LegoOS
A Disseminated Distributed OS for Hardware Resource Disaggregation

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

OS / Hypervisor
Problems?
Resource Utilization

Heterogeneity

Elasticity

Fault Tolerance

Server 1
Available Space

Server 2
Required Space

Job 2

TPU
FPGA

Hard to add, remove, or reconfigure devices in a servers after deployment.

No extra PCIe slots
How to improve resource utilization, elasticity, heterogeneity, and fault tolerance?

Go beyond physical server boundary!
Hardware Resource Disaggregation:

Breaking monolithic servers into network-attached, independent hardware components
Elasticity
Resource Utilization
Fault Tolerance
Heterogeneity
Application
Hardware
Network
Flash
RAM
CPU
Why Possible Now?

- Network is faster
  - InfiniBand (200Gbps, 600ns)
  - Optical Fabric (400Gbps, 100ns)
- More processing power at device
  - SmartNIC, SmartSSD, PIM
- Network interface closer to device
  - Omni-Path, Innova-2
Outline

• Hardware Resource Disaggregation

• Kernel Architectures for Resource Disaggregation

• LegoOS Design and Implementation
  • Abstraction
  • Design Principles
  • Implementation and Emulation

• Conclusion
Can Existing Kernels Fit?

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

Multikernel
(e.g., Barrelish, Helios, fos)

msg passing over local bus

Shared Main Memory

Monolithic Server
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 Management
- Virtual Memory System
- 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
Outline

- Hardware Resource Disaggregation
- Kernel Architectures for Resource Disaggregation
- LegoOS Design and Implementation
  - Abstraction
  - Design Principles
  - Implementation and Emulation
- Conclusion
How Should LegoOS Appear to Users?

As a set of hardware devices? As a giant machine?

- Our answer: as a set of virtual Nodes (*vNodes*)
  - Similar semantics to virtual machines
  - Unique vID, vIP, storage mount point
  - Can run on multiple processor, memory, and storage components
One vNode can run multiple hardware components
One hardware component can run multiple vNodes
Abstraction

• Appear as vNodes to users

• Linux ABI compatible
  • Support *unmodified* Linux system call interface (common ones)
  • A level of *indirection* to translate Linux interface to LegoOS interface
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

Virtual Memory System

Processor

CPU $ CPU $

Last-Level Cache

Network

TLB MMU

DRAM PT

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
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
Distributed Resource Management

1. Coarse-grain allocation
2. Load-balancing
3. Failure handling

Global Process Manager (GPM)
Global Memory Manager (GMM)
Global Storage Manager (GSM)

network messaging across non-coherent components

Process Monitor (CPU)
GPU Monitor (GPU)
Global Resource Mgmt
Memory Monitor
NVM Monitor
HDD Monitor
SSD Monitor

Memory
NVM
Hard Disk
SSD
Distributed Memory Management

- **GMM assigns vRegions to mem components**
  - On virtual mem alloc syscalls (e.g., mmap)
  - Make decisions based on global loads
- **Owner of a vRegion**
  - Fine-grained virtual memory allocation
  - **On-demand** physical memory allocation
  - Handle memory accesses

fix-sized, **coarse-grain** virtual region (vRegion) (e.g., 1GB)

GMM

Processor

Memory (M1)

Memory (M2)

mmap 1.5GB
write 1GB

max

vRegion 1 vRegion 2 vRegion 3

User Virtual Address Space

0

Used Used

(Physical Memory)

Memory (M1)

Used Used

(Physical Memory)

Memory (M2)
Implementation and Emulation

- **Status**
  - 206K SLOC, runs on x86-64, **113** common Linux syscalls

- **Processor**
  - Reserve DRAM as ExCache (4KB page as cache line)
  - h/w only on hit path, s/w managed miss path

- **Memory**
  - Limit number of cores, kernel-space only

- **Storage/Global Resource Monitors**
  - Implemented as kernel modules on Linux

- **Network**
  - RDMA RPC stack based on LITE [SOSP’17]
Performance Evaluation

- Unmodified TensorFlow, running CIFAR-10
  - Working set: 0.9G
  - 4 threads
- Systems in comparison
  - Baseline: Linux with unlimited memory
- InfiniSwap [NSDI’17]

LegoOS Config: 1P, 1M, 1S

Only 1.3x to 1.7x slowdown when disaggregating devices with LegoOS
To gain better resource packing, elasticity, and fault tolerance!
Conclusion

• Hardware resource disaggregation is promising for future datacenters

• The splitkernel architecture and LegoOS demonstrate the feasibility of resource disaggregation

• Great potentials, but many unsolved challenges!
Thank you!
Questions?

Open source @ LegoOS.io

Poster Tonight. Number 11.

@WukLab.io

PURDUE UNIVERSITY