Toast

Heterogeneous Memory Management

Maurice Bailleu, Dimitrios Stavrakakis, Rodrigo Rocha,
Soham Chakraborty, Deepak Garg, Pramod Bhatotia

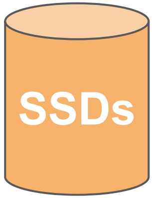
HUAWEI
THE UNIVERSITY of EDINBURGH
Technische Universität München
TU Delft
MAX PLANCK INSTITUTE FOR SOFTWARE SYSTEMS

Heterogeneity in the cloud

CPU architecture

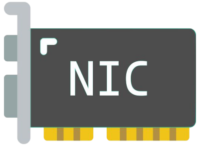
intel
AMD
ARM
RISC-V

Storage



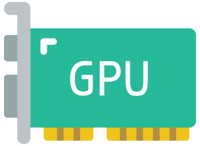
pmem.io
SPDK

Networking



DPDK

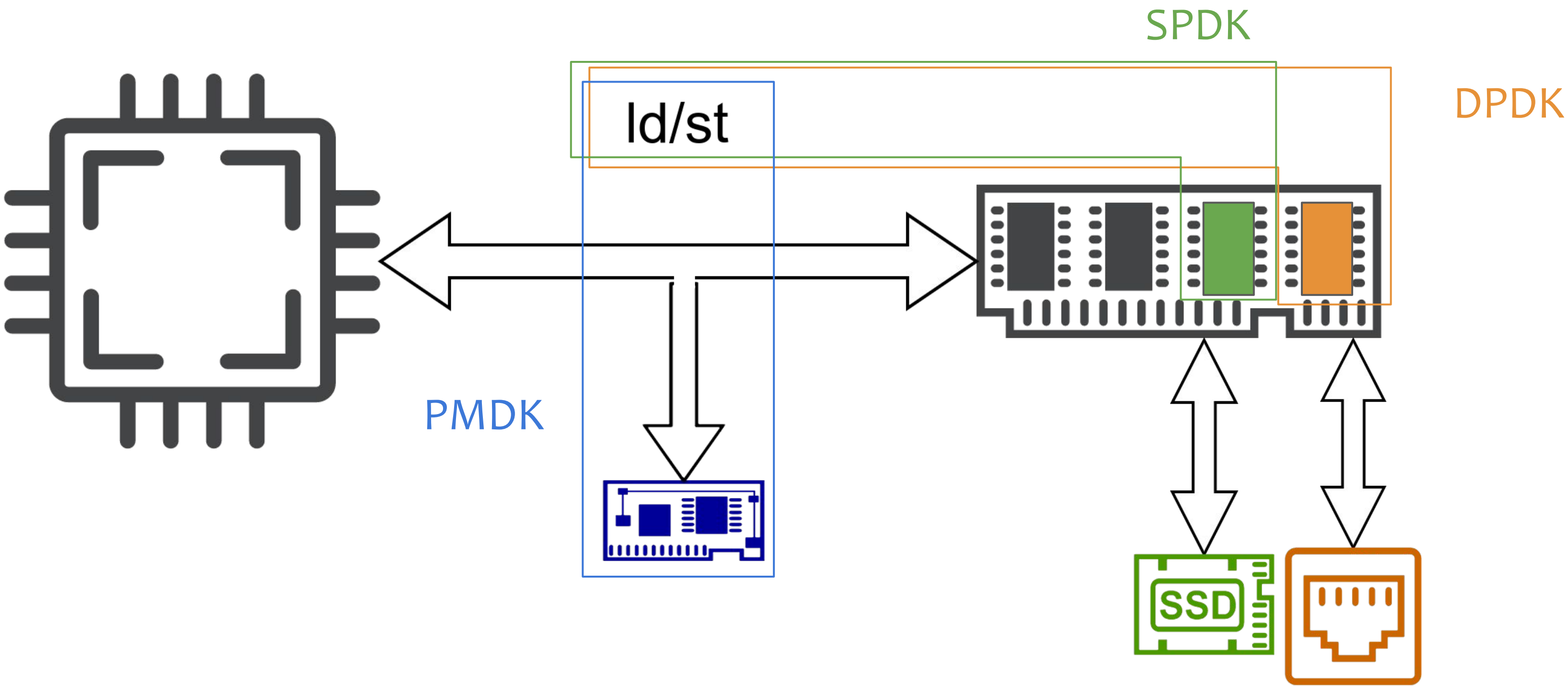
Accelerators



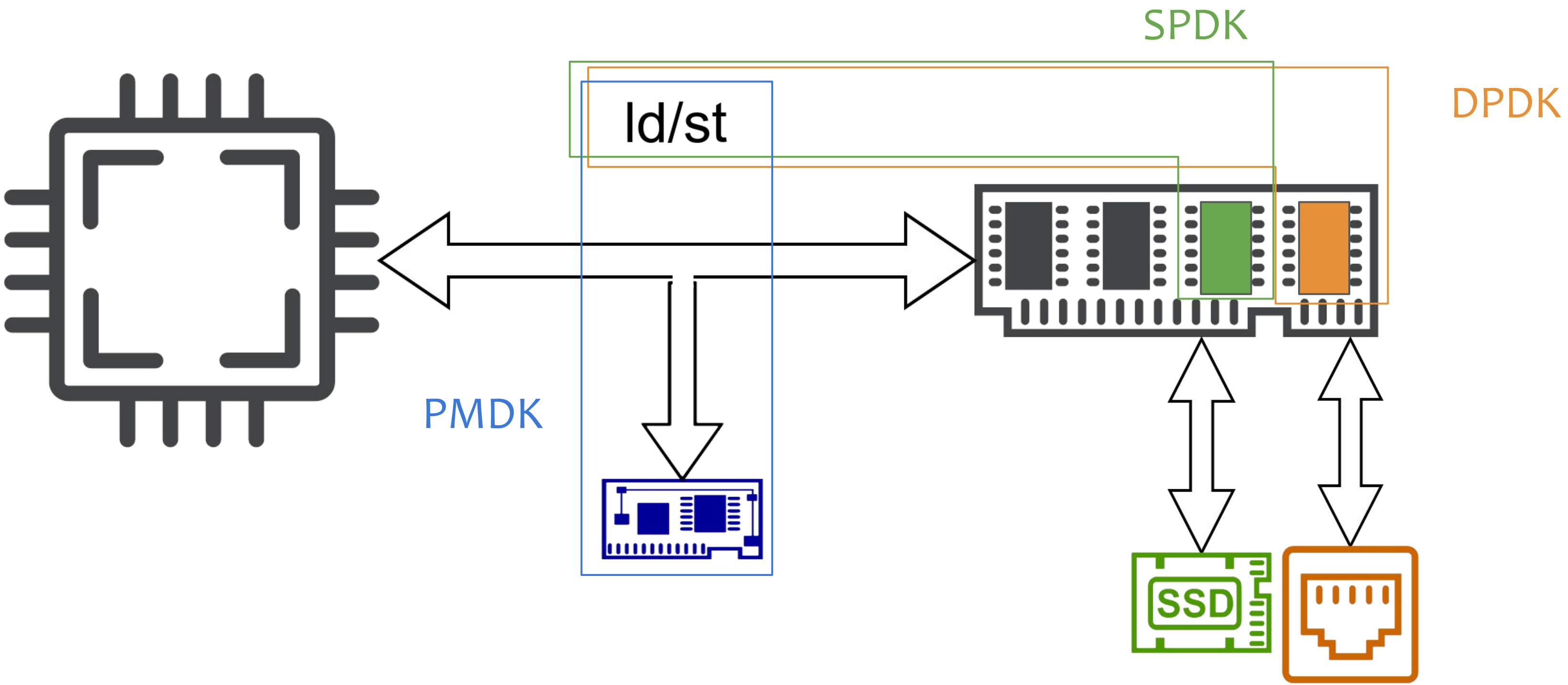
Confidential computing

intel SGX
AMD SEV-SNP

Problem statement



Problem statement



Heterogeneous memory does not offer a **unified way** to access it.

Problems with heterogeneous memory

TEEs double the amount of memory that you need to differentiate

Deal with different memory types in different ways

Difficult to upgrade between different technologies

Our proposal

Toast: A Heterogeneous Memory Management System

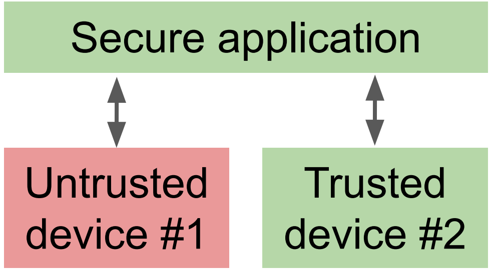
Design goals:

- Programmability
- Portability
- Performance
- Protection

Outline

- ~~Motivation~~
- Design
  - Design challenges
  - Programming model
  - Compiler
  - Protection Library
  - Overview
- Evaluation

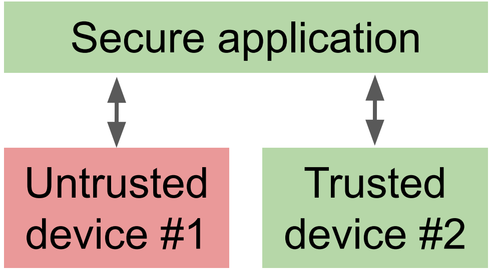
Design challenges

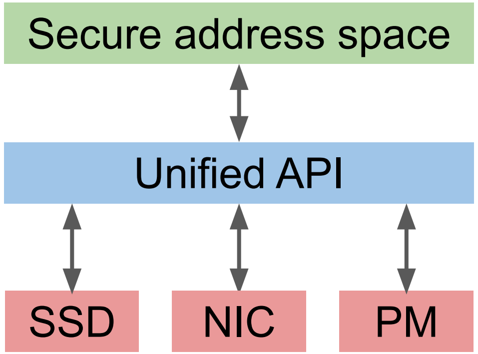


<u>#1 Memory region</u>

How to manage multiple memory regions?

Design challenges

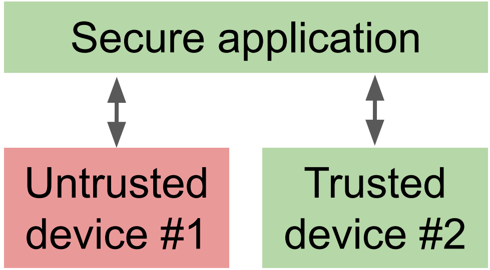




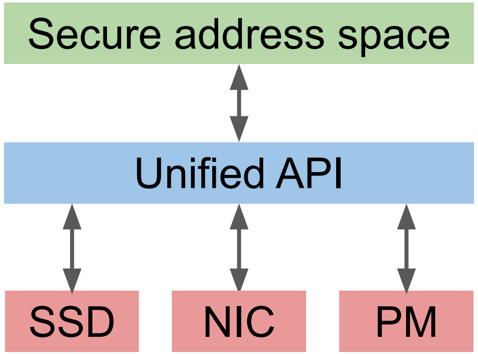
<u>#1 Memory region</u>
How to manage multiple memory regions?

<u>#2 Heterogeneity</u>
How to deal with different memory types?

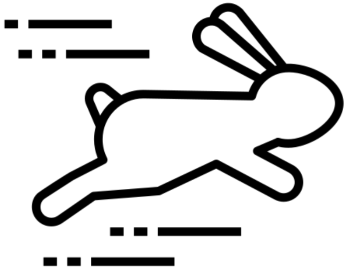
Design challenges



#1 Memory region
How to manage multiple memory regions?



#2 Heterogeneity
How to deal with different memory types?



#3 Performance
How to maximize the performance?

Example

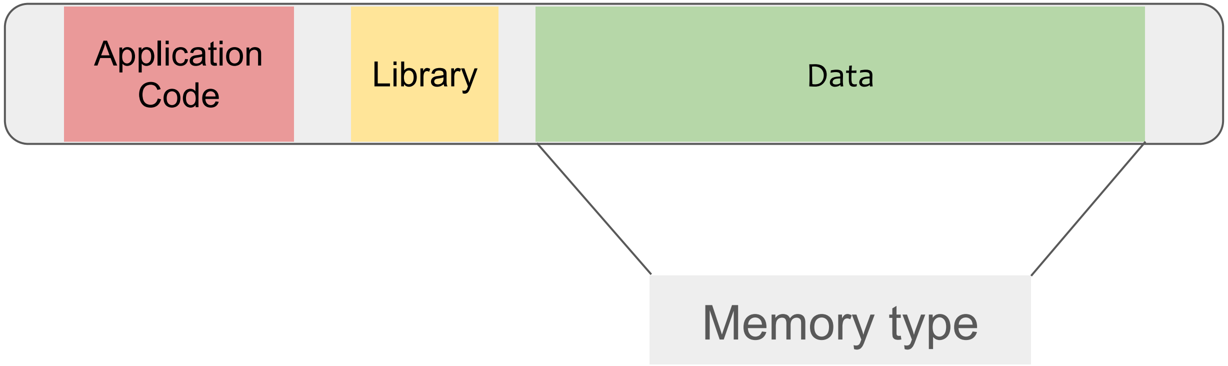
RDMA:

```
network_loop:
    //Waiting and receiving data
    poll(rx)
    char * buf = get_buf(rx)
    process(*buf)
    char * extra = next_free_buf()
    swap(buf, extra)
```

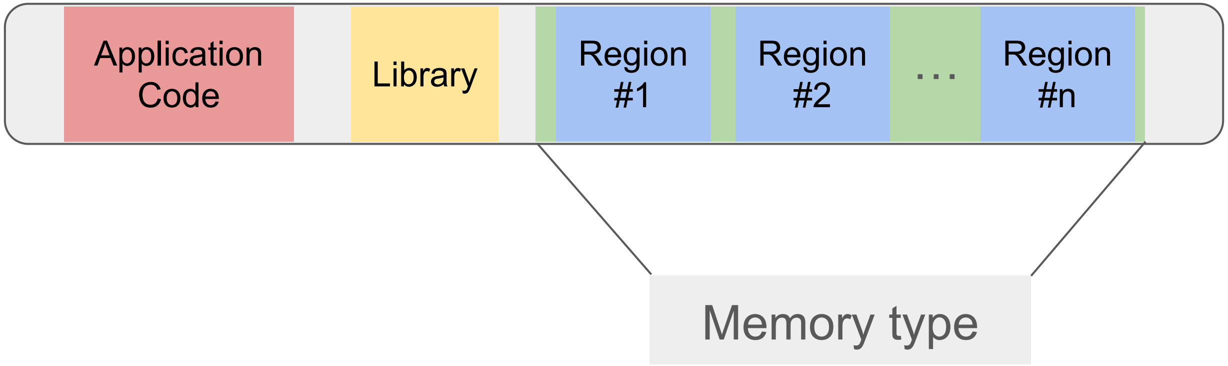
Toast:

```
network_loop:
    //Waiting and receiving data
    [[“toast::net”]] char * buf = get_buf(rx)
    process(*buf)
```

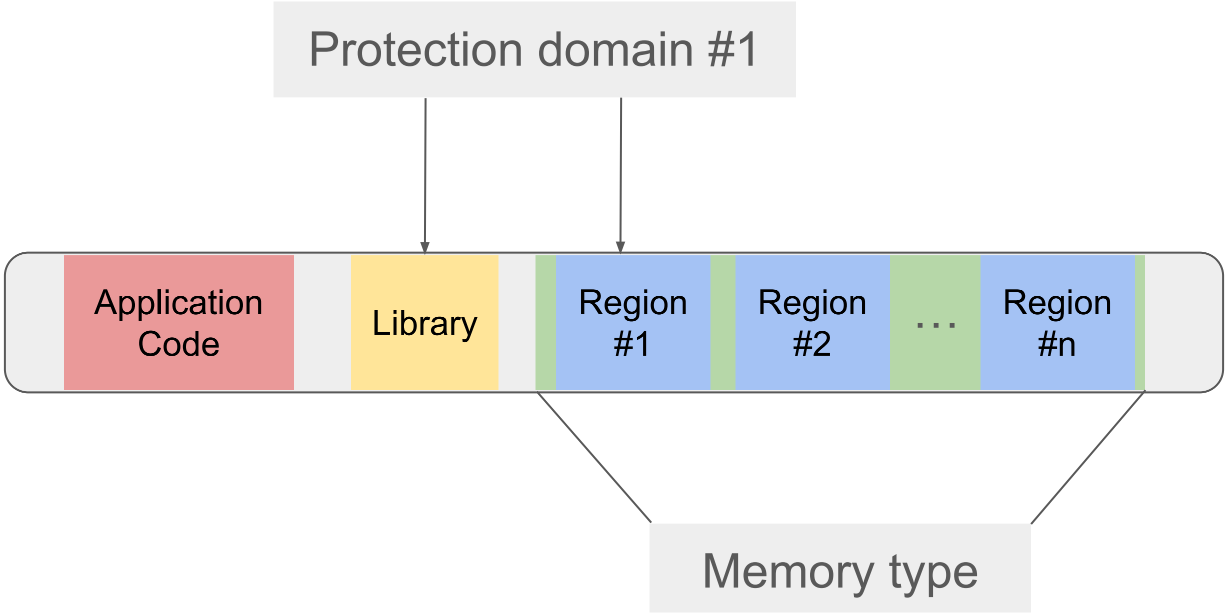
Design - Programming model



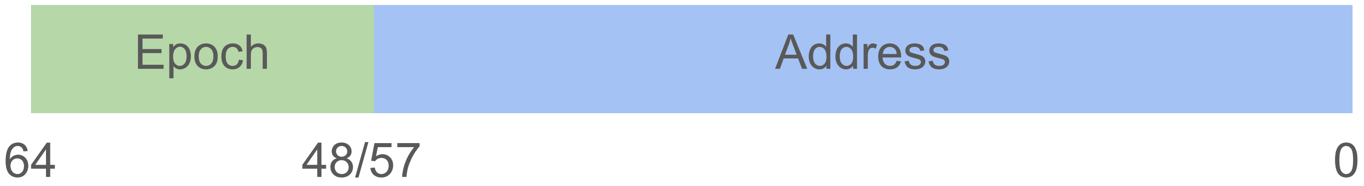
Design - Programming model



Design - Programming model

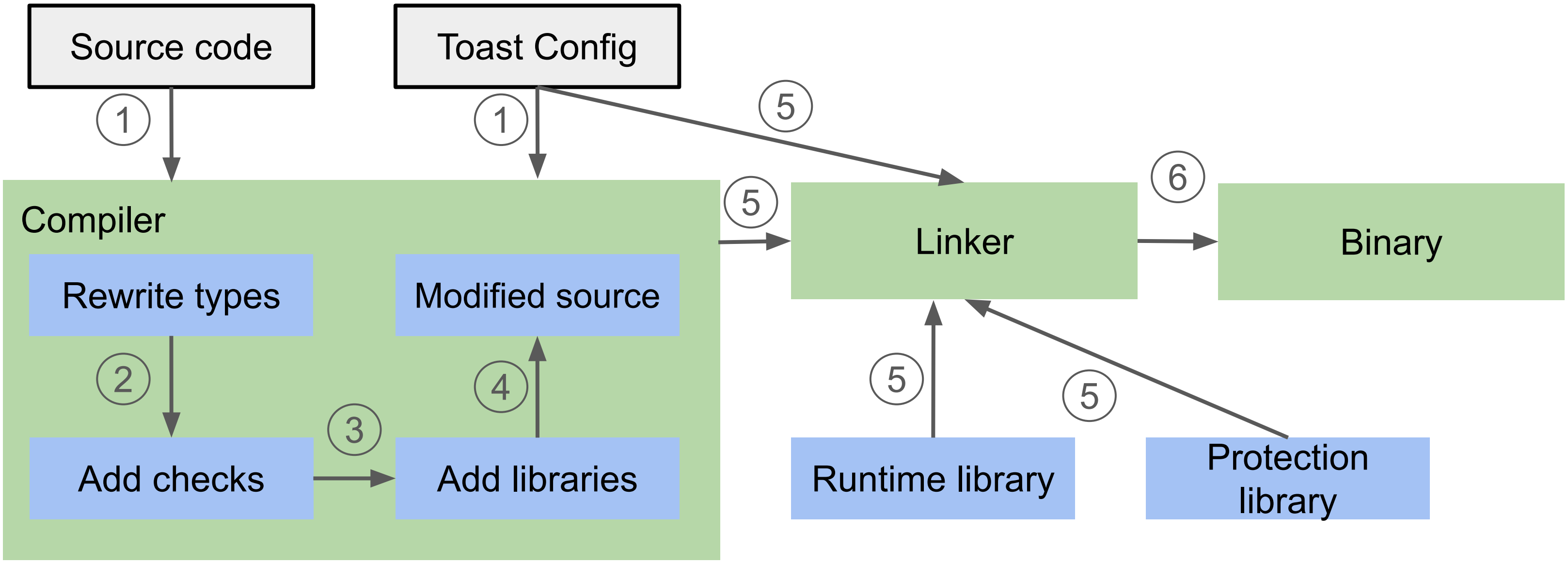


Design - ToastPtr

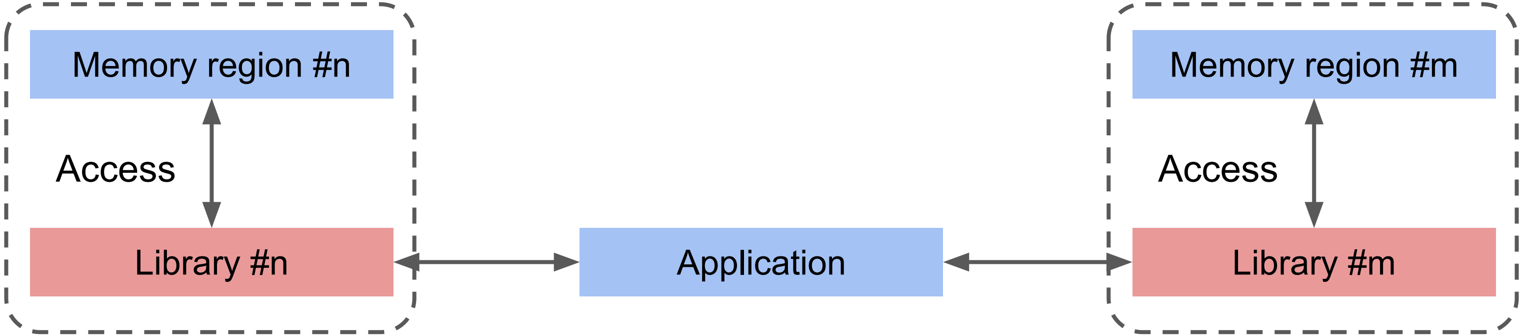


- Pointer type has memory type information
- Epoch is used for revocation
- System pointer sized

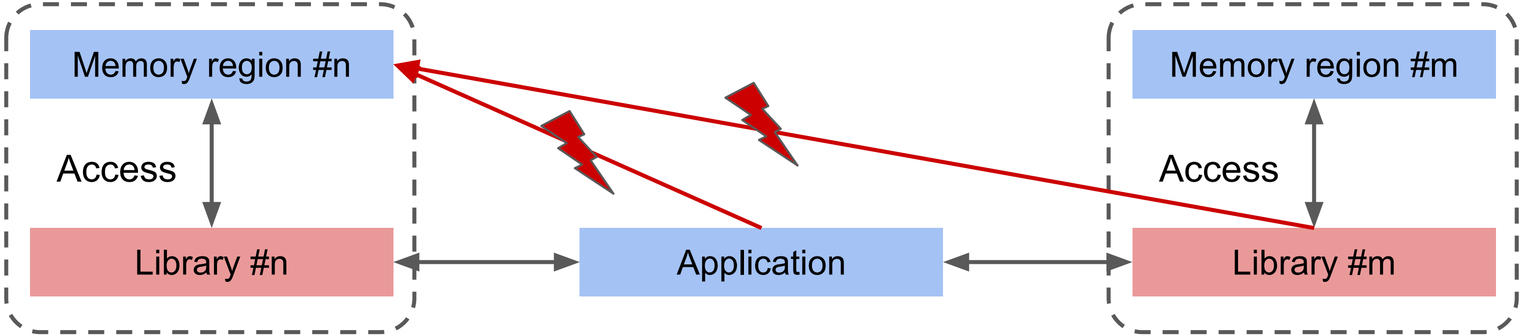
Design - Compiler



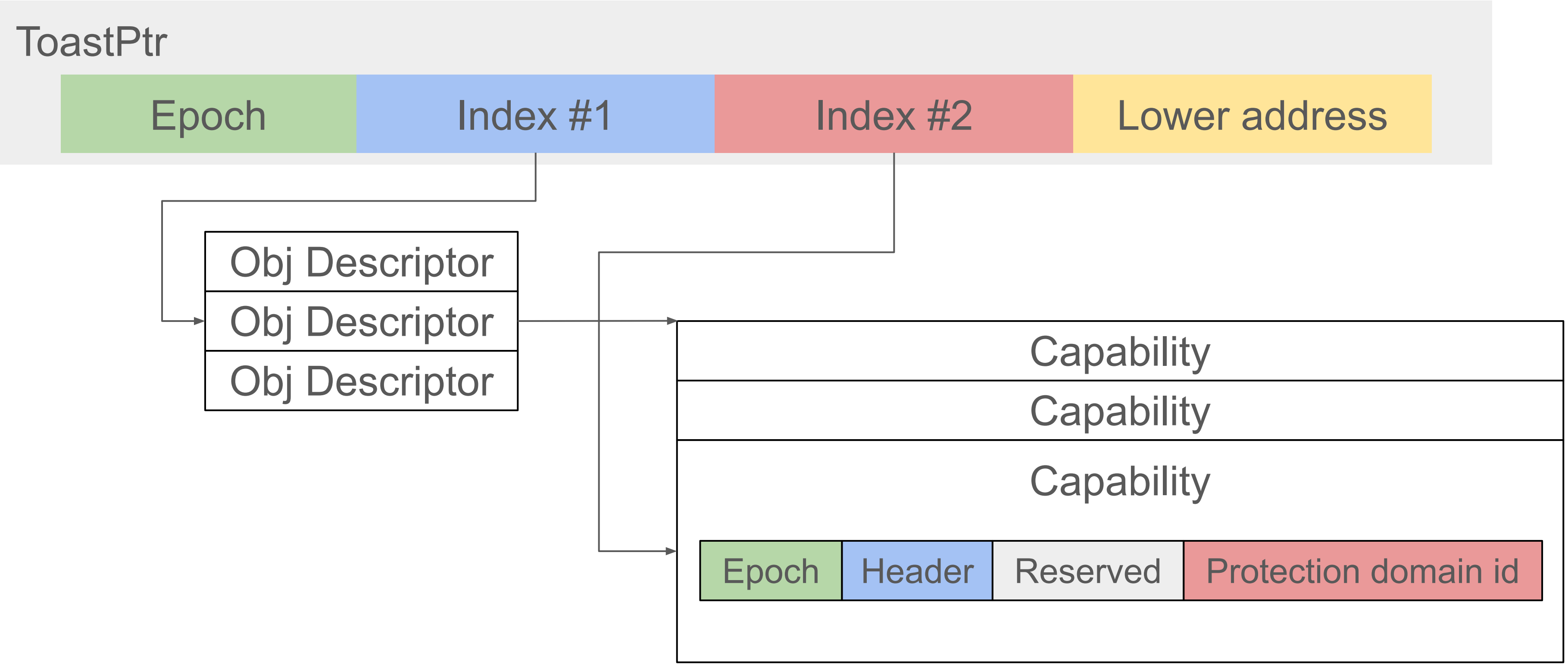
Design - Protection Library



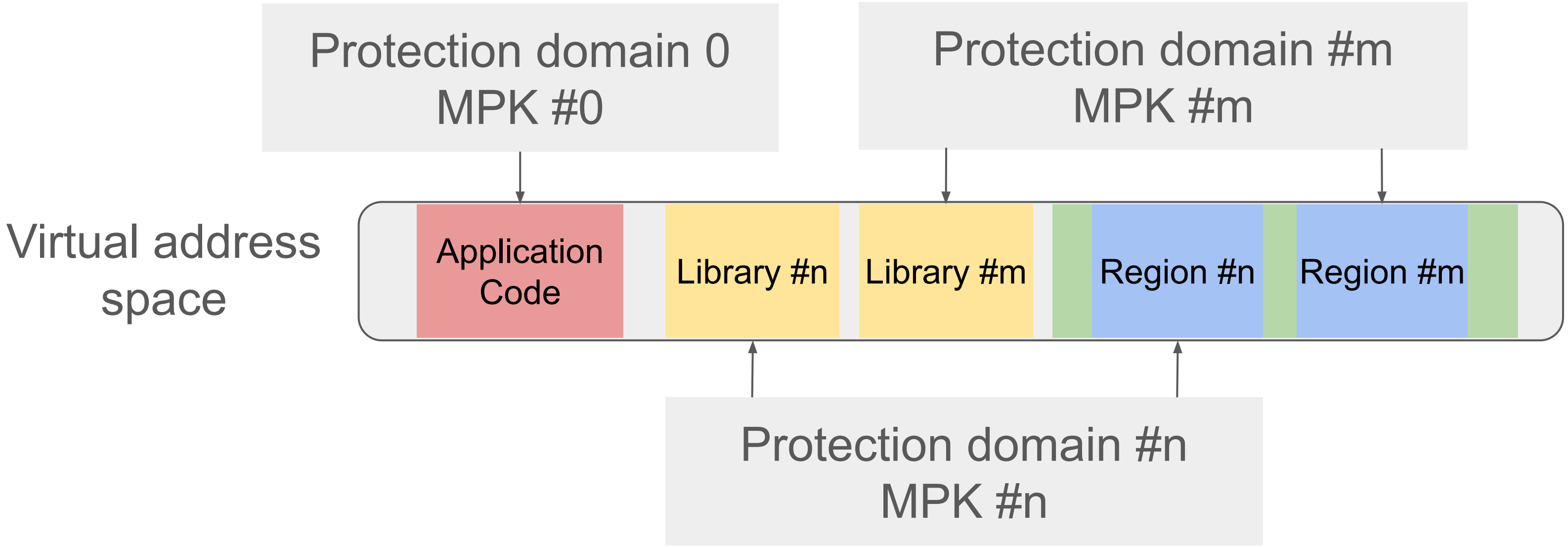
Design - Protection Library



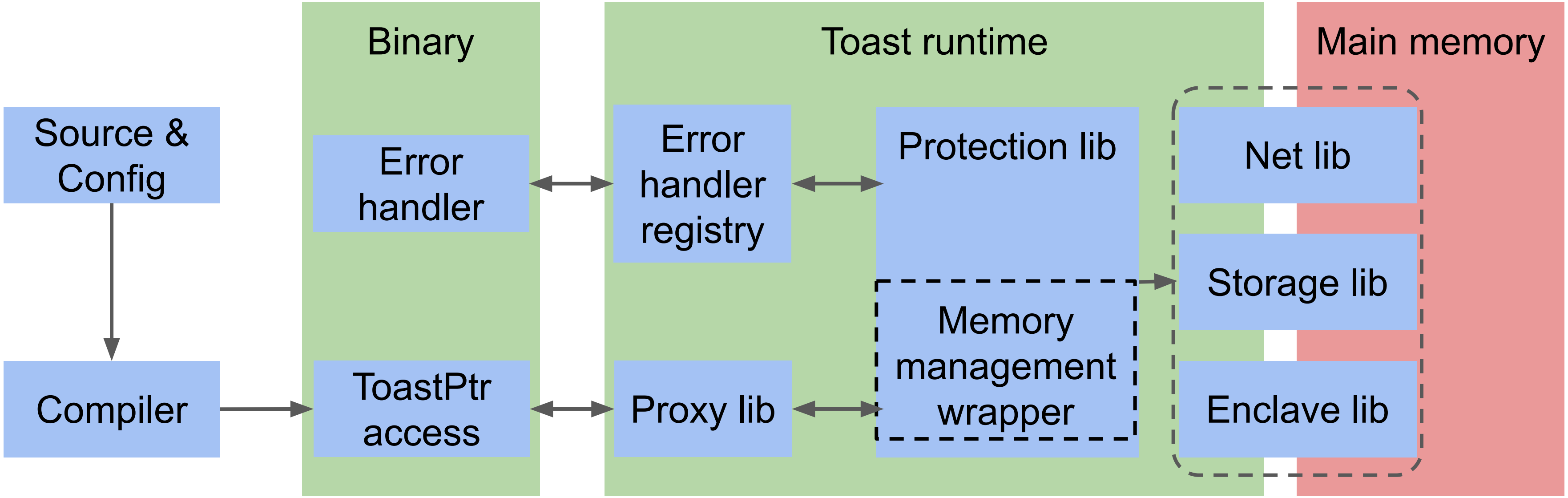
Design - Capability storage



Design - MPK protection



Design - Toast overview



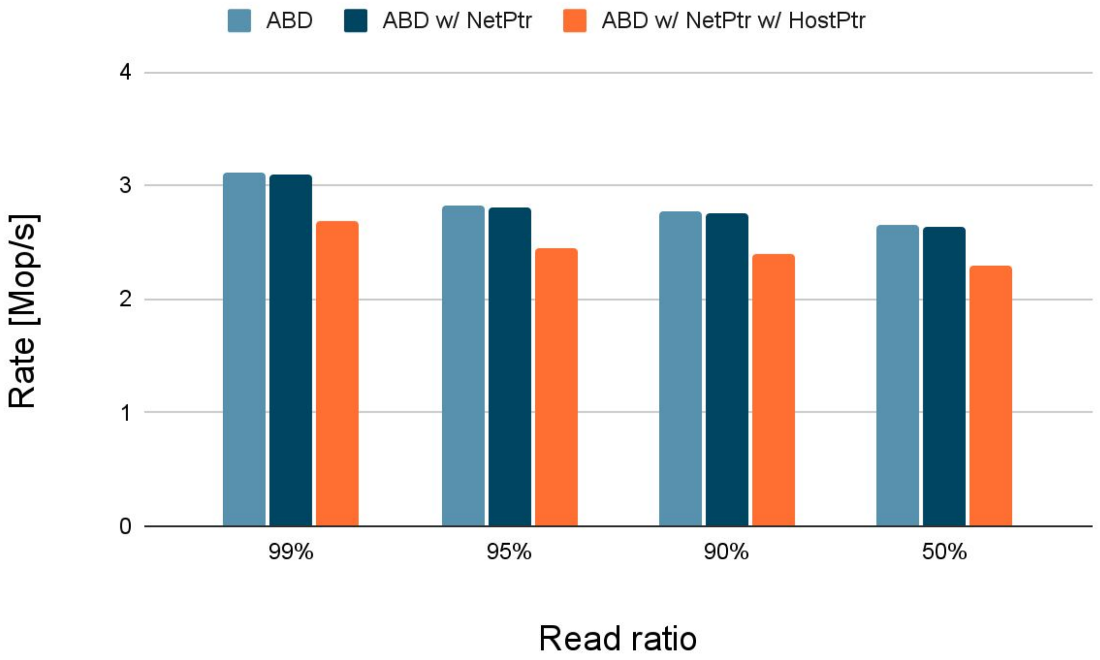
Outline

- ~~Motivation~~
- ~~Design~~
- Evaluation

Experimental setup

- 5x Server Machines
  - i9-9900K (5 GHz, 8c/16t)
  - 64 GiB RAM
  - XL710 40GbE QSFP+ (rev 02)
- 40 GbE switch
- For MPK experiments:
  - Intel Xeon Gold 5317 (3GHz, 12c/24t)
  - 256 GiB RAM

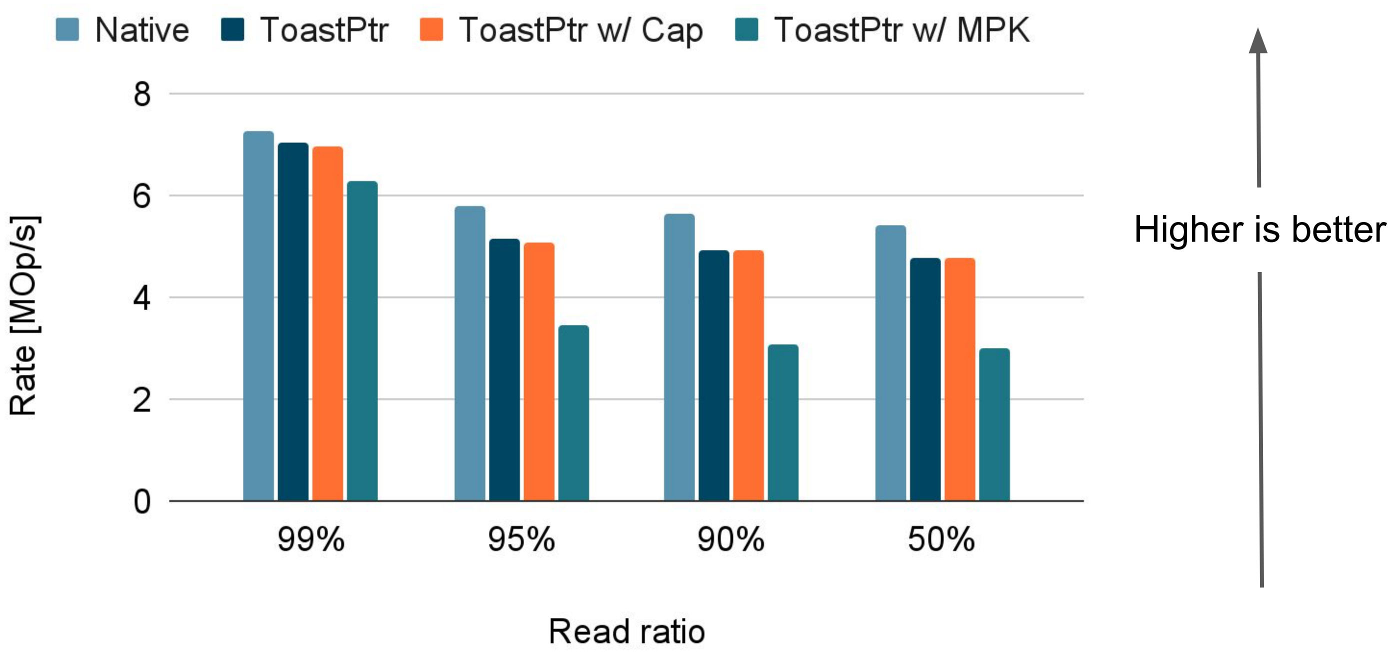
Evaluation - Secure replication protocol



Higher is better

Toast performance is comparable to hand optimized version

Evaluation - In-memory KV store



Performance trade-off between different protection levels

Evaluation - LOC

Application	Original	Toast	Reduction [%]
Secure in-memory KVS	110	105	4.5
Replication protocol	893	852	4.6
Persistent log	123	120	2.4
Persistent KVS	225	182	19.1

Toast reduces the amount of code, while increasing code maintainability

Conclusion

A compiler-based heterogeneous memory abstraction

Properties:

- Portability
- Programmability
- Safety
- Performance

Contributions:

- Uniform access
- Uniform error handling
- Protection libraries

Code available at: https://github.com/TUM-DSE/toast

Backup!

Design - Programming model

Paper diagrams

Code