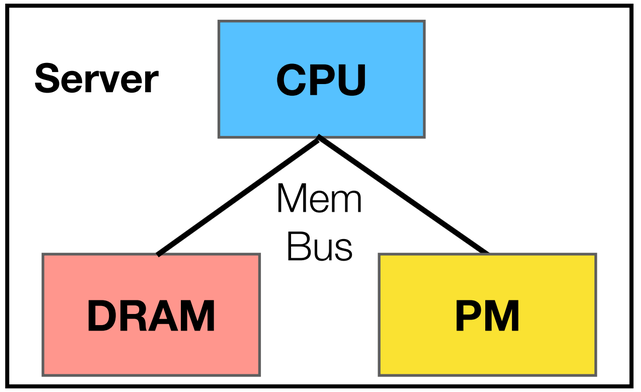
Yizhou Shan, Shin-Yeh Tsai, and Yiying Zhang

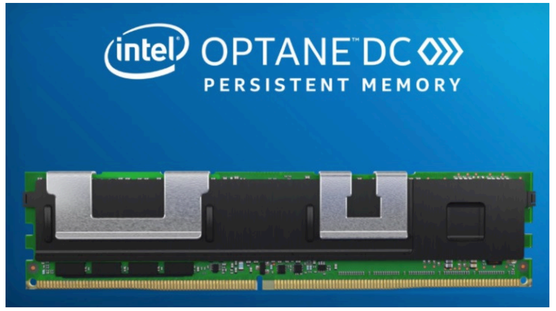
WukLab

*[1] **Yizhou Shan**, Shin-Yeh Tsai, Yiying Zhang. Distributed Shared Persistent Memory, SoCC'17.*
[Among the first to propose distributed PM + RDMA solutions]

Deploy PM in Data Centers

- **Persistent Memory (PM) was an emerging medium**
  - Byte-addressable, DRAM-alike performance
  - Persistent with large capacity
- **Very limited research on distributed PM** (circa 2017)
  - [*Mojim, ASPLOS'15]* distributed replicated PM
  - [*Octopus, ATC'17]* distributed filesystem on PM
- **It was not clear how to best utilize PM in data centers**
  - What's the right abstraction?
  - How to handle failures?
  - How to ensure good performance?

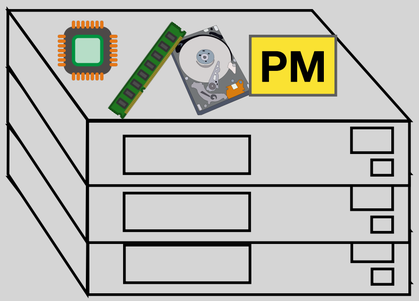




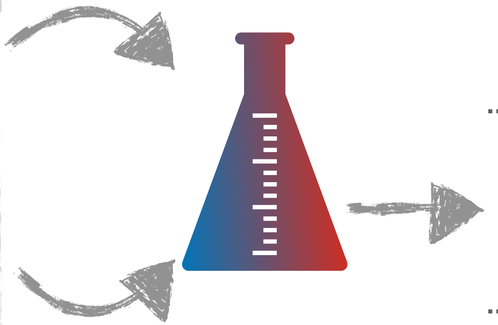
Our Objective: Deploy PM in Data Centers *Efficiently* and *Practically*

System Software:
***Hotpot*, an in-kernel distributed system managing distributed PM**

Monolithic Servers



Logical Disaggregation



Graph | **KVS** | **Filesystem**

Fault-tolerant and efficient (transactional) APIs

Distributed Shared PM (DSPM)

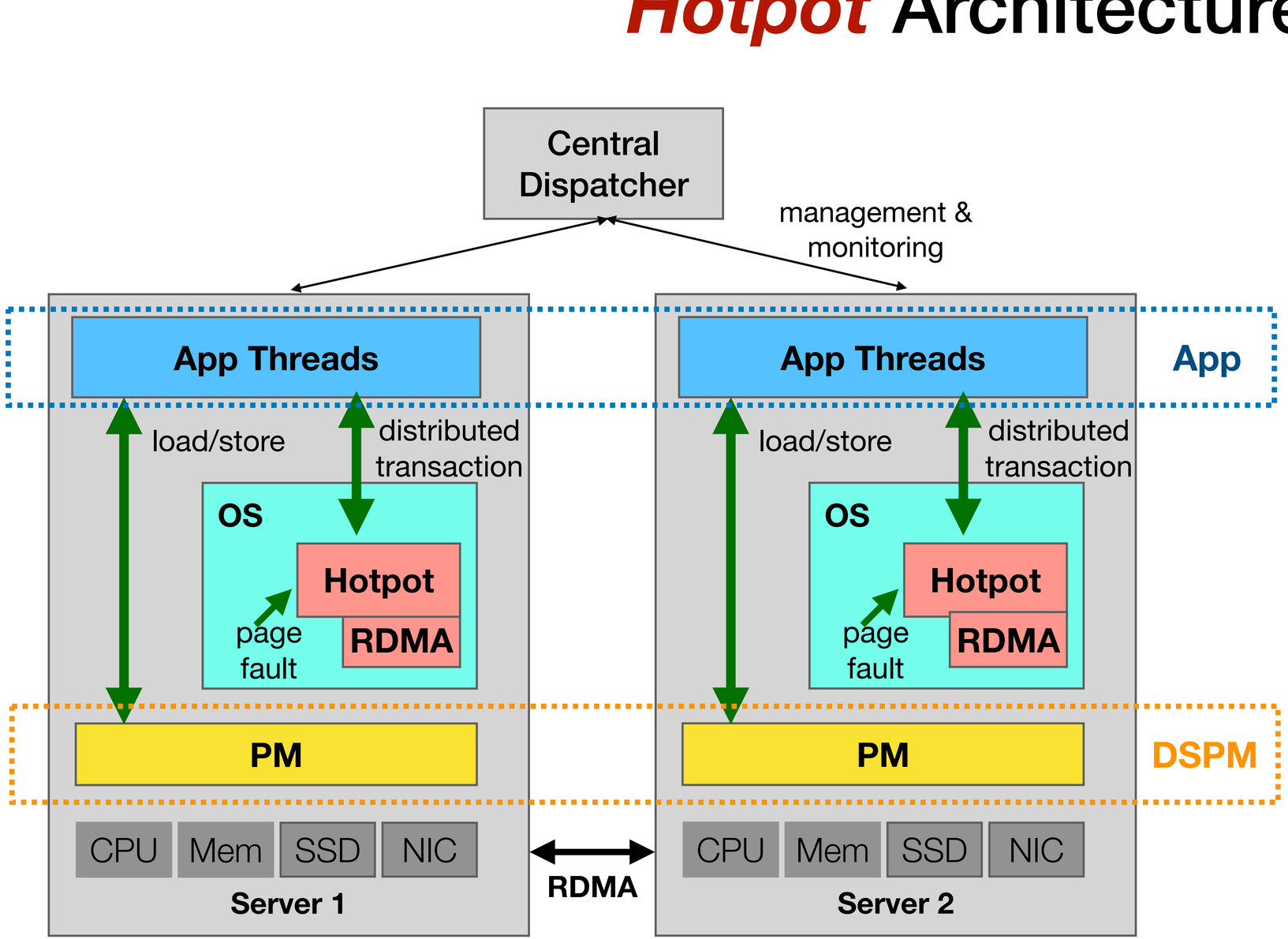
CPU | Mem | SSD

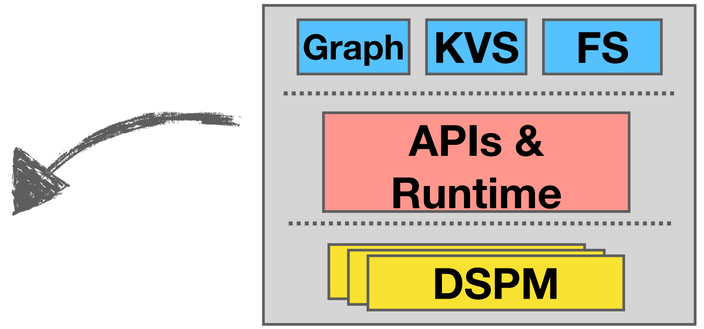
Contribution 2
Runtime & APIs

Contribution 1
A global PM Pool Abstraction

Distributed Shared Persistent Memory (DSPM)

Hotpot Architecture





Central Dispatcher for global resource mgmt & monitoring

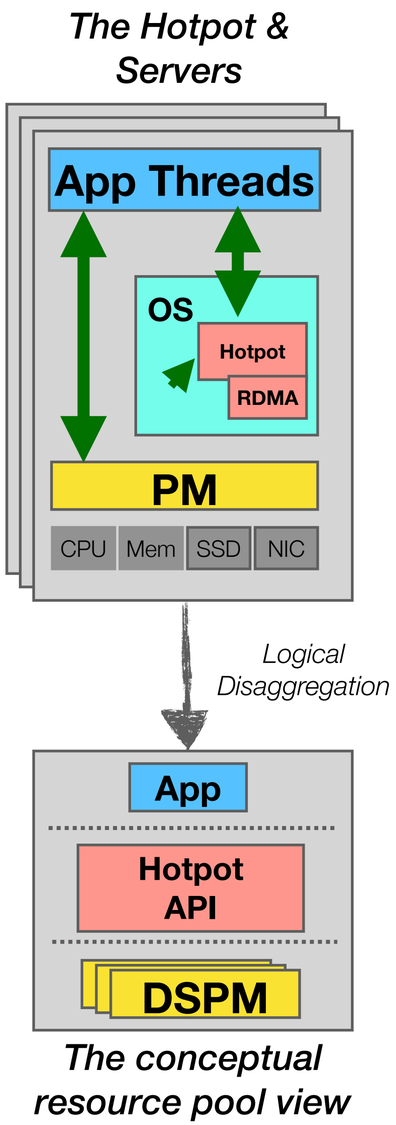
- Distributed Apps
- Hotpot sits in kernel
  - Manages local PM
  - Exposes a global virtual space
  - Unifies memory and storage
- Direct load/store with pgfault
- Distributed transaction APIs
  - MRSW: 2PL+2PC
  - MRMW: OCC+3PC

Hotpot Summary

- **Hotpot is among the first to enable distributed PM in data centers**
  - One layer unifies Distributed Share Memory and Distributed Storage
  - A kernel-level system with ACID distributed transactions
- **Logical Disaggregation inherent server limitations**
  - No independent resource scaling
  - Large fate-sharing failure domain
  - Management complexity & bin-packing

⇒ To avoid those limitations all together,

we took a radical approach: Physical Disaggregation

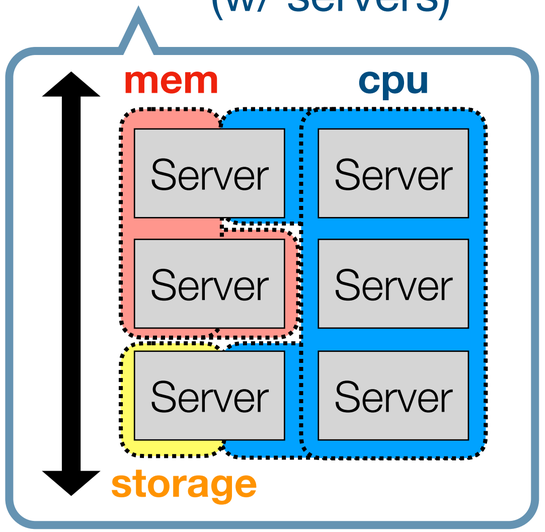


Outline

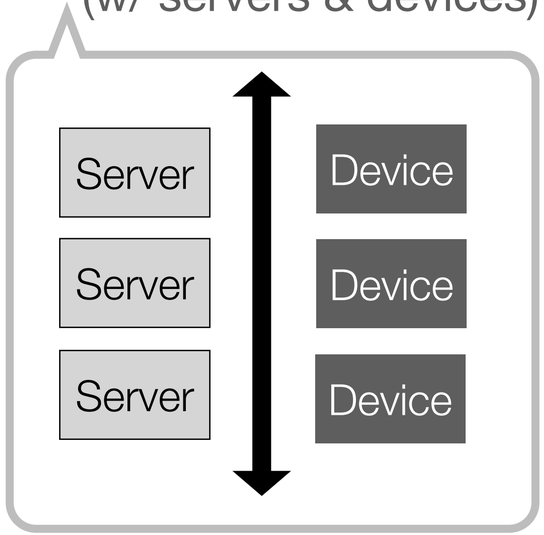
- Background on Resource Disaggregation
- Projects Conducted
  - Logical Disaggregation [Hotpot, SoCC'17]
  - **Physical Disaggregation** [LegoOS, OSDI'18]
  - **Hybrid Disaggregation** [Clio, ASPLOS'22]
  - **Network Disaggregation** [SuperNIC, arXiv'21, under submission]
- **Future Work**
- **Conclusion**

Resource Disaggregation Design (Cooking) Spectrum (Recipes)

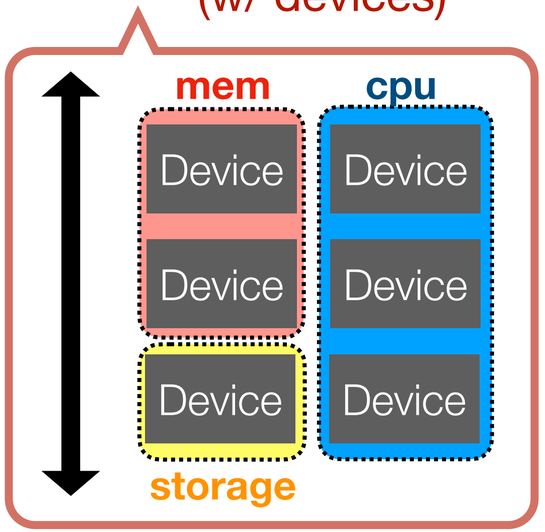
Logical
(w/ servers)



Hybrid
(w/ servers & devices)



Physical
(w/ devices)



Part 1
Distributed Shared Persistent Memory
[Hotpot, SoCC'17]

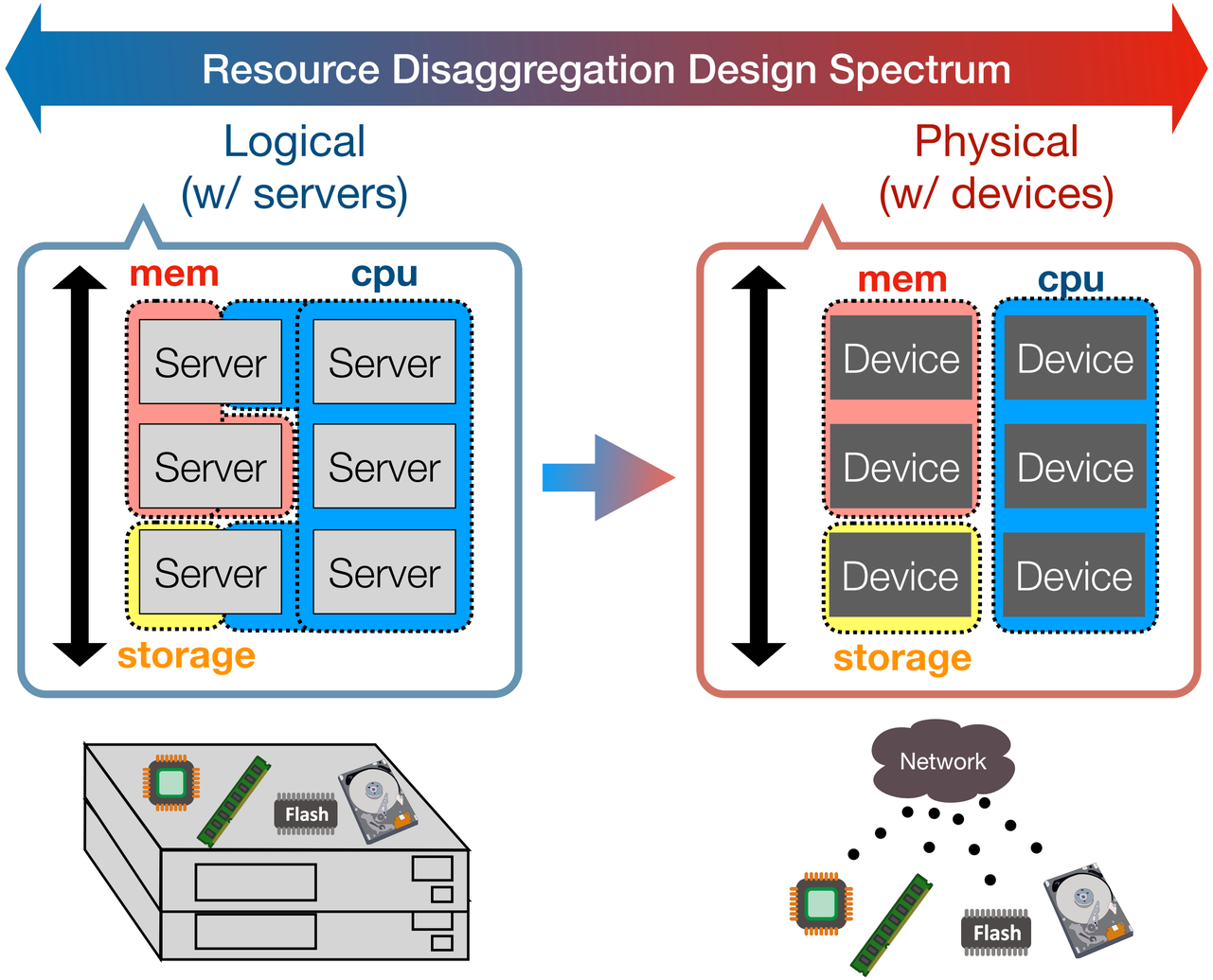
Part 3
Hardware-based Disaggregated Memory
[Clio, ASPLOS'22; Clover, ATC'20]

Part 2
Disaggregated Operating System
[LegoOS, OSDI'18]

Part 4
Disaggregated Networking For the Masses
[SuperNIC, arXiv'21]

Transition from Logical to Physical Disaggregation

(a drastic departure from the traditional computing paradigm)



Challenges

1. How could CPU work w/o memory?
2. Network is slower than memory, what about perf?
3. **How to even run the OS or apps?**

We built a new distributed OS to solve all problems at once!

LegoOS

A Disseminated Distributed OS for Hardware Resource Disaggregation

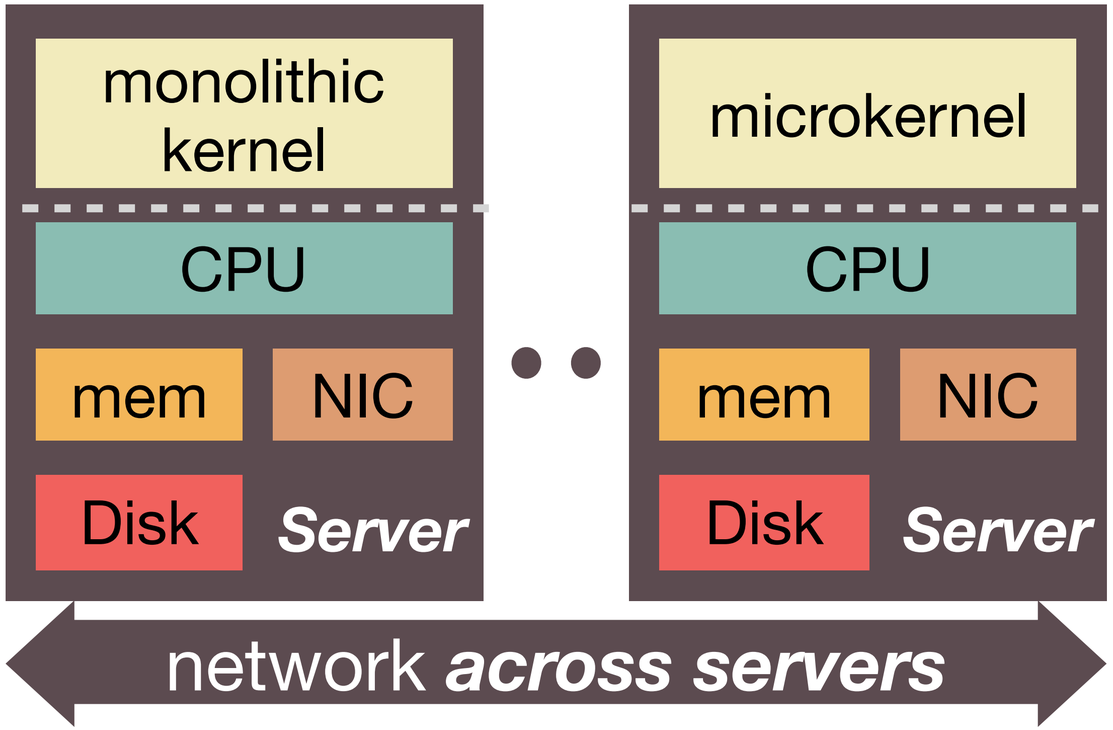
Yizhou Shan, Yutong Huang, Yilun Chen, and Yiying Zhang

Y4

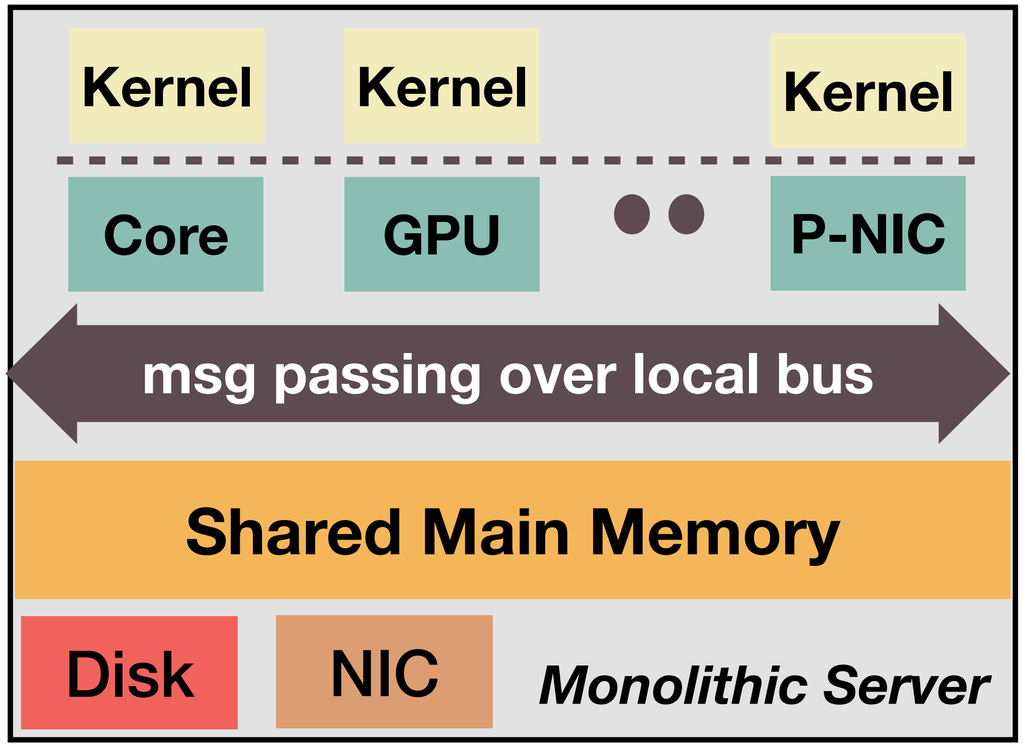
WukLab

*[2] **Yizhou Shan**, Yutong Huang, Yilun Chen, and Yiying Zhang.*
LegoOS: A Disseminated Distributed OS for Hardware Resource Disaggregation, OSDI'18. Best Paper Award.

Can Existing OSs/Kernels Fit?

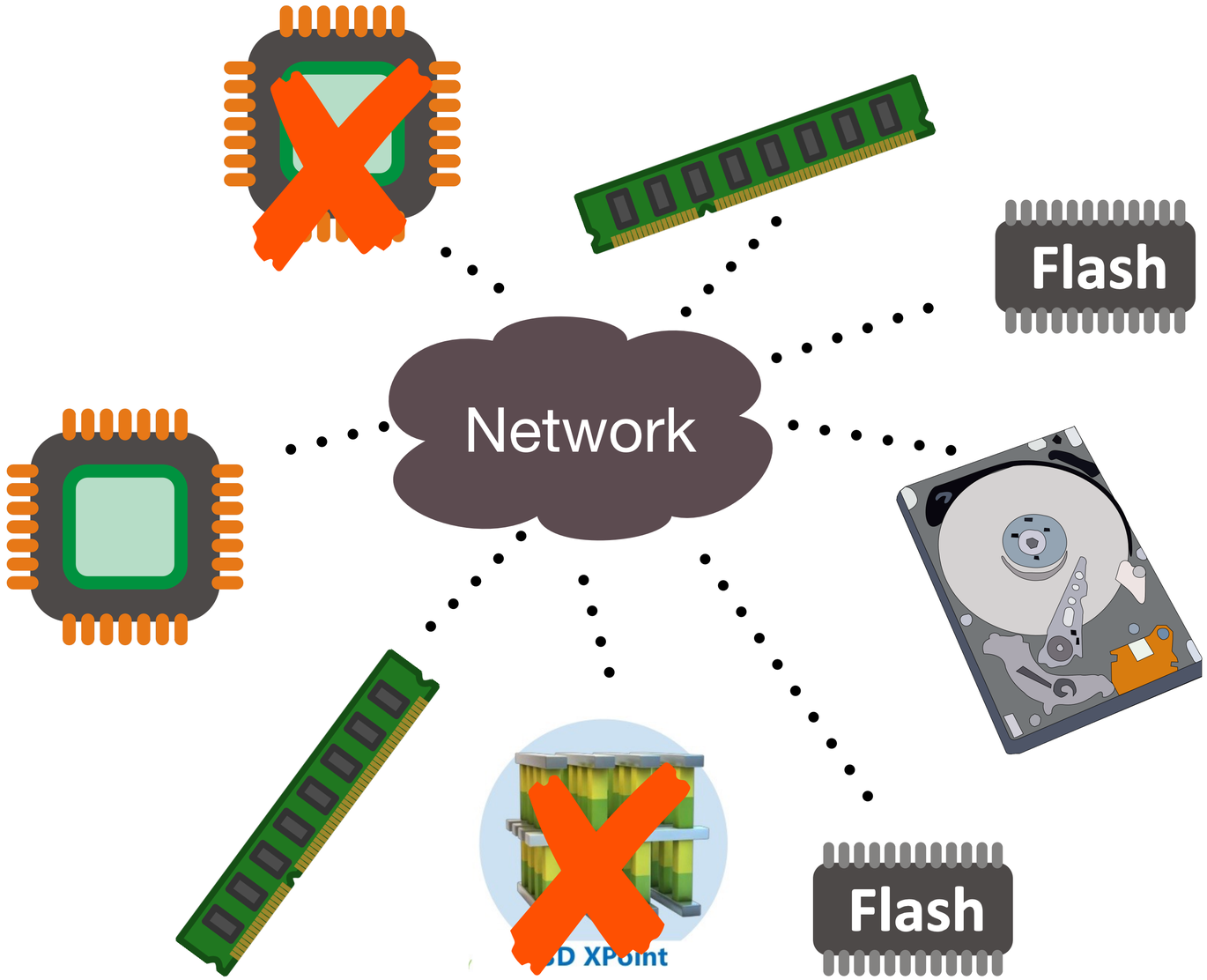


Monolithic/Micro-kernel
(e.g., Linux, L4)



Multi-kernel
(e.g., Barrelfish, Helios, fos)

Existing Kernels Don’t Fit



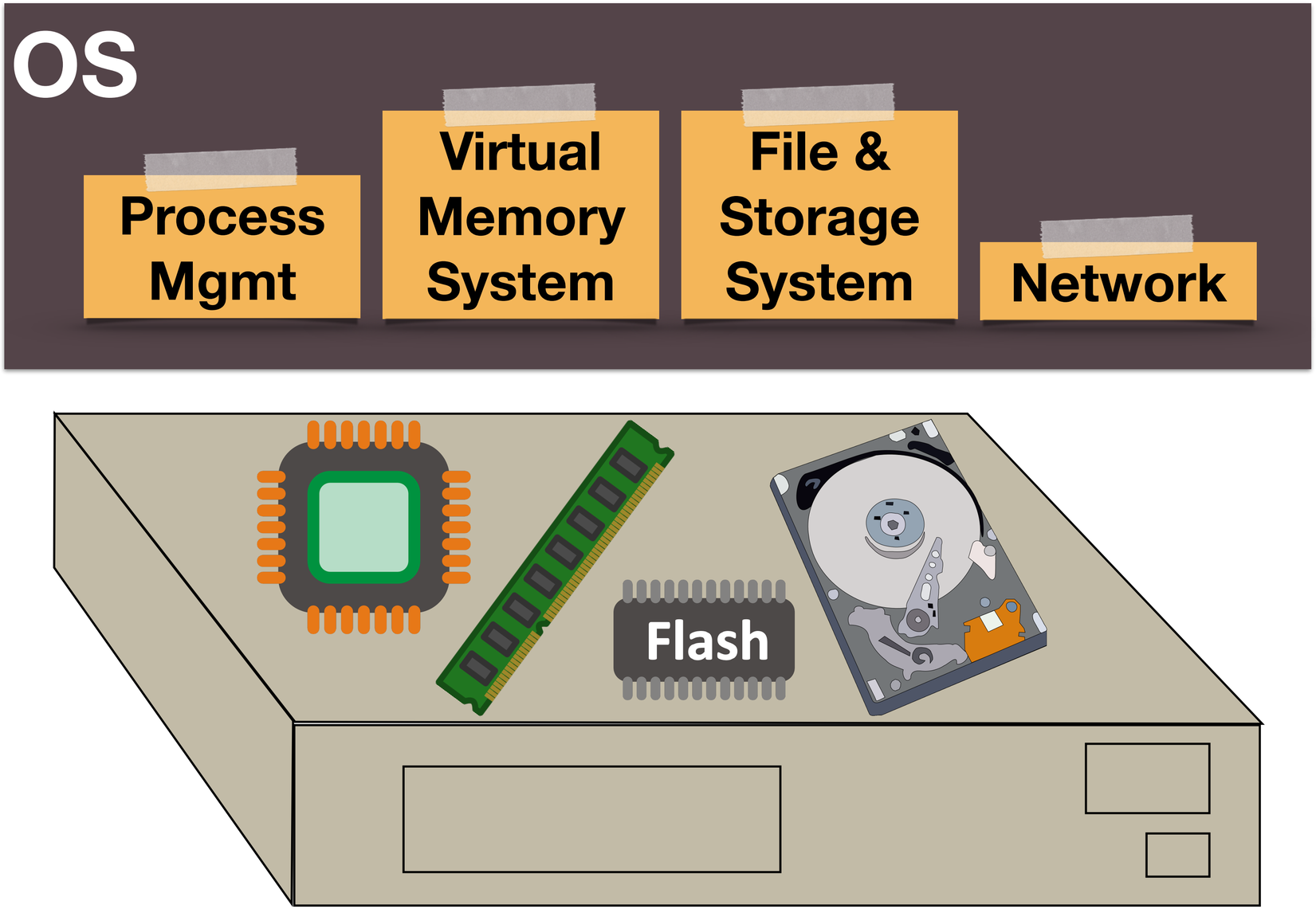
~~Access remote resources~~

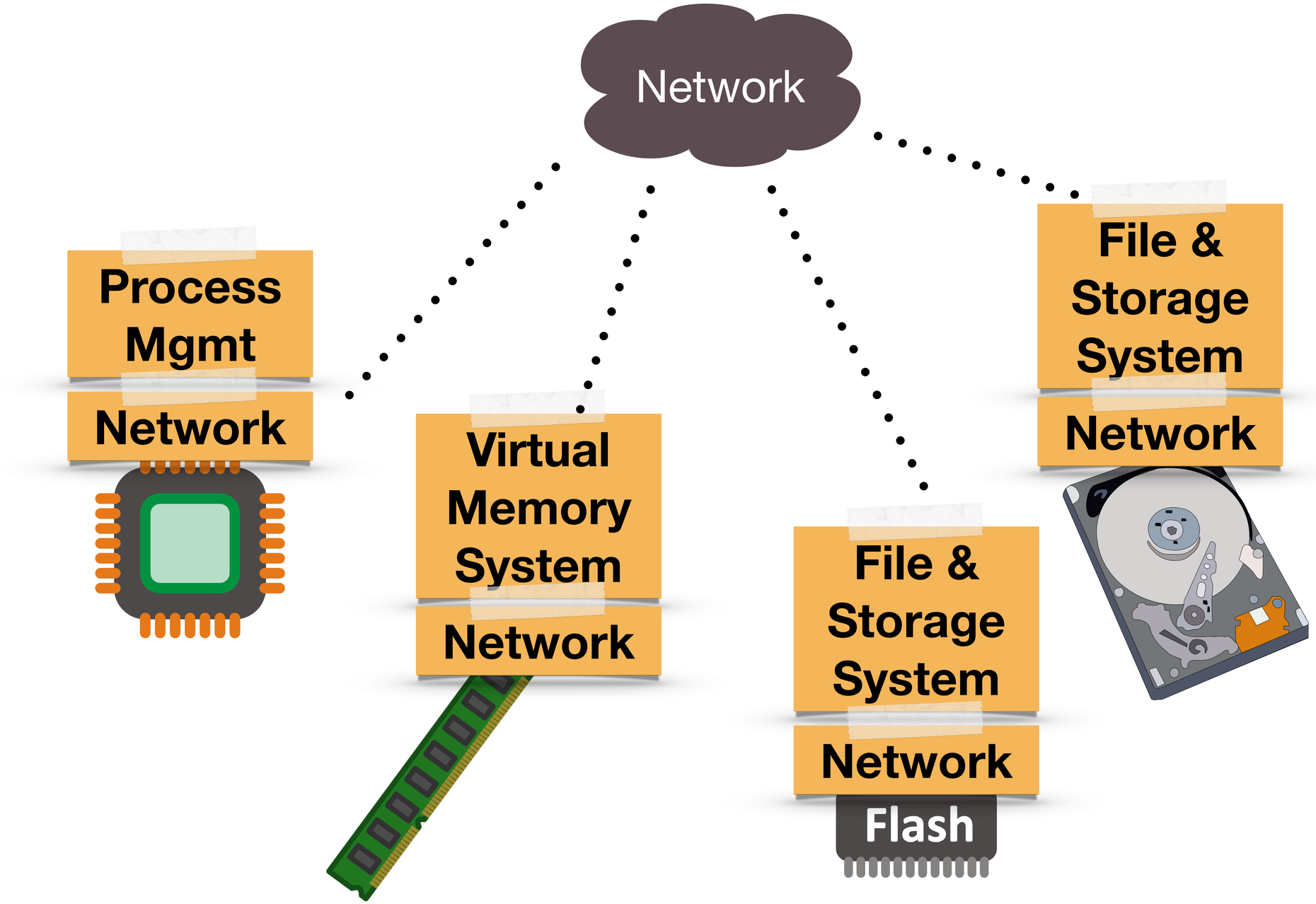
~~Distributed resource mgmt~~

~~Fine-grained failure handling~~

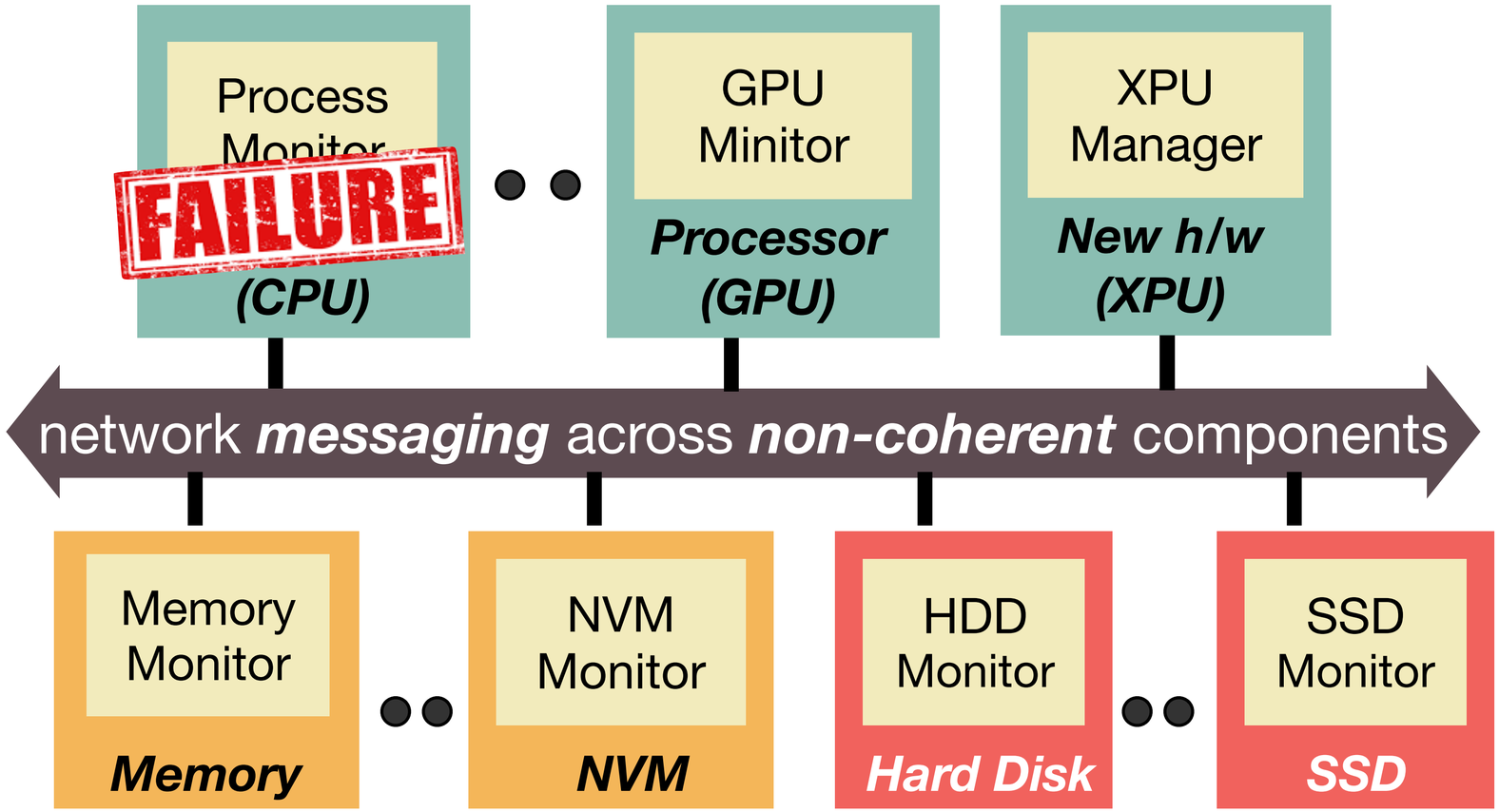
When hardware is disaggregated

The OS should be also


OS
Process Mgmt
Virtual Memory System
File & Storage System
Network
Flash


Network
Process Mgmt
Network
Virtual Memory System
Network
File & Storage System
Network
Flash
File & Storage System
Network

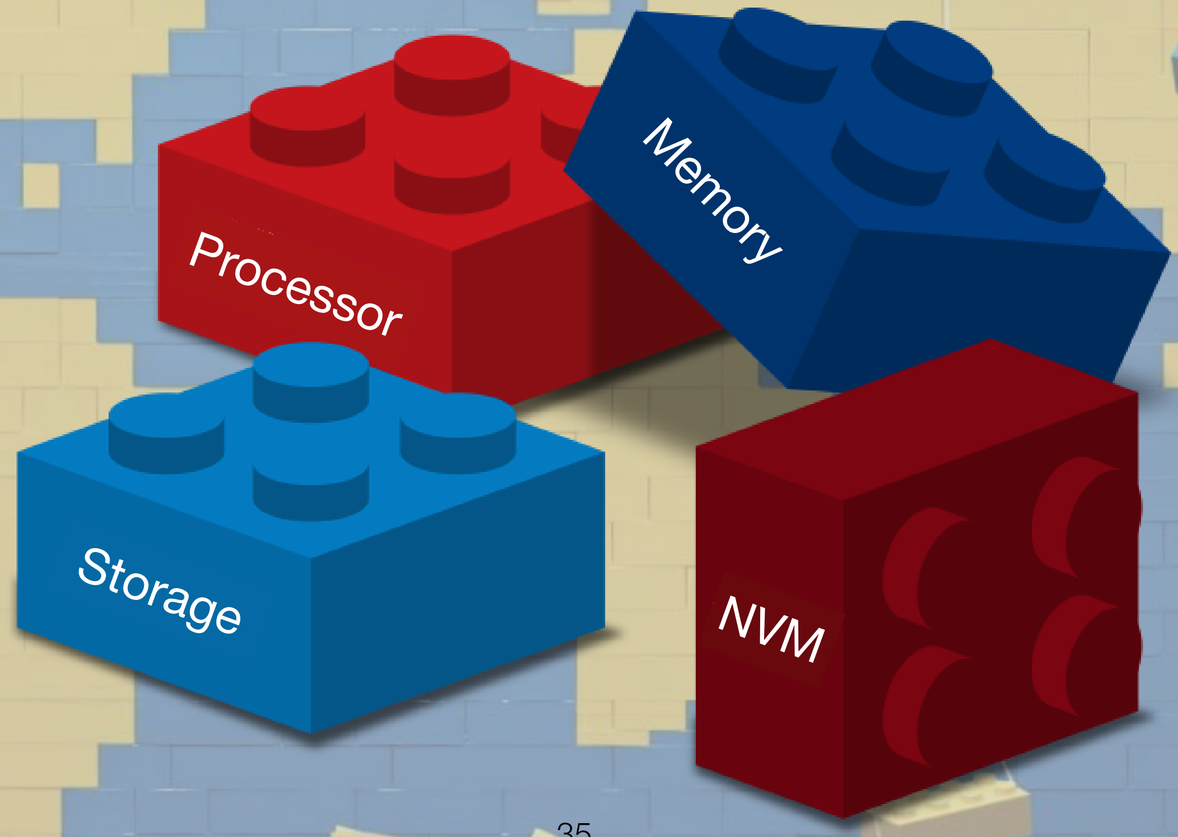
The Splitkernel Architecture



- Split OS functions into ***monitors***
- Run each monitor at h/w device
- Network messaging across non-coherent components
- Distributed resource mgmt and failure handling

LegoOS

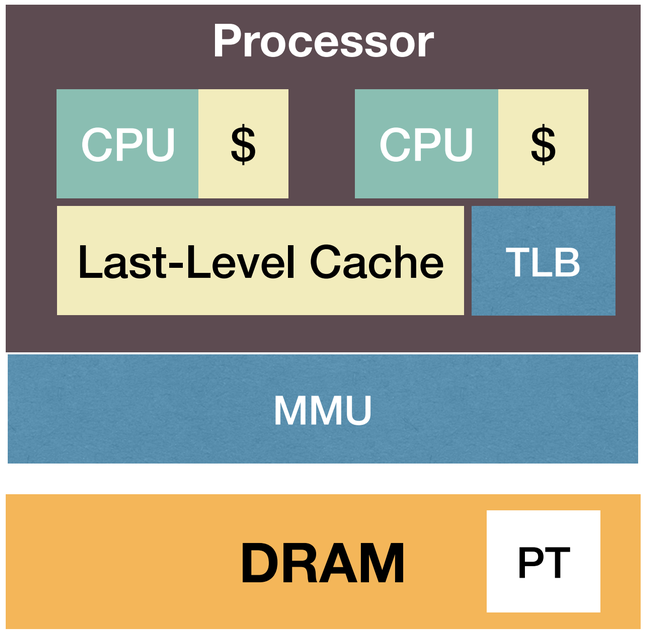
The *First* Disaggregated OS



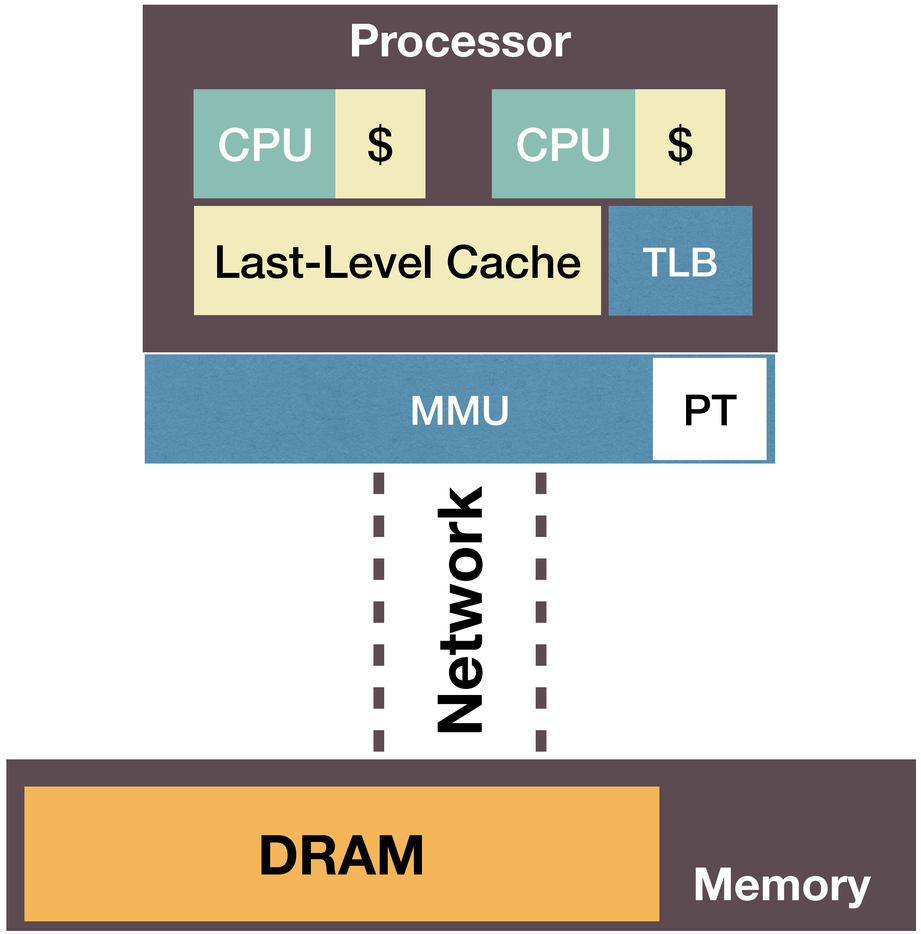
LegoOS Design

1. Clean separation of OS and hardware functionalities
2. Build monitor with hardware constraints
3. RDMA-based message passing for both kernel and applications
4. Two-level distributed resource management
5. Memory failure tolerance through replication

Separate Processor and Memory

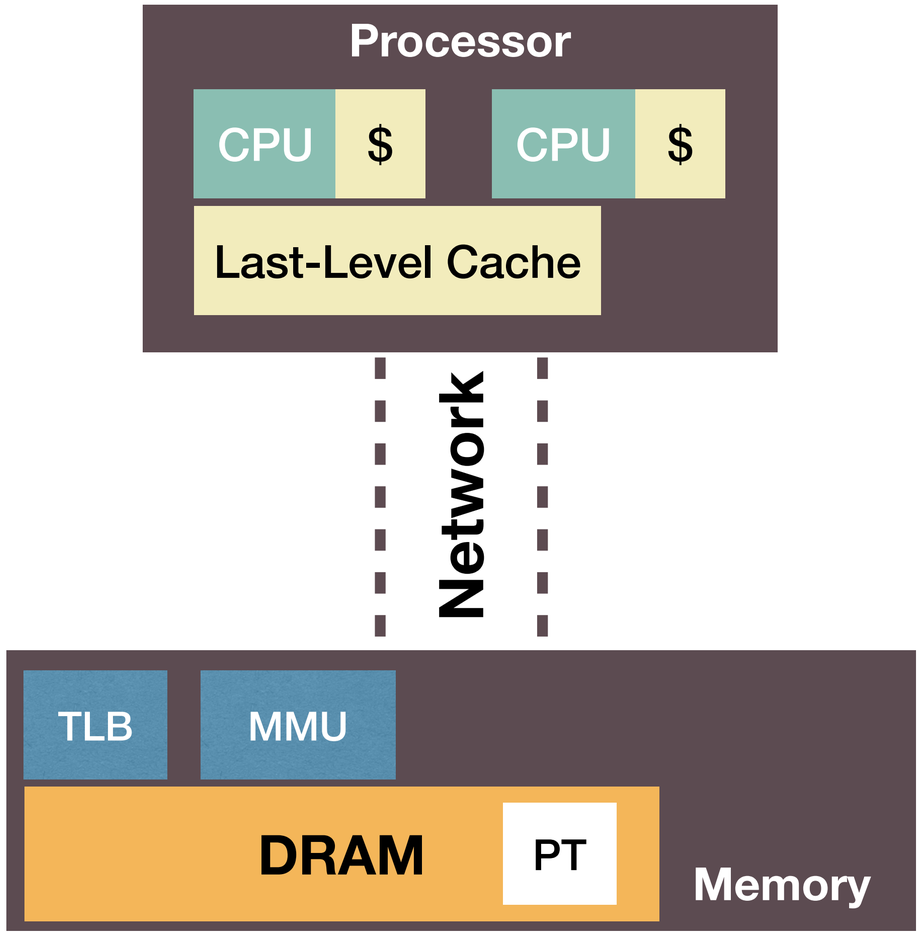


Separate Processor and Memory



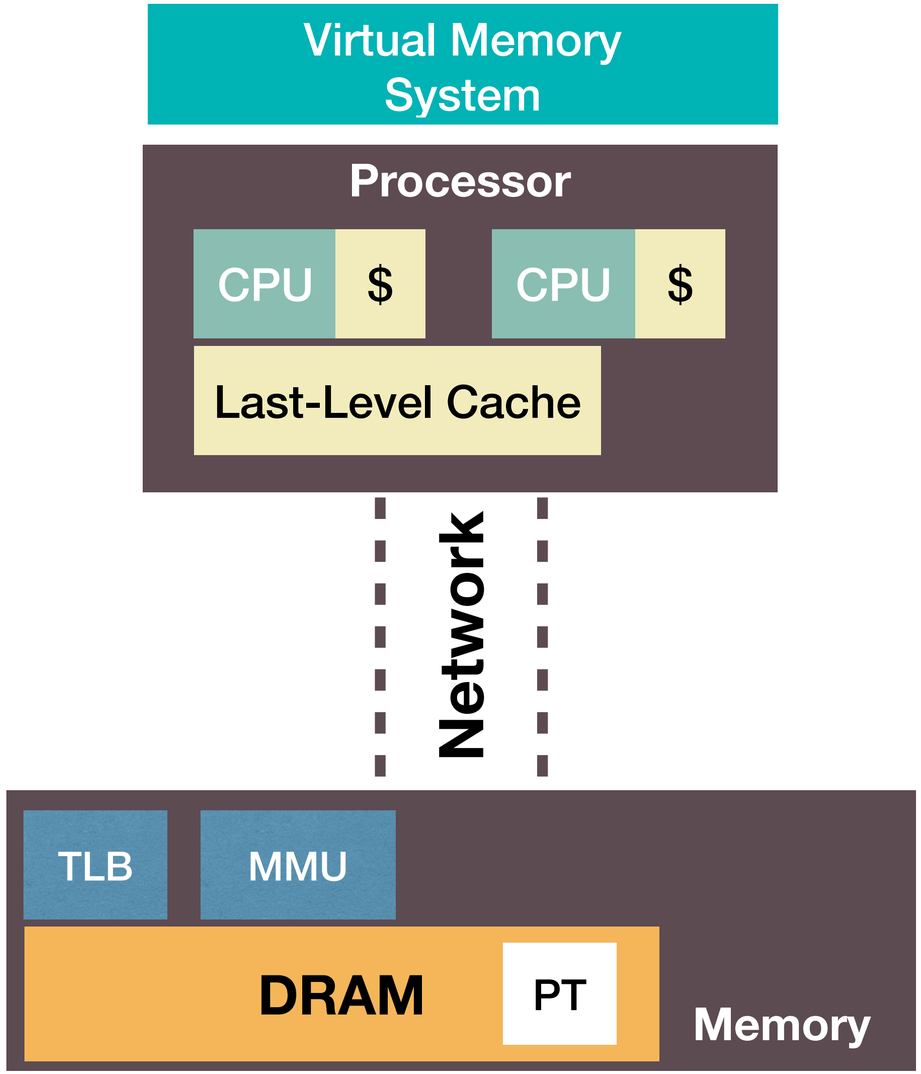
Disaggregating DRAM

Separate Processor and Memory

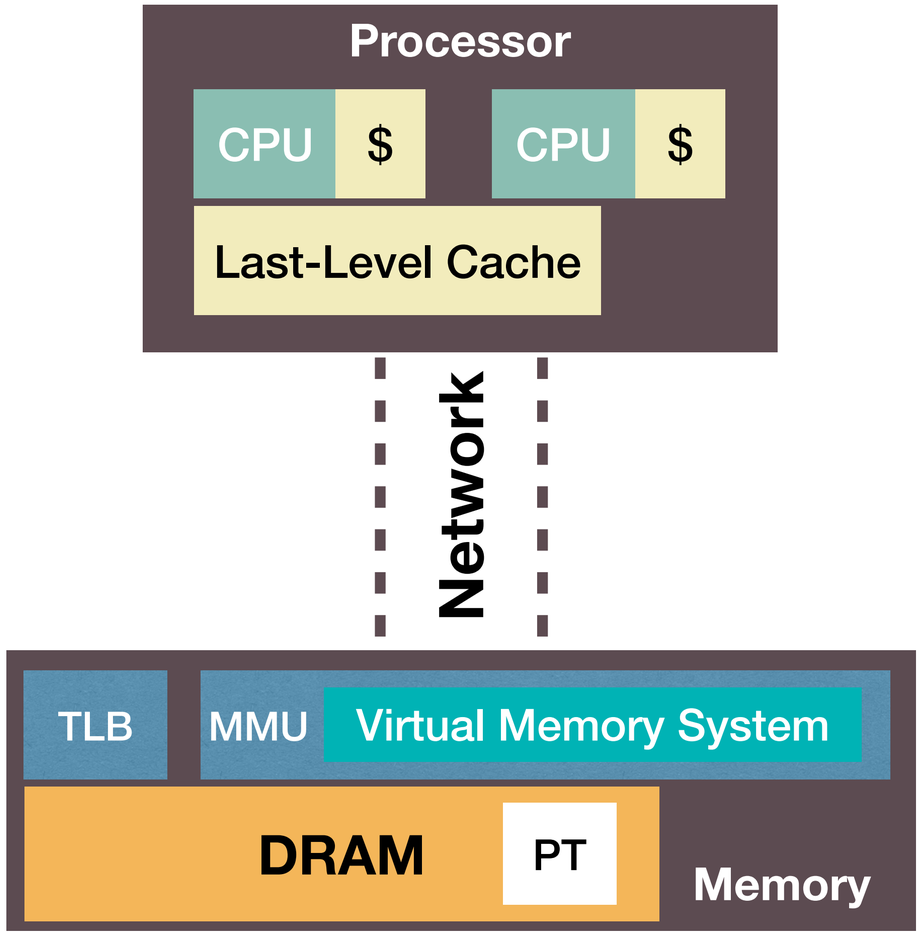


Separate and move
hardware units
to memory component

Separate Processor and Memory

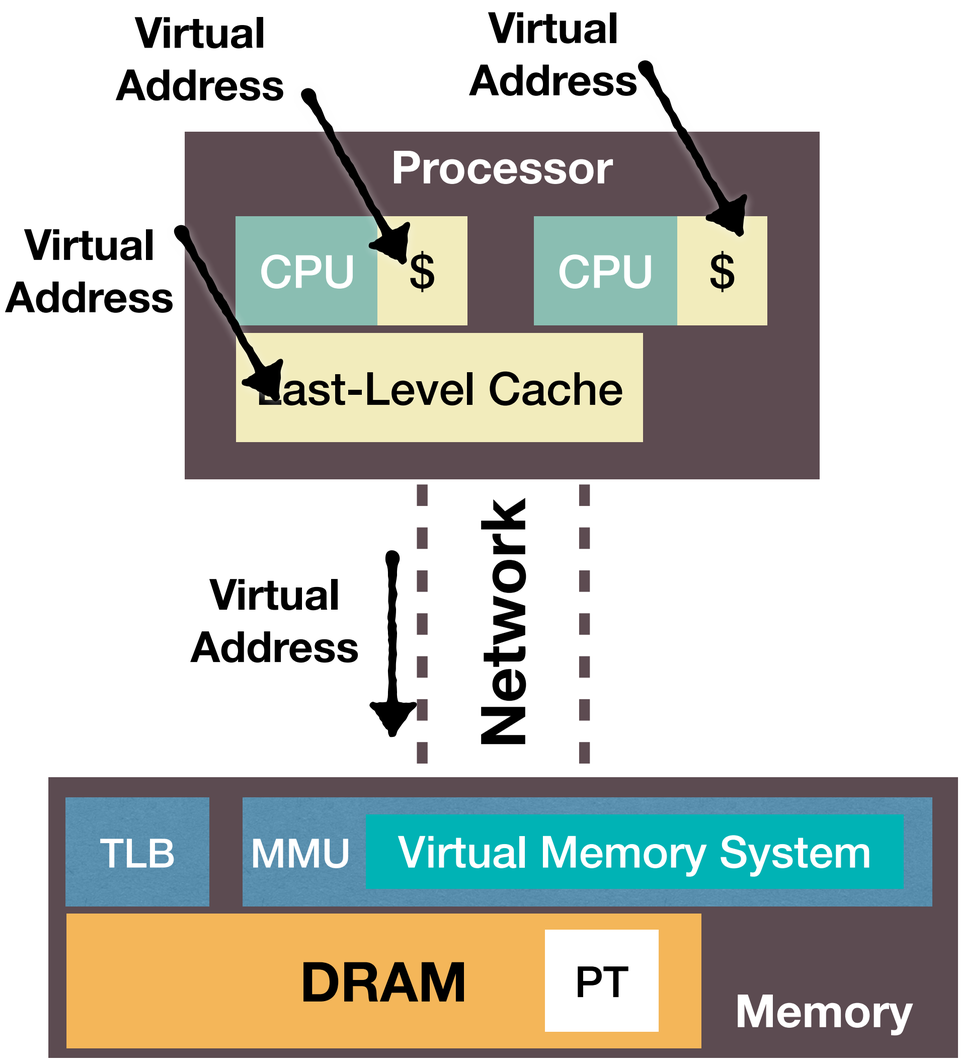


Separate Processor and Memory



Separate and move *virtual memory system* to memory component

Separate Processor and Memory



Processor components only see virtual memory addresses

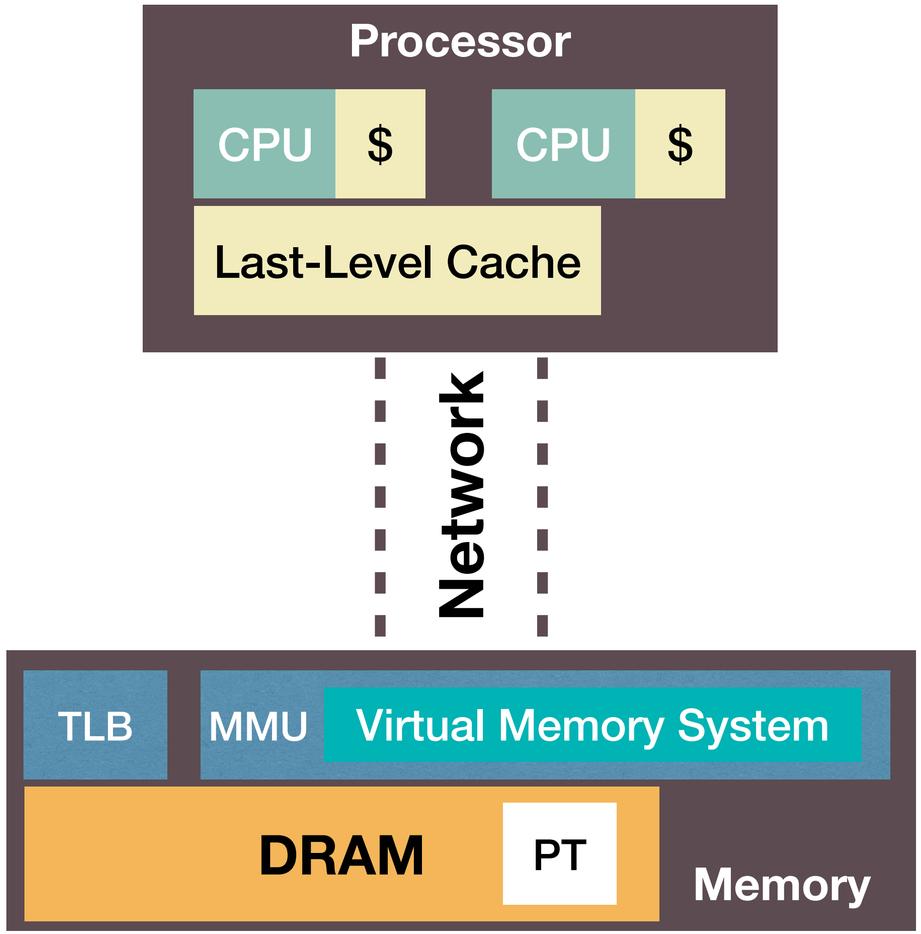
All levels of cache are ***virtual cache***

Memory components manage virtual and physical memory

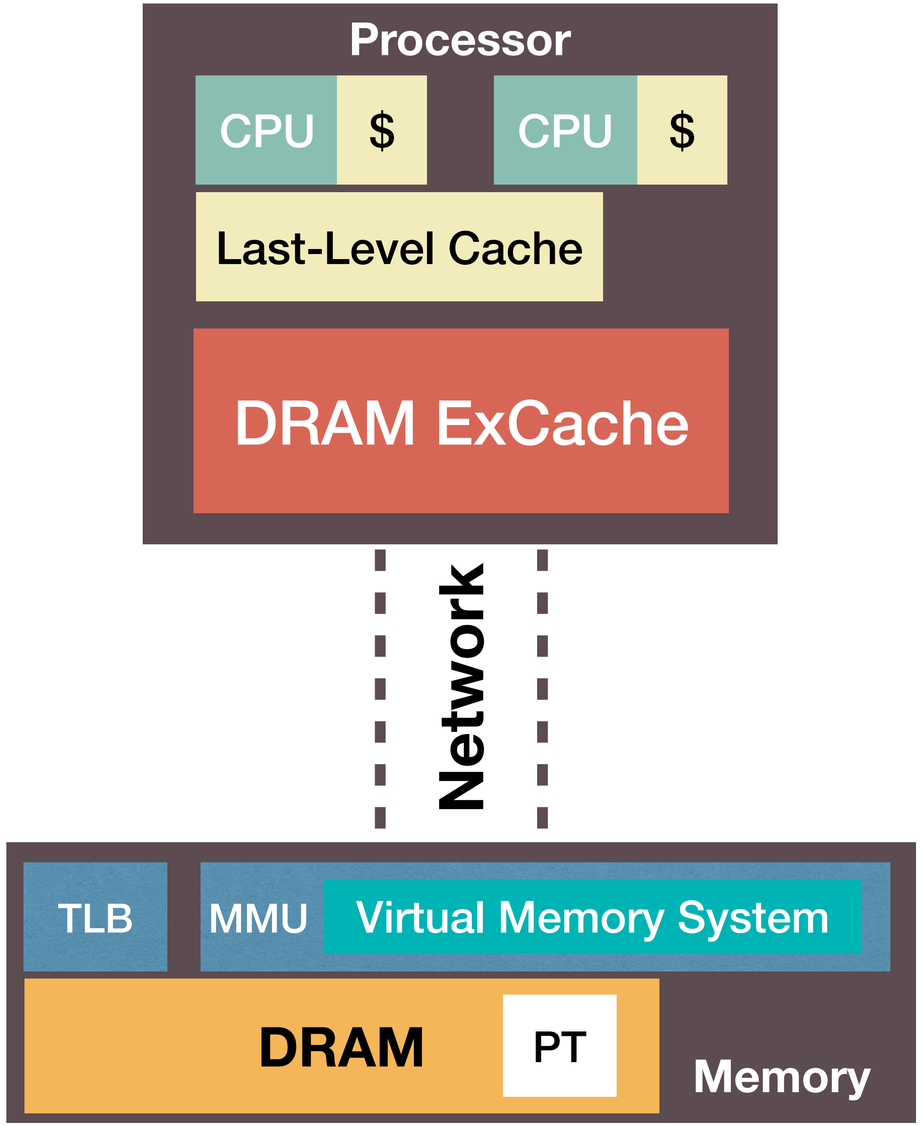
Challenge: Remote Memory Accesses

- Network is still slower than local memory bus
  - Bandwidth: 2x - 4x slower, improving fast
  - Latency: ~12x slower, and improving slowly

Add Extended Cache at Processor

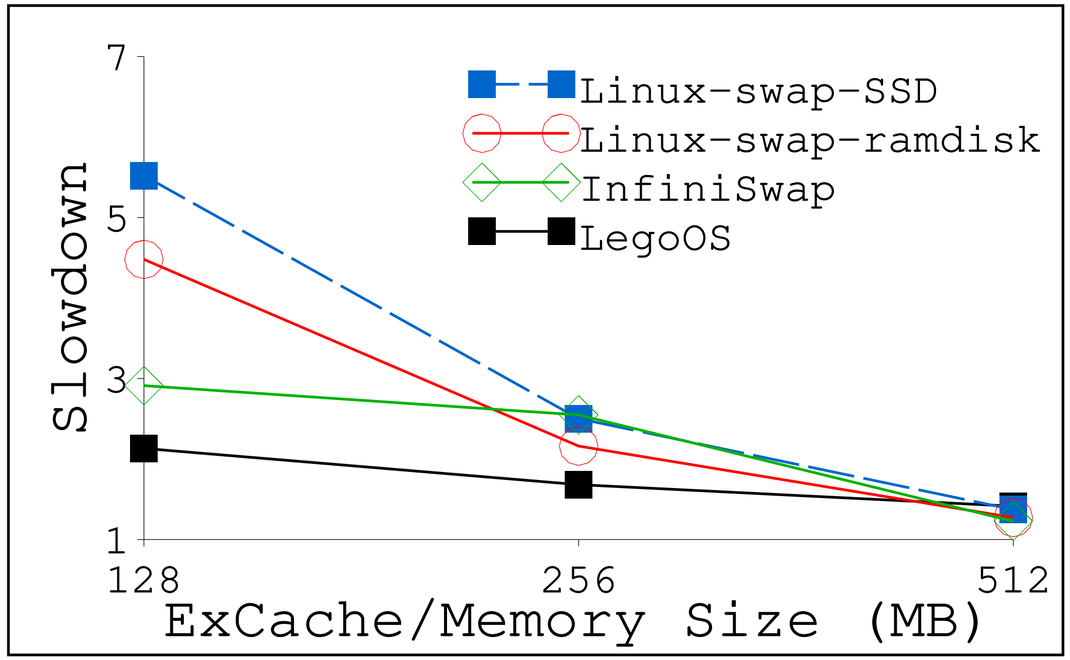


Add Extended Cache at Processor



- Add small DRAM/HBM at processor
- Use it as Extended Cache, or ***ExCache***
  - Software and hardware co-managed
  - Inclusive
  - Virtual cache

Performance Evaluation



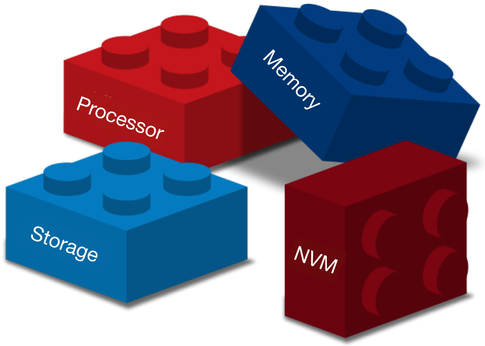
LegoOS Config: 1P, 1M, 1S

- Unmodified TensorFlow, running CIFAR-10
  - Working set: 0.9G
  - 4 threads
- Systems in comparison
  - Baseline: Linux with unlimited memory

Only 1.3x to 1.7x slowdown when disaggregating devices with LegoOS

To gain better resource packing, elasticity, and fault tolerance!

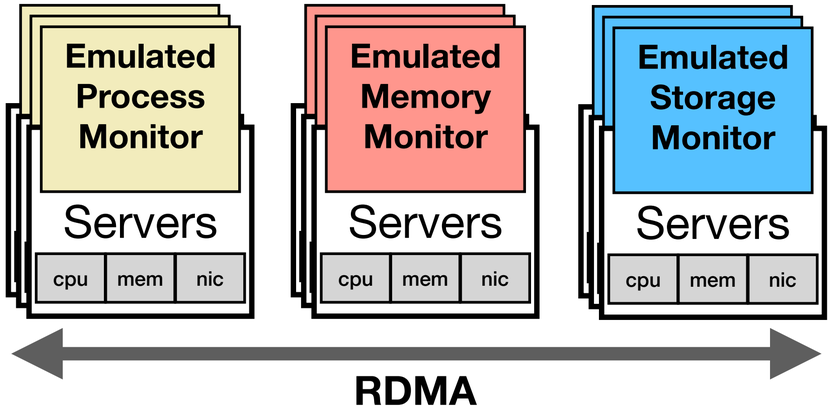
LegoOS Summary



- **LegoOS shows that Physical Disaggregation is feasible**
  - It is possible to disaggregate resources like CPU and memory
  - Decent perf slowdown (30%-70%), but with overall improved [*perf* / *$*]
  - Improved utilization, cost, failure (MTTF), and manageability
- **Key enabling techniques**
  - The Splitkernel architecture for module & failure isolation
  - Extended Cache for performance
  - Two-level approach for resource management

Problems?

- **Devices are *emulated using RDMA and CPU***
  - Non-trivial overheads
  - Limited parallelism
- **Radical approach, hard to deploy**
  - extensive hardware and network changes
  - uncertain system software and app changes



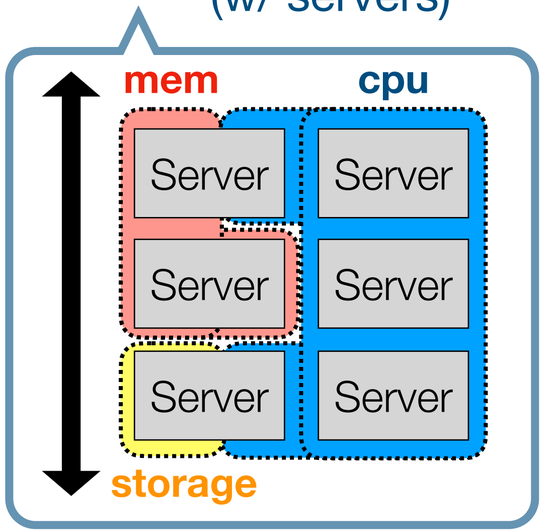
⇒ This motivates us to build a real hardware-based disaggregated device that could actually be deployed

Outline

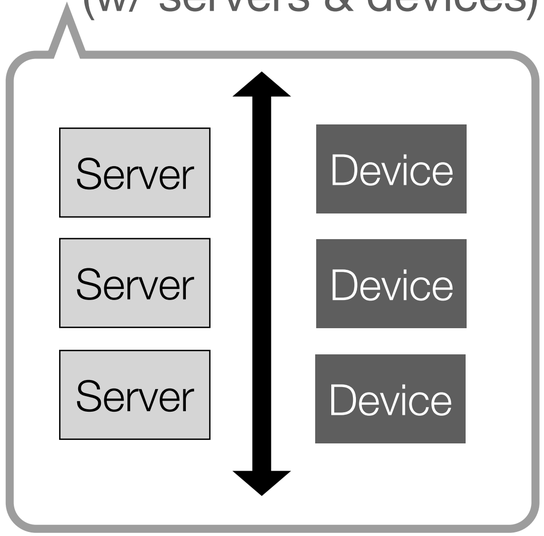
- Background on Resource Disaggregation
- Projects Conducted
  - Logical Disaggregation [Hotpot, SoCC'17]
  - Physical Disaggregation [LegoOS, OSDI'18]
  - **Hybrid Disaggregation** [Clio, ASPLOS'22]
  - **Network Disaggregation** [SuperNIC, arXiv'21, under submission]
- **Future Work**
- **Conclusion**

Resource Disaggregation Design (Cooking) Spectrum (Recipes)

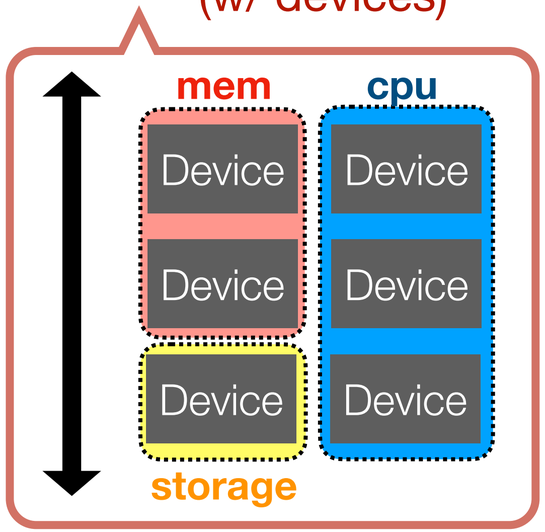
Logical
(w/ servers)



Hybrid
(w/ servers & devices)



Physical
(w/ devices)

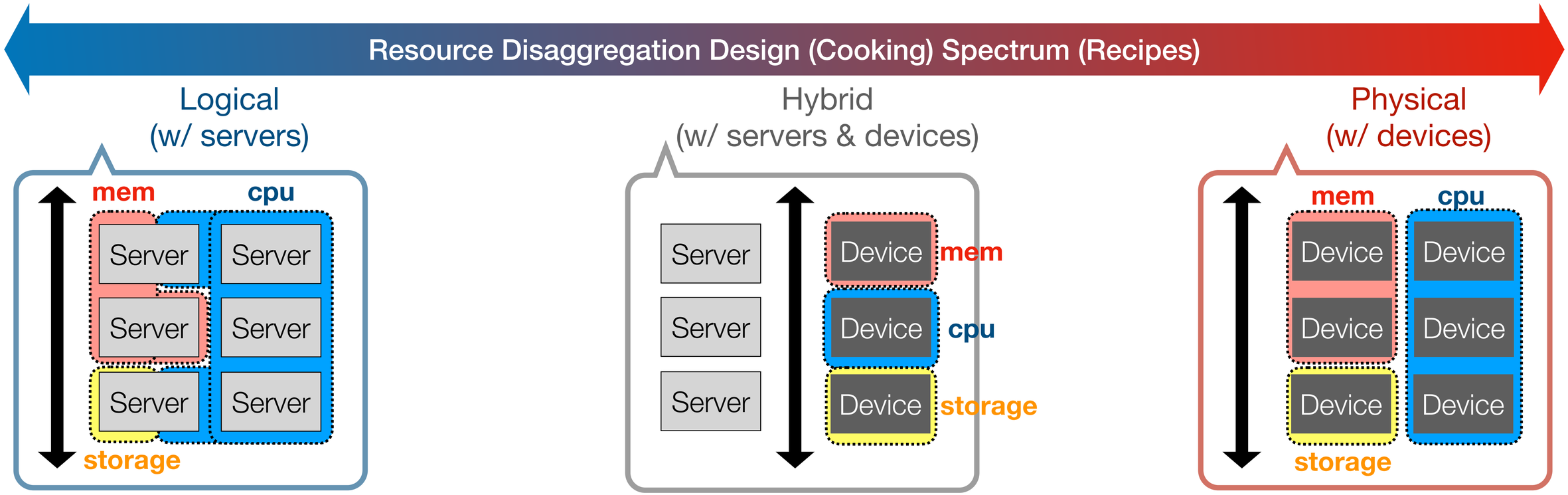


Part 1
Distributed Shared Persistent Memory
[Hotpot, SoCC'17]

Part 3
Hardware-based Disaggregated Memory
[Clio, ASPLOS'22]

Part 2
Disaggregated Operating System
[LegoOS, OSDI'18]

Part 4
Disaggregated Networking For the Masses
[SuperNIC, arXiv'21]

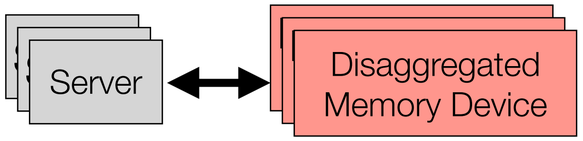


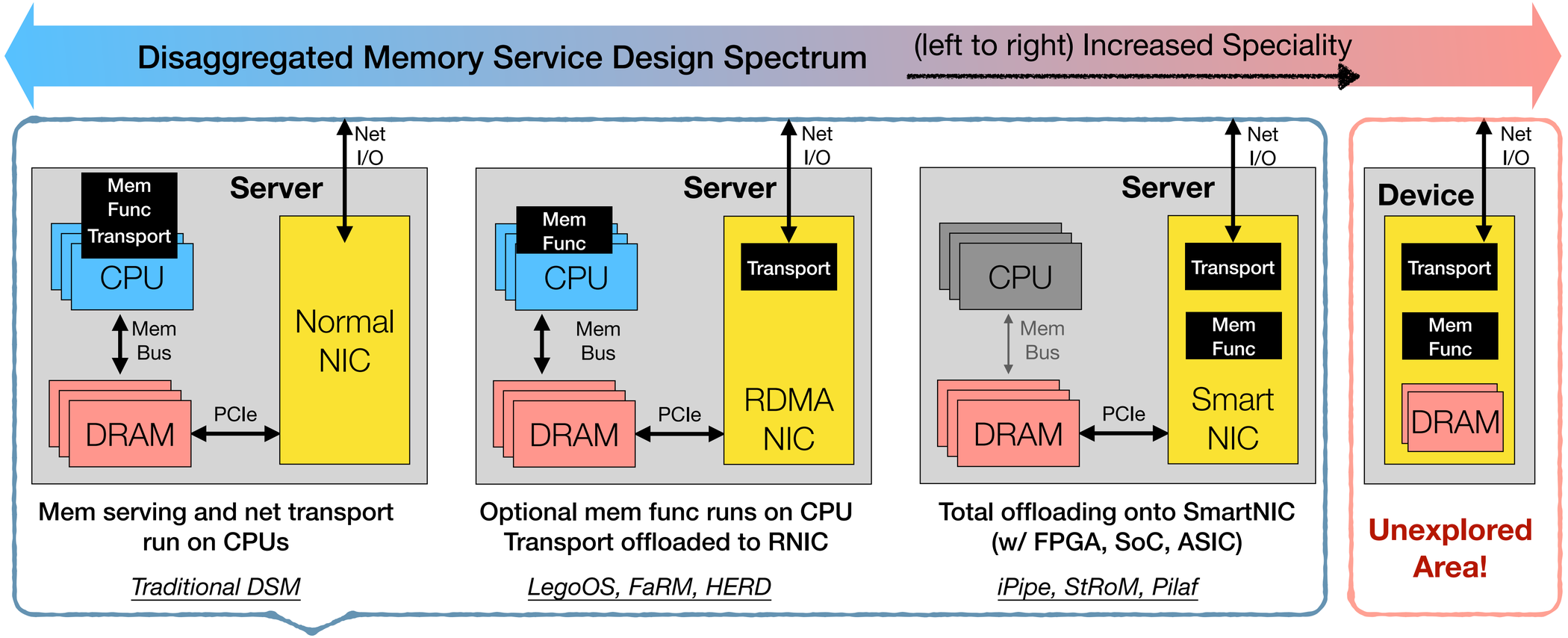
A much easier transition from the current data center infrastructure

Our goal here is to build a real disaggregated device and integrate it with the existing infrastructure

We start from the most challenging resource to disaggregate: memory.
(high perf demand, large capacity, security)

How to Design Disaggregated Memory Service?





Server box is an overkill for memory disaggregation

- Unused resource
- Limited capacity
- Limited RDMA functionalities
- Limited PCIe performance

Clio: A Hardware-Software Co-Designed Disaggregated Memory System

Zhiyuan Guo*, **Yizhou Shan***, (** equal contribution*)
Xuhao Luo, Yutong Huang, and Yiying Zhang

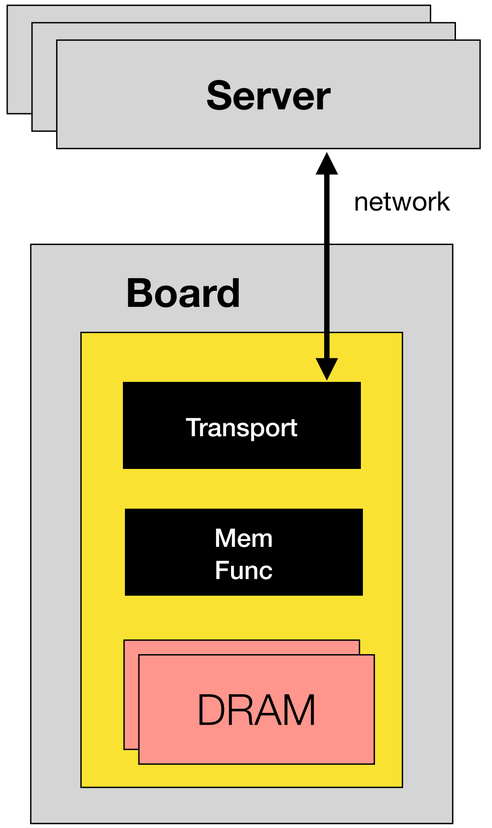
UCSDCSE
Computer Science and Engineering

WukLab

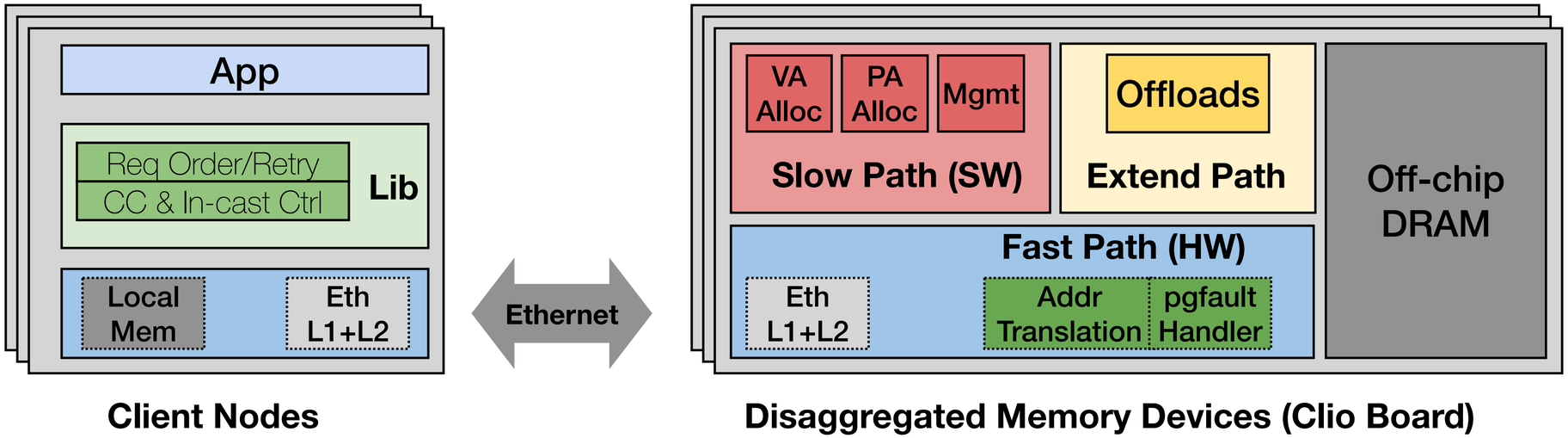
[3] Zhiyuan Guo, **Yizhou Shan***, Xuhao Luo, Yutong Huang, and Yiying Zhang (* equal contribution).
Clio: A Hardware-Software Co-Designed Disaggregated Memory System, ASPLOS'22*

Our Vision and Design Principles

- **Goals**
  - **Scalable**: able to support 1K-10K connections
  - **Huge Memory**: able to host TBs of memory
  - **Performant**: low and predictable (tail) latency
  - **Extensible**: able to run user-specific functions
- **Principles**
  - Eliminate states whenever possible
  - Move non-critical ops/states to SW, simplify HW design
  - Shift ops/states to client side



Clio Architecture



- **A new customized transport**
  - RPC-based abstraction
  - Sender-driven retransmission, congestion, and in-cast control
  - Flexible ordering and consistency model
- **Hash table-based Virtual Memory System**
  - Flat & conflict-free hash table-based virtual memory
  - in-hardware in-line page fault handling
- **Framework to deploy user-specific logic**

Slow Path
- SoC
- Software

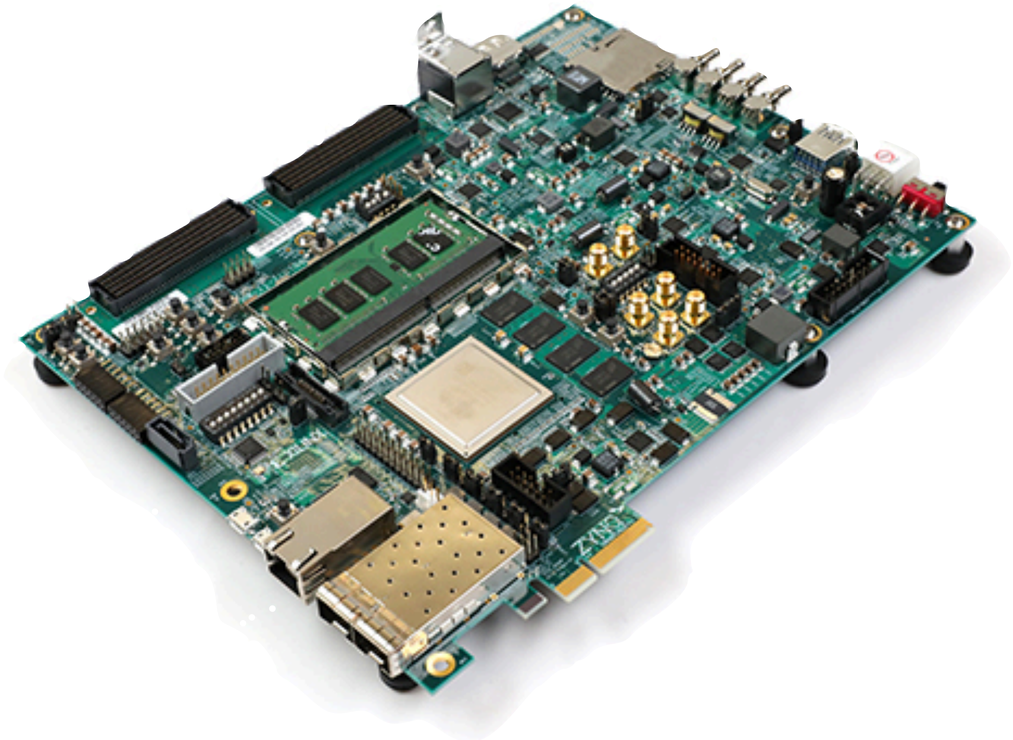
Fast Path
- ASIC + FPGA
- Transport + VM

Extend Path
- FPGA
- User logic

Implementation

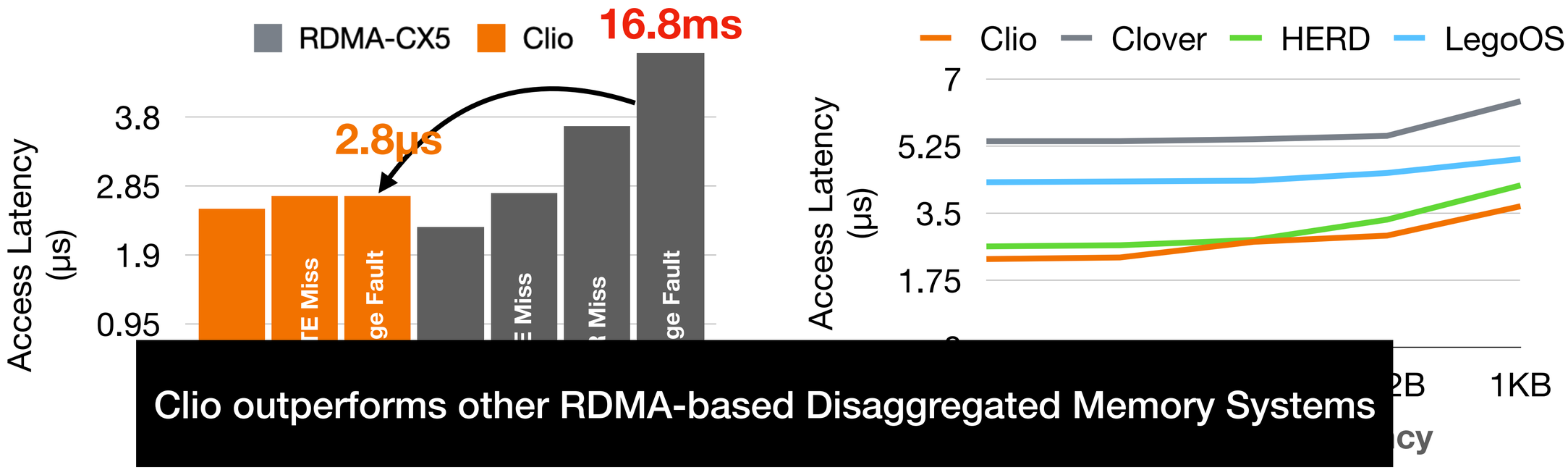
- **Xilinx ZCU106 ARM-FPGA board**
  - Shell adopted from Corundum
  - Fast & extended path in SpinalHDL
  - Slow path runs on ARM SoC
- **Applications**
  - Image compression
  - Multi-version object store
  - KVS
  - Pointer-chasing

Clio prototype on the Xilinx ZCU106 board



Clio Eval - Basic Numbers

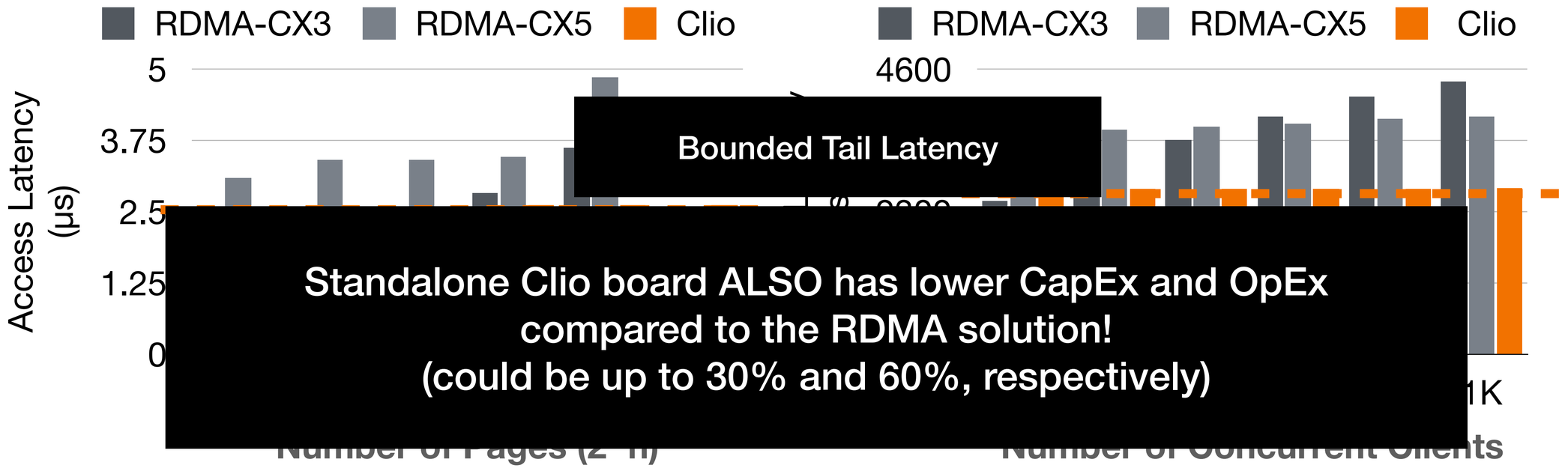
- **100Gbps** throughput, **2.8µs (avg) 3.2µs (p99)** latency
- **Orders of magnitude lower tail latency than RDMA**
- **Outperforms Clover [ATC'20], LegoOS [OSDI'18], and HERD [SIGCOMM'14]**



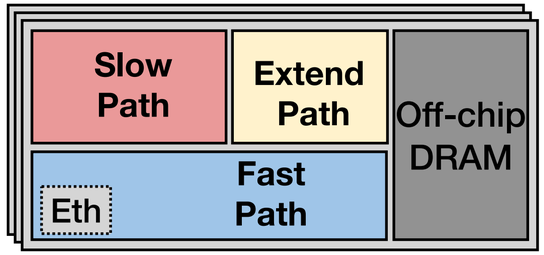
Clio outperforms other RDMA-based Disaggregated Memory Systems

Clio Eval - Scalability

- **Clio provides bounded access time for data requests**
- **Clio scales well with concurrent clients and total memory size**



Clio Summary



- **Clio shows that building disaggregated devices is REWARDING**
  - Hardware-software co-design is important for disaggregated devices
  - Overhaul the network transport and virtual memory system
  - ==> Better performance, Lower CapEx and OpEx than commercial solutions!
- **Problems?**
  - Do we need to do the exact same thing for each disaggregated device?
  - Will vendors adopt our networking solution in their products?

⇒ ***We turned out attention to the long overlooked resource***

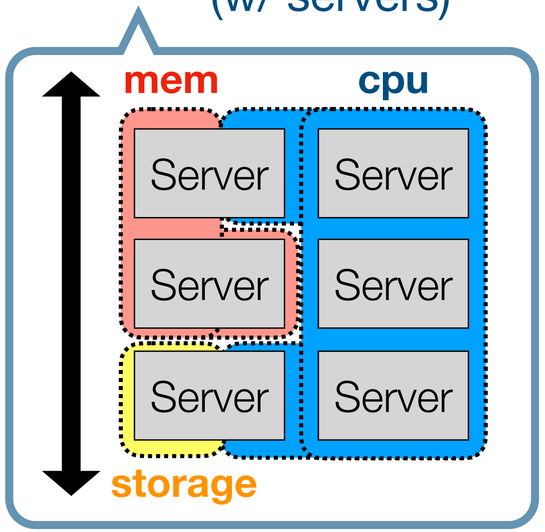
Network

Outline

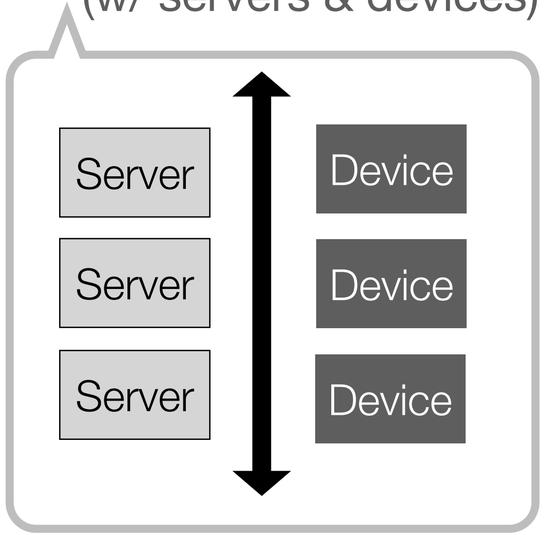
- Background on Resource Disaggregation
- Projects Conducted
  - Logical Disaggregation [Hotpot, SoCC'17]
  - Physical Disaggregation [LegoOS, OSDI'18]
  - Hybrid Disaggregation [Clio, ASPLOS'22]
  - **Network Disaggregation** [SuperNIC, arXiv'21, under submission]
- **Future Work**
- **Conclusion**

Resource Disaggregation Design (Cooking) Spectrum (Recipes)

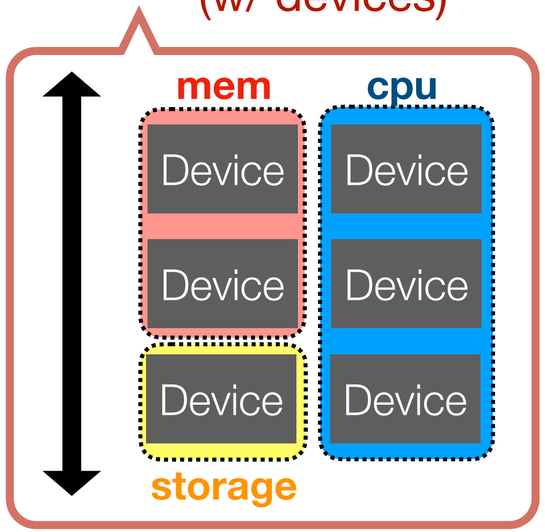
Logical
(w/ servers)



Hybrid
(w/ servers & devices)



Physical
(w/ devices)



Part 1
Distributed Shared
Persistent Memory
[Hotpot, SoCC'17]

Part 3
Hardware-based
Disaggregated Memory
[Clio, ASPLOS'22; Clover, ATC'20]

Part 2
Disaggregated
Operating System
[LegoOS, OSDI'18]

Part 4
Disaggregated Networking
For the Masses
[SuperNIC, arXiv'21]

Disaggregating and Consolidating Network Functionalities with SuperNIC

Yizhou Shan, Will Lin, Ryan Kosta,
Arvind Krishnamurthy, and Yiying Zhang

WukLab

UCSDCSE
Computer Science and Engineering

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

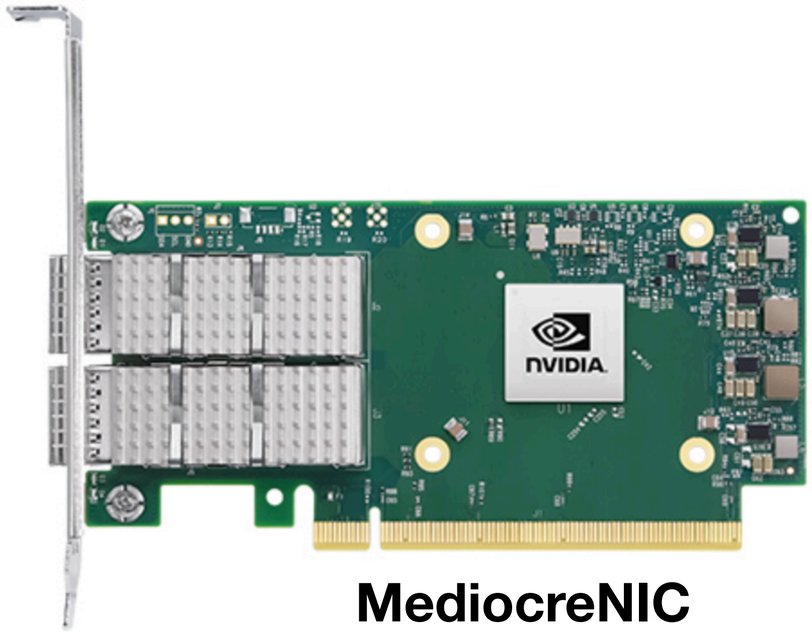
What others say about SuperNIC

- Colleague A: This is THE most elegant solution I’ve ever seen
- Colleague B: I can’t agree more
- Colleague C: I wish all my projects could be like this one
- Colleague D: I wish all my students were like you

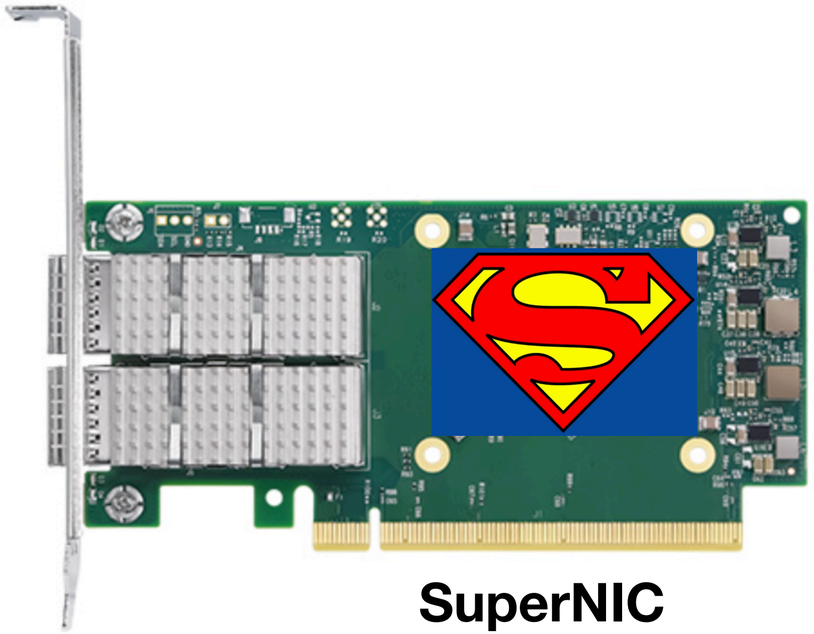
“The Problem”

- **Professor**: Those NICs, they are a problem for disaggregation.
- **Me**: How come?
- **Professor**: Well, they are kind of slow and weak. Just.. mediocre.
- **Me**: Ok. Hold my beer.

“The Solution”



MediocreNIC



SuperNIC

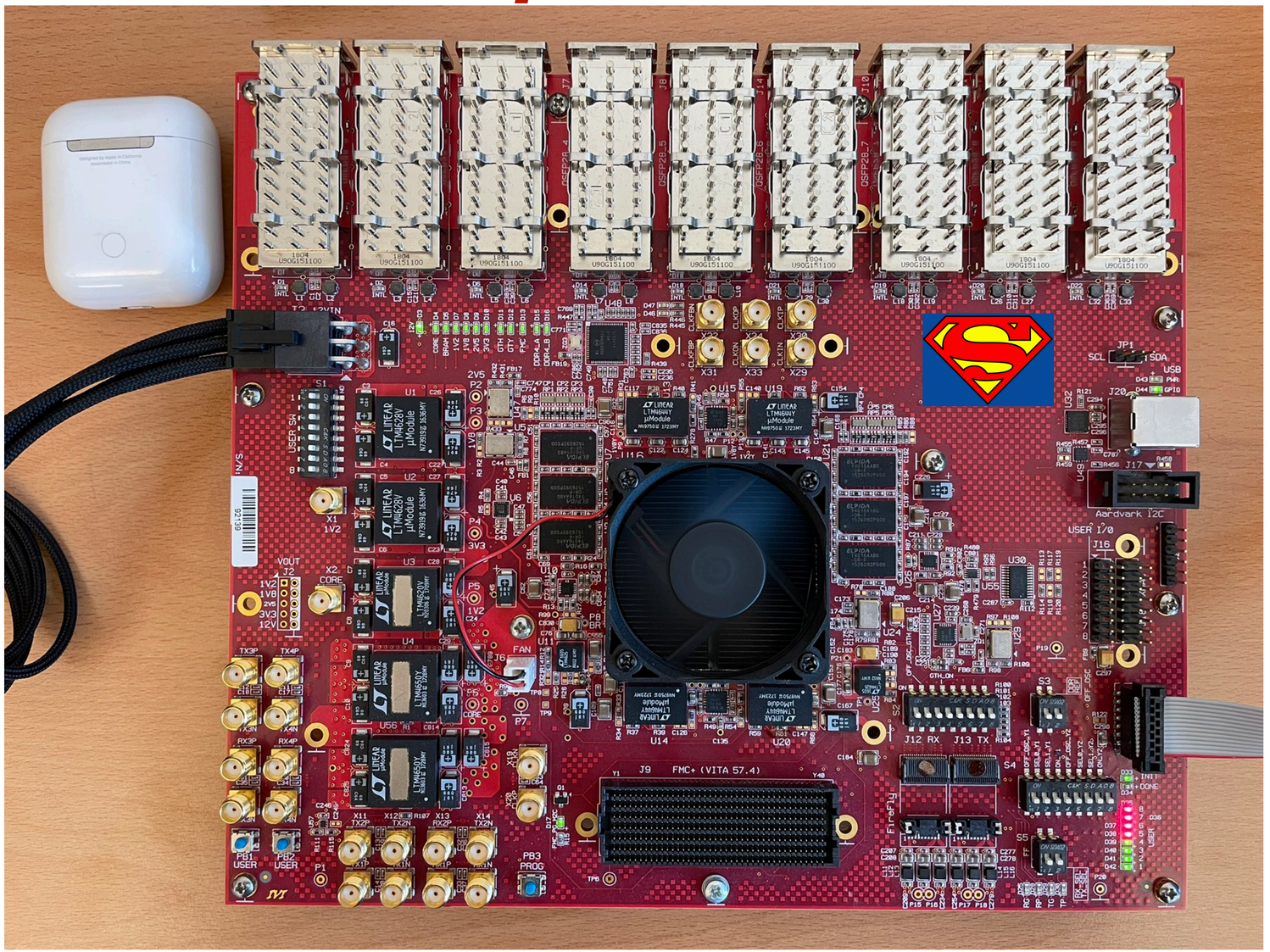
Disaggregating and Consolidating Network Functionalities with SuperNIC

Yizhou Shan[†], Will Lin[†], Ryan Kosta[†], Arvind Krishnamurthy[*], Yiying Zhang[†]
[†]*University of California San Diego,* [*]*University of Washington*

A SuperNIC is just an ordinary NIC who has found a better way to mask their NIC frailties.

Let’s talk about SuperNIC.

SuperNIC

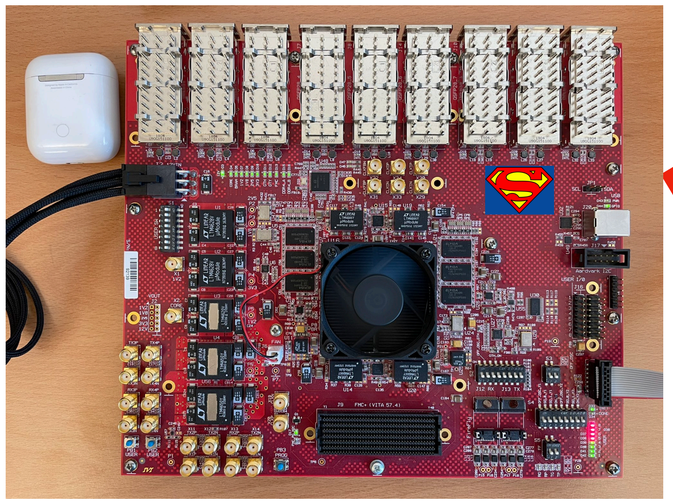


HTG-9200

- 9x100G QSFP
- Xilinx VU9P
- 8GB DRAM

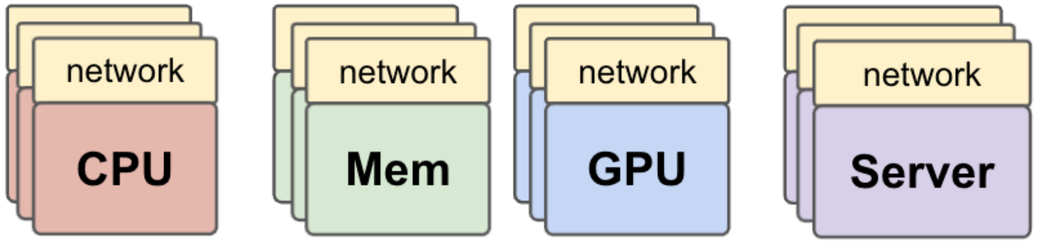
Special Thanks

- **The FPGA Ninja himself - Alex Forencich**
  - For sponsoring two boards
  - For helping us on numerous debugging sessions



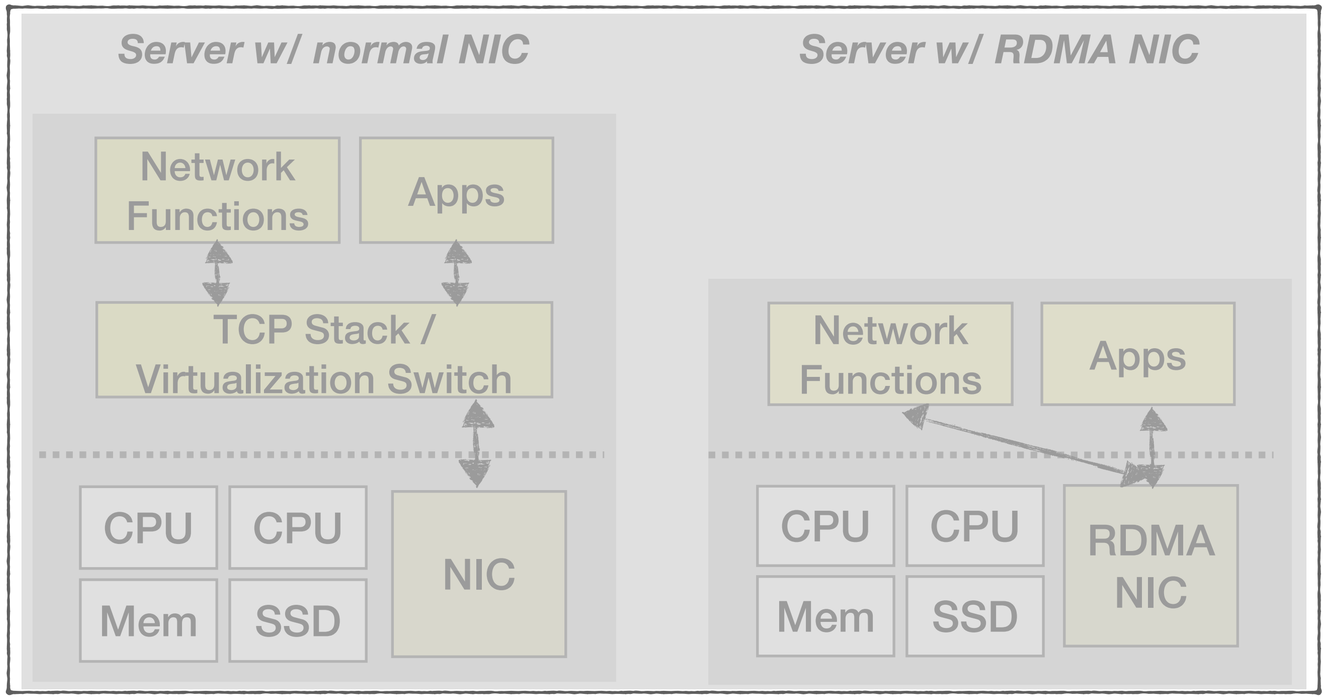
What's next for resource disaggregation?

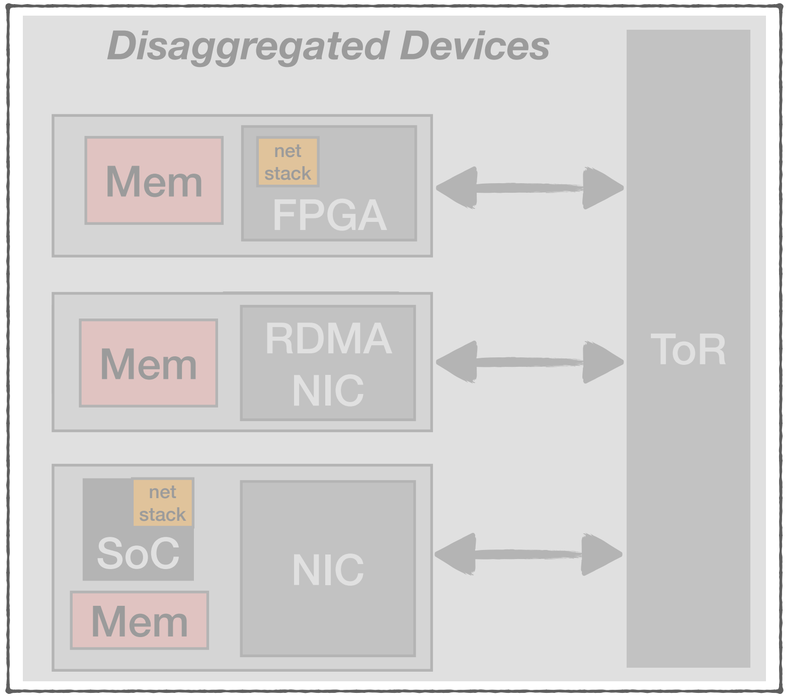
- **Resources already disaggregated**
  - Processing (e.g., CPU, GPU, TPU)
  - Memory (e.g., DRAM, PM)
  - Storage (e.g., SSD)
- **But *network* is completely left out!**

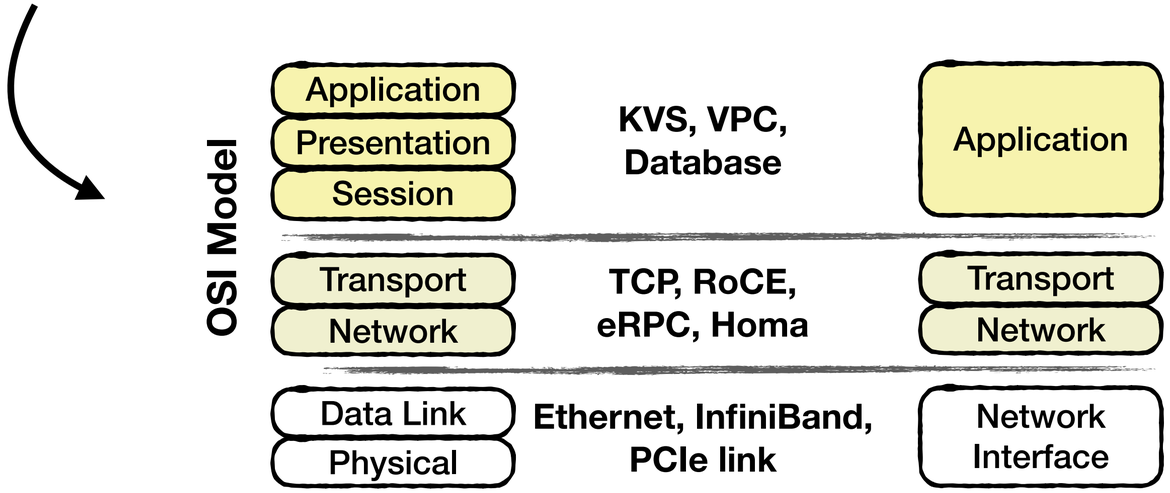


Hold on..

Can we disaggregate network?

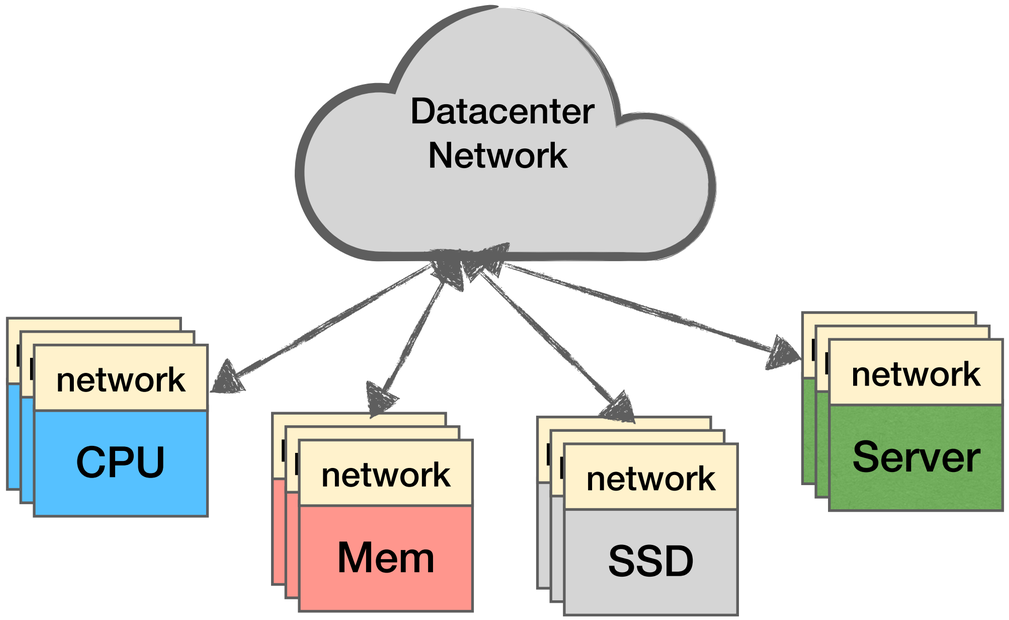


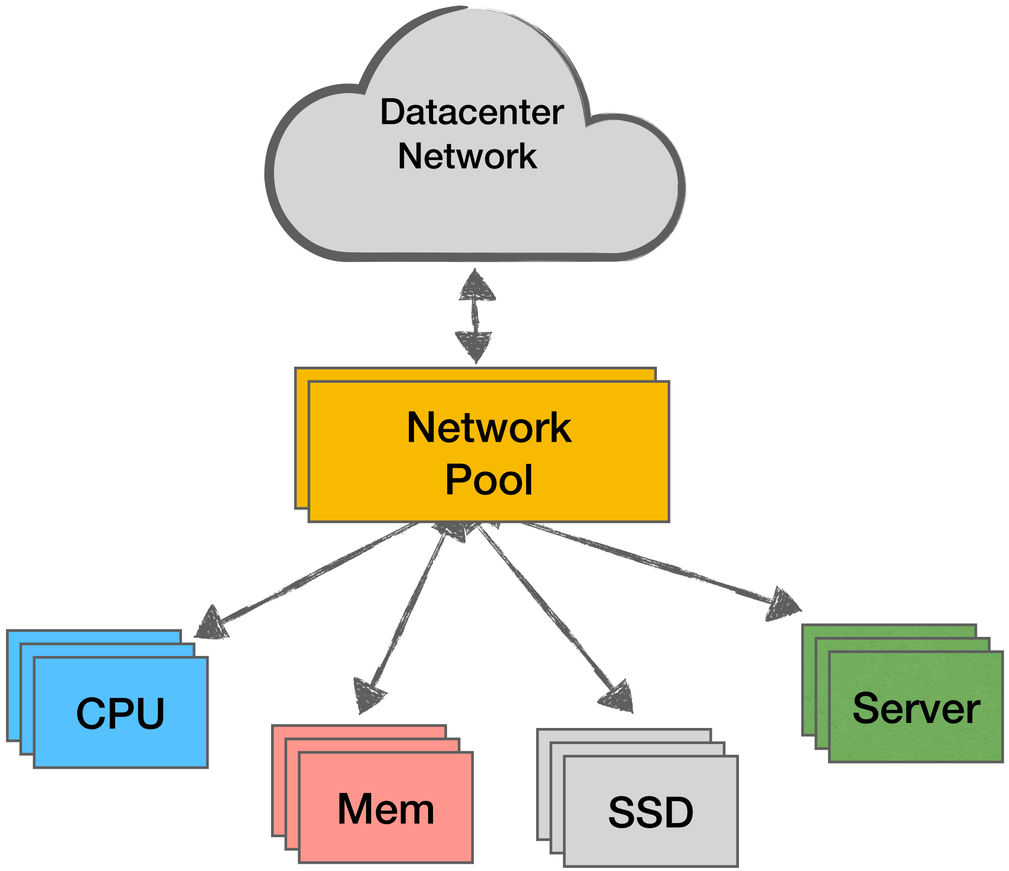


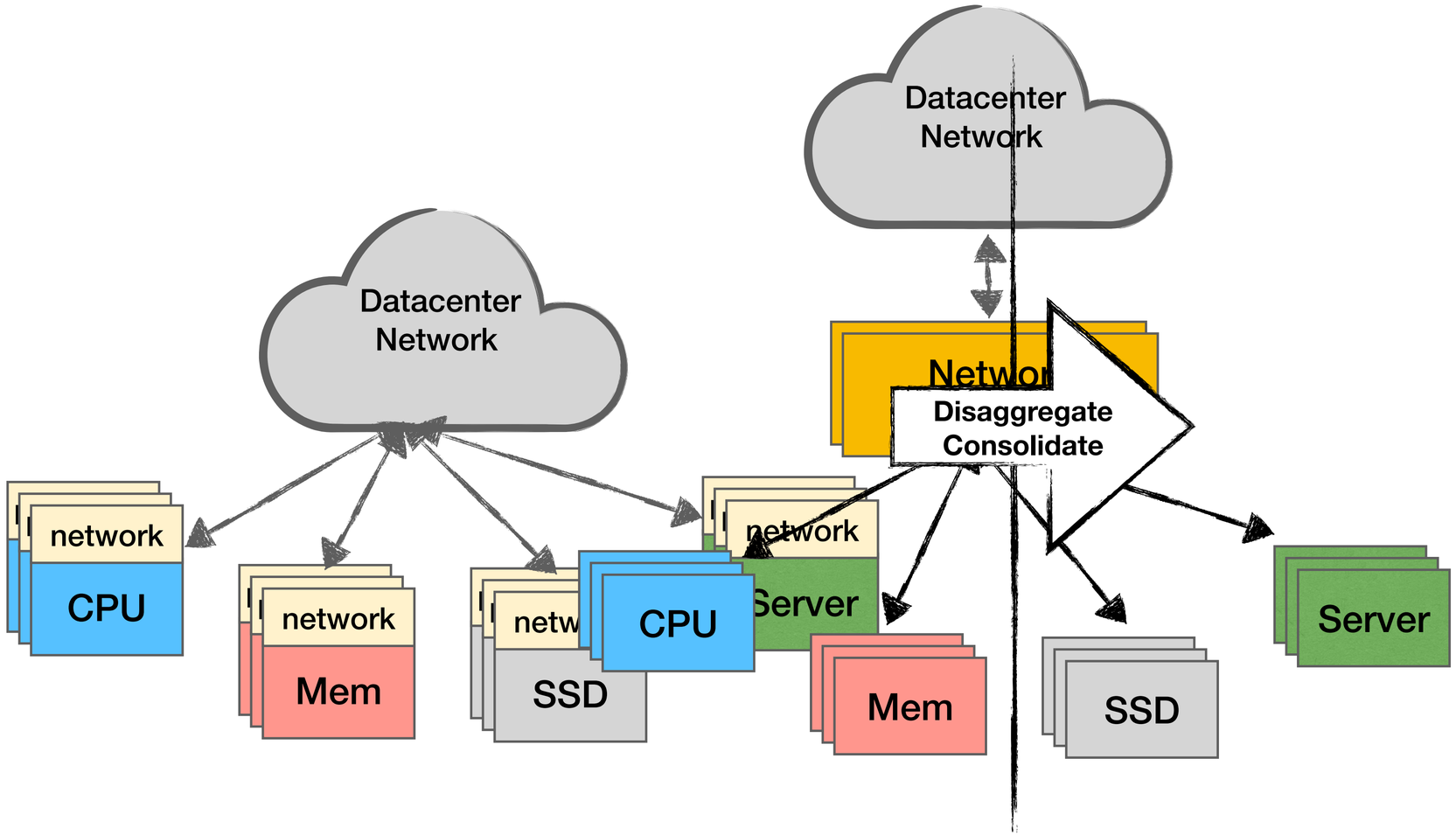


Our Insights

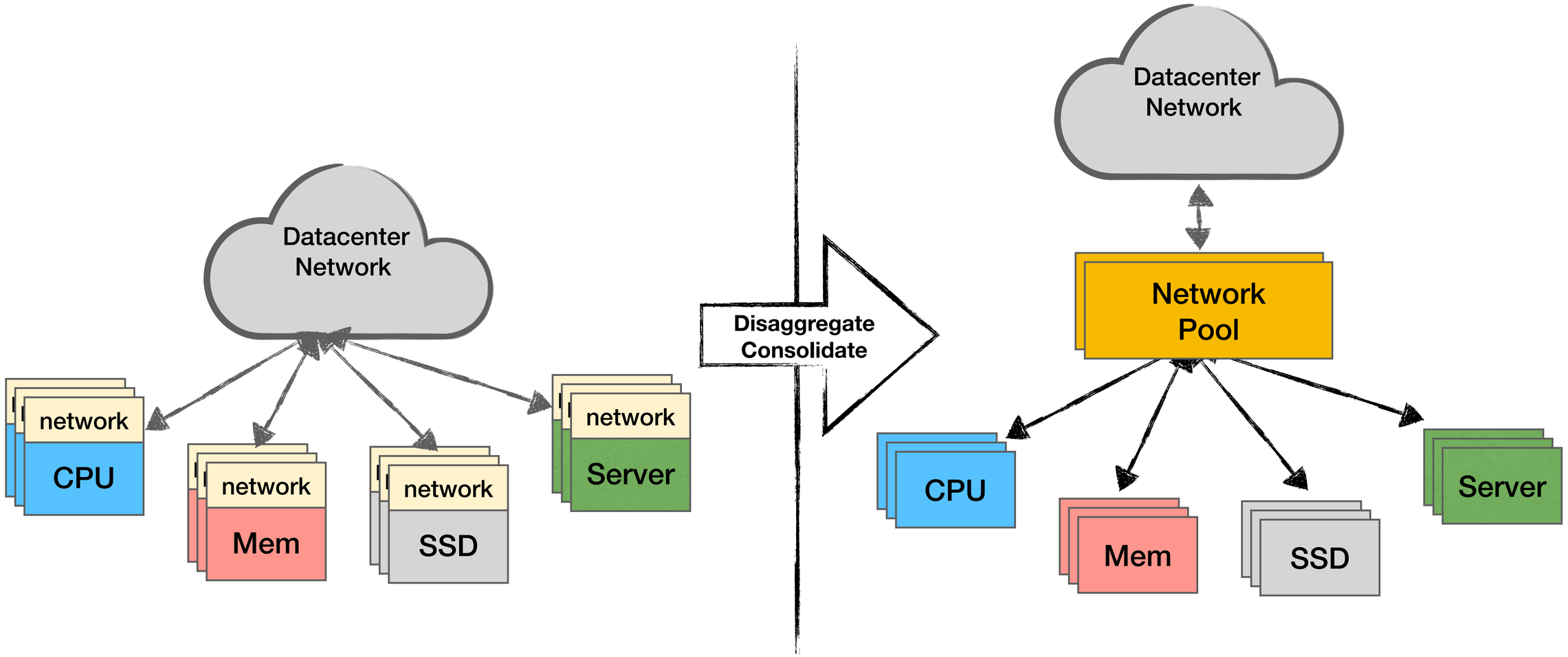
Everything above data link layer can potentially be disaggregated!


Datacenter
Network
network
CPU
network
Mem
network
SSD
network
Server


Datacenter
Network
Network
Pool
CPU
Mem
SSD
Server



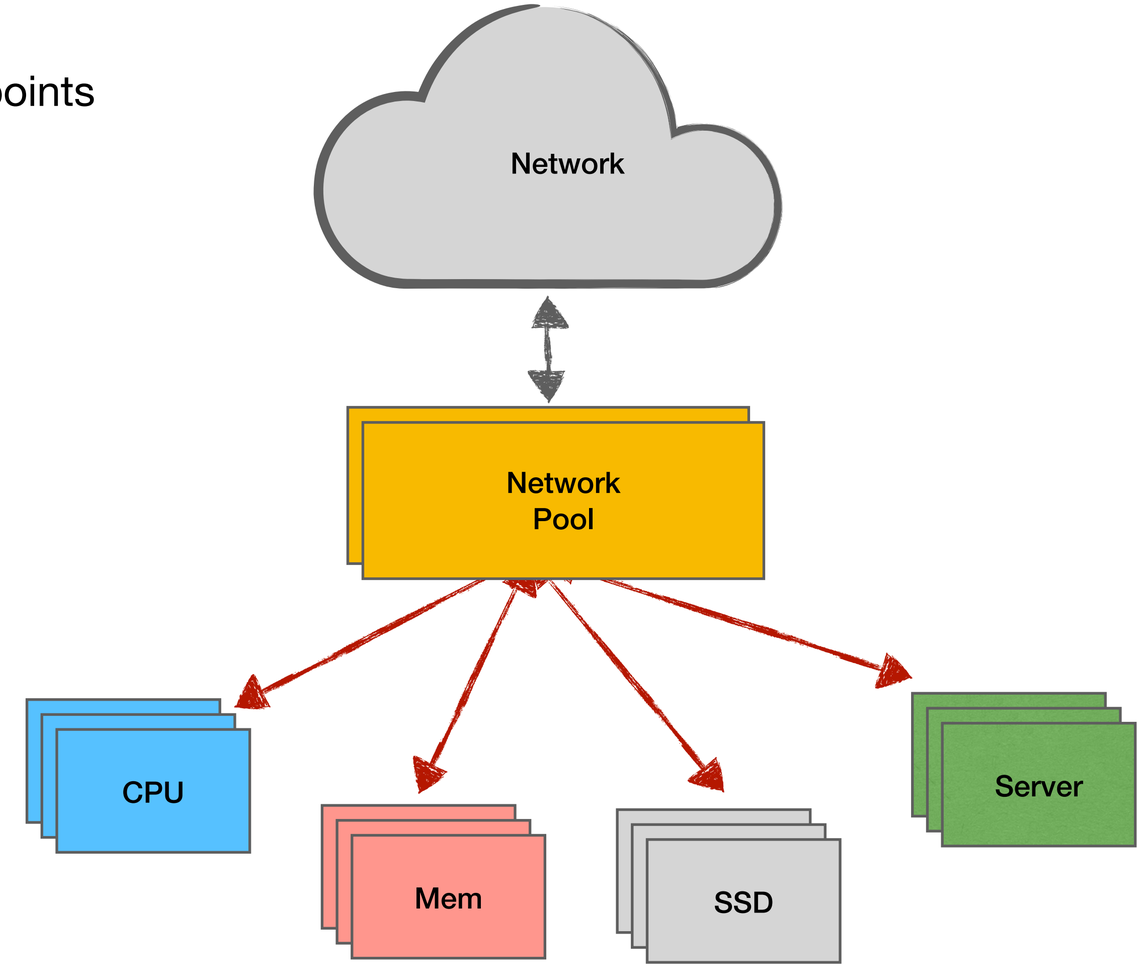
Disaggregate Network Modules from Endpoints and Consolidate Them Into a Network Resource Pool Providing Network-as-a-Service



Disaggregate Network Modules from Endpoints and Consolidate Them Into a Network Resource Pool Providing Network-as-a-Service

Network Disaggregation and Consolidation

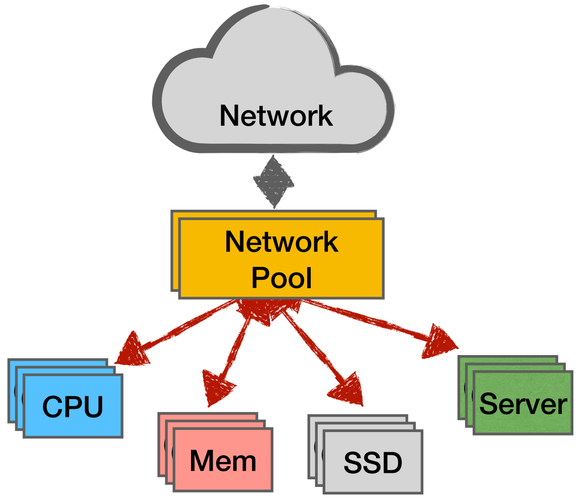
- **Definition**
  - Disaggregate **Network Tasks (NT)** from individual endpoints
  - Consolidate them into a **Network Resource Pool**
- **Network Tasks** Network Pool
  - Transports (e.g., TCP, RoCE)
  - Classical network functions (e.g., firewall, NAT)
  - Advanced in-network computation (e.g., KVS)
- **Link between endpoints and pool (→)**
  - A reliable data link (e.g., reliable Ethernet, PCIe)
  - Small buffer and simple logic



Should we disaggregate network?

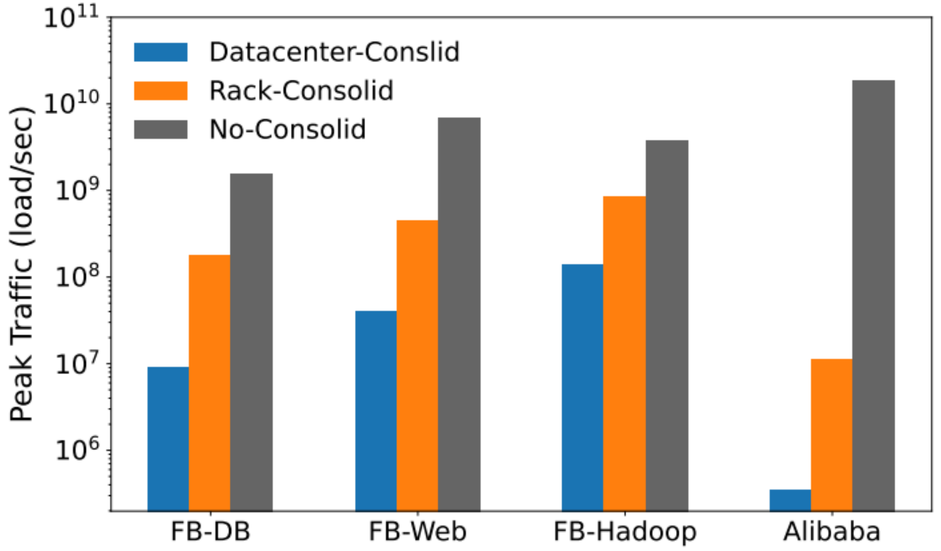
Benefits of Network Disaggregation

- **We discover three main benefits**
  - Avoids implementing ***net hw/sw*** at each device
  - Enable rack to host a large number of disaggregated devices
  - Provision for ***the peak of aggregated usage***



Provision for the peak resource usage

- **Sum-of-peak v.s. Peak-of-sum**
  - sum-of-peak: provision for each host's max usage
  - peak-of-sum: provision for the max of aggregated resource
- **Our finding**
  - Consolidation uses 2 orders of magnitude ***fewer*** resources than no consolidation

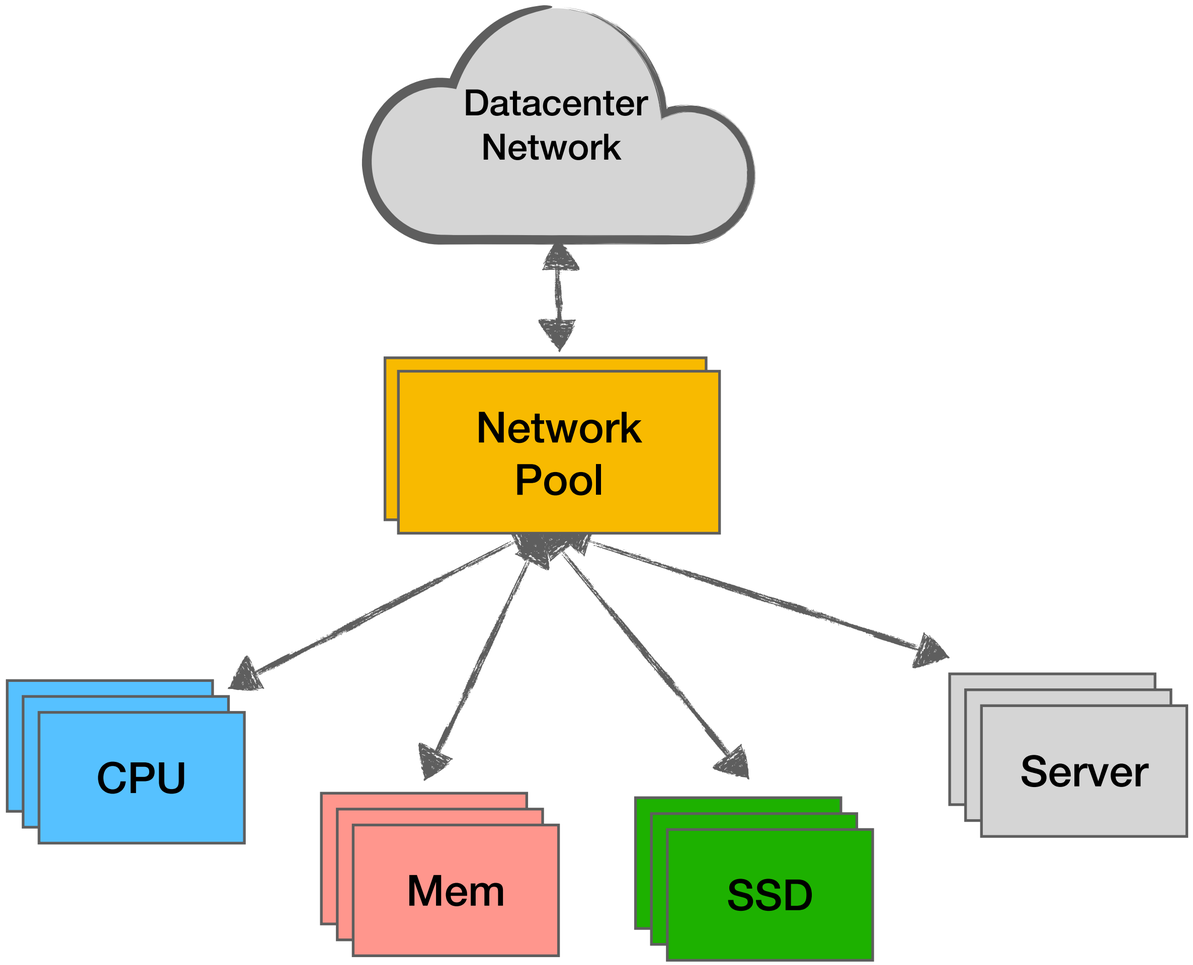


Facebook trace, SIGCOMM'15
Alibaba trace, released on GitHub early 2020

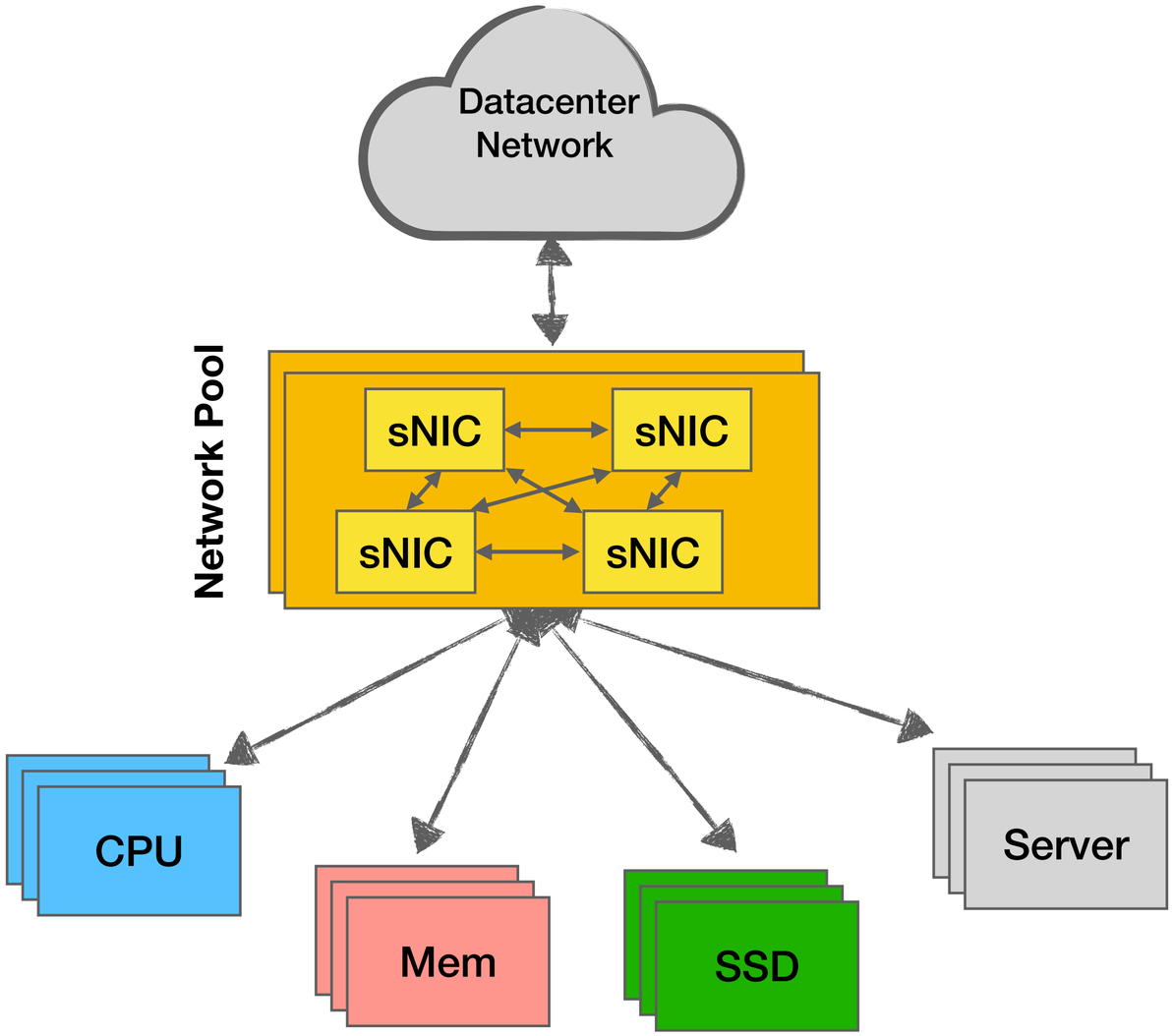
Outline

- **Network Disaggregation and Consolidation**
- **Alternative Solutions**
- **SuperNIC**
  - Overview
  - Board Architecture
  - Fast and Fair Packet Scheduling
  - Distributed SuperNIC
  - Case Studies and Results
- **Conclusion**

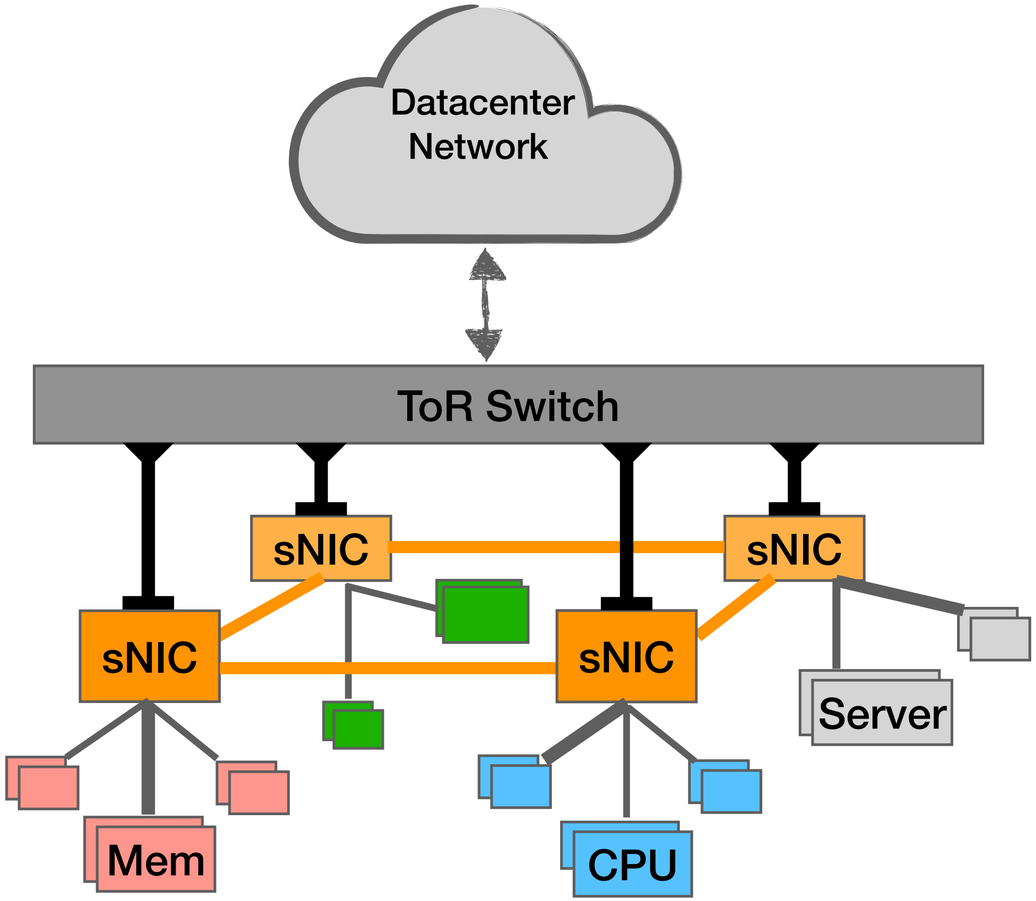
SuperNIC High-Level Architecture



SuperNIC High-Level Architecture



SuperNIC is an ideal way to realize the Network Pool for Disaggregated Datacenter



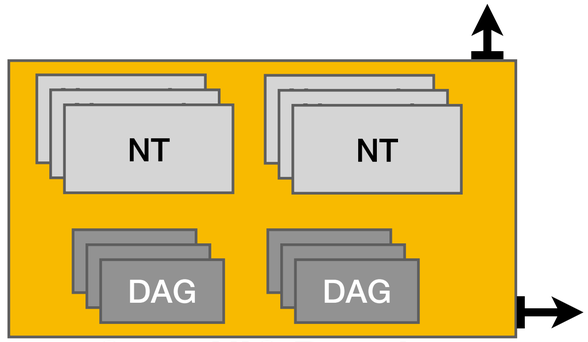
- ***SuperNIC is connected to ToR switch***
- ***SuperNICs are connected via ring or mesh***
- ***SuperNIC connects to a set of endpoints***

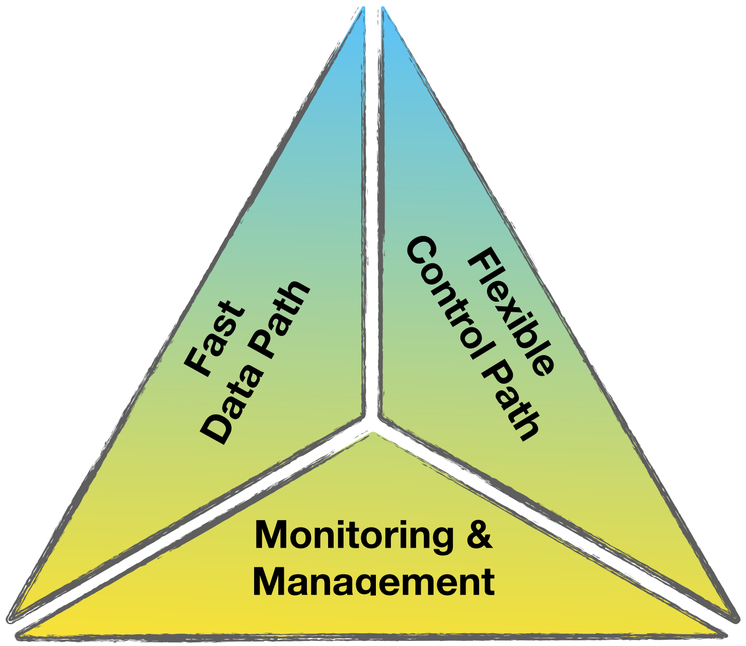
Outline

- **Network Disaggregation and Consolidation**
- **Alternative Solutions**
- **SuperNIC**
  - Overview
  - Board Architecture
  - Packet Scheduling
  - Distributed SuperNIC
  - Case Studies and Results
- **Conclusion**

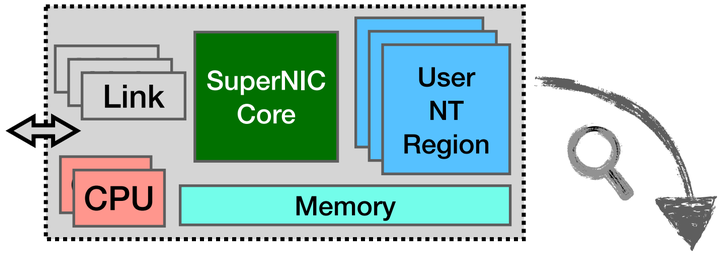
SuperNIC Board Architecture

- **Key Goals/Questions**
  - How to **efficiently and safely consolidate** tasks?
  - How to **ensure fairness** among tasks?
  - How to **design applications** for sNIC?
- **SuperNIC main features**
  - **Data Plane**: Handle packets at line rate with low latency
  - **Control Plane**: Multiplex multi-tenant network tasks
  - **Mgmt Plane**: Adapt to dynamic workload change





SuperNIC Board Architecture



Prototype on FPGA

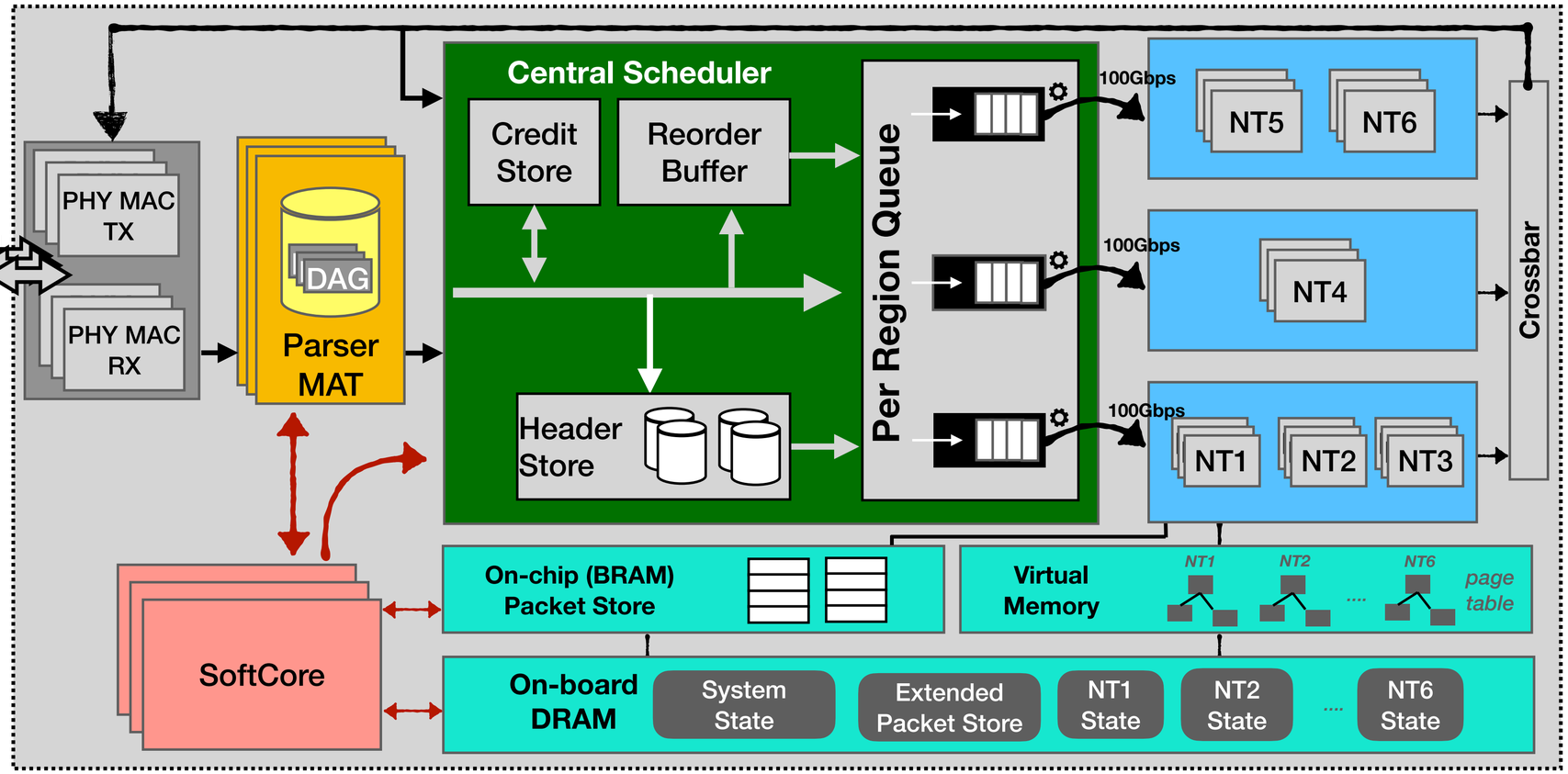
- **PHY & MAC (Ethernet)**
- **SoftCore for Monitoring and Management**
- **Parsers w/ User DAG (Core)**
- **Central Scheduler (Core)**
- **Virtual Memory Subsystem (Core)**
- **Network Task Regions run user code**

In Real Deployment

ASIC

FPGA or hardened as ASIC

FPGA



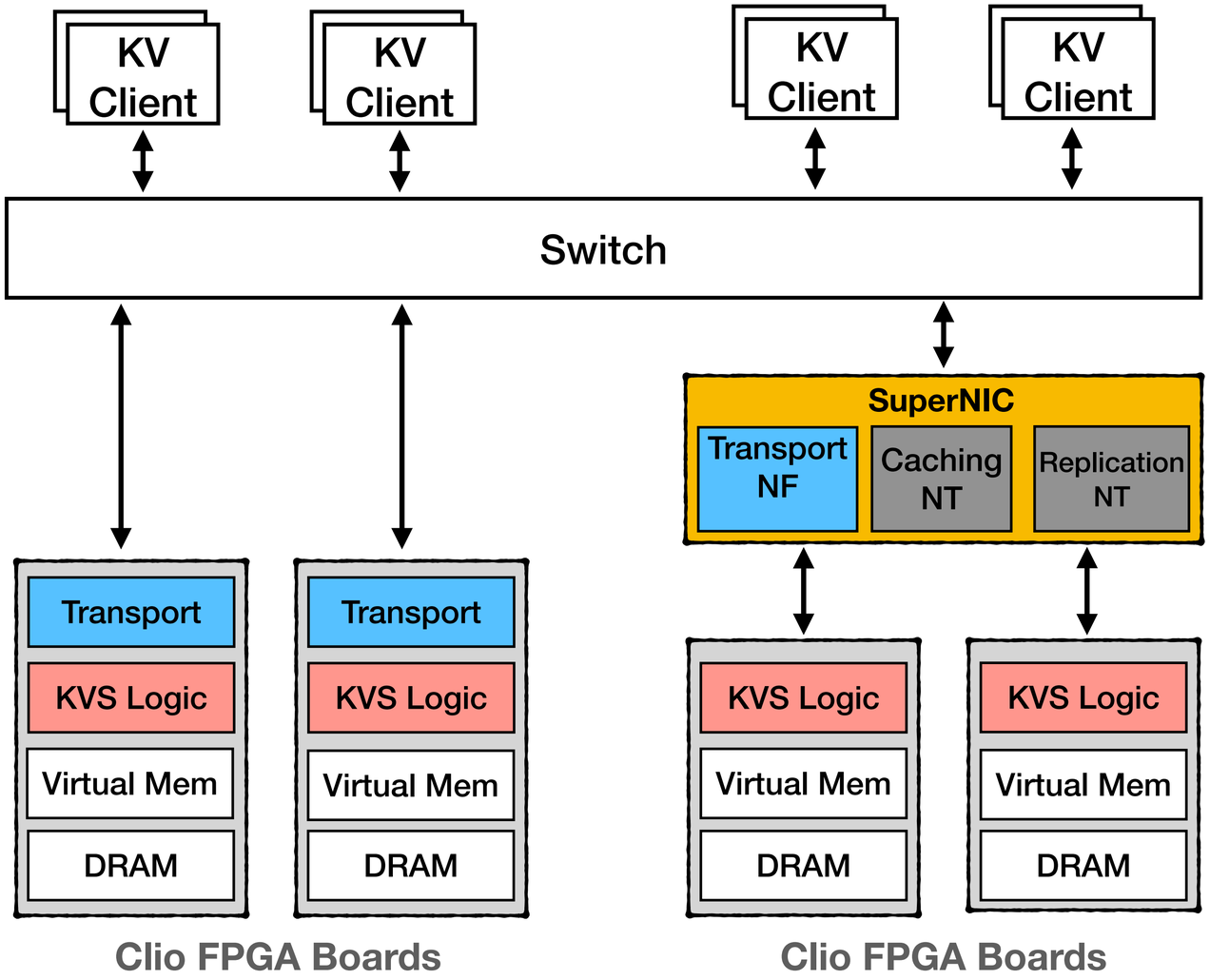
SuperNIC board design has *a fast data plane* with safe/fair sharing, *a control & mgmt plane* with great flexibility.

SuperNIC core uses 10% chip area
User Region occupies the rest 90%

Outline

- **Network Disaggregation and Consolidation**
- **Alternative Solutions**
- **SuperNIC**
  - Overview
  - Board Architecture
  - Fast and Fair Packet Scheduling
  - Distributed SuperNIC
  - Case Studies and Results
- Conclusion

Case Study on Disaggregated Devices



Clio is an FPGA-based disaggregated memory system

1. RDMA-alike Transport
2. Virtual Memory Subsystem
3. Key Value Store

We take 3 steps to integrate it with SuperNIC

1. Consolidate transport ==> Reduce CapEx/OpEx
2. Add Caching NT ==> Improve Latency
3. Add Replication NT ==> Improve distributed xact

Takeaway

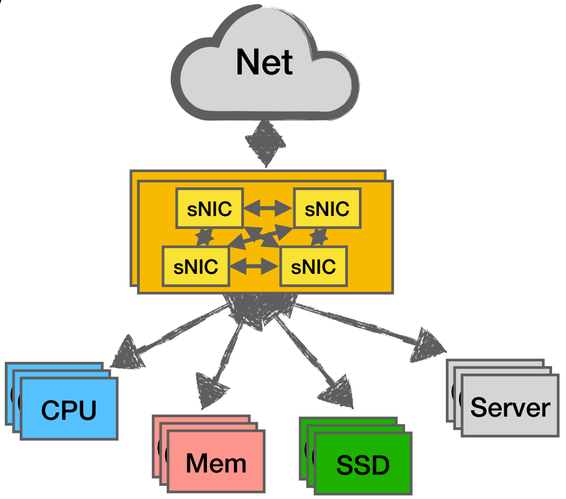
SuperNIC helps reduce CapEx and OpEx. It adds one extra hop, but helps building distributed applications!

Results

Module	Logic (LUT)	Memory (BRAM)
sNIC Core	4.36%	4.74%
Packet Store	0.91%	9.17%
PHY+MAC	0.72%	0.35%
DDR4Controller	1.57%	0.29%
MicroBlaze	0.25%	1.81%
Misc	1.52%	0.75%
Total	**9.33%**	**17.11%**

- **FPGA Utilization**
  - Our shell uses roughly **10% chip area**
  - Leave most of the on-board logic/memory to application logic
- **Cost of an extra hop**
  - sNIC core only has roughly **100-200 ns** latency cost per packet (~1us total)
  - All units are pipelined and able to achieve **100Gbps** line rate
- **Performance and Cost Saving**
  - Achieve **56% CapEx and OpEx saving** with **only 4% perf overhead** compared to a normal SmartNIC-based deployment model
- More results in the paper https://arxiv.org/pdf/2109.07744.pdf

SuperNIC Summary



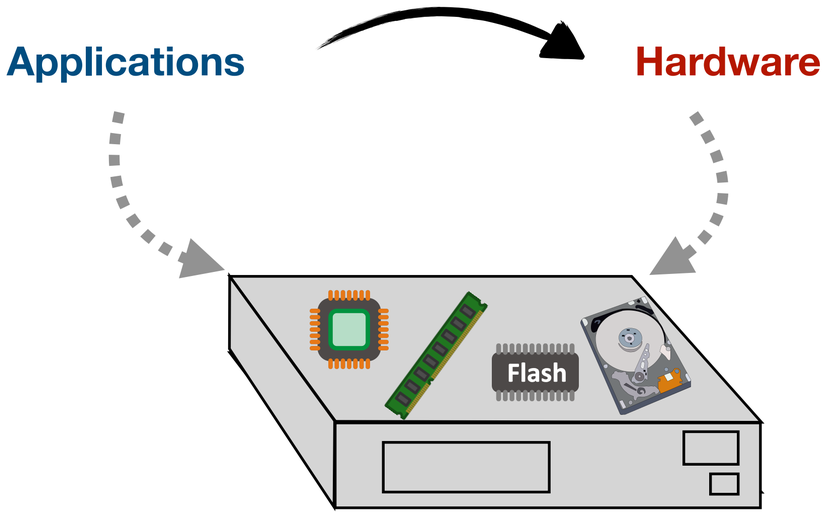
- **Network can be disaggregated and consolidated**
  - Everything above data link layer can potentially be disaggregated
  - Network pool provides Network-as-a-Service
  - SuperNIC is an ideal way to realize the pool
  - SuperNIC offers high-performance, isolated, and fair consolidation solutions

Outline

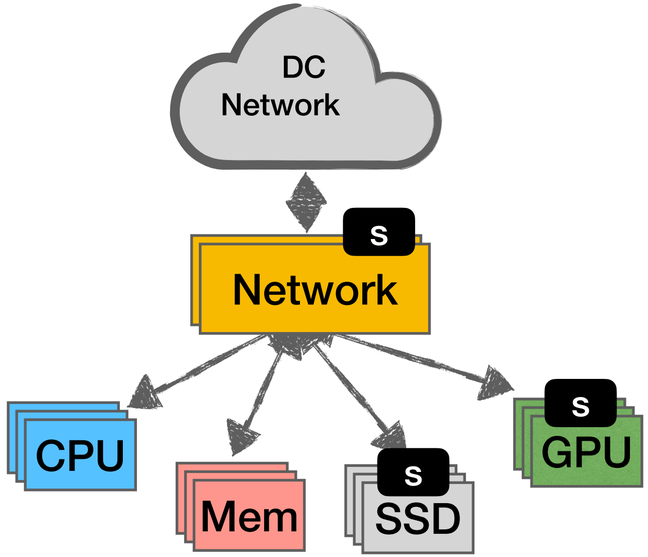
- Background on Resource Disaggregation
- Projects Conducted
  - Logical Disaggregation [Hotpot, SoCC'17]
  - Physical Disaggregation [LegoOS, OSDI'18]
  - Hybrid Disaggregation [Clio, ASPLOS'22]
  - Network Disaggregation [SuperNIC, arXiv'21, under submission]
- **Future Work**
- **Conclusion**

Future Work

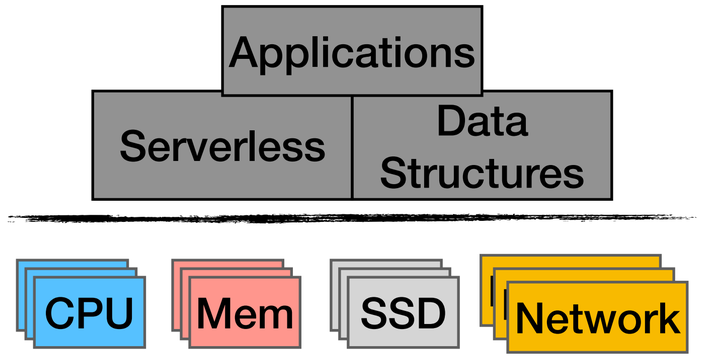
1 **Advanced Compilers and PL Frameworks**



2 **End-to-End Security and Confidential Computing**



3 **Disaggregation-Native Application and Data Structures**



Conclusion

- Disaggregation holds its promises on manageability, cost, and perf
- Disaggregation benefits “overlooked” systems/resources
- Hardware-software co-design benefits disaggregated devices
- Many open problems remained, call for more chefs!

Don’t adventures ever have an end?
I suppose not.
Someone else always has to carry on the story.
– The Fellowship of the Ring

Acknowledgements


Yiying Zhang


Arvind Krishnamurthy


Marcos K. Aguilera


Alex C. Snoeren


Geoffrey M. Voelker

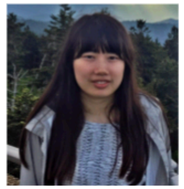

Stefan Savage


George Porter


George Papen


Mark Silberstein


Michael Swift


Ziqiao Zhou


Andrew Baumann


Shin-Yeh Tsai


Yutong Huang

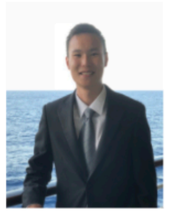

Yilun Chen


Sumukh Ravindra


Stanko Novakovic


Zhiyuan Guo


Xuhao Luo


William Lin


Zac Blanco


Alex Forencich


Moein Khazraee


Ryan Kosta


Anil Yelam


Stewart Grant


Sanidhya Kashyap

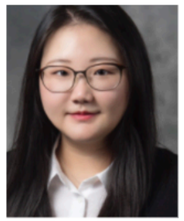

Soujanya Ponnapalli

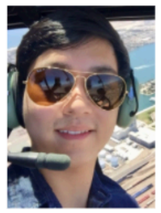

Qizhen Zhang

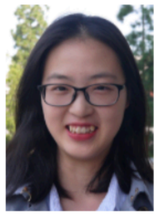

Heejin Park


Yanfang Le


Xiaoyi Huang

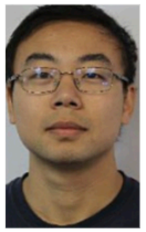

Anita Zhu


Lixiang Ao


Yi Xu

Haolan Liu


Zesen Zhang


Hanwen Yao


Jasmine Zhang

Thank you!


UCSDCSE
Computer Science and Engineering


WukLab


PURDUE
UNIVERSITY