UNIVERSITY OF CALIFORNIA SAN DIEGO

Design and Implementation of Disaggregated Datacenter Systems

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Yizhou Shan

Committee in charge:

Professor Yiying Zhang, Chair
Professor Humor Less
Professor Ironic Name
Professor Cirius Thinker

2022

DEDICATION

To my family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Thanks to whoever deserves credit for Blacks Beach, Porters Pub, and every coffee shop in San Diego.

Thanks also to hottubs.

VITA

2002	B. S. in Mathematics cum laude, University of Southern North Dakota, Hoople
2002-2007	Graduate Teaching Assistant, University of California, San Diego
2007	Ph. D. in Mathematics, University of California, San Diego

PUBLICATIONS

Your Name, “A Simple Proof Of The Riemann Hypothesis”, Annals of Math, 314, 2007.

Your Name, Euclid, “There Are Lots Of Prime Numbers”, Journal of Primes, 1, 300 B.C.

ABSTRACT OF THE DISSERTATION

Design and Implementation of Disaggregated Datacenter Systems

by

Yizhou Shan

Doctor of Philosophy in Computer Science

University of California San Diego, 2022

Professor Yiying Zhang, Chair

This dissertation will be abstract.

Chapter 1

Introduction

This is only a test.

1.1 A section

Test.

1.1.1 A Figure Example

This subsection shows a sample figure.

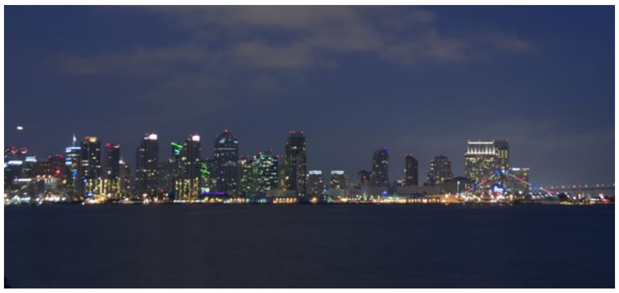


Figure 1.1: A picture of San Diego. Short figure caption must be $<$ 4 lines in the list of figures and match the start of the main figure caption verbatim. Note that figures must be on their own line (no neighboring text) and captions must be single-spaced and appear *below* the figure. Captions can be as long as you want, but if they are longer than 4 lines in the list of figures, you must provide a short figure caption.

1.1.2 A Table Example

While in Section **??** Figure **??** we had a majestic figure, here we provide a crazy table example.

Time of day	Hunger Level	Preferred Food
8am	high	IHOP (French Toast)
noon	medium	Croutons (Tomato Basil Soup & Granny Smith Chicken Salad)
5pm	high	Bombay Coast (Saag Paneer) or Hi Thai (Pad See Ew)
8pm	medium	Yogurt World (froyo!)

Table 1.1: A table of when I get hungry. Short table caption must be $<$ 4 lines in the list of tables and match the start of the main table caption verbatim. Note that tables must be on their own line (no neighboring text) and captions must be single-spaced and appear *above* the table. Captions can be as long as you want, but if they are longer than 4 lines in the list of figures, you must provide a short figure caption.

Chapter 2

Distributed Shared Persistent Memory

Hotpot.

This is only a test.

2.1 A section

Test.

2.1.1 A Figure Example

This subsection shows a sample figure.

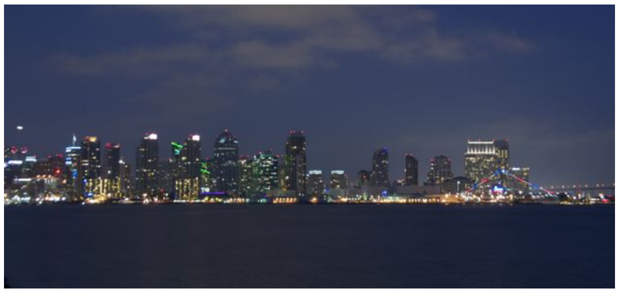
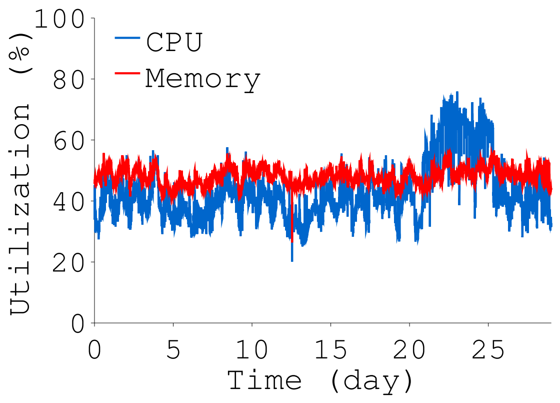


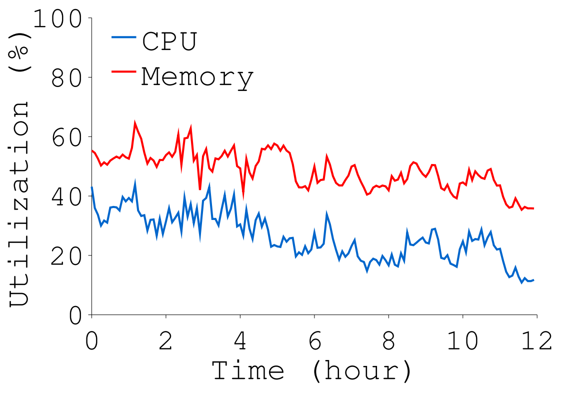
Figure 2.1: A picture of San Diego. Short figure caption must be $<$ 4 lines in the list of figures and match the start of the main figure caption verbatim. Note that figures must be on their own line (no neighboring text) and captions must be single-spaced and appear *below* the figure. Captions can be as long as you want, but if they are longer than 4 lines in the list of figures, you must provide a short figure caption.

Chapter 3

LegoOS



(a) Google Cluster.

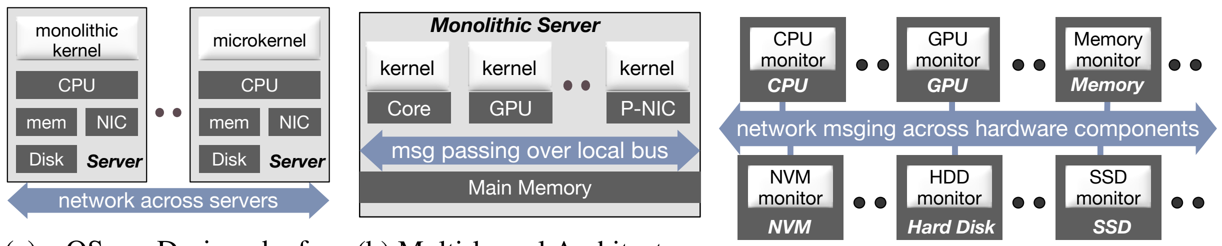


(b) Alibaba Cluster.

Figure 3.1: Data center resource utilization.

3.1 Disaggregate Hardware Resource

This section motivates the hardware resource disaggregation architecture and discusses the challenges in managing disaggregated hardware.



(a) OSes Designed for Monolithic Servers. (b) Multi-kernel Architecture. P-NIC: programmable NIC. (c) Splitkernel Architecture.

Figure 3.2: Operating System Architecture.

3.1.1 Limitations of Monolithic Servers

A monolithic server has been the unit of deployment and operation in datacenters for decades. This long-standing *server-centric* architecture has several key limitations.

Inefficient resource utilization. With a server being the physical boundary of resource allocation, it is difficult to fully utilize all resources in a datacenter [?, ?, ?]. We analyzed two production cluster traces: a 29-day Google one [?] and a 12-hour Alibaba one [?]. Figure **??** plots the aggregated CPU and memory utilization in the two clusters. For both clusters, only around half of the CPU and memory are utilized. Interestingly, a significant amount of jobs are being evicted at the same time in these traces (*e.g.*, evicting low-priority jobs to make room for high-priority ones [?]). One of the main reasons for resource underutilization in these production clusters is the constraint that CPU and memory for a job have to be allocated from the same physical machine.

Poor hardware elasticity. It is difficult to add, move, remove, or reconfigure hardware components after they have been installed in a monolithic server [?]. Because of this rigidity, datacenter owners have to plan out server configurations in advance. However, with today's speed of change in application requirements, such plans have to be adjusted frequently, and when changes happen, it often comes with waste in existing server hardware.

Coarse failure domain. The failure unit of monolithic servers is coarse. When a hardware component within a server fails, the whole server is often unusable and applications running on it can all crash. Previous analysis [?] found that motherboard, memory, CPU, power supply failures account for 50% to 82% of hardware failures in a server. Unfortunately, monolithic servers cannot continue to operate when any of these devices fail.

Bad support for heterogeneity. Driven by application needs, new hardware technologies are finding their ways into modern datacenters [**?**]. Datacenters no longer host only commodity servers with CPU, DRAM, and hard disks. They include non-traditional and specialized hardware like GPGPU [**?**, **?**], TPU [**?**], DPU [**?**], FPGA [**?**, **?**], non-volatile memory [**?**], and NVMe-based SSDs [**?**]. The monolithic server model tightly couples hardware devices with each other and with a motherboard. As a result, making new hardware devices work with existing servers is a painful and lengthy process [**?**]. Mover, datacenters often need to purchase new servers to host certain hardware. Other parts of the new servers can go underutilized and old servers need to retire to make room for new ones.

3.1.2 Hardware Resource Disaggregation

The server-centric architecture is a bad fit for the fast-changing datacenter hardware, software, and cost needs. There is an emerging interest in utilizing resources beyond a local machine [**?**], such as distributed memory [**?**, **?**, **?**, **?**] and network swapping [**?**]. These solutions improve resource utilization over traditional systems. However, they cannot solve all the issues of monolithic servers (*e.g.*, the last three issues in §**??**), since their hardware model is still a monolithic one. To fully support the growing heterogeneity in hardware and to provide elasticity and flexibility at the hardware level, we should *break the monolithic server model.*

We envision a *hardware resource disaggregation* architecture where hardware resources in traditional servers are disseminated into network-attached *hardware components*. Each component has a controller and a network interface, can operate on its own, and is an *independent, failure-isolated* entity.

The disaggregated approach largely increases the flexibility of a datacenter. Applications can freely use resources from any hardware component, which makes resource allocation easy and efficient. Different types of hardware resources can *scale independently*. It is easy to add, remove, or reconfigure components. New types of hardware components can easily be deployed in a datacenter — by simply enabling the hardware to talk to the network and adding a new network link to connect it. Finally, hardware resource disaggregation enables fine-grain failure isolation, since one component failure will not affect the rest of a cluster.

Three hardware trends are making resource disaggregation feasible in datacenters. First, network speed has grown by more than an order of magnitude and has become more scalable in the past decade with new technologies like Remote Direct Memory Access (*RDMA*) [?] and new topologies and switches [?, ?, ?], enabling fast accesses of hardware components that are disaggregated across the network. InfiniBand will soon reach 200Gbps and sub-600 nanosecond speed [?], being only $2\times$ to $4\times$ slower than main memory bus in bandwidth. With main memory bus facing a bandwidth wall [?], future network bandwidth (at line rate) is even projected to exceed local DRAM bandwidth [?].

Second, network interfaces are moving closer to hardware components, with technologies like Intel OmniPath [?], RDMA [?], and NVMe over Fabrics [?, ?]. As a result, hardware devices will be able to access network directly without the need to attach any processors.

Finally, hardware devices are incorporating more processing power [?, ?, ?, ?, ?, ?], allowing application and OS logics to be offloaded to hardware [?, ?]. On-device processing power will enable system software to manage disaggregated hardware components locally.

With these hardware trends and the limitations of monolithic servers, we believe that future datacenters will be able to largely benefit from hardware resource disaggregation. In fact, there have already been several initial hardware proposals in resource disaggregation [?], including disaggregated memory [?, ?, ?], disaggregated flash [?, ?], Intel Rack-Scale System [?], HP "The Machine" [?, ?], IBM Composable System [?], and Berkeley Firebox [?].

3.1.3 OSes for Resource Disaggregation

Despite various benefits hardware resource disaggregation promises, it is still unclear how to manage or utilize disaggregated hardware in a datacenter. Unfortunately, existing OSes and distributed systems cannot work well with this new architecture. Single-node OSes like Linux view a server as the unit of management and assume all hardware components are local (Figure **??**). A potential approach is to run these OSes on processors and access memory, storage, and other hardware resources remotely. Recent disaggregated systems like soNUMA [**?**] take this approach. However, this approach incurs high network latency and bandwidth consumption with remote device management, misses the opportunity of exploiting device-local computation power, and makes processors the single point of failure.

Multi-kernel solutions [**?**, **?**, **?**, **?**, **?**] (Figure **??**) view different cores, processors, or programmable devices within a server separately by running a kernel on each core/device and using message passing to communicate across kernels. These kernels still run in a single server and all access some common hardware resources in the server like memory and the network interface. Moreover, they do not manage distributed resources or handle failures in a disaggregated cluster.

There have been various distributed OS proposals, most of which date decades back [**?**, **?**, **?**]. Most of these distributed OSes manage a set of monolithic servers instead of hardware components.

Hardware resource disaggregation is fundamentally different from the traditional monolithic server model. A complete disaggregation of processor, memory, and storage means that when managing one of them, there will be no local accesses to the other two. For example, processors will have no local memory or storage to store user or kernel data. An OS also needs to manage distributed hardware resource and handle hardware component failure. We summarize the following key challenges in building an OS for resource disaggregation, some of which have previously been identified [**?**].

- How to deliver good performance when application execution involves the access of network-partitioned disaggregated hardware and current network is still slower than local buses?
- How to locally manage individual hardware components with limited hardware resources?
- How to manage distributed hardware resources?
- How to handle a component failure without affecting other components or running applications?
- What abstraction should be exposed to users and how to support existing datacenter applications?

Instead of retrofitting existing OSes to confront these challenges, we take the approach of designing a new OS architecture from the ground up for hardware resource disaggregation.

Chapter 4

Clio

This is only a test.

4.1 A section

Test.

4.1.1 A Figure Example

This subsection shows a sample figure.

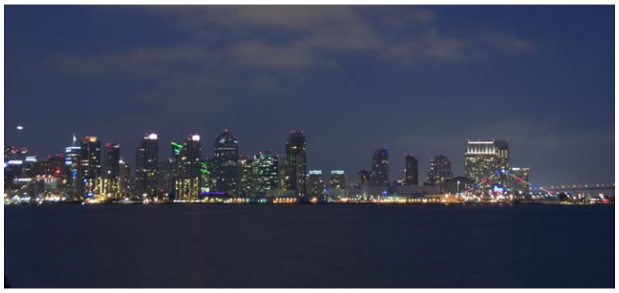


Figure 4.1: A picture of San Diego. Short figure caption must be $<$ 4 lines in the list of figures and match the start of the main figure caption verbatim. Note that figures must be on their own line (no neighboring text) and captions must be single-spaced and appear *below* the figure. Captions can be as long as you want, but if they are longer than 4 lines in the list of figures, you must provide a short figure caption.

Chapter 5

SuperNIC

Chapter 6

Conclusion

TODO.

Appendix A

Final notes

What to do about things [?]. What did he say [?]. Remove me in case of abdominal pain.