Disaggregated Applications using *Uniservices*

Xinwen Wang
Robbert van Renesse

Cornell University

Road Map

Disaggregated Applications

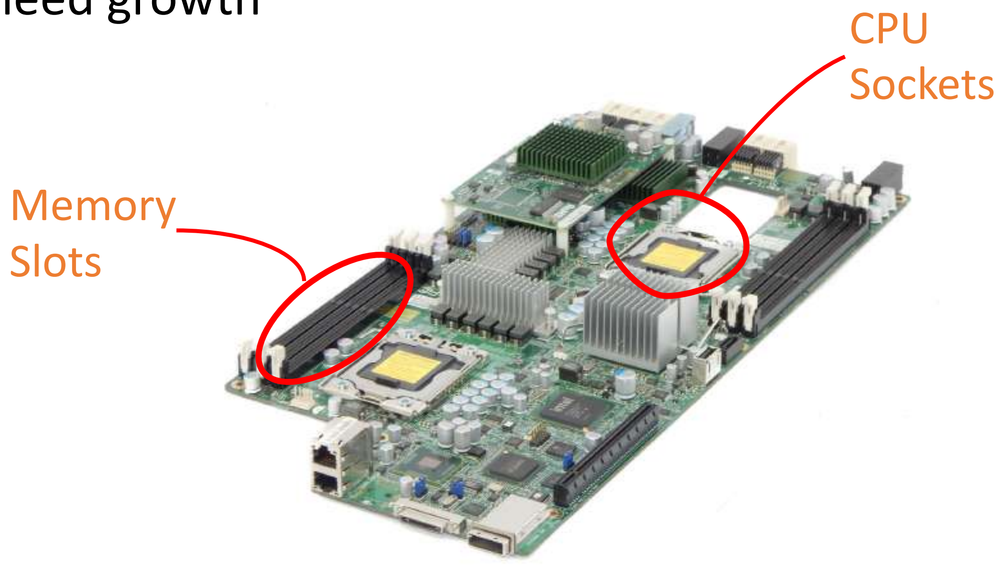
- ❑ Disaggregated Architecture
- ❑ Uniservices
- ❑ Evaluation
- ❑ Challenges

What is Resource Disaggregation?

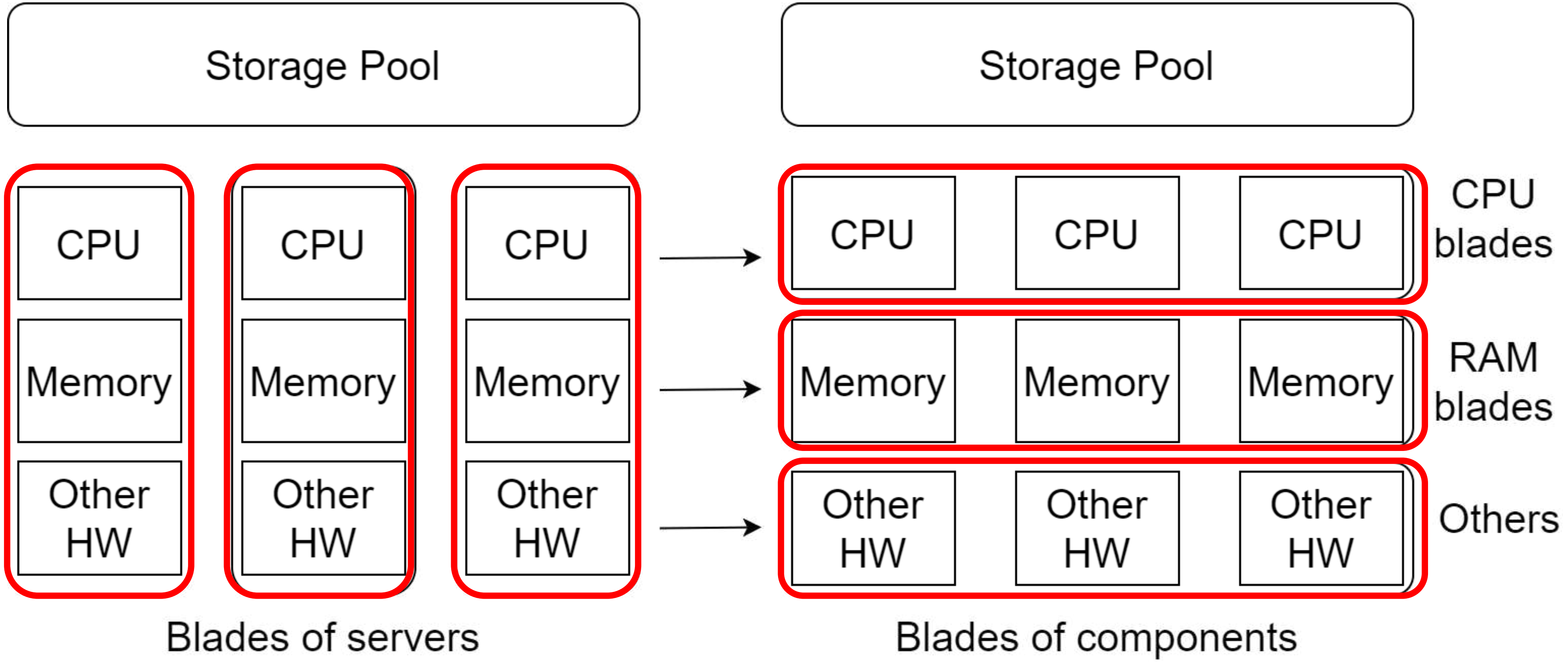
Before Resource Disaggregation

- Monolithic server or “blade”
  - The de facto deployment unit at current data centers
  - Contains large supply of different resource types in one server
    - CPUs, memory, disks, GPUs, NICs, FPGA, etc.
  - Scale out by adding more monolithic servers
    - even if only a subset of resource types need growth

  - Resource underutilization
  - Poor elasticity
  - Poor energy efficiency
  - Fate-sharing failure
  - Hard to adopt new hardware



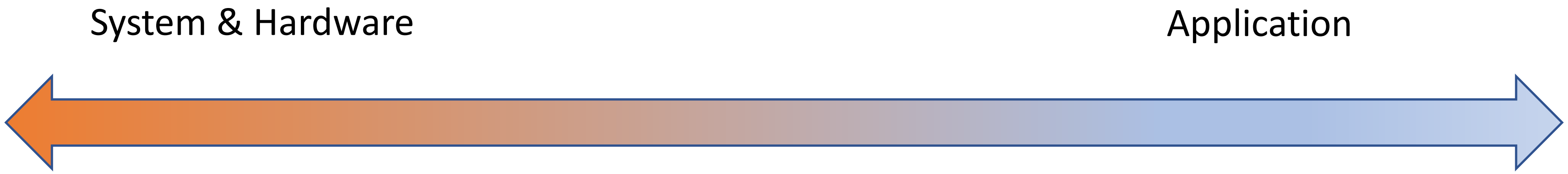
Transformation to Disaggregated Servers



Disaggregated Servers

- Disaggregated Servers
  - Contain mostly one type of resources: CPU servers, memory servers, …
  - Connected by fast interconnects such as RDMA
  - Scale out independently to other resource types
  - Heterogeneity-friendly
  - Better utilization, energy efficiency
- Monolithic server
  - The de facto deployment unit at current data centers
  - Contains different resource types in one server
  - Scale out by adding more monolithic servers
    - even if only a subset of components is needed.

Prior Work on Resource Disaggregation



System & Hardware

Application

LegoOS [OSDI’18]

Leap [ATC’20]

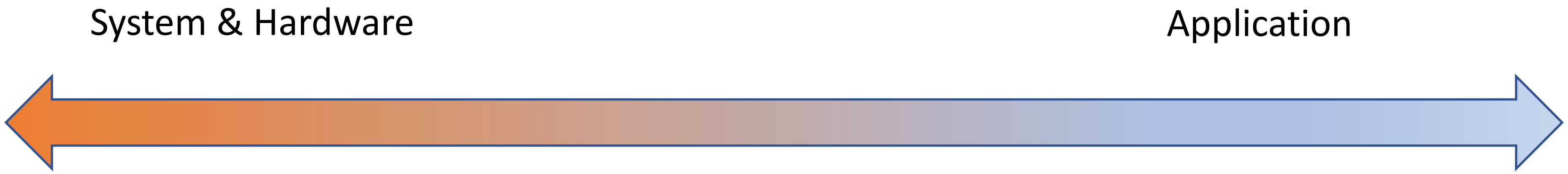
Infiniswap [NSDI’17]

Clover [ATC’20]

The Machine [HP]

etc.

Prior Work on Resource Disaggregation



System & Hardware

Application

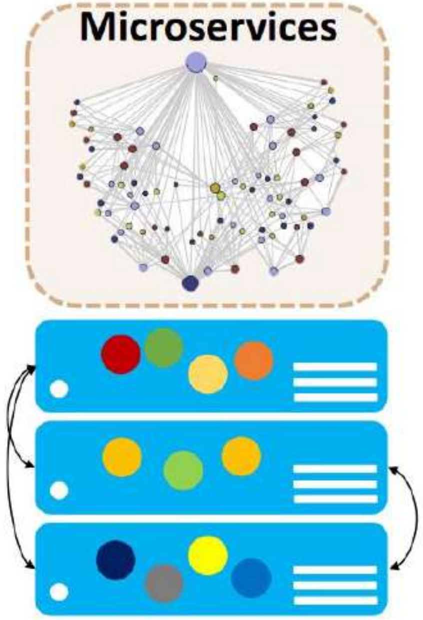
LegoOS [OSDI’18]
Leap [ATC’20]
Infiniswap [NSDI’17]
Clover [ATC’20]
The Machine [HP]
etc.

?

Today’s Cloud Applications

- Microservices and Serverless computing
  - Highly modularized
  - Separate state and functions

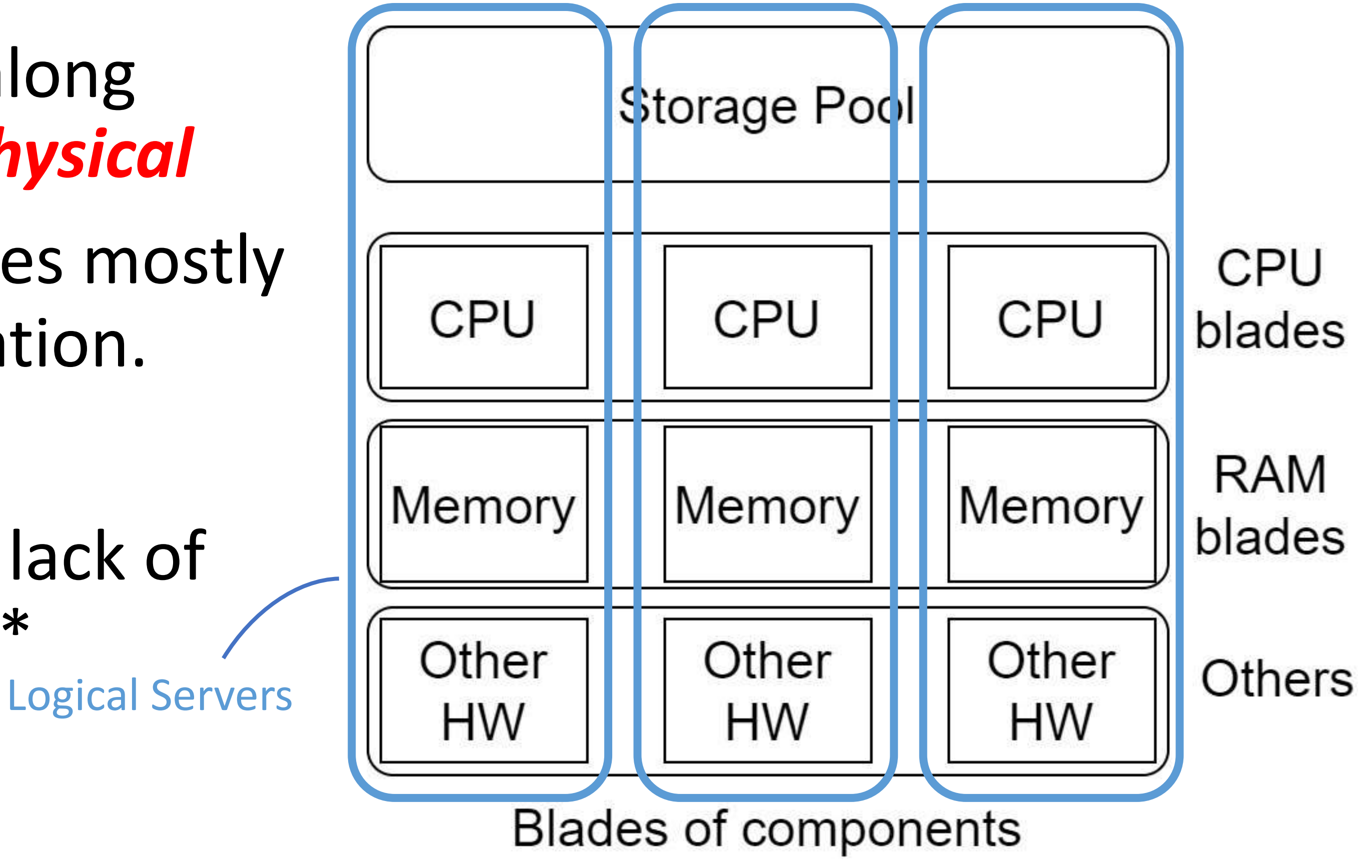
- Advantages
  - Pay only for what you need
  - Better elasticity
  - Relatively easy failure recovery



DeathStarBench [ASPLOS’19]

Current work: Disaggregated Logical Servers

- Applications are divided along ***logical*** boundaries, not ***physical***
- Current disaggregated OSes mostly **hide** hardware disaggregation.
- Complicates OS design
- Poor performance due to lack of leveraging disaggregation*



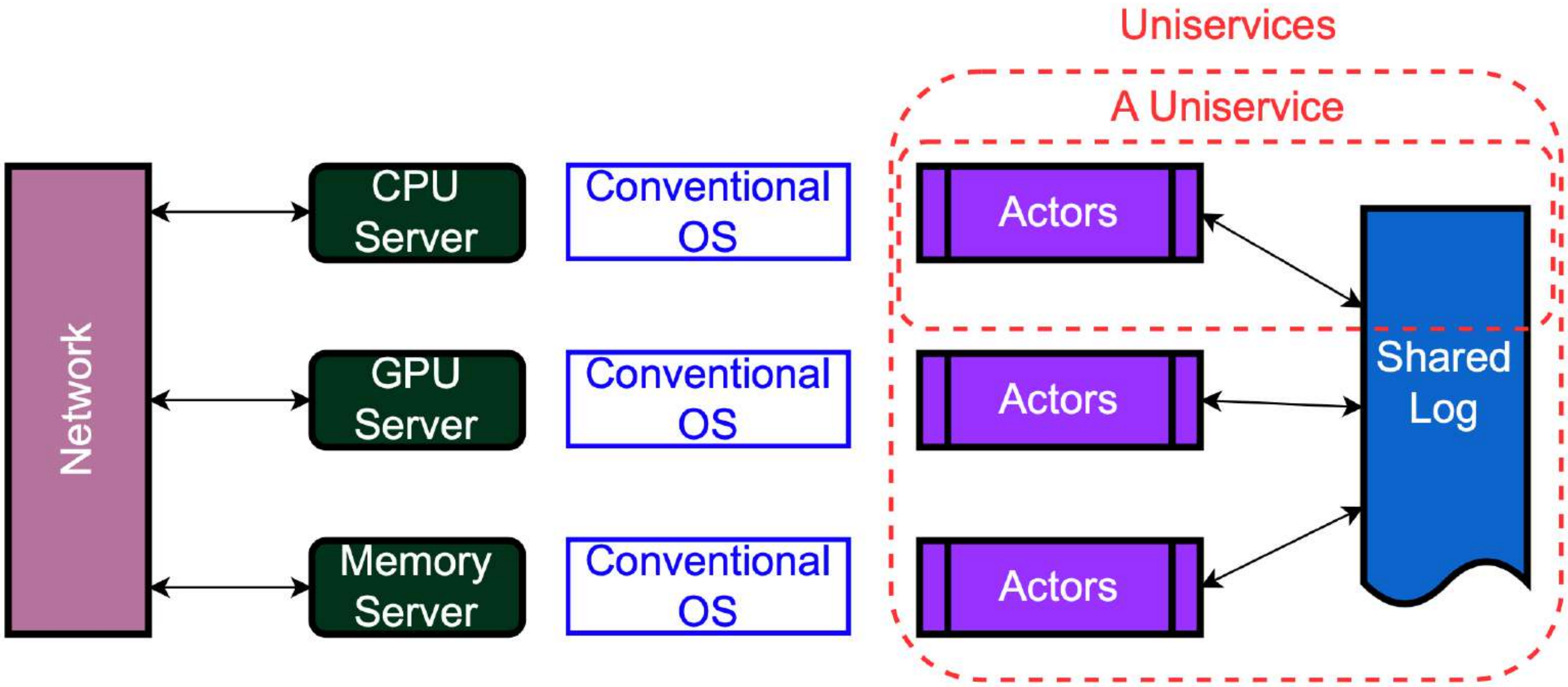
*"Understanding the Effect of Data Center Resource Disaggregation on Production DBMSs" [Zhang et. al, VLDB'20]

Exposing Disaggregation

Hardware resource disaggregation should be exposed to applications (using new abstractions) rather than hidden

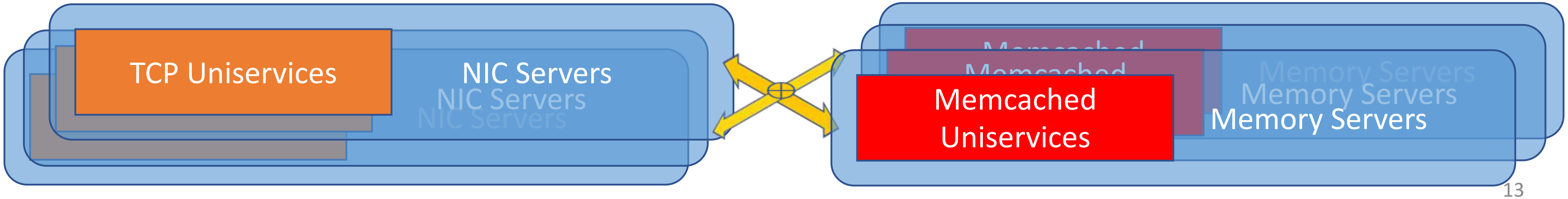
Lampson's Hints: Don't Hide Power

Leveraging Disaggregation: Overview



Uniservices

- Microservice specialized for a particular type of hardware resource
- Deployed and scaled out along ***physical*** boundaries
- Uses a fast communication backbone such as RDMA
- Running on conventional operating systems
- Reusable and shareable



Types of Uniservices

- Control-path Uniservices
  - Focusing on application logic
  - Examples include load balancing uniservices and web server uniservices

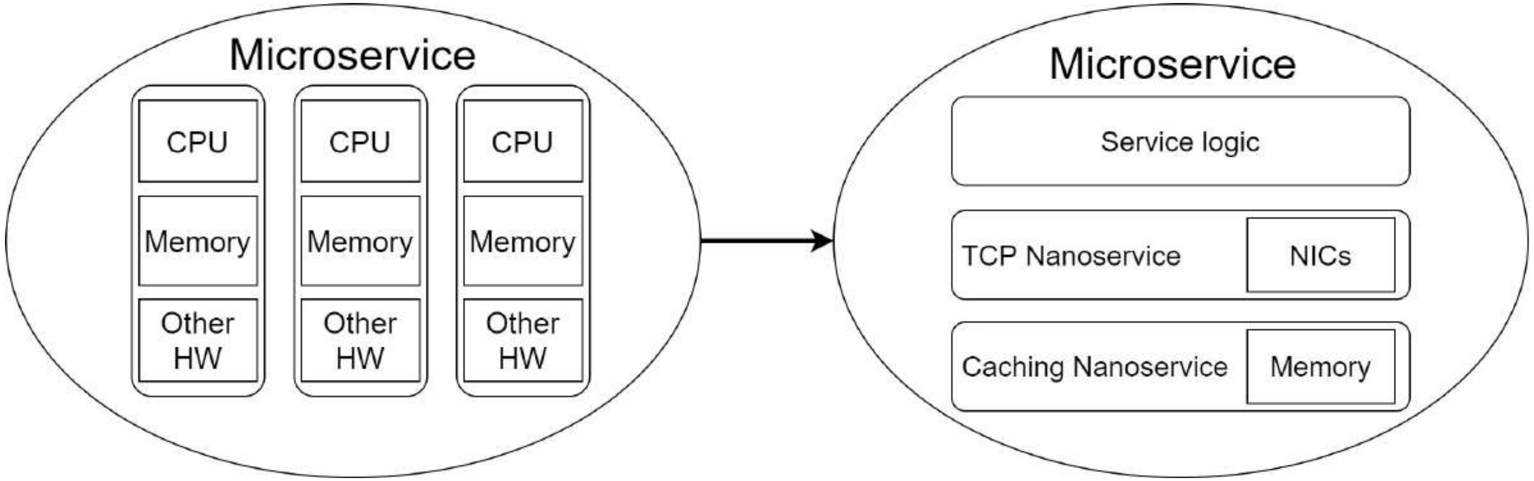
- Data-path Uniservices
  - Focusing on data processing and transfer
  - Examples include TCP uniservices and caching uniservices

Reusable Uniservices

- Cache uniservices
- Persistent key-value store uniservices
- TCP/TLS uniservices
- ML related uniservices

Comparing Microservices and Uniservices

	Microservices	Uniservices
Deployment unit	A (logical) server, such as a virtual machine or a container	A physical hardware platform with a specialized resource
Modularity level	Coarse-grained, combining various functionalities and using multiple resource types	Fine-grained, focusing on managing a single hardware resource
Network stack	Usually TCP/IP	RDMA or optical switching network



Actors

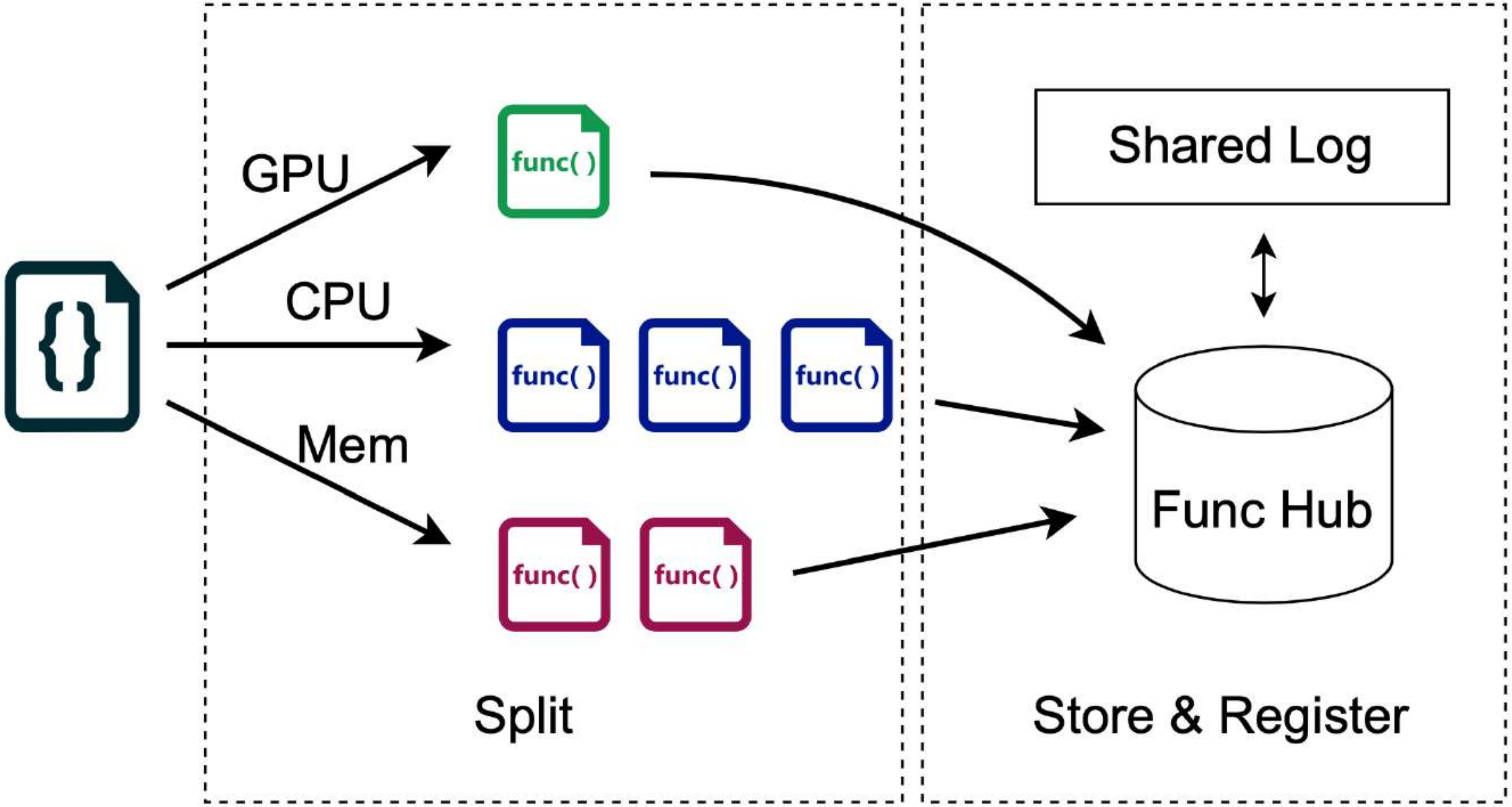
A Uniservice can comprise two types of actors:

- C-functions
  - stateless
  - “one shot”
  - non-blocking
  - predictable running time
  - easily re-startable after failure
- M-functions
  - stateful
  - long-running
  - can block waiting for external services
  - requires replication or re-computation of state

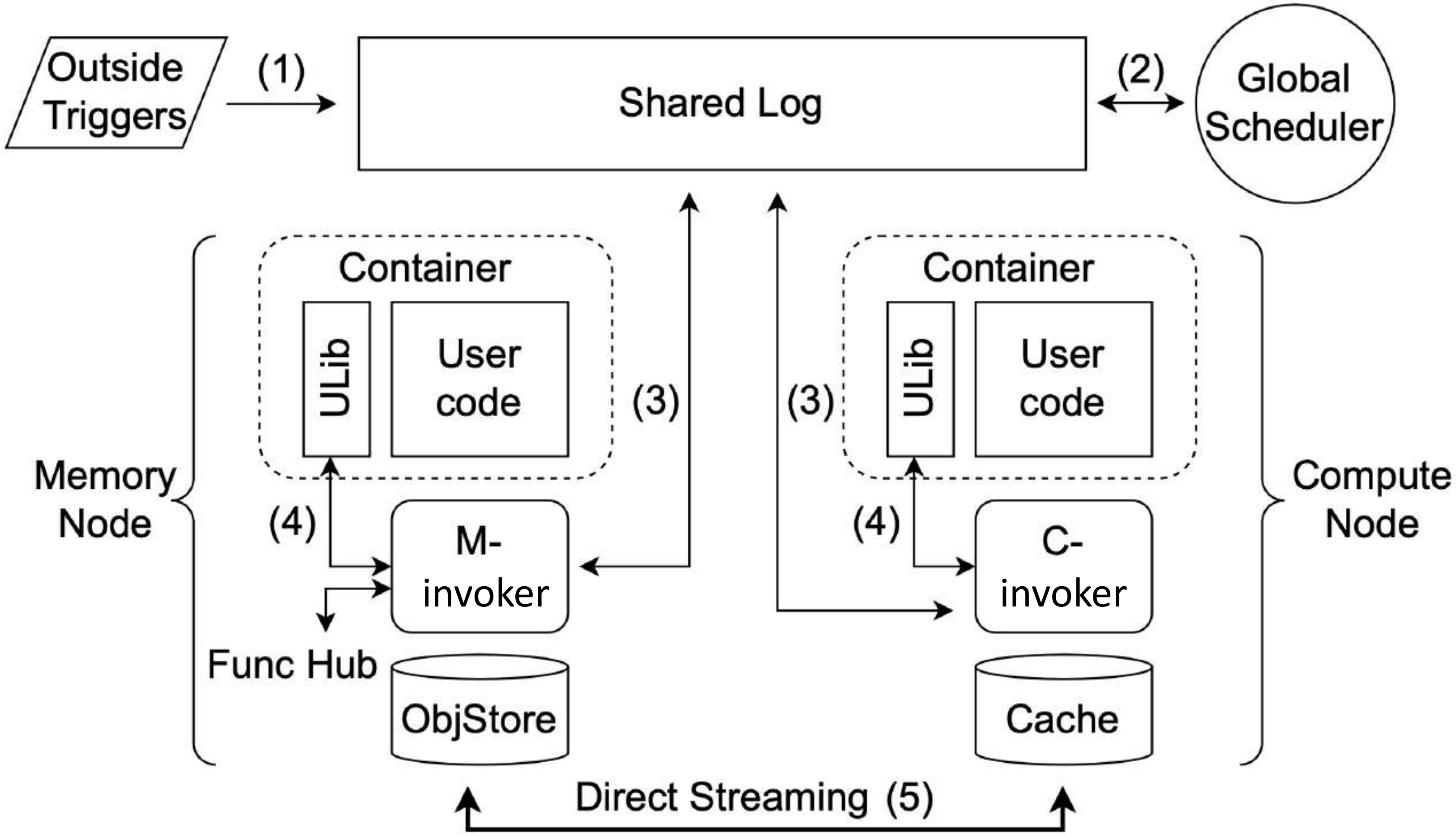
Object Stores

- Goals
  - provide input to C-functions and store output of C-functions
  - minimize copying through flexible object references
  - optimize cache locality
  - enable prefetching / warm-up
- Object Types
  - Streamable
  - Random Access
- Object Modes
  - Read-only
  - Copy-on-Write
  - Mutable

Application Decomposition

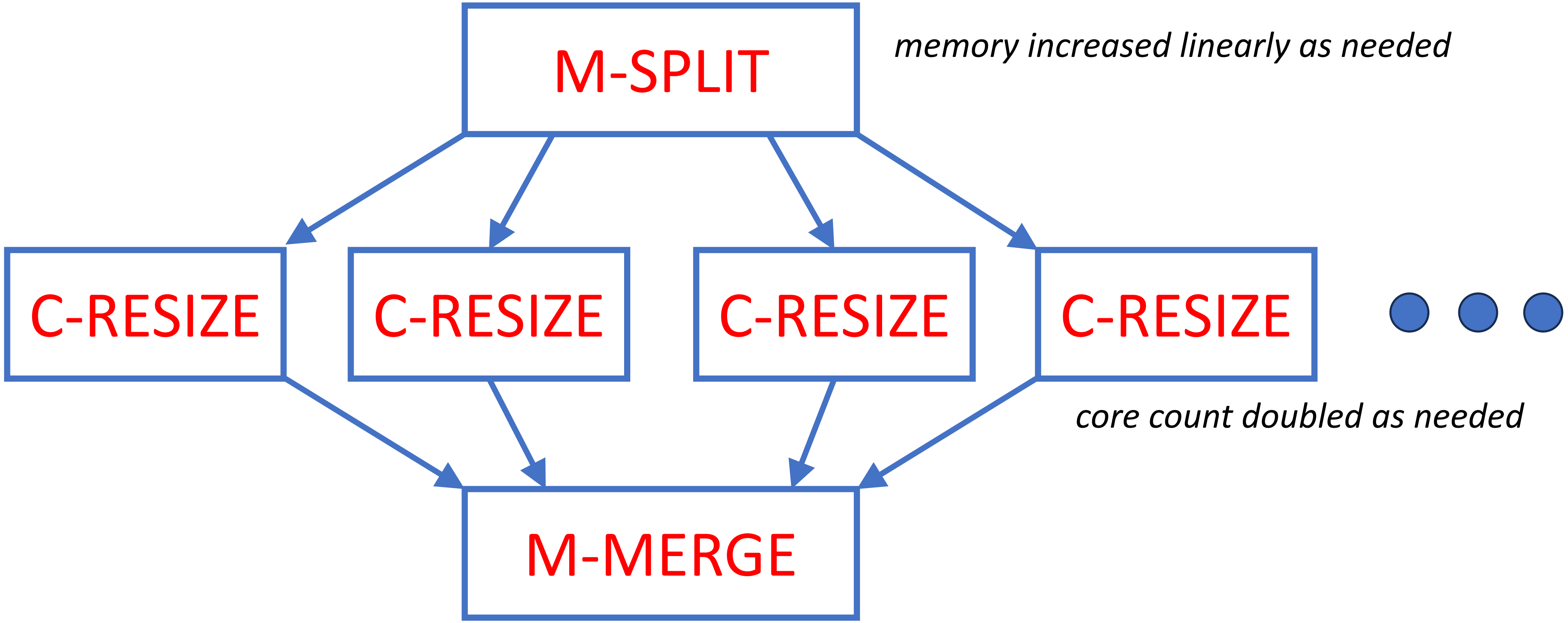


Uniservices Architecture



1. insert triggers into log
2. scheduler assigns functions to *invokers*
3. invokers load code from hubs
4. invokers launch functions
5. data movement over RDMA

Use Case: Video rescaling


M-SPLIT

memory increased linearly as needed

C-RESIZE

C-RESIZE

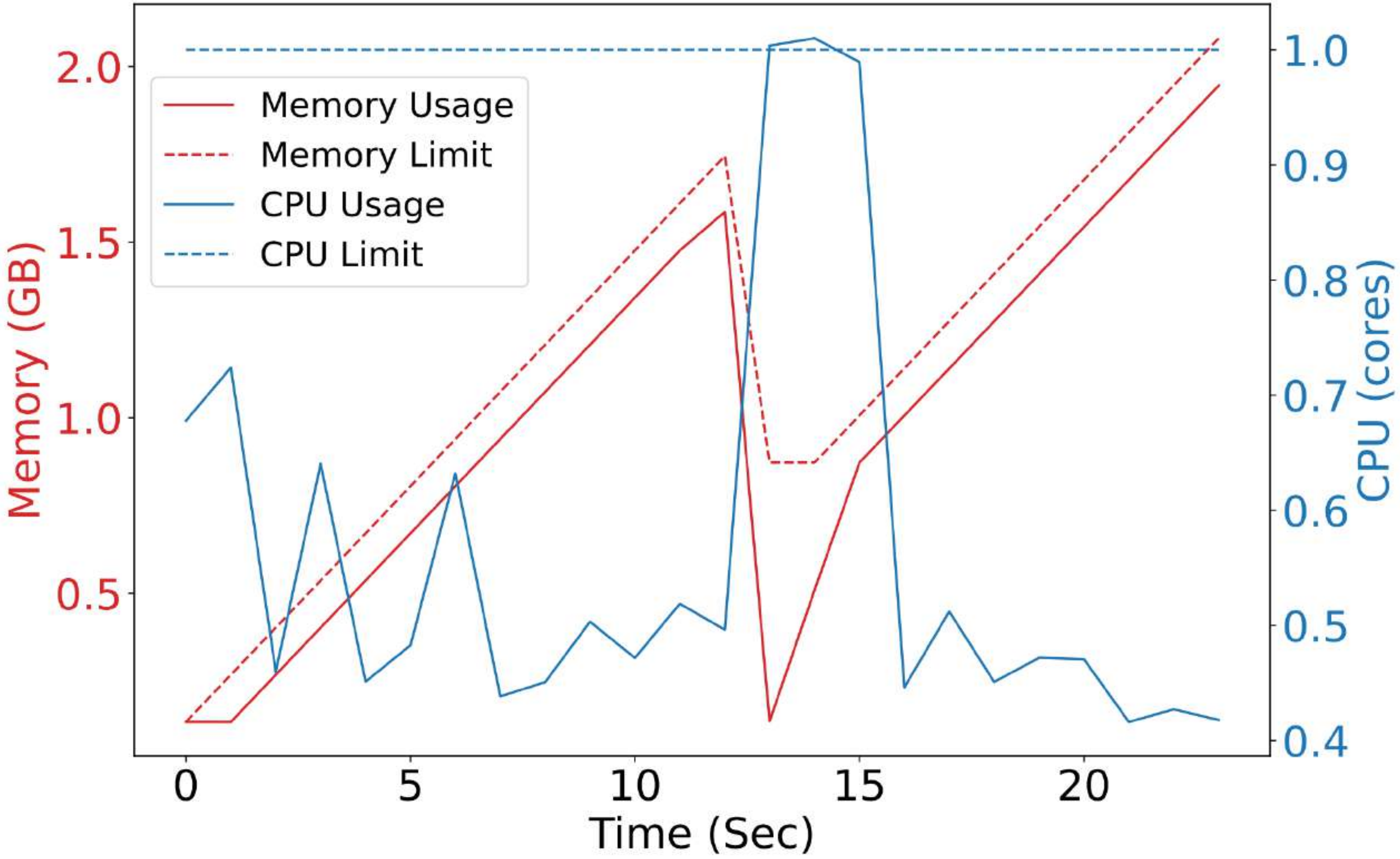
C-RESIZE

C-RESIZE

core count doubled as needed

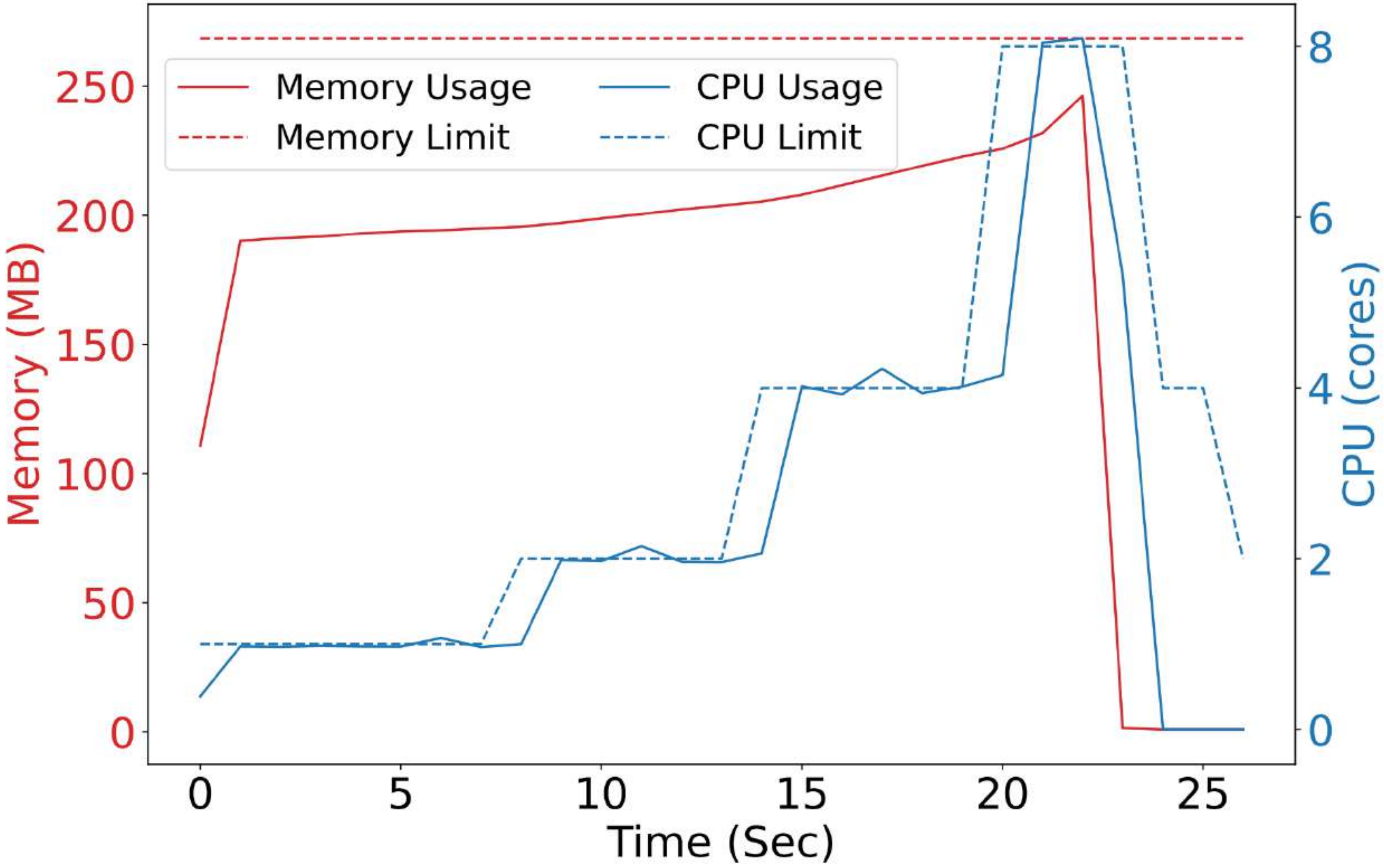
M-MERGE

M-SPLIT scaling



- single core enough
- AWS requires 2 cores to obtain sufficient memory

C-RESIZE scaling



- Mostly compute bound
- AWS would require allocating **10G** of memory (at 6 cores)

Challenges

- Converting existing applications to disaggregated applications with uniservices requires effort
  - Solution: providing basic reusable uniservices as building blocks
- Communication among uniservices requires fast interconnects
  - Leverage disaggregated memory to reduce memory copying
  - Support for sending data from remote memory to sockets directly

Challenges

Right abstractions are needed for balancing portability of uniservices and performance

Summary

- Hardware resource disaggregation should be exposed to applications rather than hidden from them

- On disaggregated architecture, applications themselves should be disaggregated as well
  - Decompose applications into uniservices for scaling along physical boundaries