Lego: A Distributed, Decomposed OS for Resource Disaggregation

Yizhou Shan†∗, Yilun Chen∗, Yutong Huang∗, Sumukh Hallymysore∗, Yiying Zhang
Purdue University, ∗ are students, † will present the poster

Recently, there is an emerging interest in datacenter *resource disaggregation* [1, 2, 3], an architecture that breaks *monolithic* servers into independent hardware components connected through network. OSes built for monolithic computers can not handle the distributed nature of disaggregated hardware components. Datacenter distributed systems are built for managing clusters of monolithic computers, not individual hardware components. When traditional OS operations spread across hardware components over the network, these distributed systems fall short. Clearly, we need a new operating system for the disaggregated architecture.

We propose the concept of *decomposed operating system* for the disaggregated datacenter architecture. The basic idea is simple: *When hardware is disaggregated, the operating system should be also*. There are three main challenges in building a disaggregated OS:

1) How to cleanly separate OS services and map them to hardware components? Traditional OS services are tightly coupled, which assume all hardware resources are accessible within the same machine. Resource disaggregation calls for a clean separation of OS services; only when OS services are cleanly separated can disaggregated components be flexibly deployed and reconfigured. Furthermore, different hardware components have different constraints. For example, processor component will only have limited memory, while memory component will only have limited processing power to handle network and virtual memory requests.

2) How to ensure failure independence? We envision the scale of disaggregated architecture to have at least thousands of components. With this scale, failure is inevitable. Since an application running on disaggregated architecture can be using a set of hardware components and one component can host multiple applications, a component failure should not affect application execution or make the whole system unavailable.

3) How to support existing datacenter applications? All current datacenter applications are designed to run on monolithic servers. To provide complete transparency and backward compatibility, we need to hide the resource disaggregation nature in a disaggregated OS.

We are building Lego, a distributed, loosely-coupled, failure-independent OS, designed and built from scratch for disaggregated hardware architecture. Lego runs a *component manager* at each hardware component and appears to applications as a set of distributed servers. It currently consists of three types of component managers: processor, memory, and storage. These component managers can be heterogeneous and can be added, restarted, or reconfigured dynamically without affecting the rest of the disaggregated system. Lego uses coarse-grained, global resource management to allocate, schedule, and coordinate across components.

Lego cleanly separates the functionalities of different component managers, with zero or only minimal dependencies across them. Our approach is to build each Lego manager as a *stateless* service. All communication across managers is performed with RPC (we implemented a customized, RDMA-based RPC stack) and each RPC request contains all the information needed to complete the corresponding operation. For example, our storage manager is built in the NFS stateless server style.

Lego processor and memory managers are also cleanly separated in that memory managers manage (de)-allocation, mapping of all physical memory and virtual memory addresses. Processors only view virtual memory addresses assigned by memory managers. To assist this clean separation and make processor hardware simple, we propose a hardware design where processors use *virtual caches*, caches that are virtually indexed and virtually tagged, at all levels. Address mappings are moved completely to memory components, which are free to choose their own virtual memory organization (*e.g.*, paging or segmentation). Network communication is still slower than local memory accesses and will be so in the near future. Therefore, we propose to use a small DRAM attached to the processor as a last-level cache (LLC). Lego manages this DRAM-based LLC in software.

Lego uses a combination of process checkpointing and memory replication to handle component failure. Different from traditional process checkpointing, we store checkpoints in memory components and selectively replicate checkpoints in memory. Replication in memory also has the benefits of sustain memory component failure besides processor failure. We move checkpoints to storage when memory space is scarce.

We plan to evaluate resource utilization, application performance, and failure recovery performance of Lego. To evaluate performance, we will emulate hardware components using commodity x86 servers, for example, by limiting number of active cores to emulate a memory component.

References

[1] P. Faraboschi, K. Keeton, T. Marsland, and D. Milojicic. Beyond processor-centric operating systems. In *15th Workshop on Hot Topics in Operating Systems (HotOS '15)*, Kartause Ittingen, Switzerland, May 2015.

[2] P. X. Gao, A. Narayan, S. Karandikar, J. Carreira, S. Han, R. Agarwal, S. Ratnasamy, and S. Shenker. Network requirements for resource disaggregation. In *OSDI'16)*, pages 249–264, GA, 2016. USENIX Association.

[3] K. Lim, J. Chang, T. Mudge, P. Ranganathan, S. K. Reinhardt, and T. F. Wenisch. Disaggregated memory for expansion and sharing in blade servers. In *ISCA '09*, New York, NY, USA, 2009. ACM.

Lego: A Distributed, Decomposed OS for Resource Disaggregation

PURDUE UNIVERSITY

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

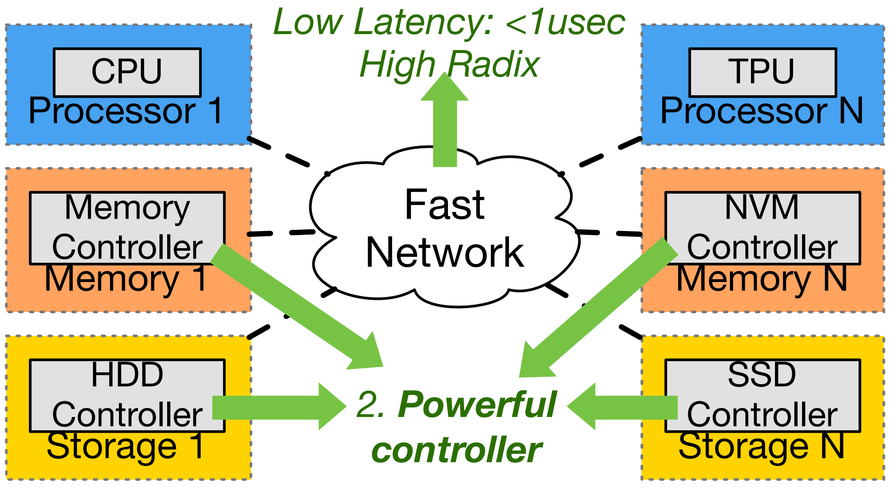
WukLab

Wuklab.io

When hardware is disaggregated, the OS should be also!

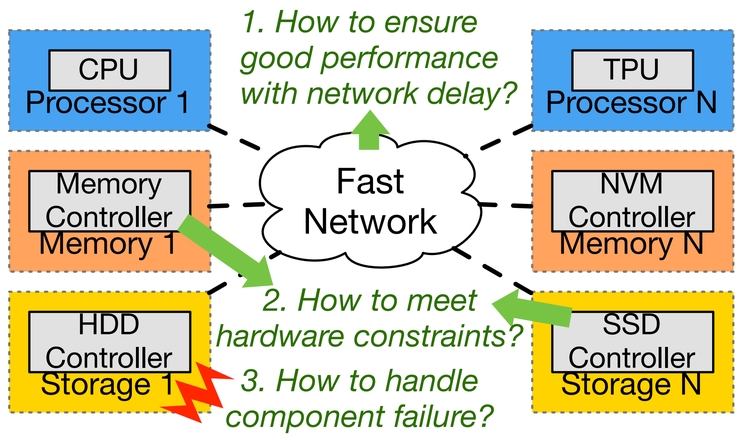
Resource Disaggregation

Breaking *monolithic* servers into network-attached, independent hardware components



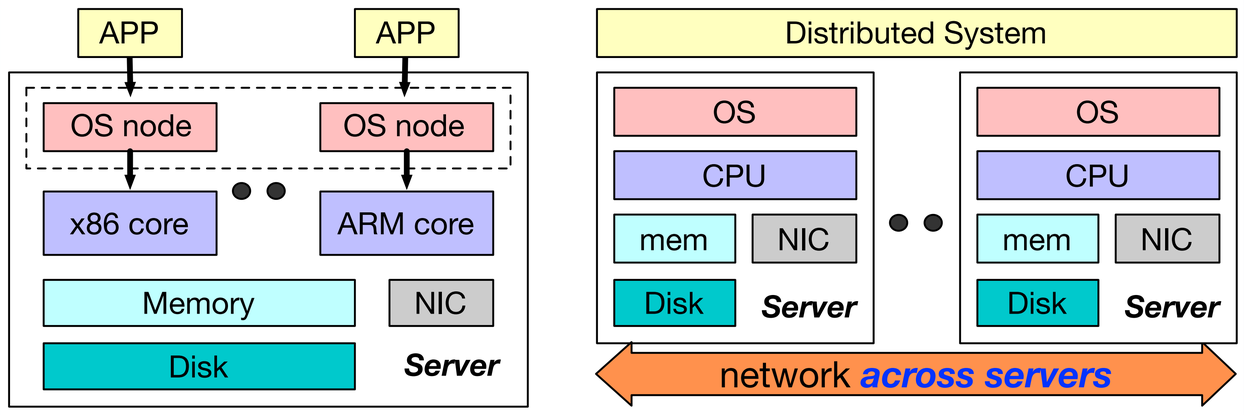
- Efficient resource utilization
- Finer-grained failure
- Easy to add/remote/change hardware

Unique Challenges



No existing OS solves the unique challenges of resource disaggregation!

Problems with Existing OSes



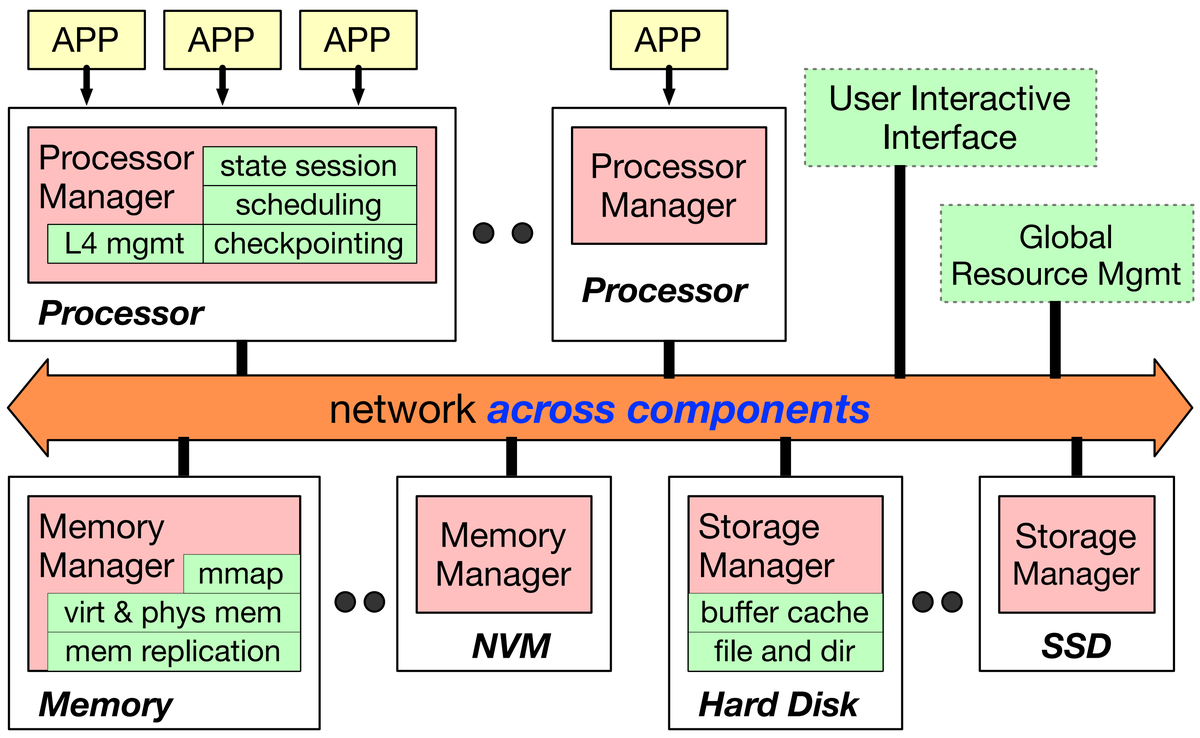
Multikernel: manages cores, not other components

Monolithic/micro- kernel: built for monolithic server

Distributed OS: manages distributed monolithic servers, but not distributed hardware components

Lego: a Disaggregated OS

How to build an OS for resource disaggregation?



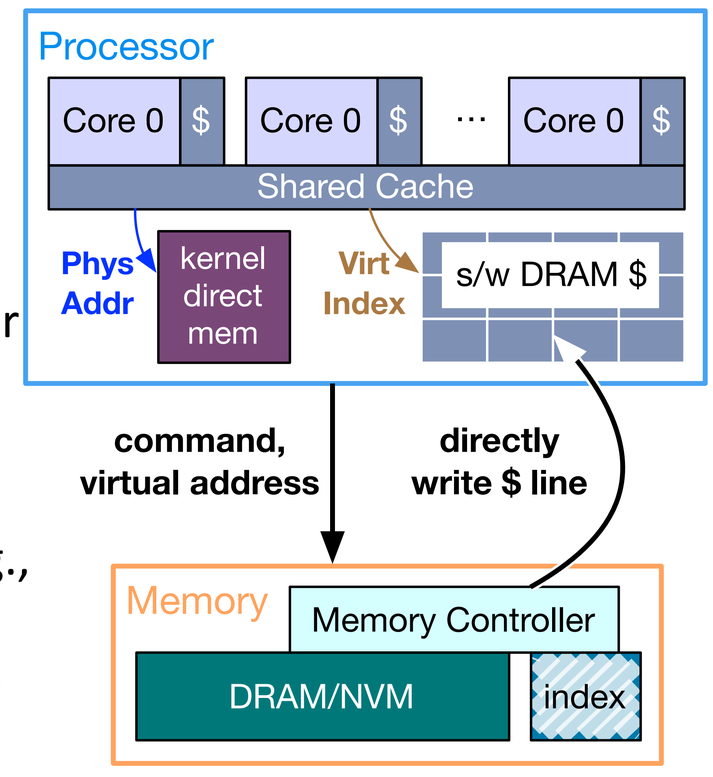
- Split kernel to stateless managers
- Handle failures transparently
- No memory sharing across processors
- Built from scratch; ongoing, open-source project
- Supports x86-64 and unmodified Linux binaries

Hardware Design

How to cleanly separate hardware components while ensuring good performance?

Processor

- Only knows virtual addresses, caches are all 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 Memory Failure

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

Handle Processor Failure

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