Distributed Data-Driven Operating System (D3OS)

Michael Schöttner

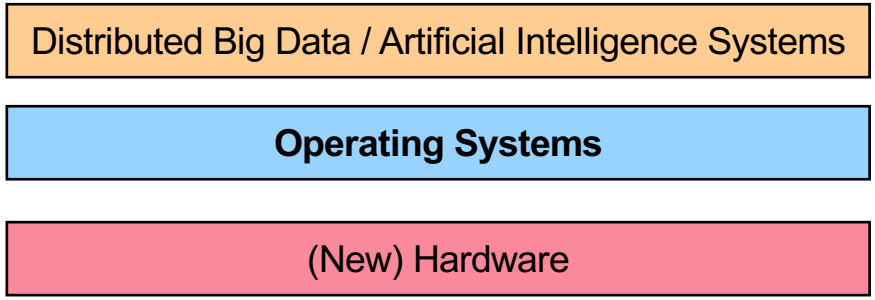
Heinrich-Heine University Duesseldorf

27.9.2022

Motivation

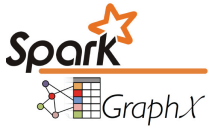
Motivation Overview

- The following motivation will address different layers of the software and hardware stack
- From big-data / AI systems and data centers
- Down to the operating system and new emerging hardware



Big-Data Systems

- Many distributed big-data systems have been developed (in Java) …
  - Batch processing systems, e.g. Hadoop MapReduce
  - In-memory processing systems, e.g. Apache Spark
  - Stream processing systems, e.g. Apache Flink
  - Graph processing systems, e.g. Apache GraphX



- These systems provide sophisticated APIs and each its own runtime
  - Storage management
  - Fast network communication
  - Distributed task and local thread management

Distributed AI Systems

Heinrich Heine Universität Düsseldorf

- Recently, more and more distributed AI platforms are emerging
  - Aiming at parallelizing training for huge amounts of training data
  - As well as training deep learning network which have up to billions of parameters
  - Examples are: Tensor flow, Ray, Horovod, as well as cloud offerings

- These systems provide sophisticated APIs (in Python) and each its own runtime
  - Using GPUs
  - Storage management
  - Fast network communication
  - Distributed task and local thread management

Complexity of BD/AI system stacks

Heinrich Heine
Universität Düsseldorf

- The systems mentioned before are typically a very complex layered middleware
  - Each of them re-implements resource management like a distributed operating system
  - Lines of code are typically one million and more

- They run on top of complex operating systems, mostly based on Linux
  - The Linux kernel has more than 40 million lines of code

→ This is also calling for a distributed operating system providing basic data centric abstractions as well as fast and direct access to modern hardware

→ Objective: reduce complexity

Cloud Data Centers

Heinrich Heine
Universität Düsseldorf

- Big-data and distributed AI systems are designed to run in data centers
- Large amounts of resources
- Fast reliable network
- Trusted environment

NVRAM

Heinrich Heine
Universität Düsseldorf

- Fast byte addressable persistent memory

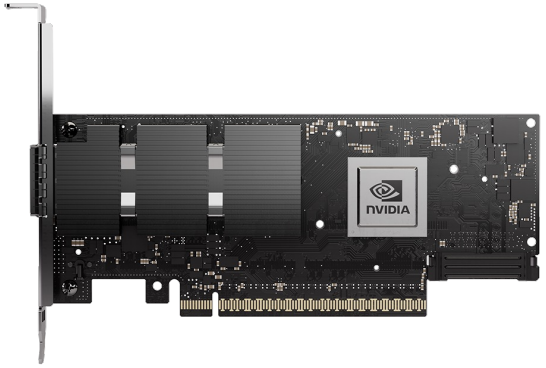
- Intel Optane DIMM
  - Used like DRAM
  - 128 GB – 512 GB Module
  - AES 256-bit encryption in Hardware

- Potential for simplifying many things
  - On the OS level: we can reduce the complexity of the memory hierarchy
    - No disks, no DRAM, no paging, no file systems, ...
  - On the application level:
    - No need for two data representations
  - But errors are persistent!

InfiniBand

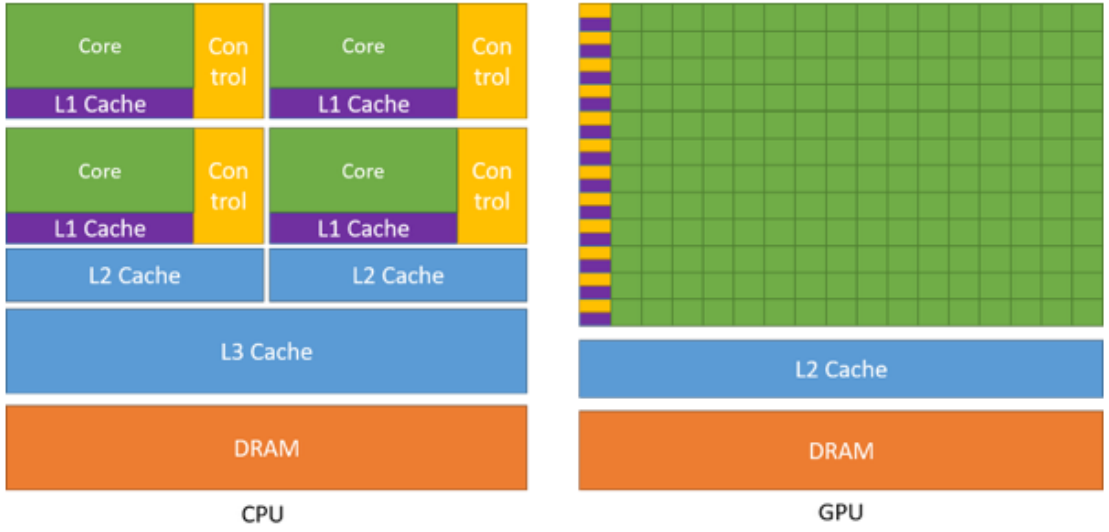
Heinrich Heine
Universität Düsseldorf

- Very fast network technology
- Up to 400 Gbit/s and 500ns latency (one way)
- Offers message passing and Remote Direct Memory Access (RDMA)
- Network protocol stacks are implemented in firmware



GPUs

- Provide a lot of vector processors (SIMD)
- For many applications beyond gaming
  - Artificial intelligence
  - Machine learning
  - Crypto mining
  - …
- Up to 10.000 cores and 80 GB VRAM per graphic card



Programmable NICs

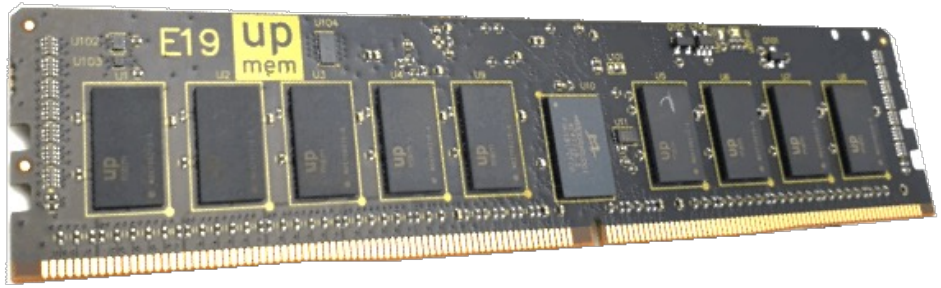
Heinrich Heine
Universität Düsseldorf

- Nvidia Bluefield Data Processing Units
  „data center infrastructure-on-a-chip“
- Ethernet or InfiniBand up to 400 Gbits/s
- Up to 8 ARM cores (64 Bit)
- Max. 32 GB DRAM on board
- Optional NVMe storage

Programmable DRAM modules

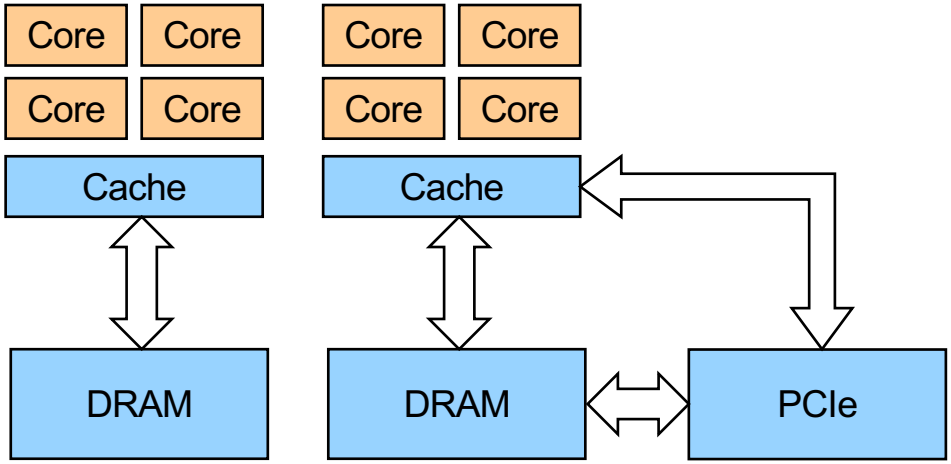
Heinrich Heine
Universität Düsseldorf

- UPEM = DRAM with DPUs (data processors) integrated in DIMM module
- Objective: data processing near memory → near-memory computing
- Application example: parallel word search in a data base

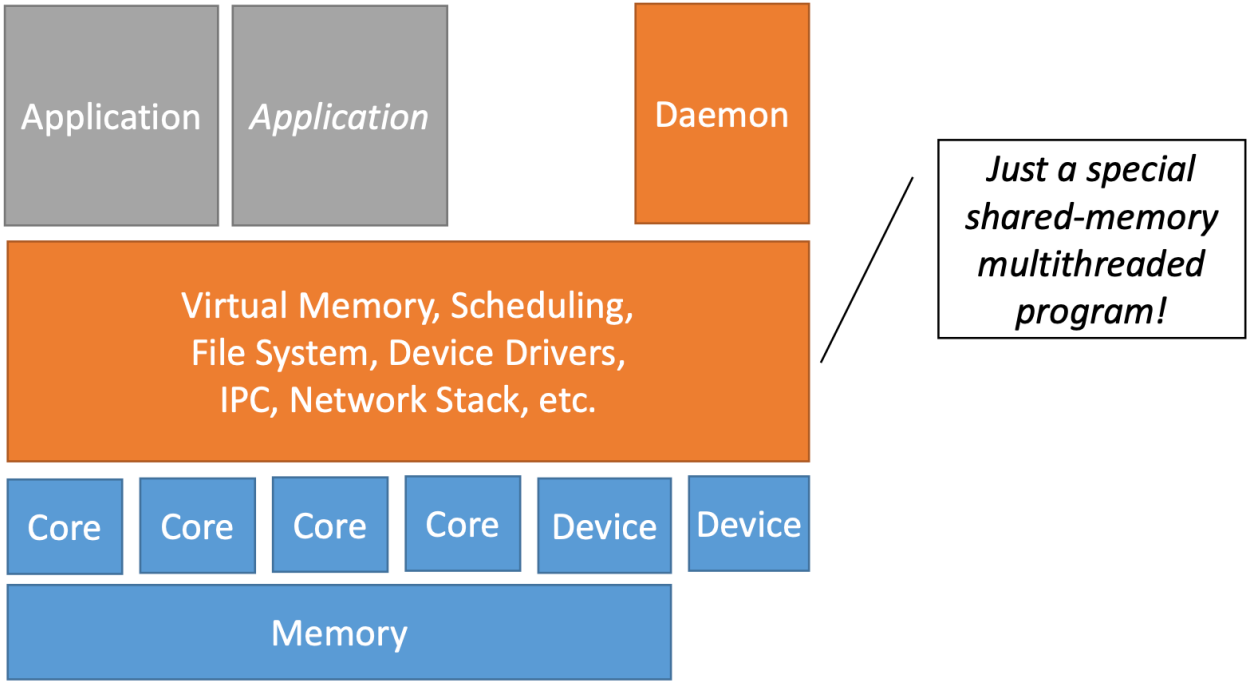


Hardware view of todays server OSs

- Most of todays server operating systems are designed for ccNUMA machines (= cache coherent Non Uniform Memory Access)

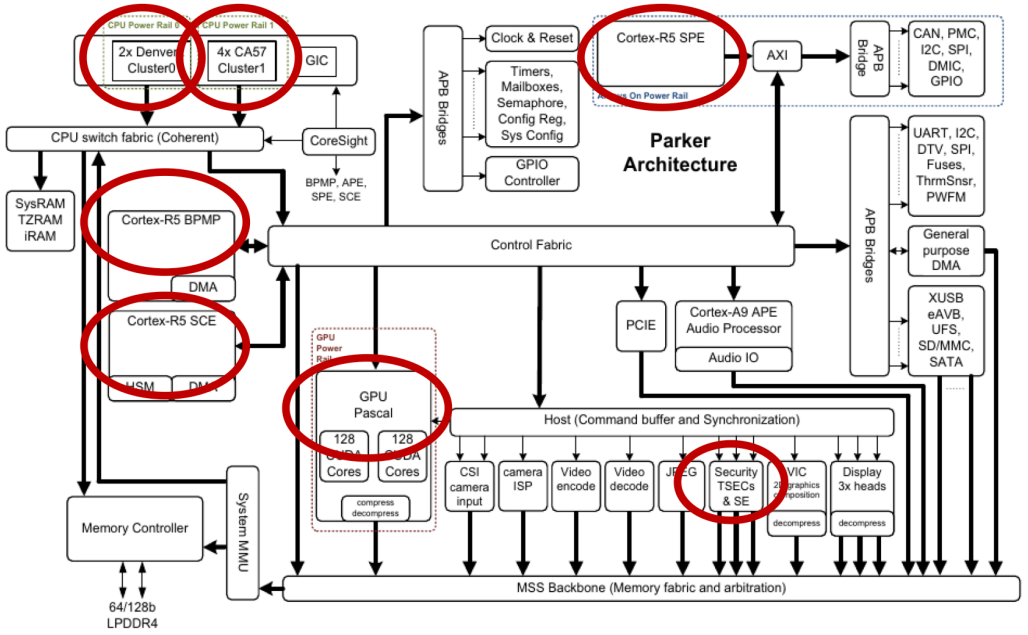


Resulting Architecture of traditional OSs



Todays Hardware looks different

- Fig. shows NVIDIA parker SoC
  - A deep learning computing platform for autonomous machines
- Each read circle is a set of processors
  - Have a different view of the physical address space
  - May be cache coherent or not



Todays Hardware looks different

Heinrich Heine
Universität Düsseldorf

- Also x86_64 servers with SoC devices mentioned before look different
- Each SoC device comes with its own firmware has its own Cache & RAM

- The operating system:

- Runs only on a (small) part of the hardware the central processing unit
- Acts as a coordinator for all the different devices
- Looks like a distributed system with a coordinator

Summary

Heinrich Heine
Universität Düsseldorf

- Big-data and AI frameworks re-implement distributed operating system functionality
- The emerging hardware makes servers look more and more like a distributed system which is calling for distributed operating system concepts
- However, most operating system efforts try to extend or mimic UNIX/Linux
- But Linux is neither a distributed OS nor is it designed for the new hardware and the Linux kernel is very complex and contiguously growing

D3OS

Objectives

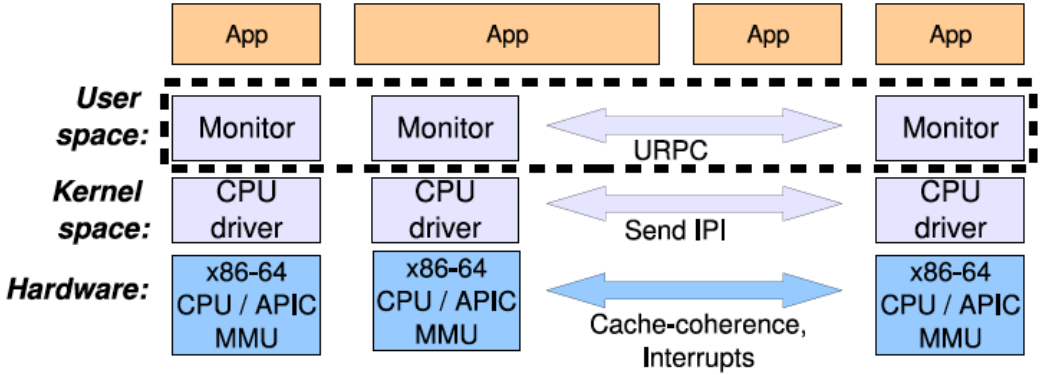
Heinrich Heine
Universität Düsseldorf

- Distributed Data-Driven OS for data centers / clusters
- Target modern hardware:
  - Many core, heterogeneous cores (NUMA, GPUs, programmable NICs, ...)
  - Persistent memory, RDMA, …
- Reduce complexity
- Improve reliability by using Rust

Multikernel approach of Barrelfish OS

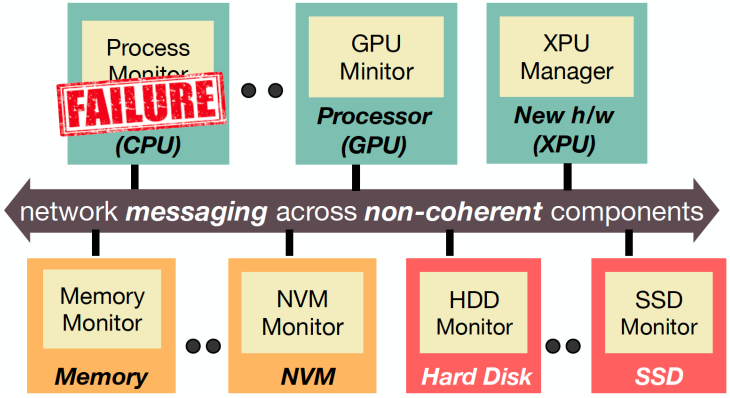
Heinrich Heine
Universität Düsseldorf

- Use message passing between cores instead of shared state
- Improve scalability (message passing vs distributed shared memory)



Splitkernel approach of LegoOS

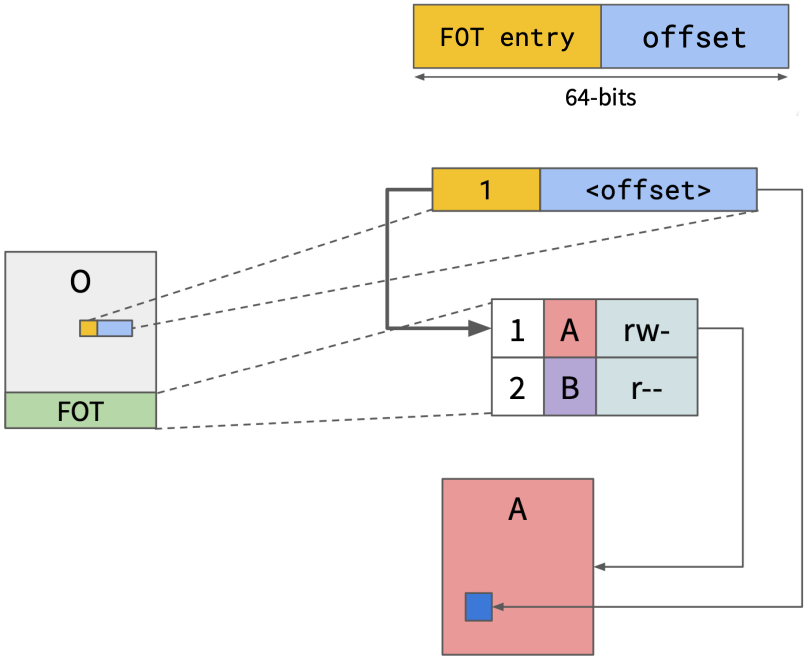
- A splitkernel breaks traditional OS functionalities into loosely-coupled monitors, each running at and managing a hardware component.
- Each splitkernel monitor operates locally for its own functionality and only communicates with other monitors when there is a need to access resources there.
- There are only two global tasks in a splitkernel: orchestrating resource allocation across components and handling component failure



Native support for Pmem of Twizzler OS

Heinrich Heine Universität Düsseldorf

- Objects may be trees or lists or …
- We need persistent pointers → last forever
- FOT = Foreign Object Table
  - FOT entry = 0: reference within object
  - FOT entry > 0: cross-object reference
- Indirektion erlaubt
  - Late binding, e.g. updating shared libraries



Status

- We are at the beginning