Lego: A Distributed, Decomposed OS PURDUE for Resource Disaggregation WukLab

Yizhou Shan, Yilun Chen, Yutong Huang, Sumukh Hallymysore, Yiying Zhang

underutilization

in datacenters

WukLab.io

When hardware is disaggregated, the OS should be also!

Resource Disaggregation Breaking monolithic servers into networkattached, independent hardware components 1. Powerful network High Bandwidth: 200Gbps Low Latency: <1usec High Radix CPU TPU 4. Applications' Processor N Processor 1 dynamic resource Fast NVM Memory requirements Controller Controller / Network Memory N Memory 1 5. Resource

SSD

Controller

Storage N

3. Heterogeneous hardware

2. Powerful

controller

Efficient resource utilization

HDD

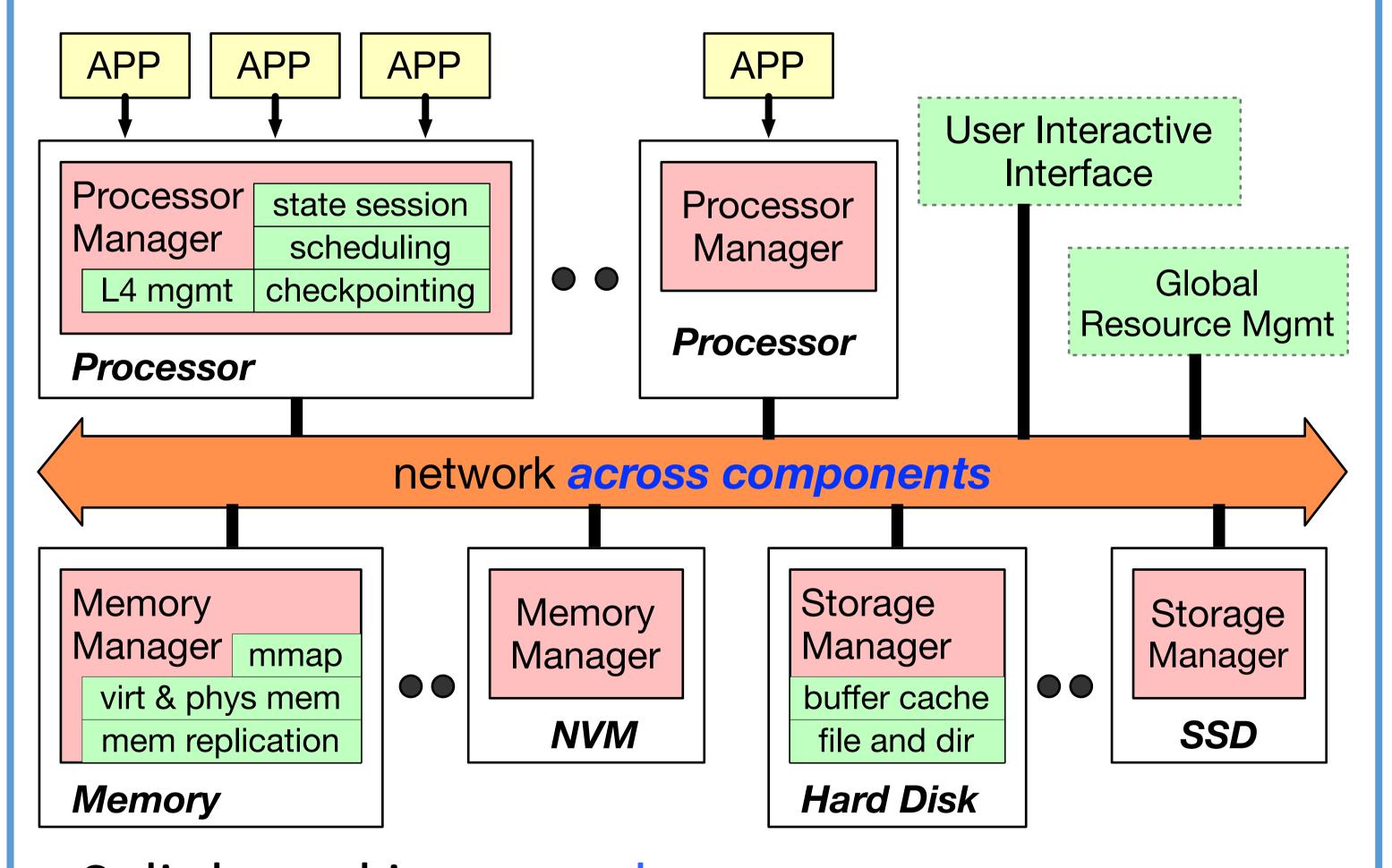
Controller

Storage 1

- Finer-grained failure domain
- Easy to add/remove/change hardware

Lego: a Disaggregated OS

How to build an OS for resource disaggregation?



- Split kernel into stateless managers
- Handle failures transparently
- No memory sharing across processors
- Built from scratch; ongoing, open-source project
- Supports x86-64 and unmodified Linux binaries

Unique Challenges 1. How to ensure good performance CPU No existing OS with network delay? solves the unique Fast Memory challenges of Controller Network IVICITIOTY I IVICITION IN resource 2. How to meet SSD HDD hardware constraints? Controller Storage N disaggregation! Controller 3. How to handle Storage 1 component failure? 4. How to manage resources and run applications? **Problems with Existing OSes** APP APP Distributed System OS OS OS node OS node CPU CPU ARM core x86 core NIC NIC mem mem NIC Memory Disk Disk Server Server Server Disk network across servers Multikernel: manages cores Monolithic/micro- kernel: separately, not other components built for monolithic server

Hardware Design

Distributed OS: manages distributed monolithic servers,

but not distributed hardware components

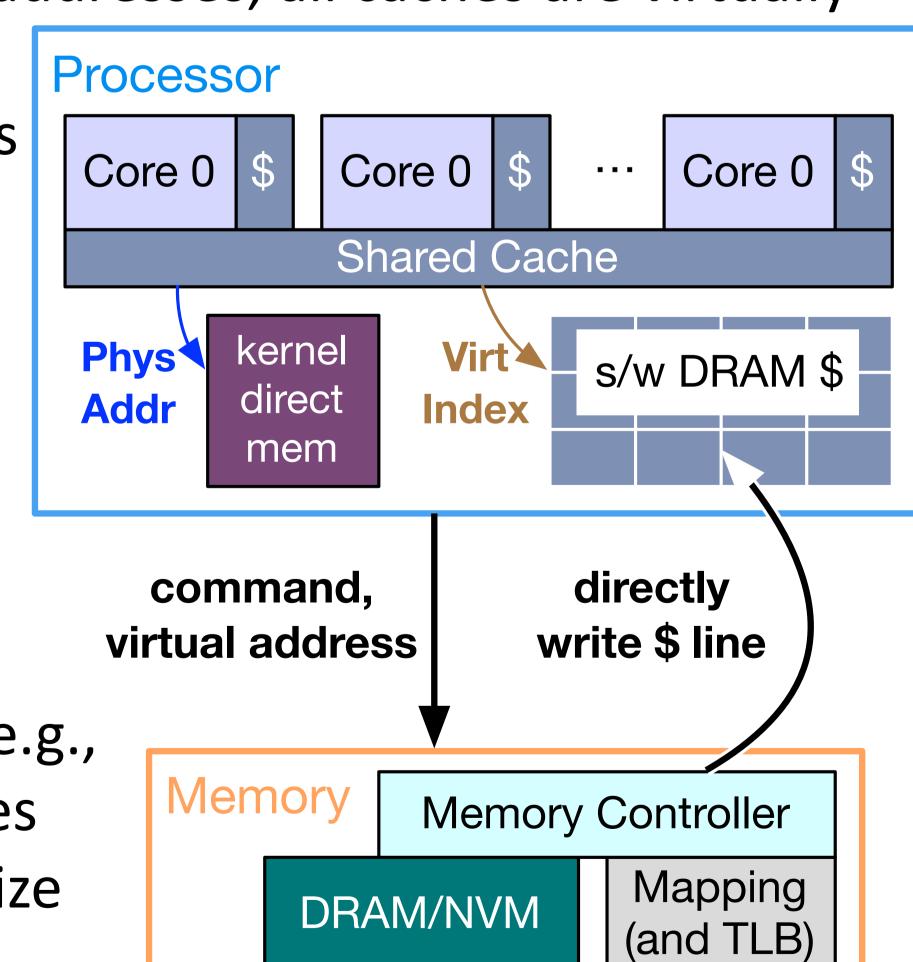
How to cleanly separate hardware components while ensuring good performance?

Processor

- Only knows virtual addresses, all caches are virtually indexed and tagged
- Small local DRAM as software-managed
- last-level cache Small physically-
- addressed memory for kernel usages

Memory/Storage

- Device controllers (e.g., ASIC) run OS services
- Manage and virtualize DRAM/NVM/HDD



Failure Handling

Handle Processor Failure

- Per-process coordinated checkpointing
- Stateless managers \rightarrow efficient checkpointing
- Minimal dependencies across processes

Handle Memory Failure

- Replicate memory at checkpointing time
- Selective dirty memory replication \rightarrow < 2X space
- Use replicated memory for better parallelism
- Lazily compact replicas and checkpoint to storage