Imperial College London

LSDS
Large-Scale Data & Systems Group

Slashing the Disaggregation Tax in Heterogeneous Data Centers with FractOS

Joint work with:

TECHNION Israel Institute of Technology

barkhausen institut

TECHNISCHE UNIVERSITÄT DRESDEN

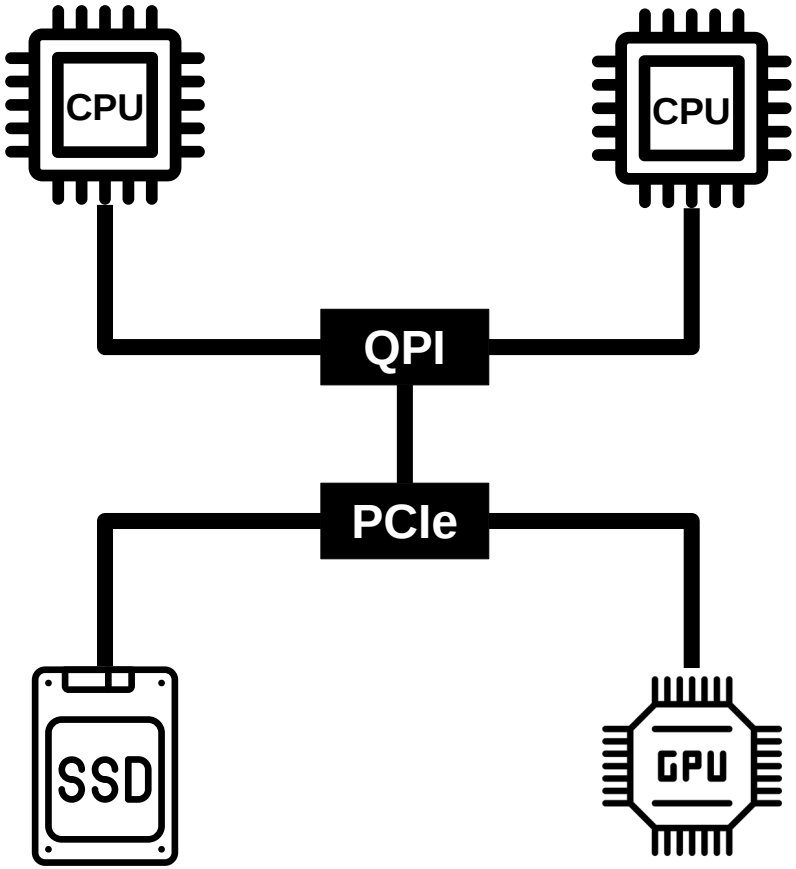
Lluís Vilanova, Lina Maudlej, Shai Bergman, Till Miemietz, Matthias Hille, Nils Asmussen, Michael Roitzsch, Hermann Härtig, Mark Silberstein

https://lsds.doc.ic.ac.uk/projects/fractos
<vilanova@imperial.ac.uk>

From PCIe to Disaggregation

PCIe

- ↓ Scale
- ↑ Performance (128GB/sec, 1µsec)
  (PCIe v6)
- ↓ Congestion and variability



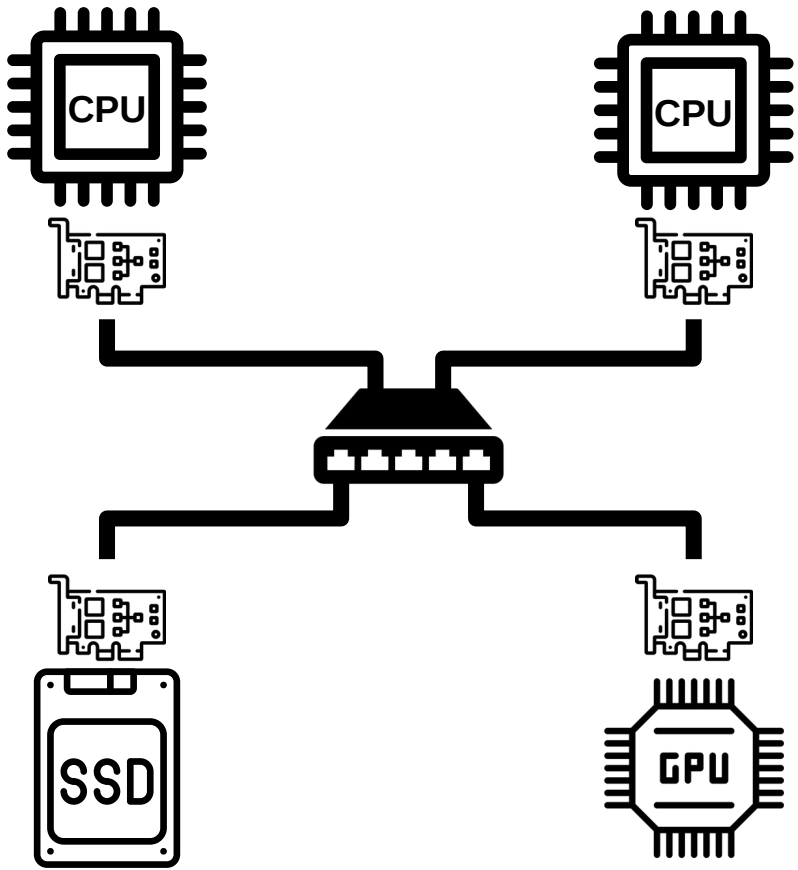
From PCIe to Disaggregation

PCIe

- ↓ Scale
- ↑ Performance (128GB/sec, 1µsec)
  (PCIe v6)
- ↓ Congestion and variability

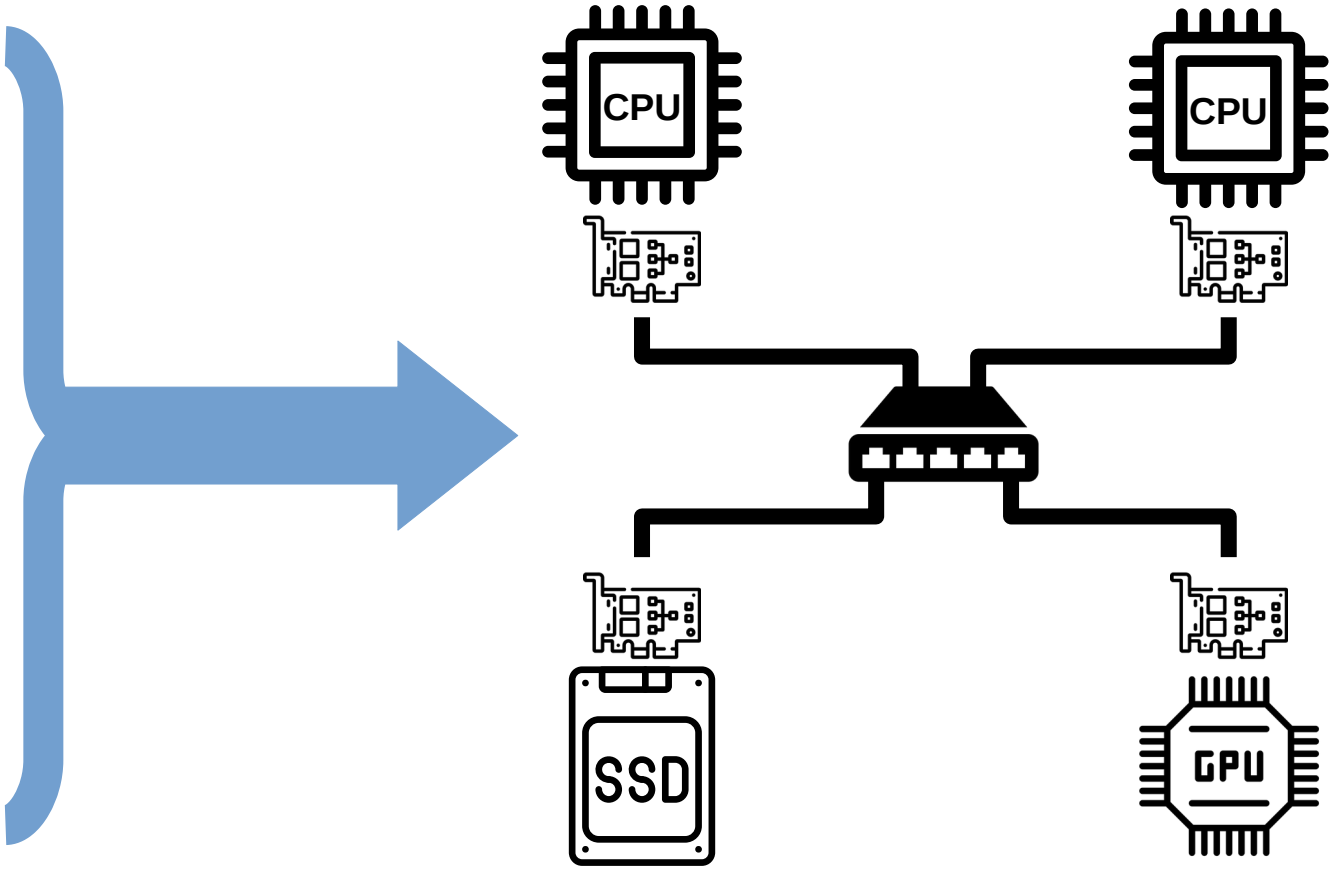
Disaggregation

- ↓ Data center cost
- ↑ Scale
- ↓ Performance (50GB/sec, 24µsec)
  (400 GbE + RoCE RDMA)
- ↑ Congestion and variability

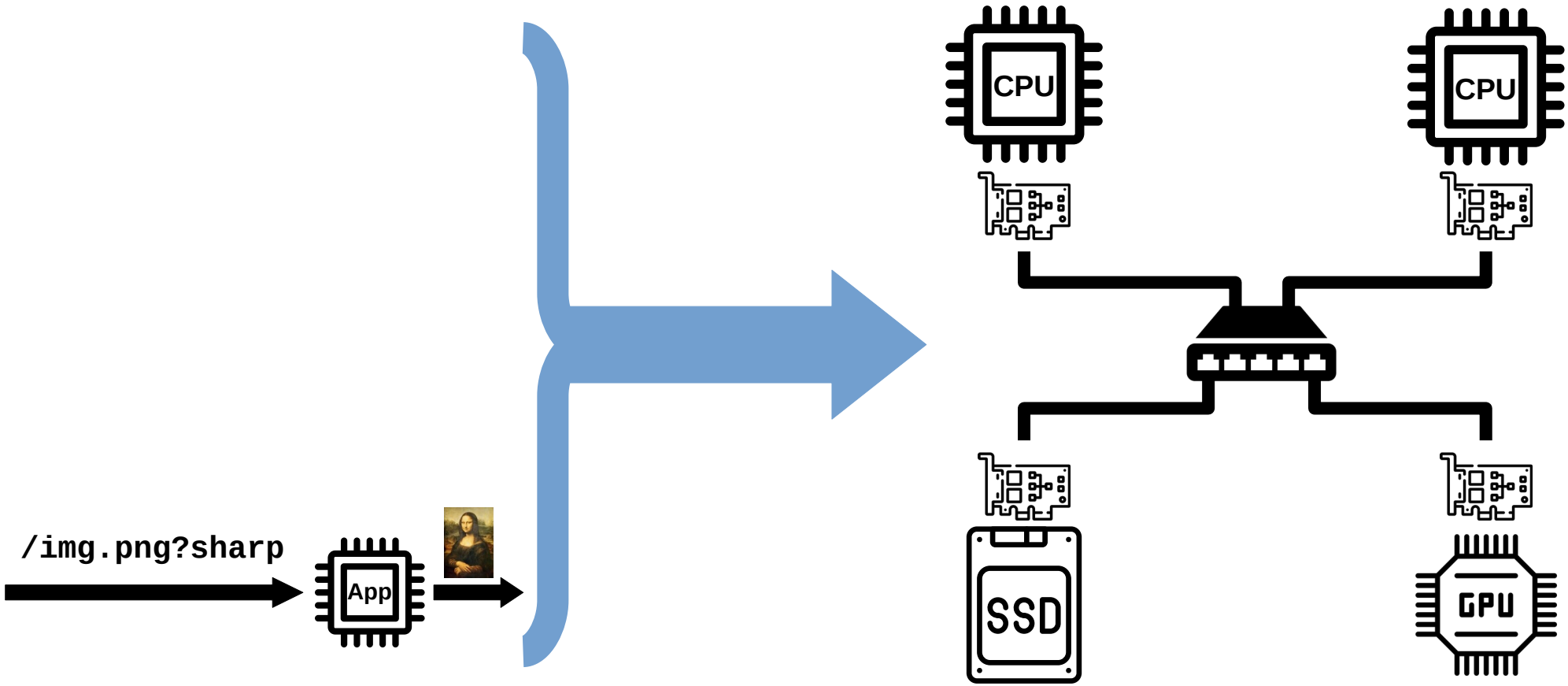


Disaggregation brings new trade-offs

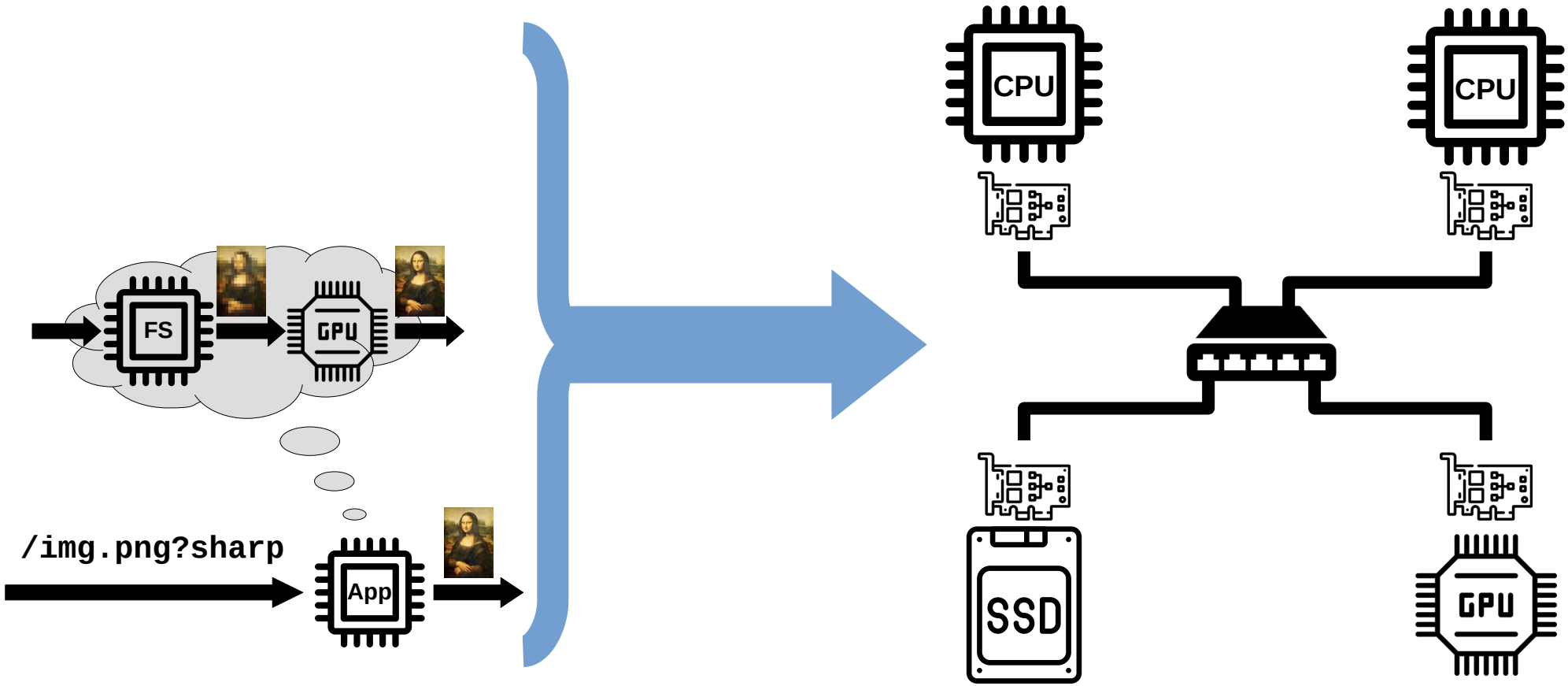
Application Example



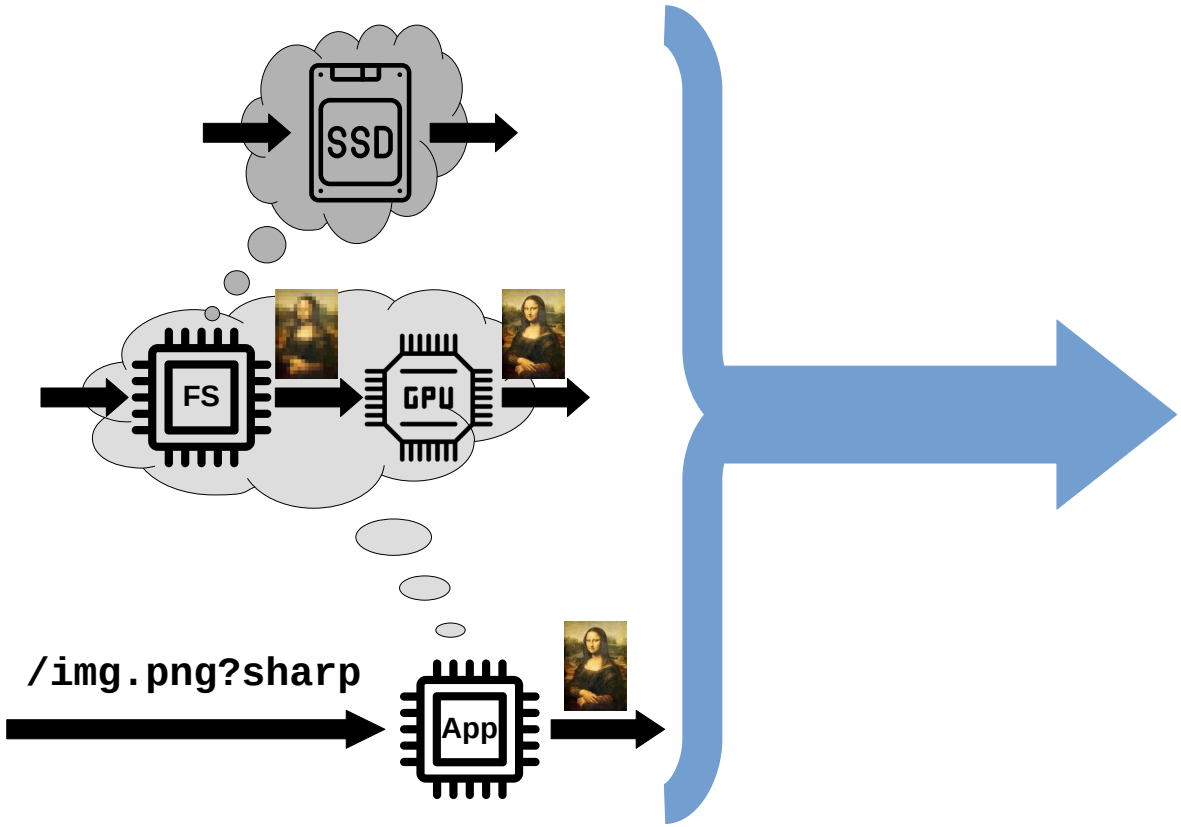
Application Example

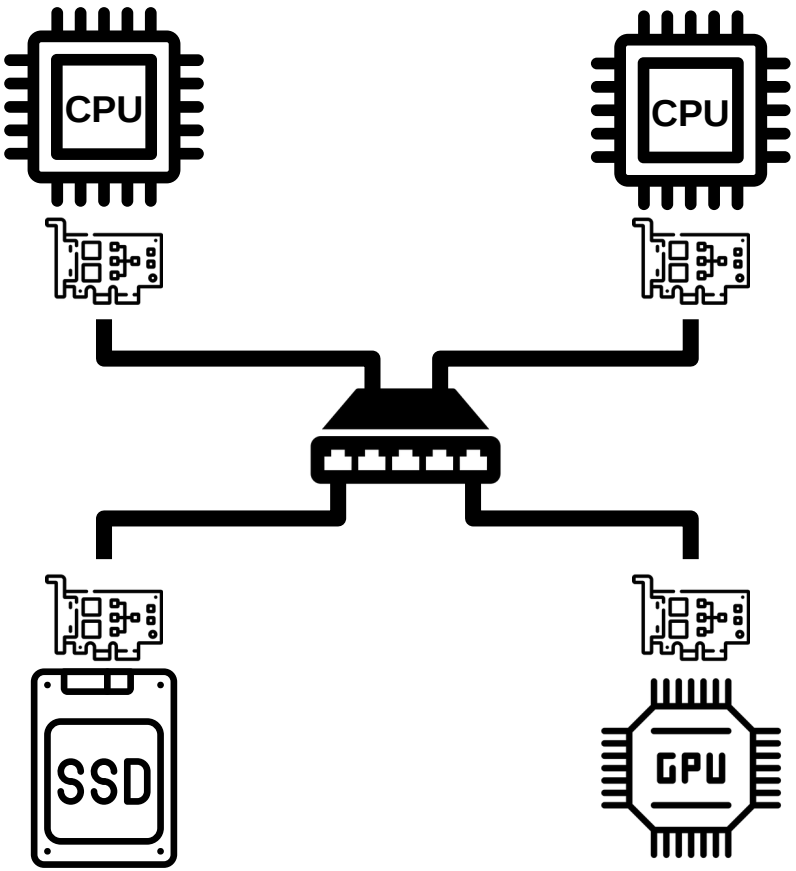


Application Example

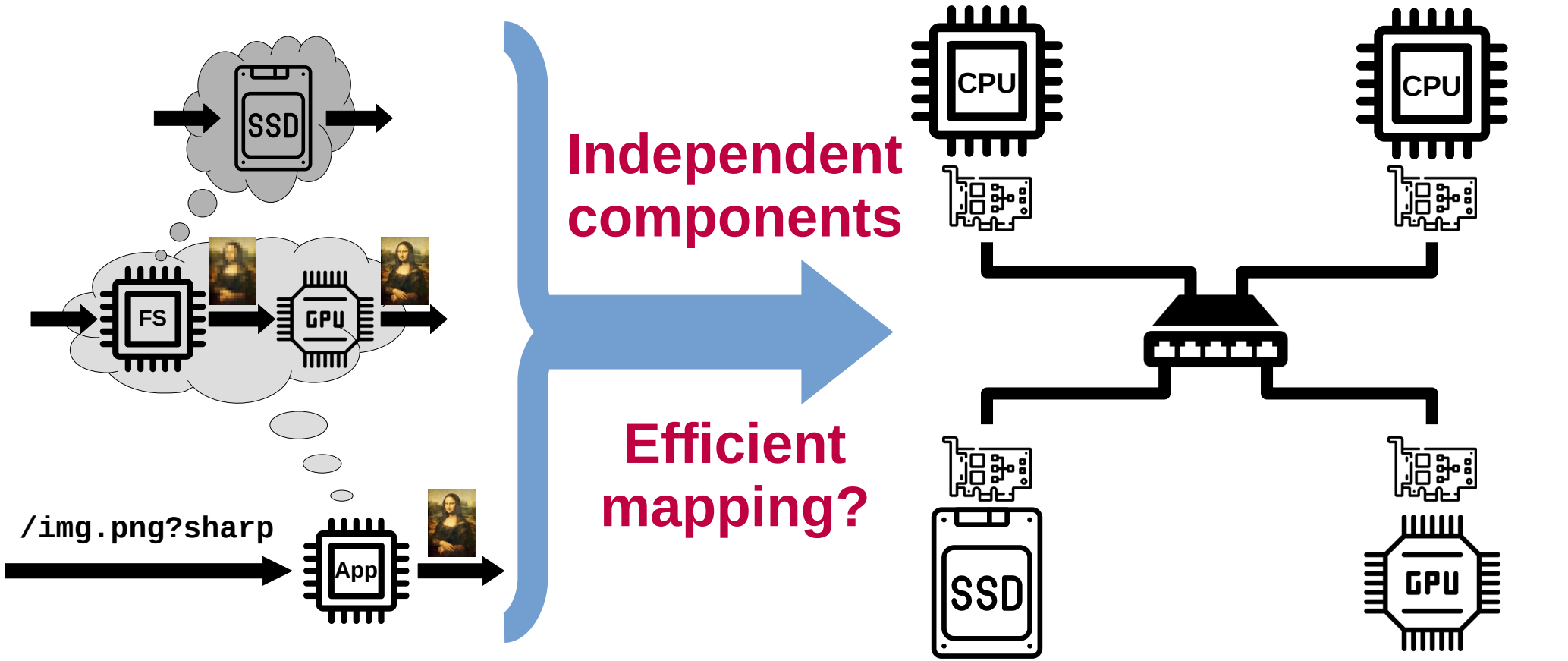


Application Example

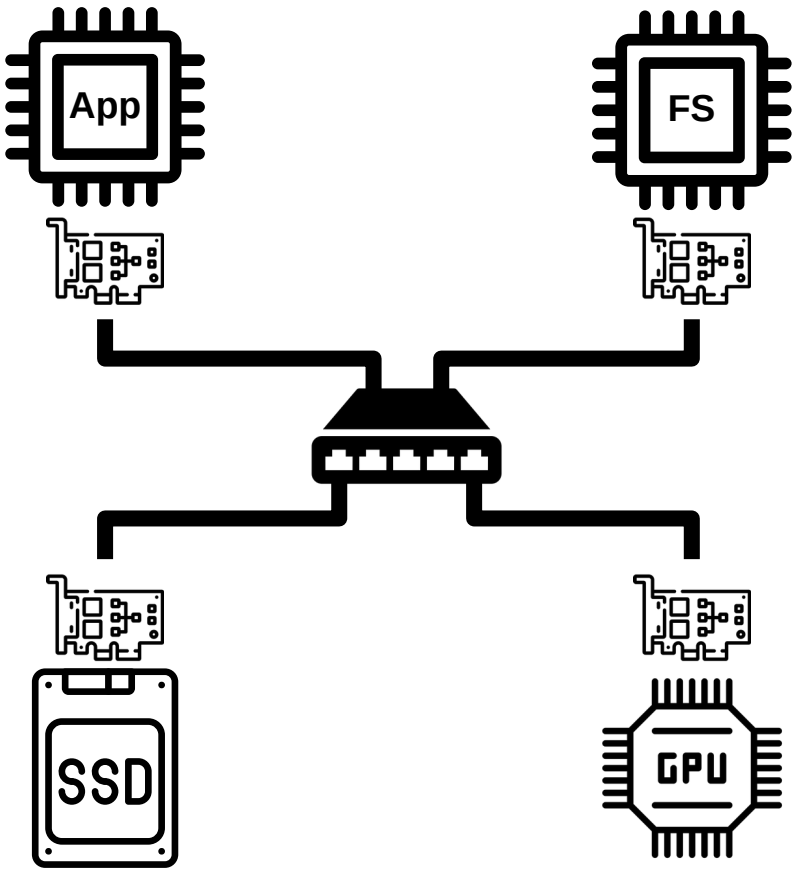




Application Example

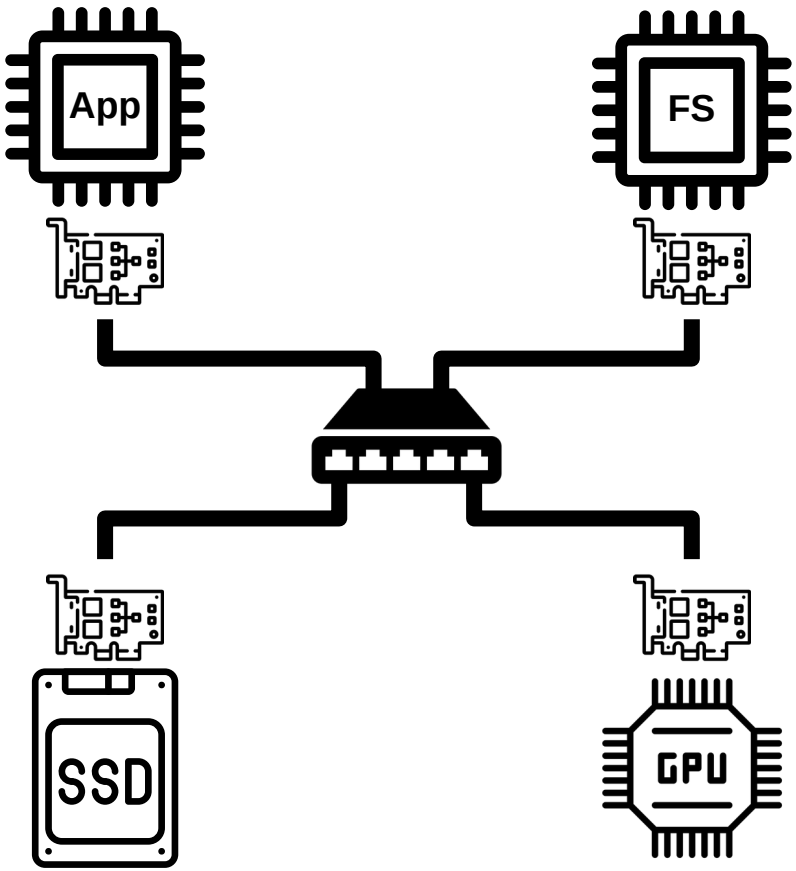


The Disaggregation Tax



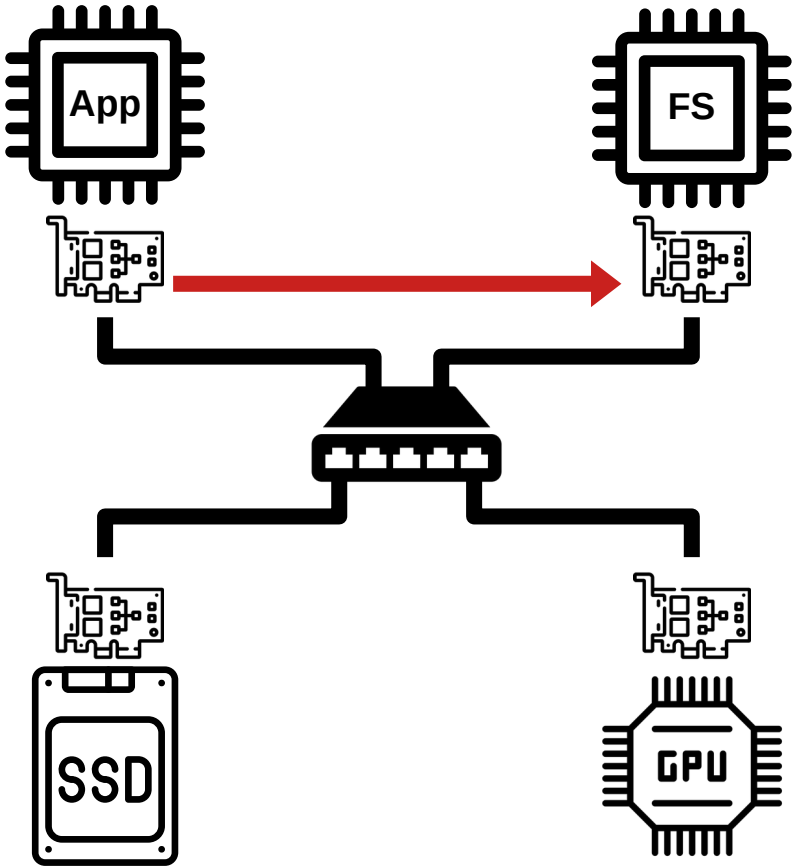
The Disaggregation Tax

- Centralized application
  - Redundant transfers vs. shared network



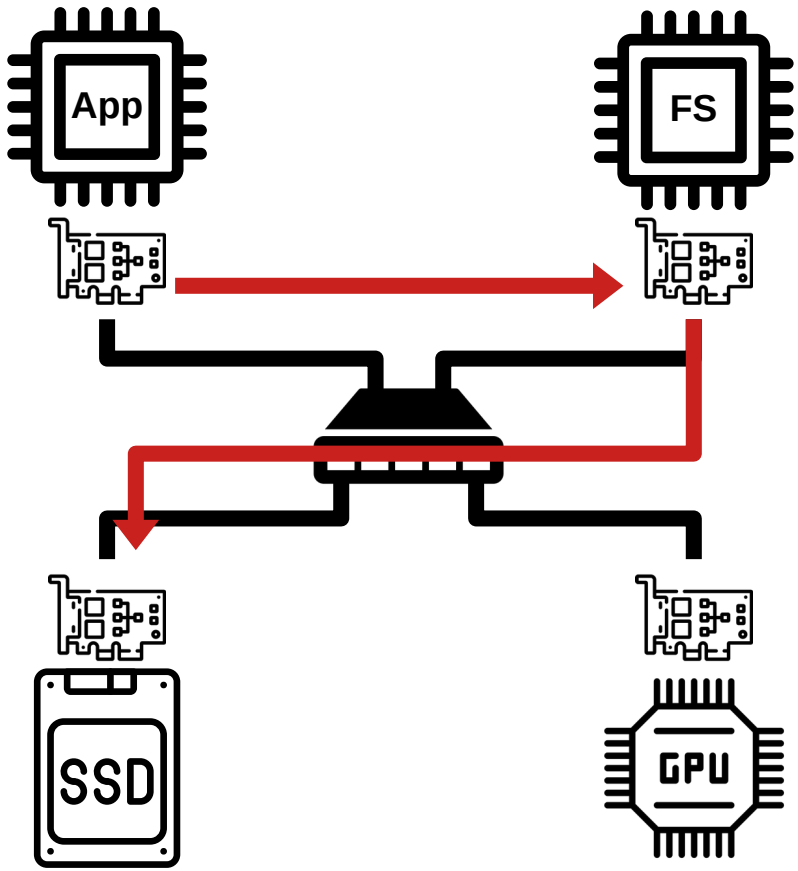
The Disaggregation Tax

- Centralized application
  - Redundant transfers vs. shared network



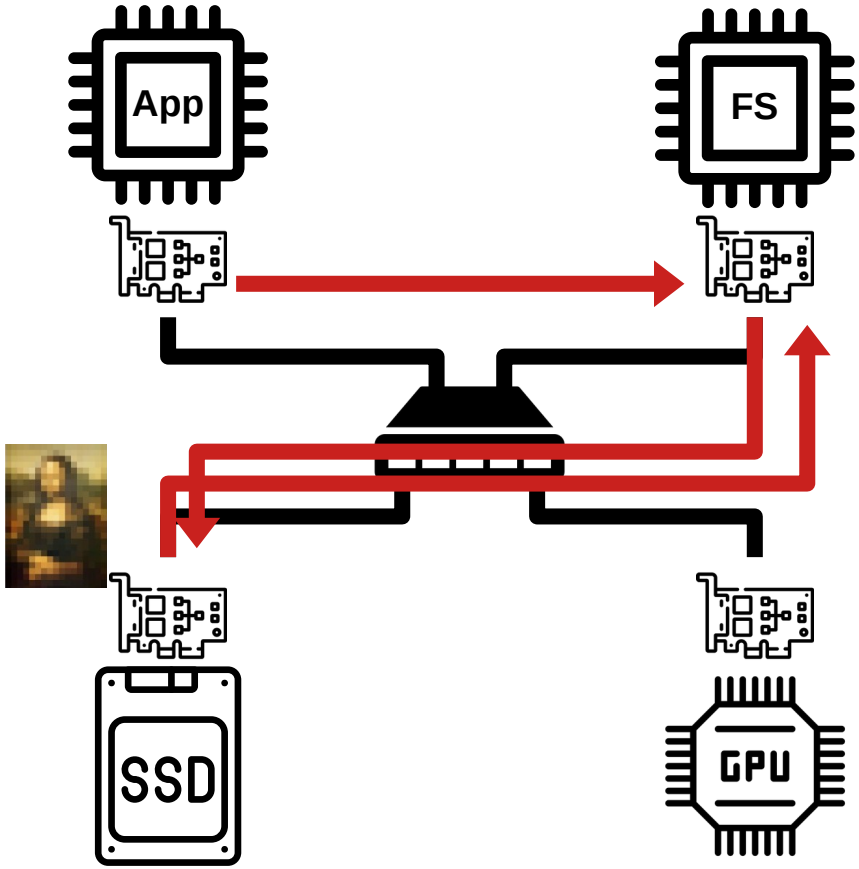
The Disaggregation Tax

- Centralized application
  - Redundant transfers vs. shared network



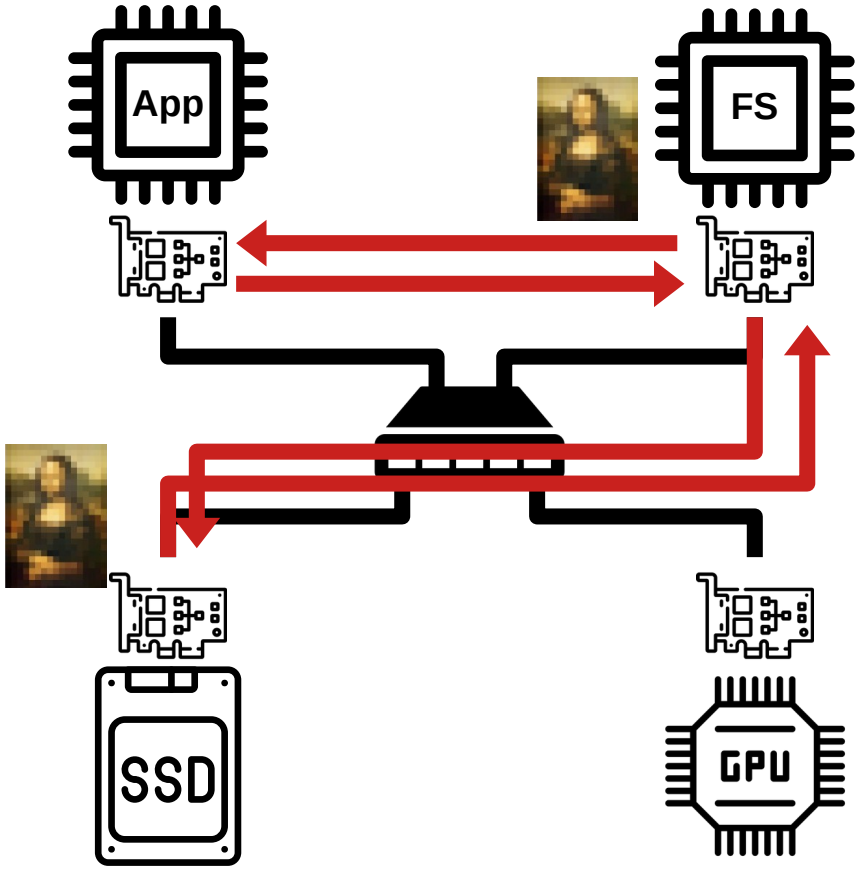
The Disaggregation Tax

- Centralized application
  - Redundant transfers vs. shared network



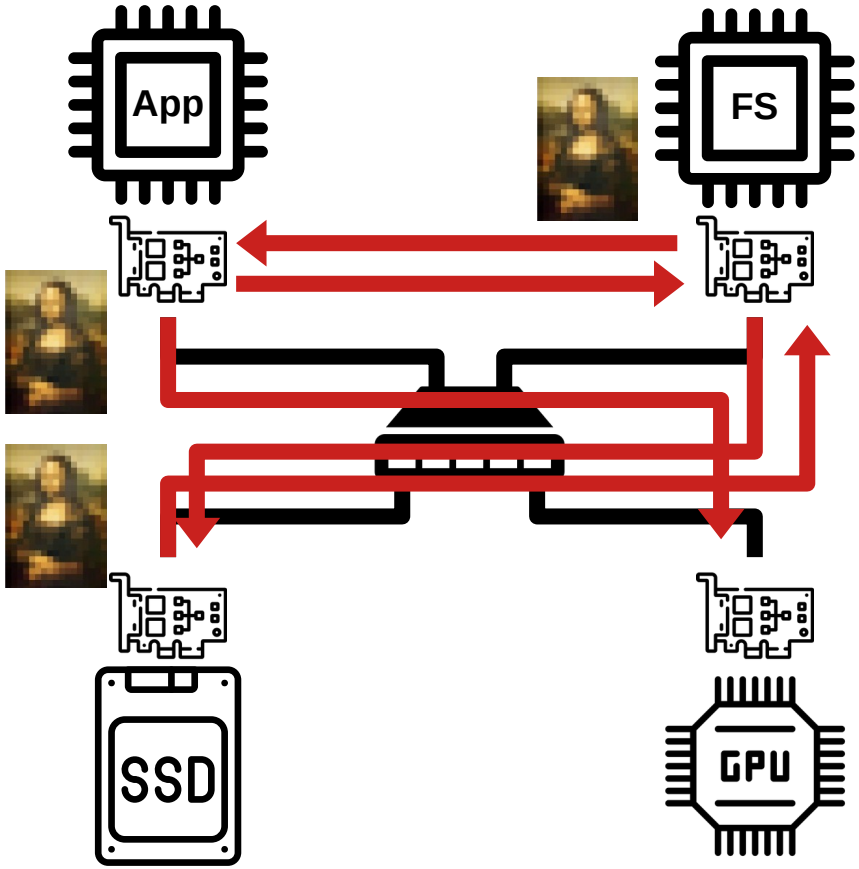
The Disaggregation Tax

- Centralized application
  - Redundant transfers vs. shared network



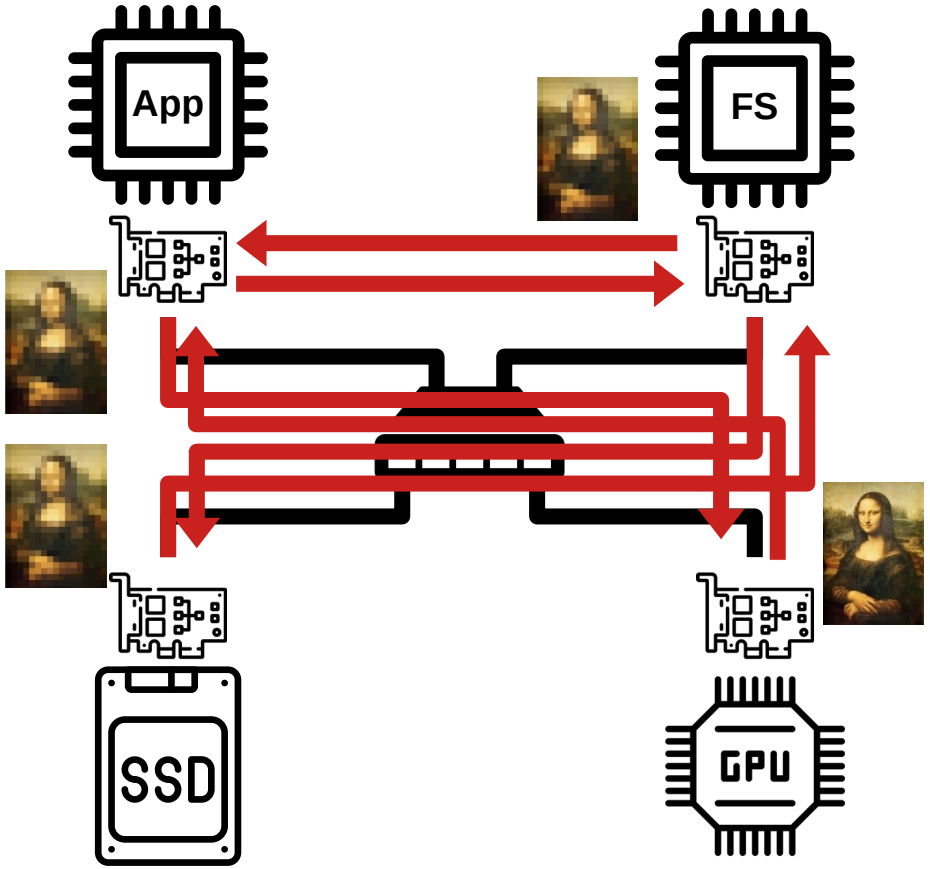
The Disaggregation Tax

- Centralized application
  - Redundant transfers vs. shared network



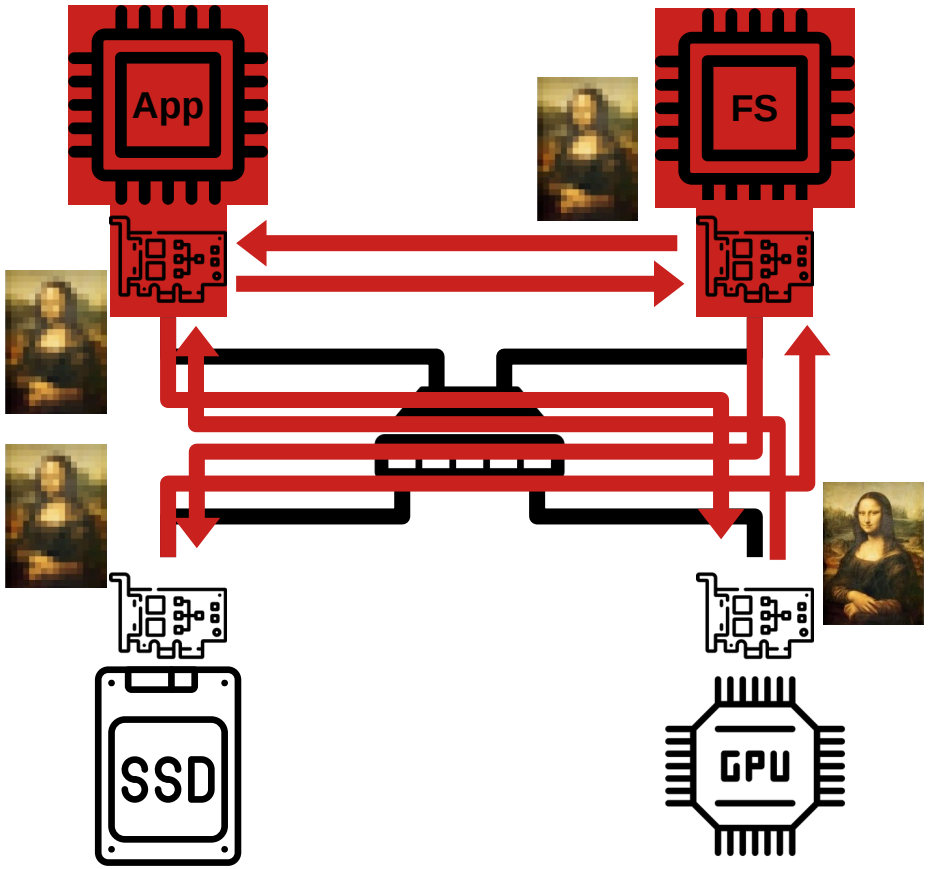
The Disaggregation Tax

- Centralized application
  - Redundant transfers vs. shared network



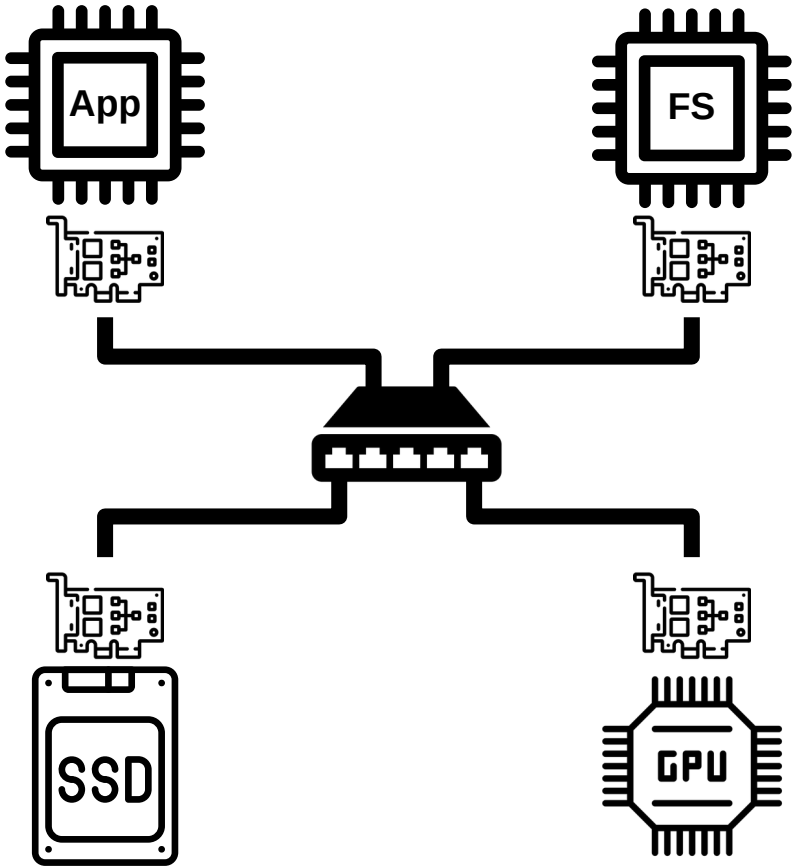
The Disaggregation Tax

- Centralized application
  - Redundant transfers vs. shared network



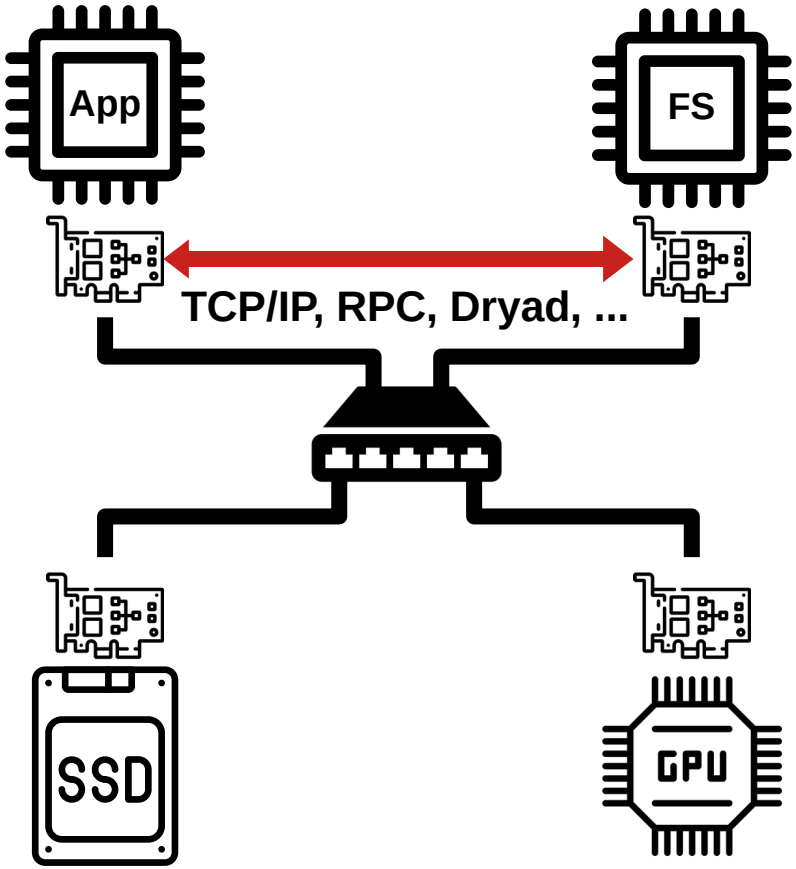
The Disaggregation Tax

- Centralized application
  - Redundant transfers vs. shared network
  - CPU-centric system



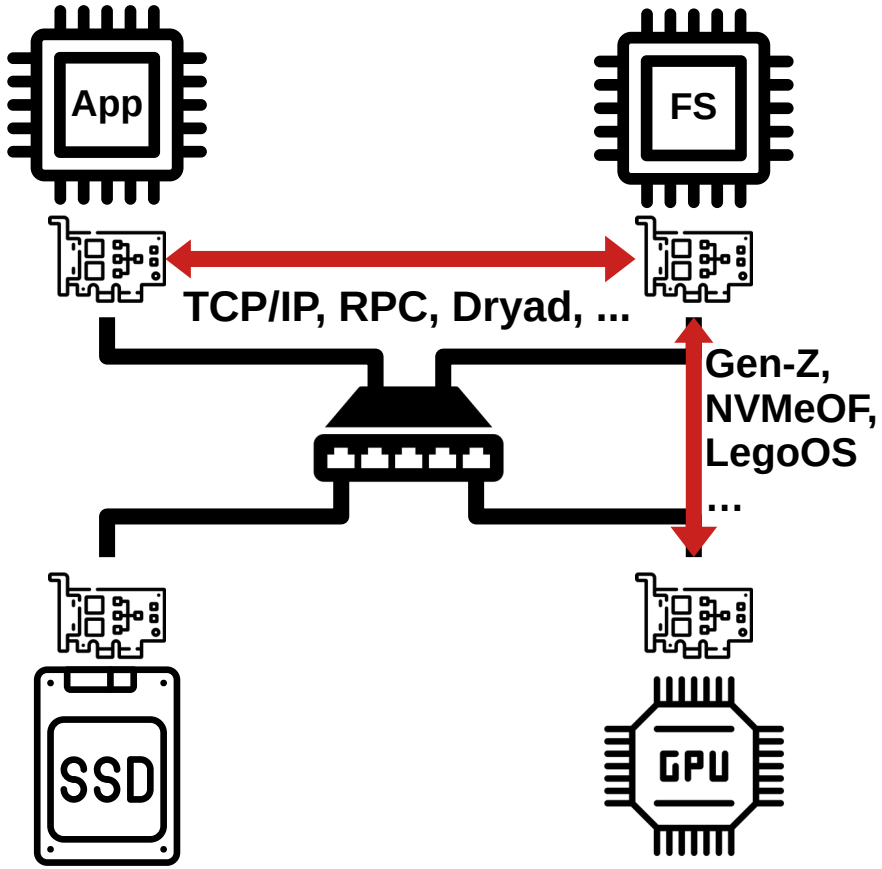
The Disaggregation Tax

- Centralized application
  - Redundant transfers vs. shared network
  - CPU-centric system



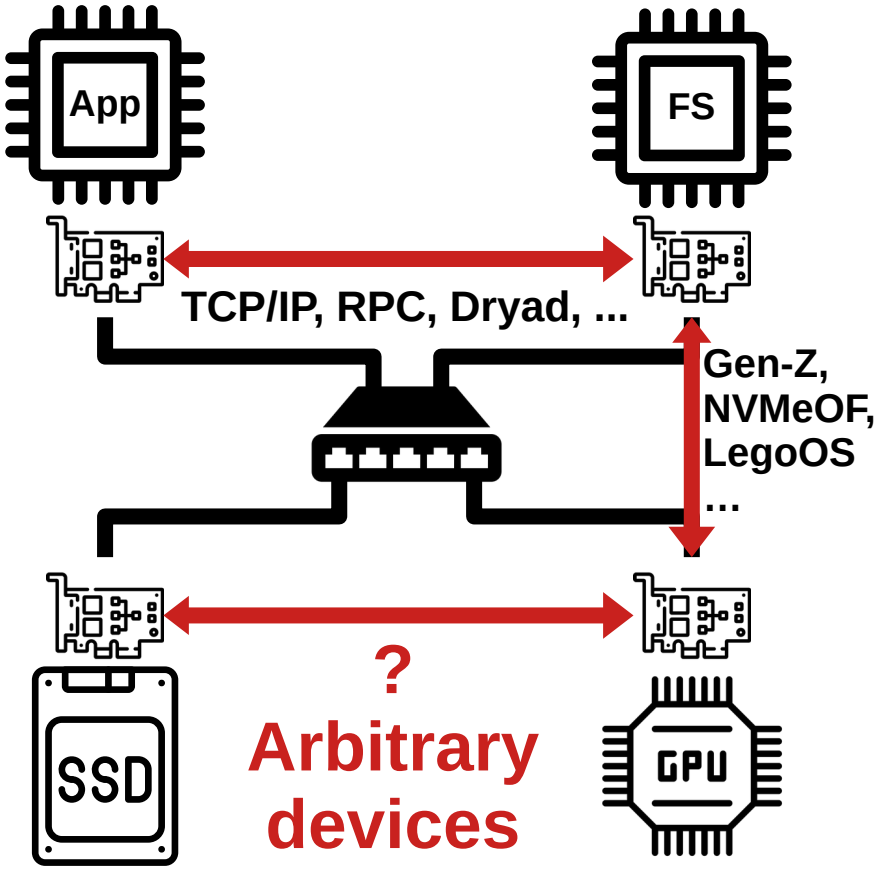
The Disaggregation Tax

- Centralized application
  - Redundant transfers vs. shared network
  - CPU-centric system



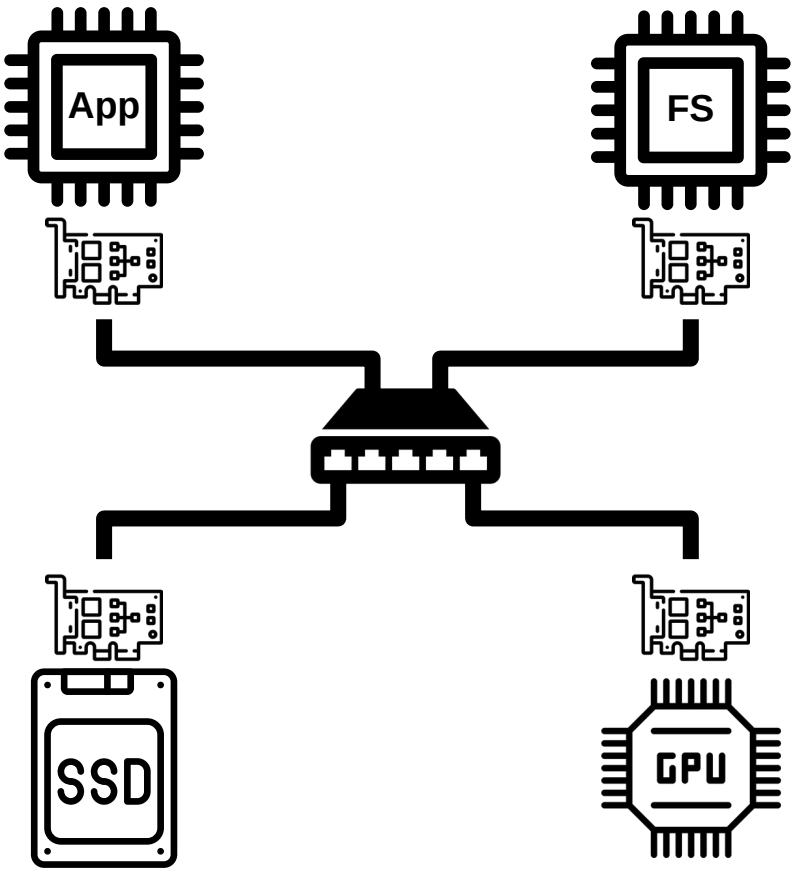
The Disaggregation Tax

- Centralized application
  - Redundant transfers vs. shared network
  - CPU-centric system



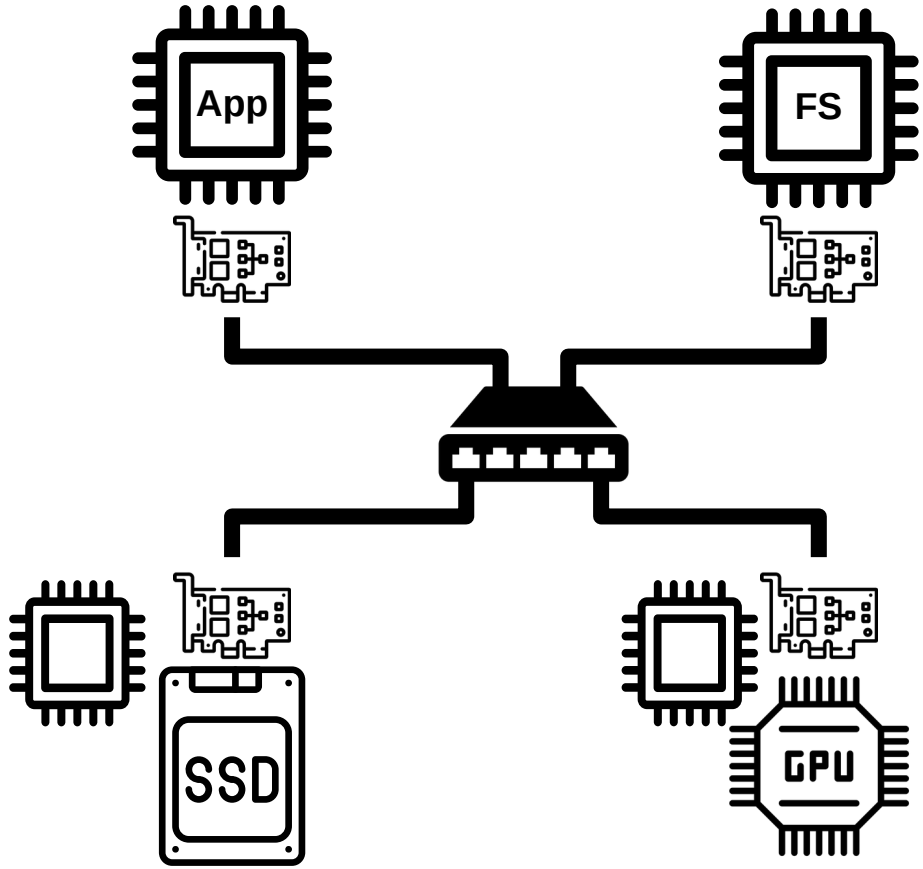
The Disaggregation Tax

- Centralized application
  - Redundant transfers vs. shared network
  - CPU-centric system
- Naive app. distribution
  - CPU over-provisioning vs. disaggregation



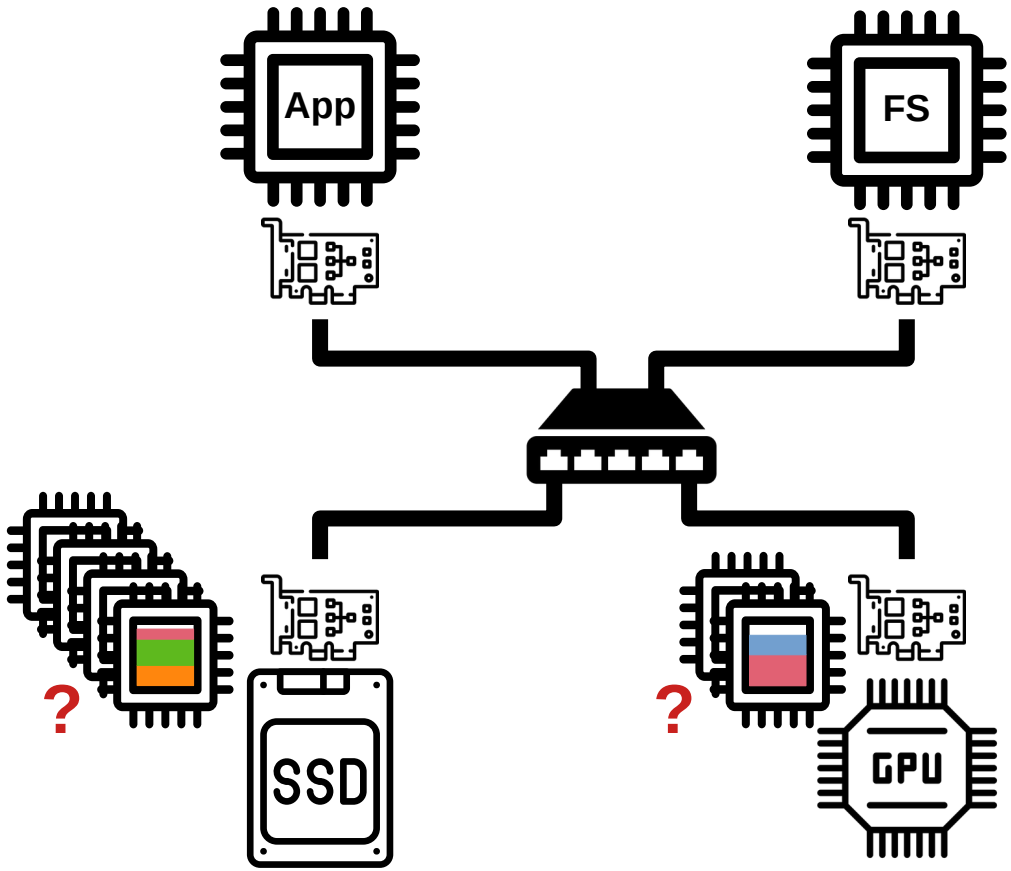
The Disaggregation Tax

- Centralized application
  - Redundant transfers vs. shared network
  - CPU-centric system
- Naive app. distribution
  - CPU over-provisioning vs. disaggregation

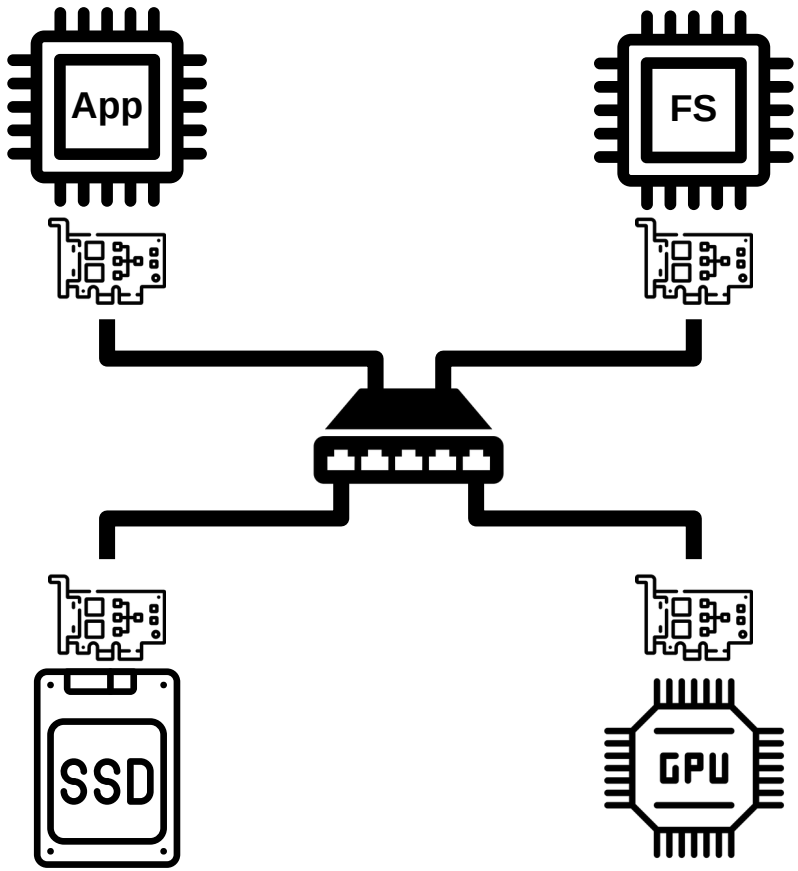


The Disaggregation Tax

- Centralized application
  - Redundant transfers vs. shared network
  - CPU-centric system
- Naive app. distribution
  - CPU over-provisioning vs. disaggregation

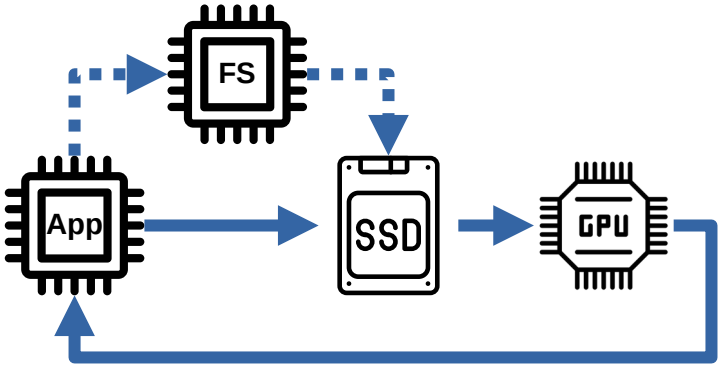


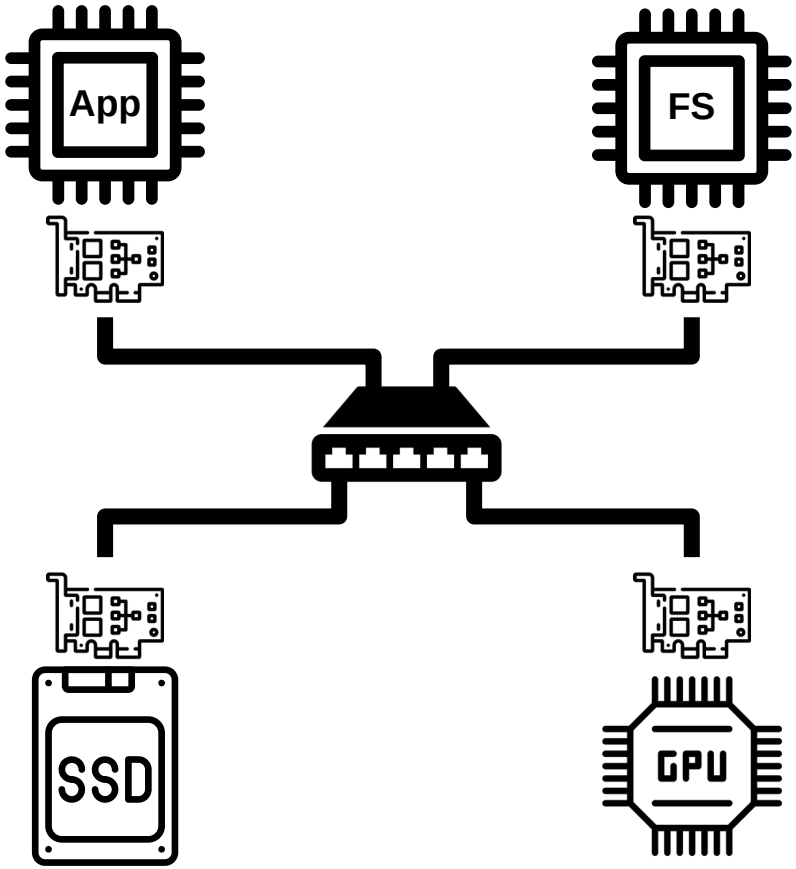
FractOS Goals



FractOS Goals

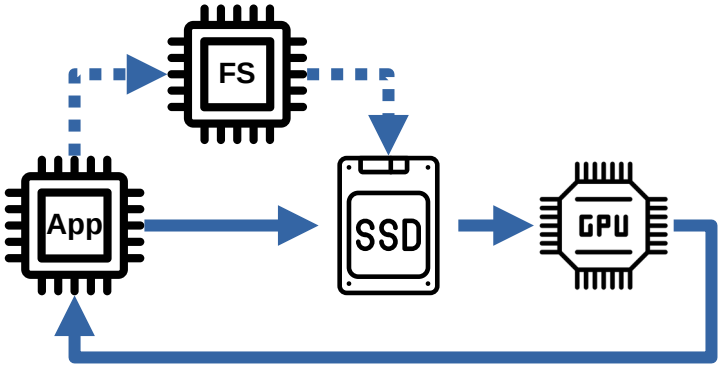
- Decentralized execution
  - Device-to-device, short-circuiting



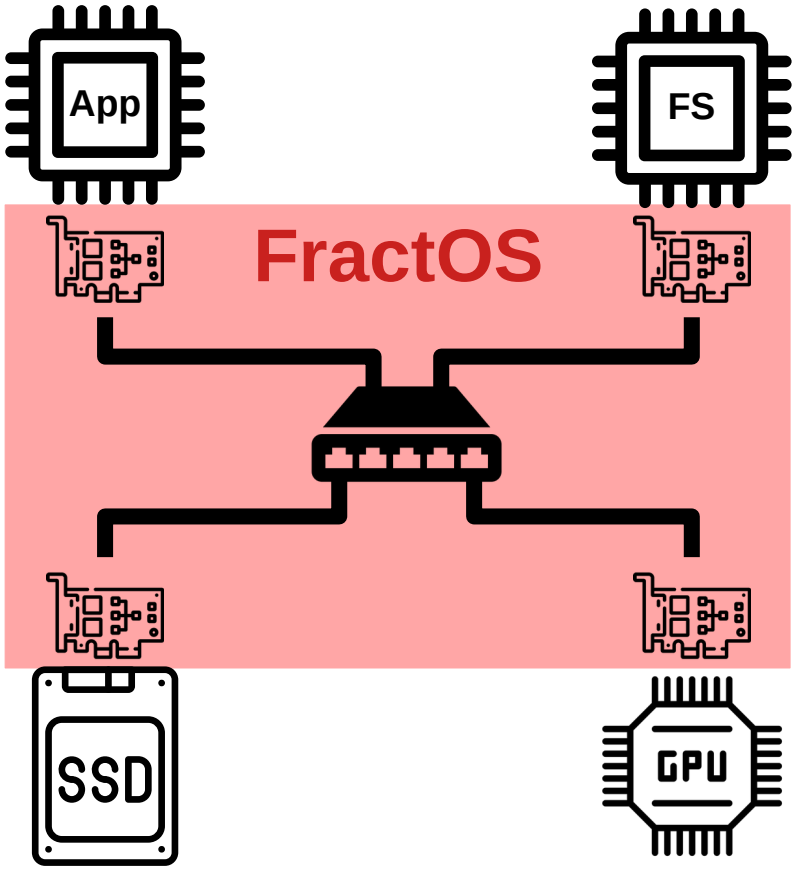


FractOS Goals

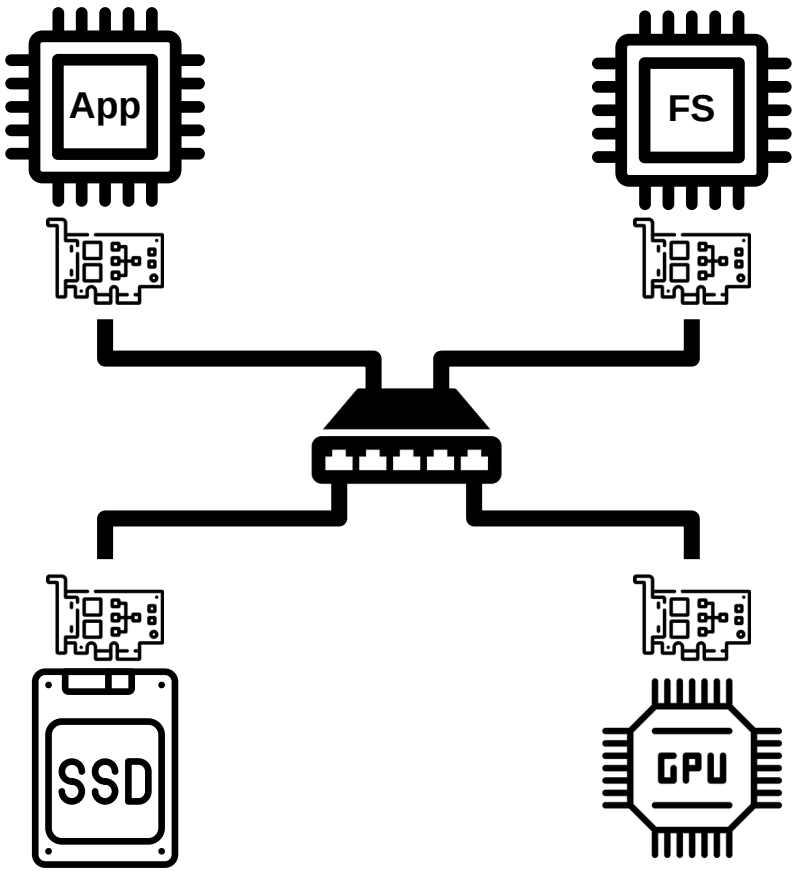
- Decentralized execution
  - Device-to-device, short-circuiting



- First-class devices
  - Abstractions for device-agnostic P2P data and control operations

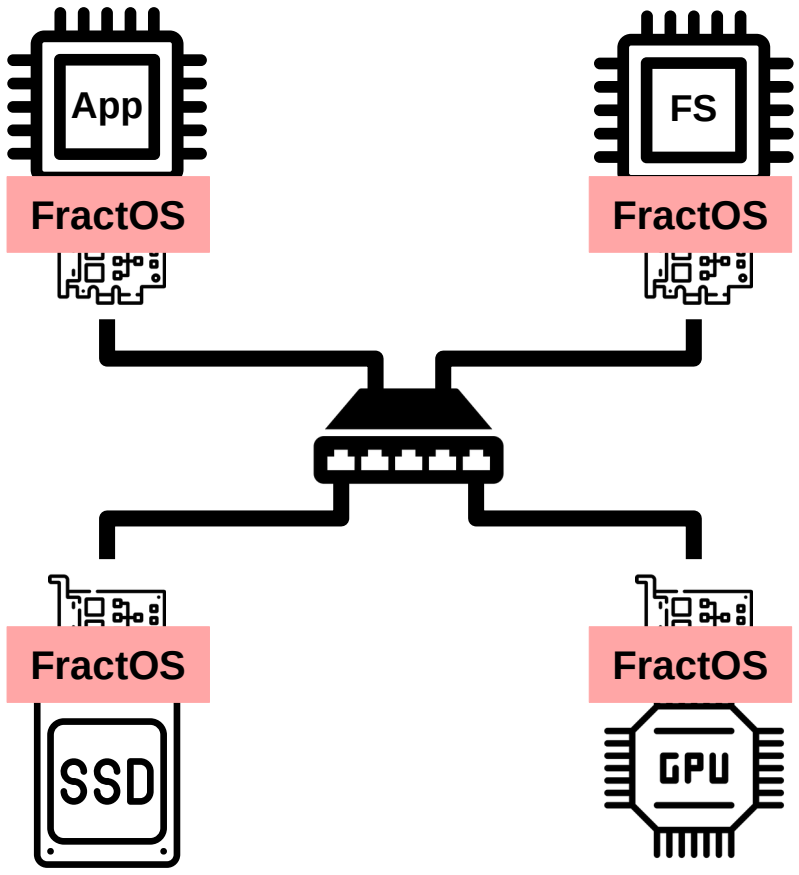


FractOS Design



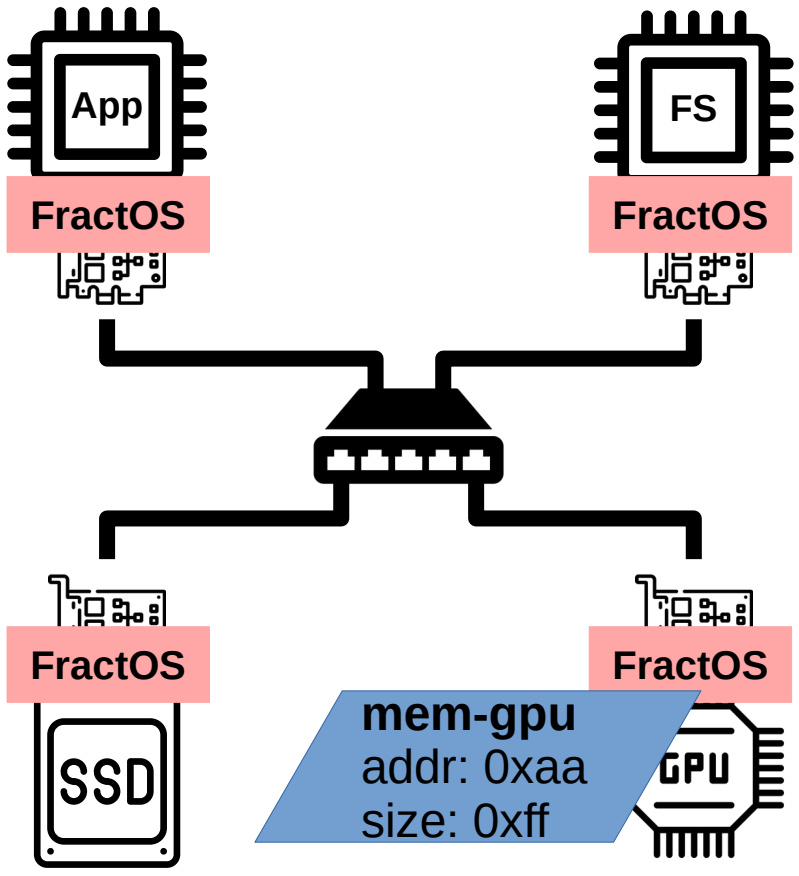
FractOS Design

- Distributed “micro-kernel” architecture



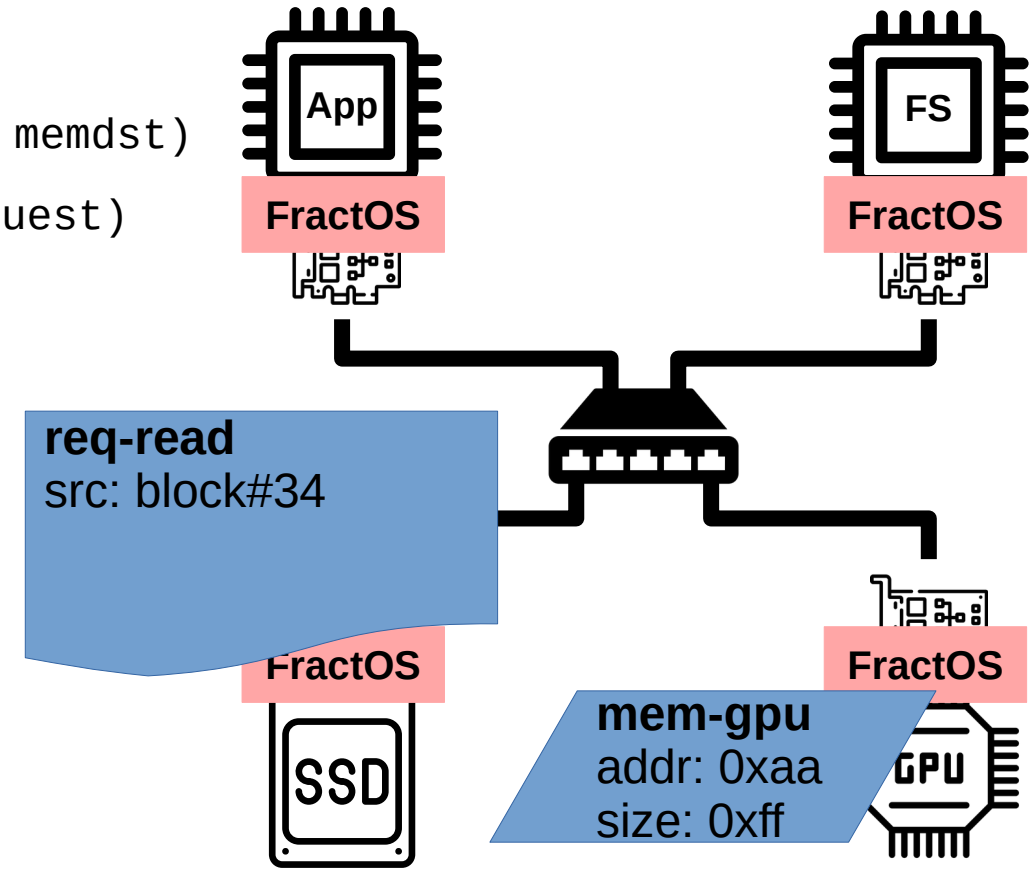
FractOS Design

- Distributed “micro-kernel” architecture
  - Data: *Memory* objects → `copy(memsrc, memdst)`



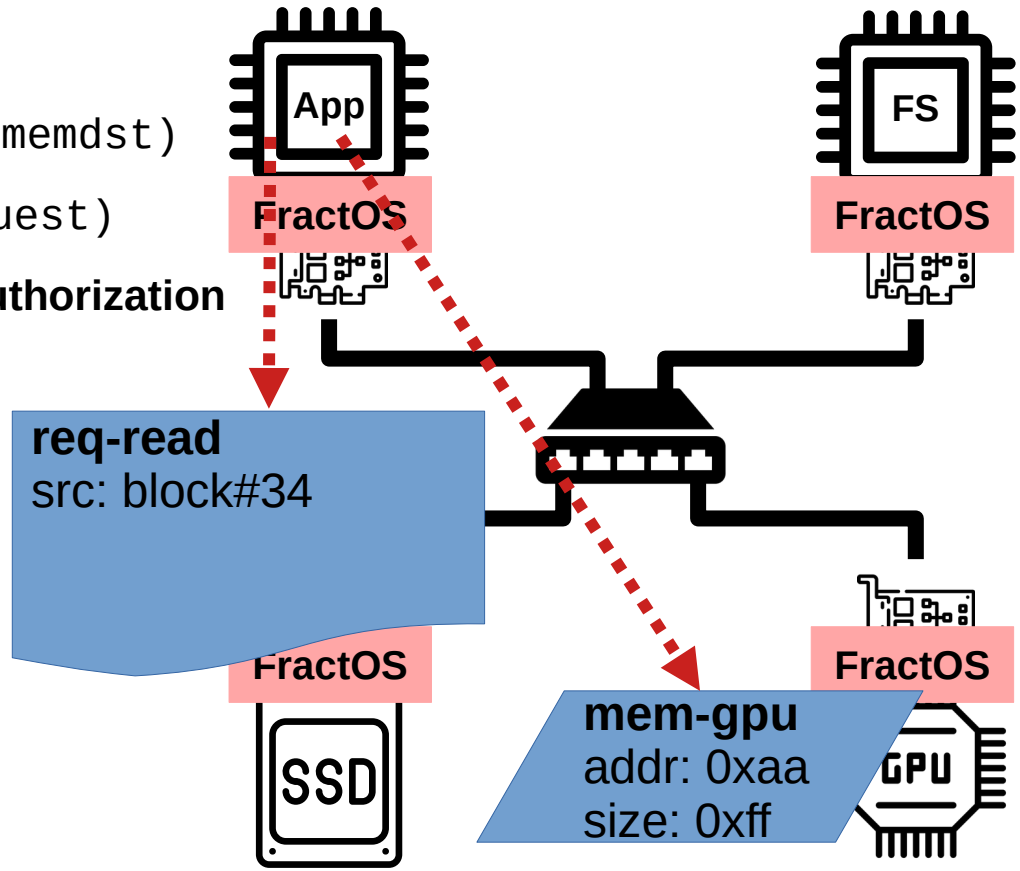
FractOS Design

- Distributed “micro-kernel” architecture
  - Data: *Memory* objects → `copy(memsrc, memdst)`
  - Control: *Request* objects → `invoke(request)`



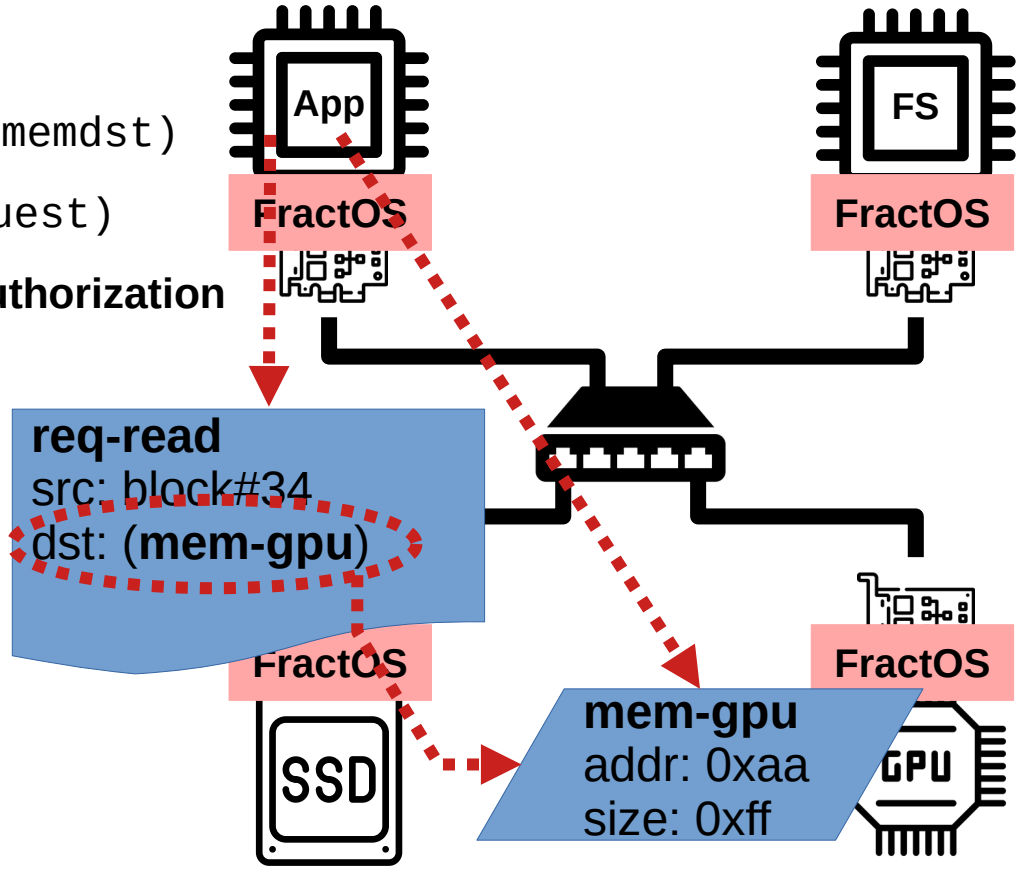
FractOS Design

- Distributed “micro-kernel” architecture
  - Data: *Memory* objects → `copy(memsrc, memdst)`
  - Control: *Request* objects → `invoke(request)`
  - Via capabilities: **dynamic + distributed authorization**



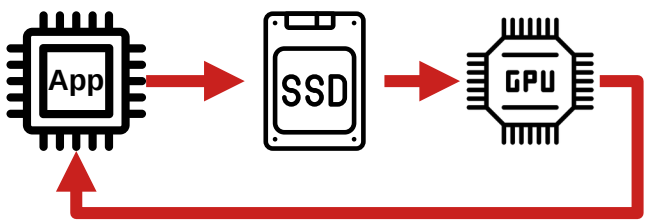
FractOS Design

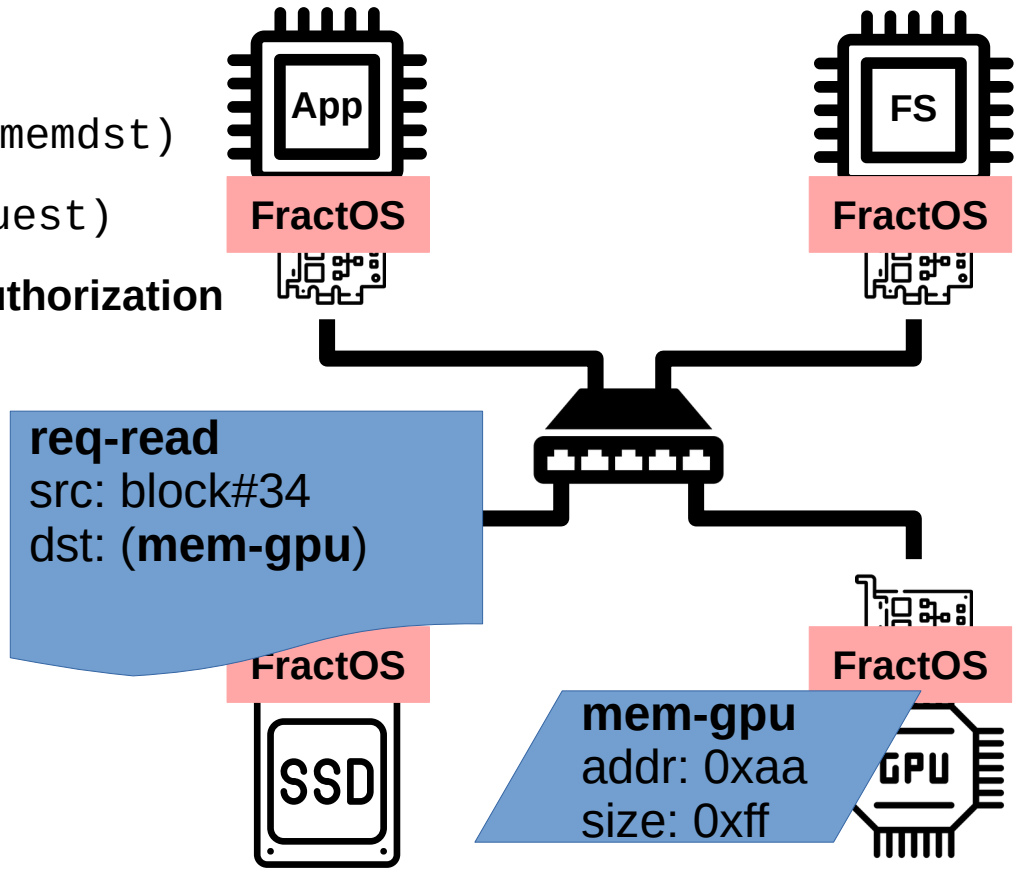
- Distributed “micro-kernel” architecture
  - Data: *Memory* objects → `copy(memsrc, memdst)`
  - Control: *Request* objects → `invoke(request)`
  - Via capabilities: **dynamic + distributed authorization**



FractOS Design

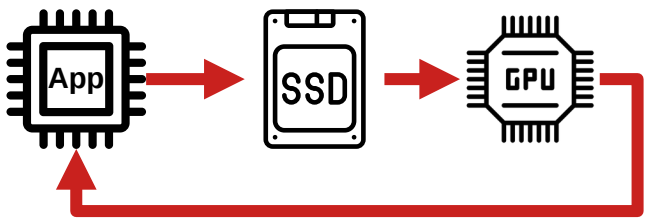
- Distributed “micro-kernel” architecture
  - Data: *Memory* objects → `copy(memsrc, memdst)`
  - Control: *Request* objects → `invoke(request)`
  - Via capabilities: **dynamic + distributed authorization**
- Decentralized execution

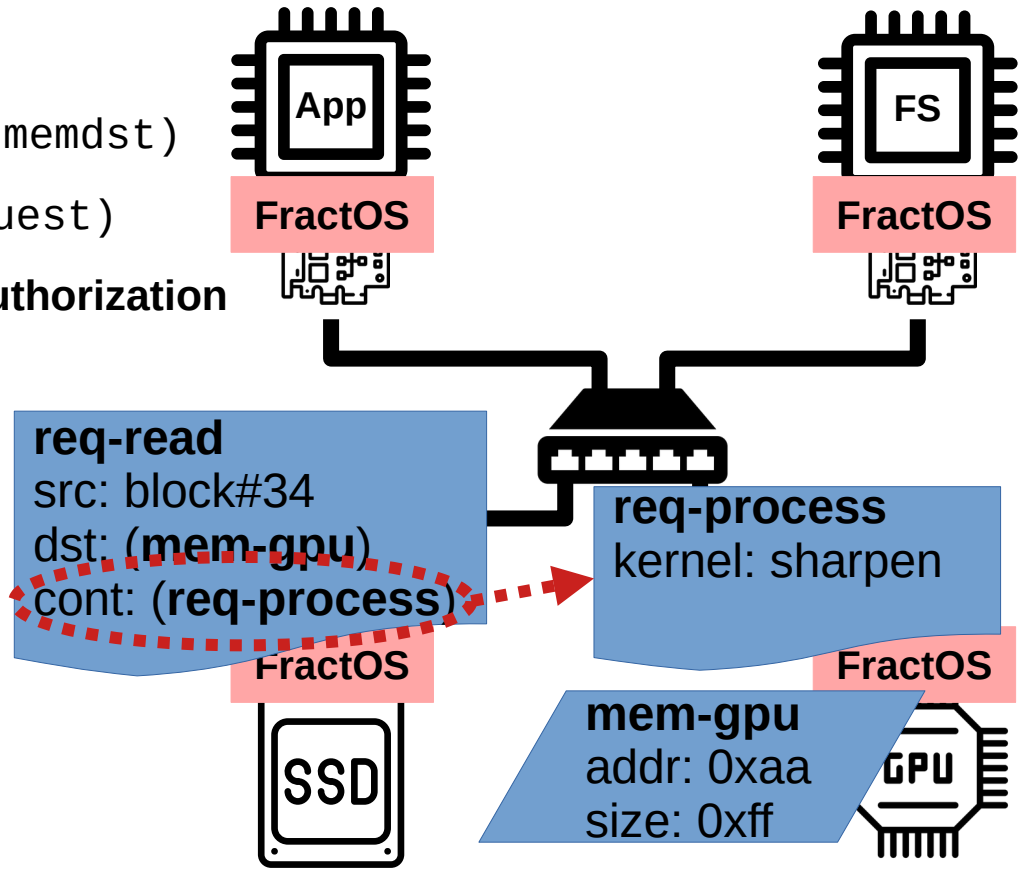




FractOS Design

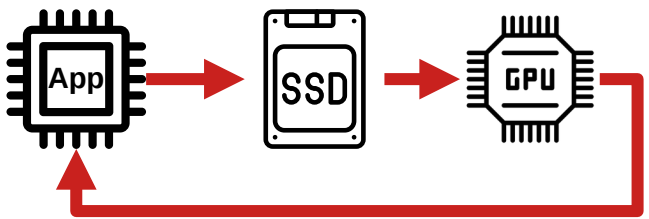
- Distributed “micro-kernel” architecture
  - Data: *Memory* objects → `copy(memsrc, memdst)`
  - Control: *Request* objects → `invoke(request)`
  - Via capabilities: **dynamic + distributed authorization**
- Decentralized execution

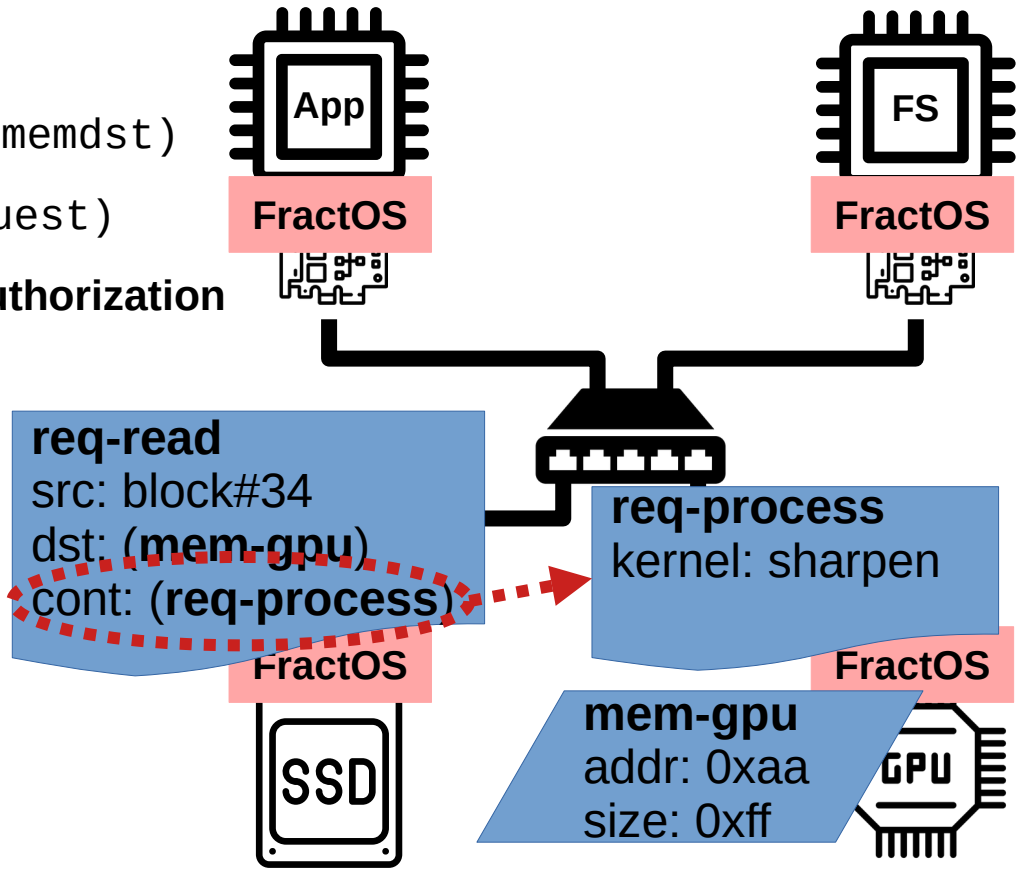




FractOS Design

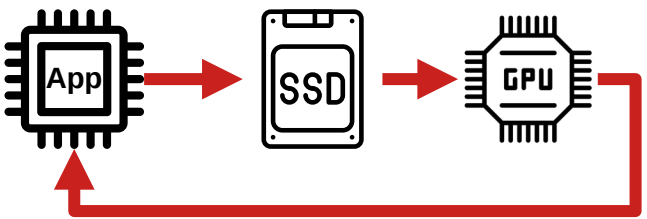
- Distributed “micro-kernel” architecture
  - Data: *Memory* objects → `copy(memsrc, memdst)`
  - Control: *Request* objects → `invoke(request)`
  - Via capabilities: **dynamic + distributed authorization**
- Decentralized execution
  - Requests as “continuations”
  - From RPCs to distributed dataflow

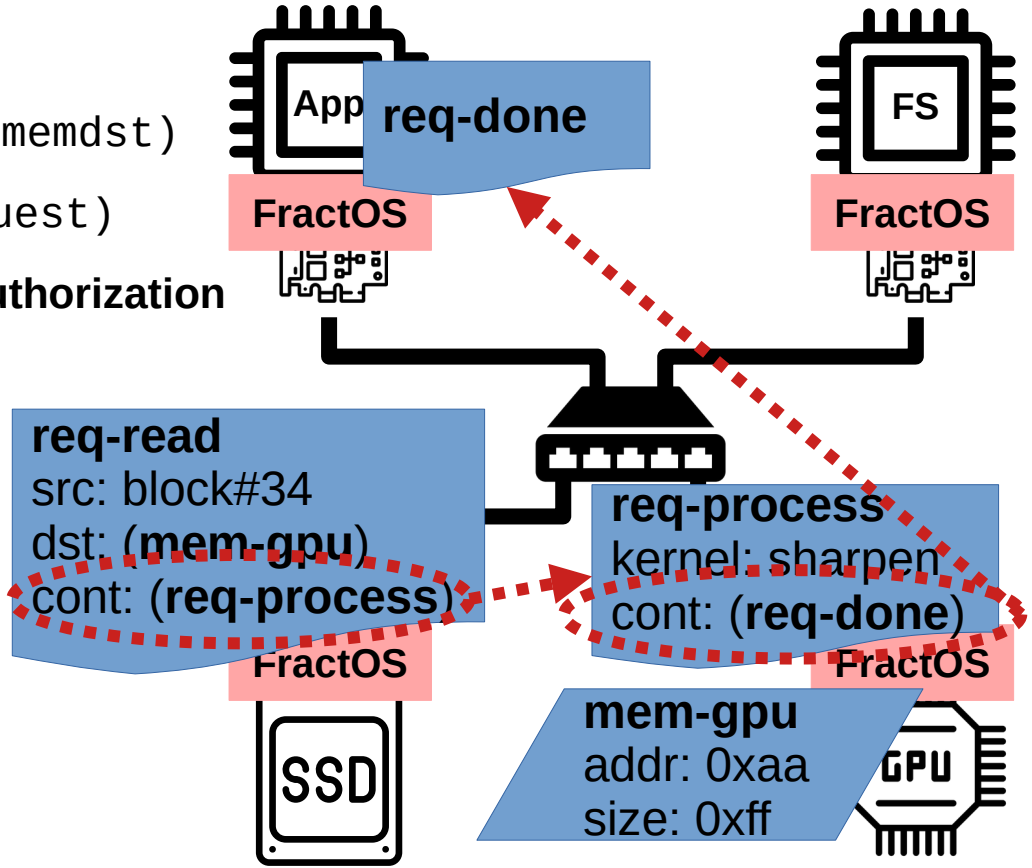




FractOS Design

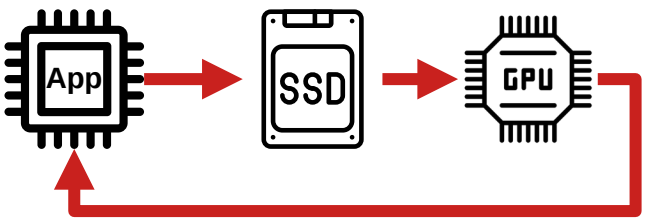
- Distributed “micro-kernel” architecture
  - Data: *Memory* objects → `copy(memsrc, memdst)`
  - Control: *Request* objects → `invoke(request)`
  - Via capabilities: **dynamic + distributed authorization**
- Decentralized execution
  - Requests as “continuations”
  - From RPCs to distributed dataflow

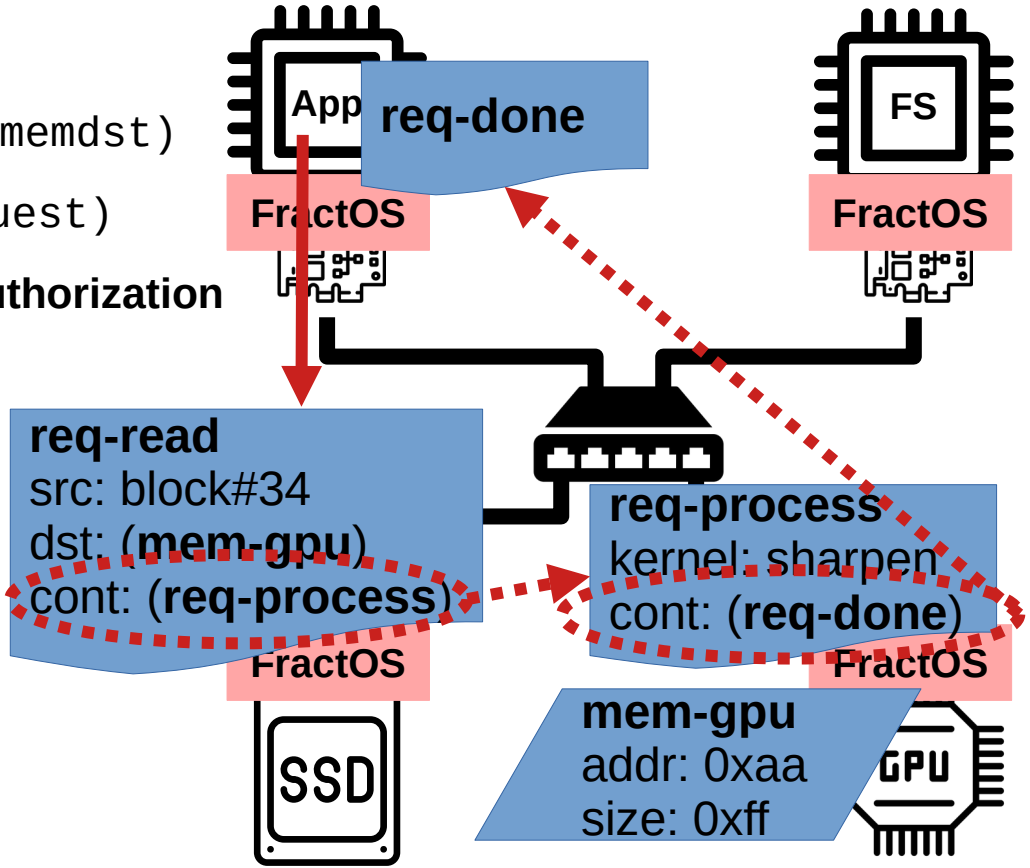




FractOS Design

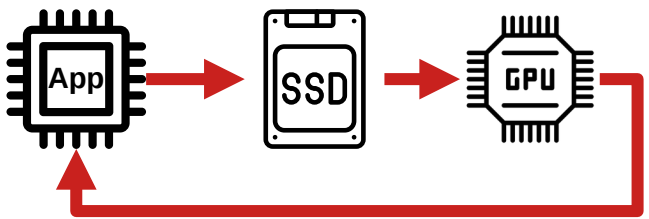
- Distributed “micro-kernel” architecture
  - Data: *Memory* objects → `copy(memsrc, memdst)`
  - Control: *Request* objects → `invoke(request)`
  - Via capabilities: **dynamic + distributed authorization**
- Decentralized execution
  - Requests as “continuations”
  - From RPCs to distributed dataflow

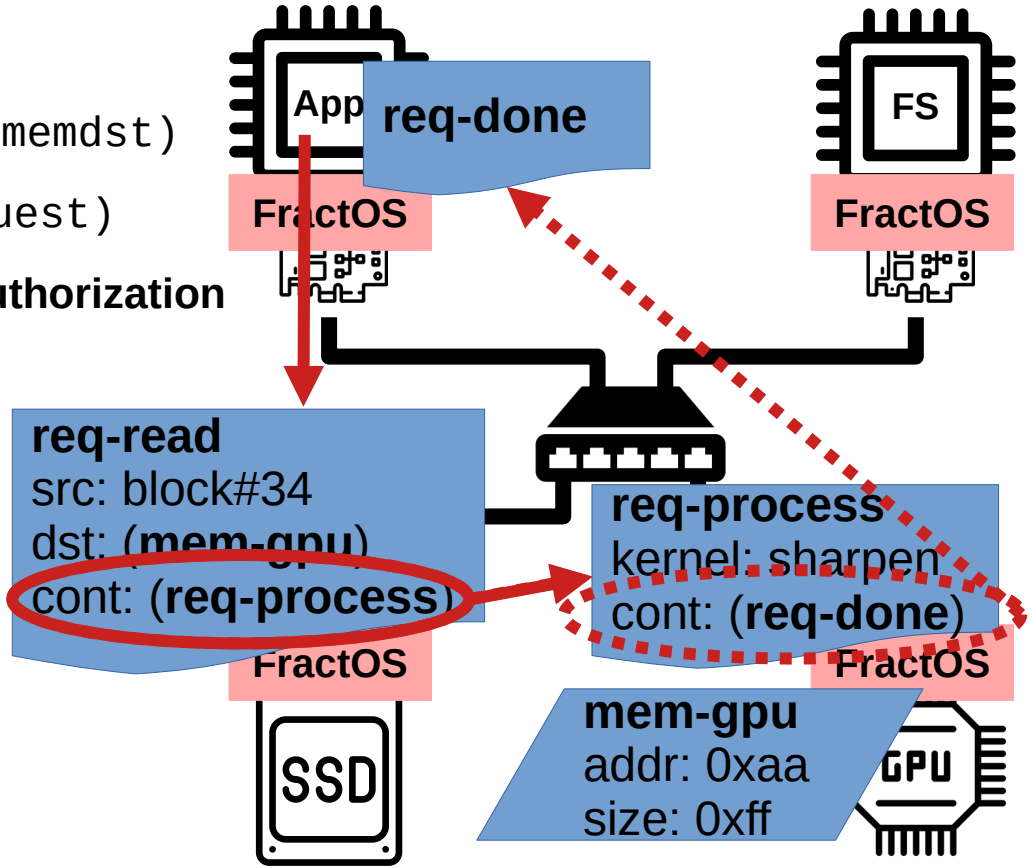




FractOS Design

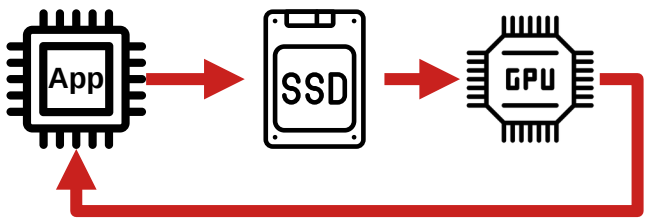
- Distributed “micro-kernel” architecture
  - Data: *Memory* objects → `copy(memsrc, memdst)`
  - Control: *Request* objects → `invoke(request)`
  - Via capabilities: **dynamic + distributed authorization**
- Decentralized execution
  - Requests as “continuations”
  - From RPCs to distributed dataflow

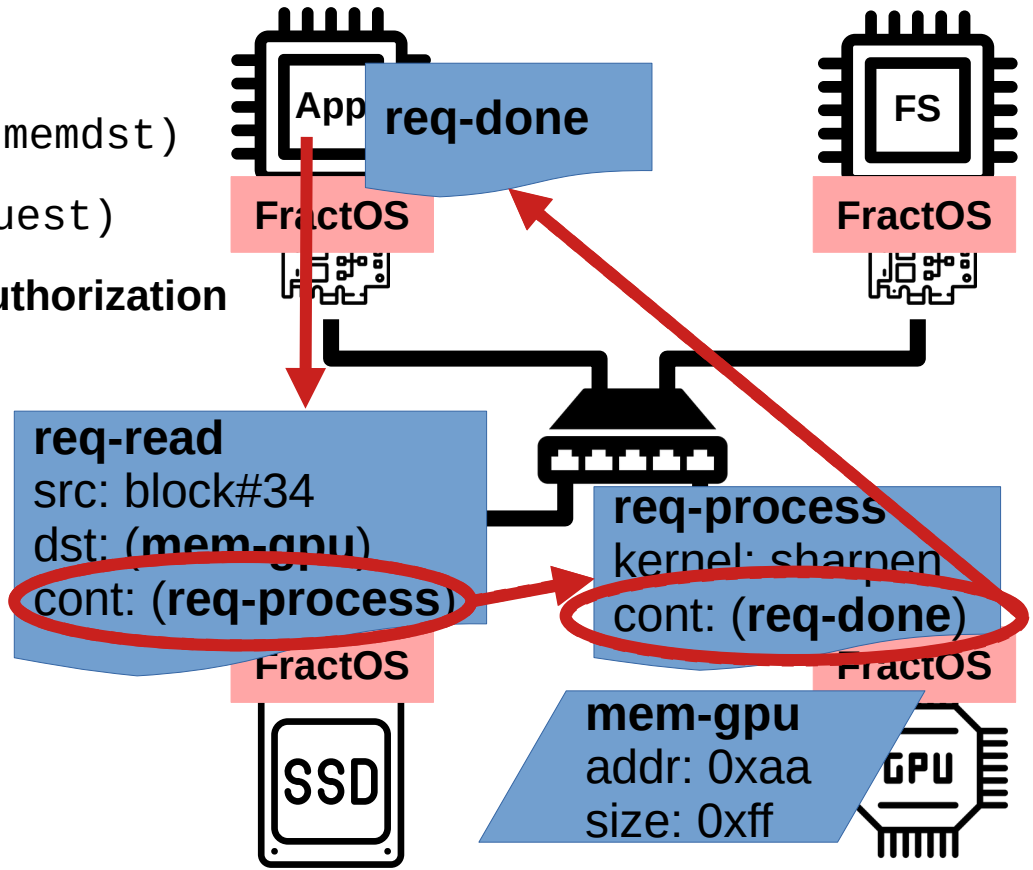




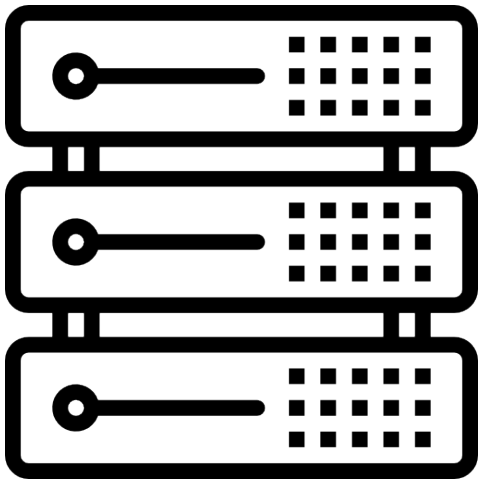
FractOS Design

- Distributed “micro-kernel” architecture
  - Data: *Memory* objects → `copy(memsrc, memdst)`
  - Control: *Request* objects → `invoke(request)`
  - Via capabilities: **dynamic + distributed authorization**
- Decentralized execution
  - Requests as “continuations”
  - From RPCs to distributed dataflow



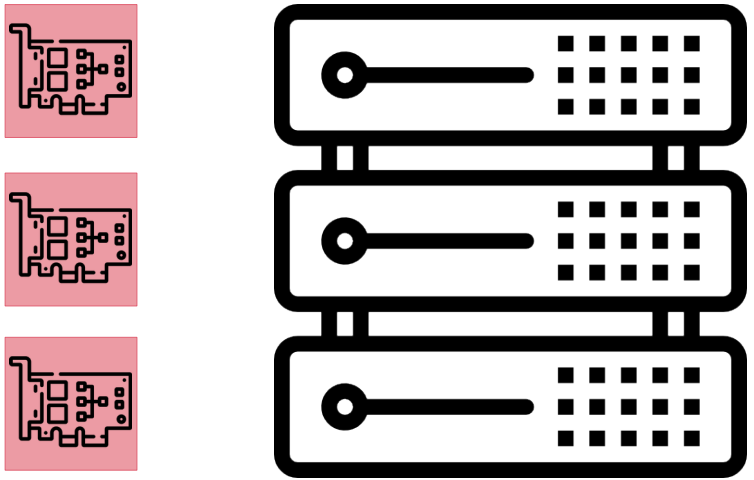


FractOS and Data Center Architecture



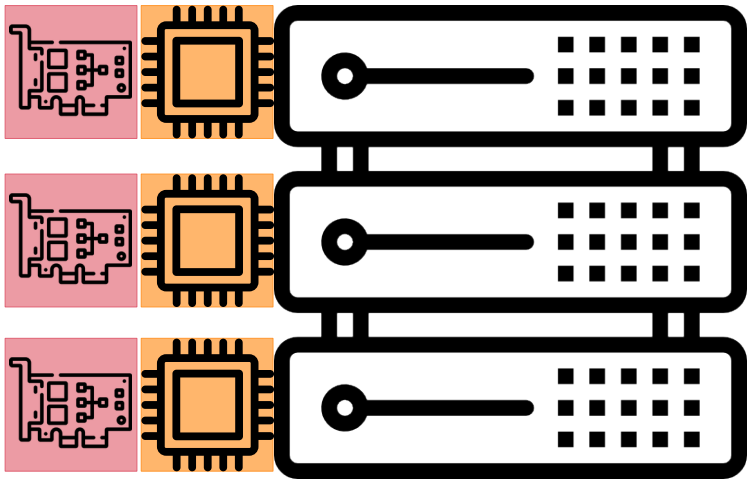
FractOS and Data Center Architecture

- Per-node FractOS Controller
  - Trusted “micro-kernel” instance
  - Serves **FractOS Processes** via asynchronous req/resp queues



FractOS and Data Center Architecture

- Per-node FractOS Controller
  - Trusted “micro-kernel” instance
  - Serves **FractOS Processes** via asynchronous req/resp queues
- Per-node device service adaptor
  - FractOS “device driver” Process
  - CPUs with Linux + existing drivers
  - **No client application code**



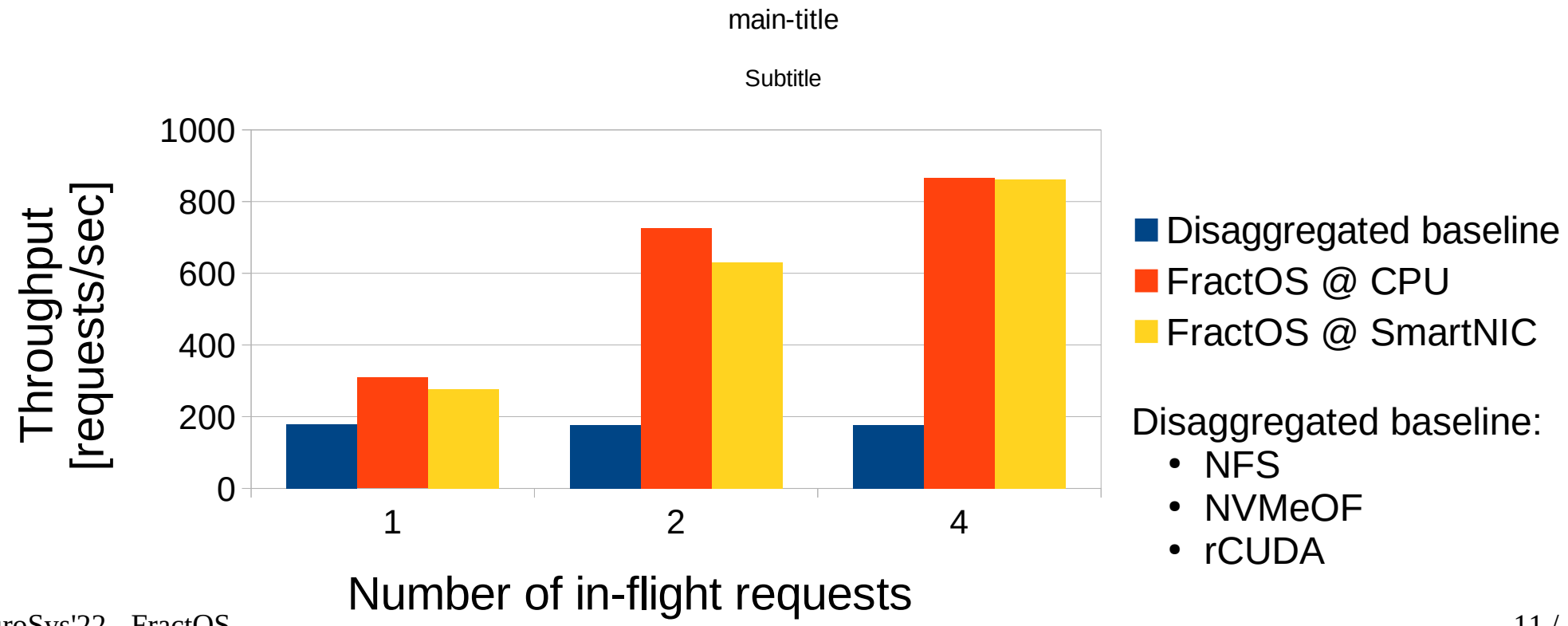
req-read
src: block#34
dst: (mem-gpu)
cont: (req-process)

cmd: NVME_CMD_READ
block: 34
buffer: 0xcafe
…

copy(0xcafe, req.dst)
invoke(req.cont)

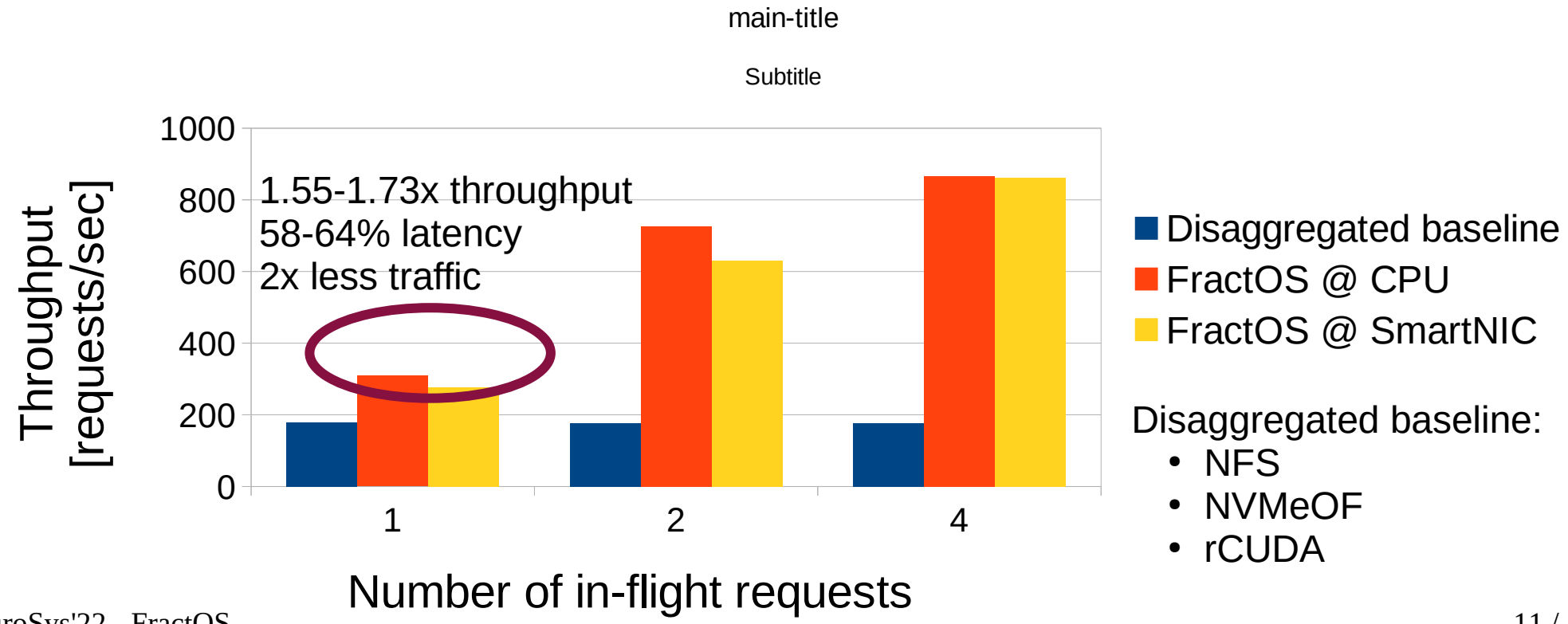
Evaluation

- Face-verification application (similar to example)



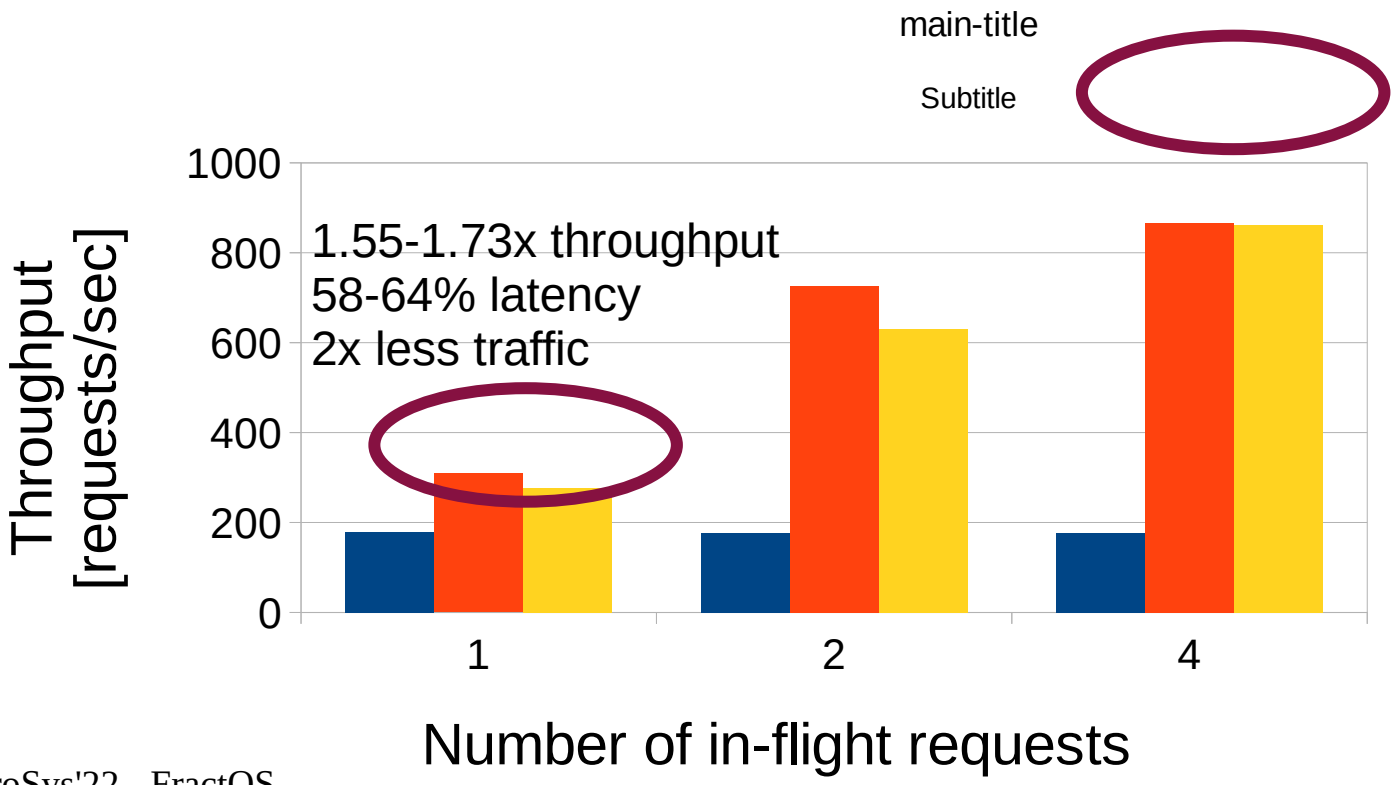
Evaluation

- Face-verification application (similar to example)



Evaluation

- Face-verification application (similar to example)



FractOS saturates GPU (vs. rCUDA: centralized + multiple messages)

- Disaggregated baseline
- FractOS @ CPU
- FractOS @ SmartNIC

Disaggregated baseline:

- NFS
- NVMeOF
- rCUDA

Conclusions

- A path to disaggregation-native systems
  - Device-to-device operations → latency speedups + network traffic reduction
  - Decoupled Controller and Process →CPUs/SmartNICs, FractOS-native devices, …
- More details on the paper
  - Secure and dynamic composition of third-party services
  - Application and deployment-specific optimizations
  - Distributed capability management algorithms
  - Micro-benchmark and application evaluation… and more!

A small step within a wide area of research, let's talk!

Lluís Vilanova
<vilanova@imperial.ac.uk>

https://lsds.doc.ic.ac.uk/projects/fractos