CXL benchmarking

Viacheslav Dubeyko
Adam Manzanares

Content

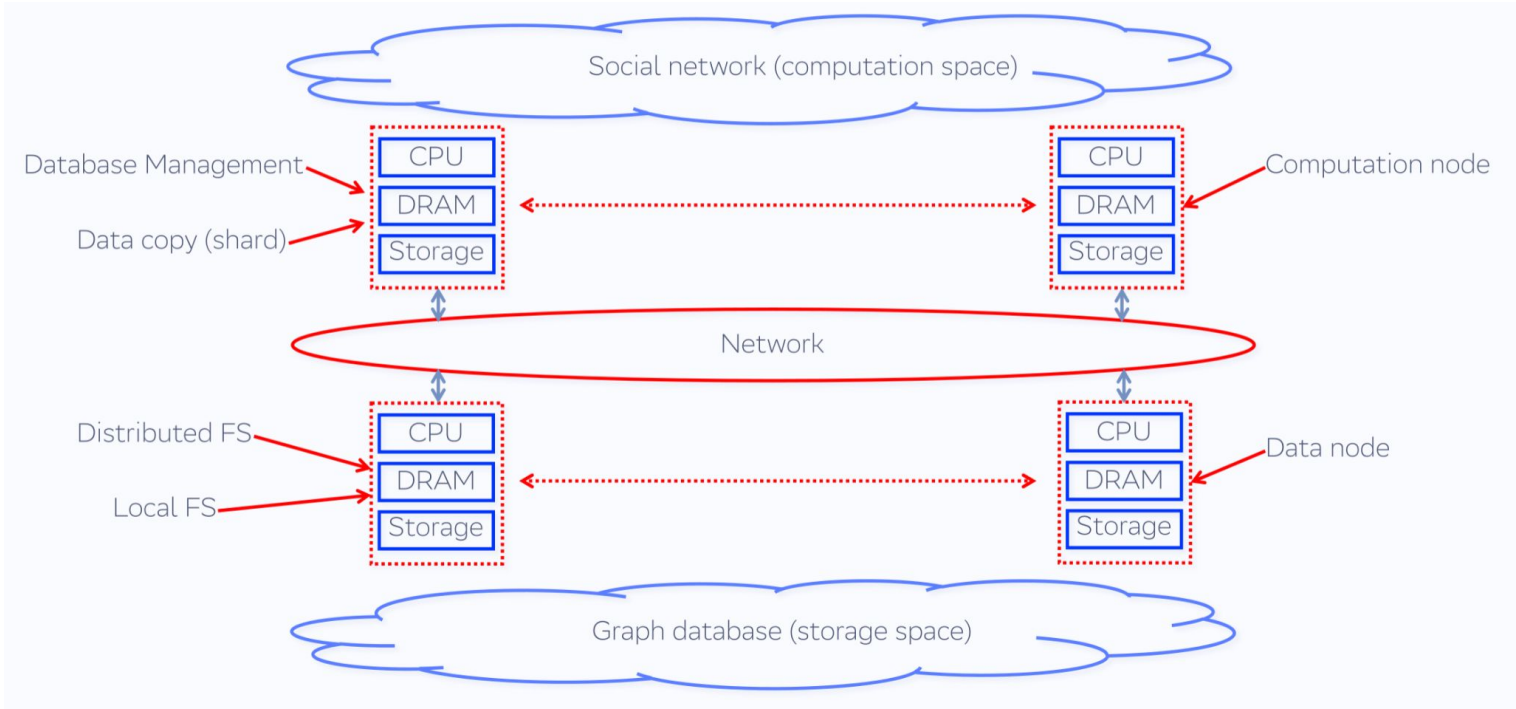
1. Problem declaration
2. Potential target use-cases
3. Emulation of target use-cases
4. Emulation of different types of CXL memory and CXL device types
5. Benchmarking framework
6. Benchmarking results interpretation
7. Open questions

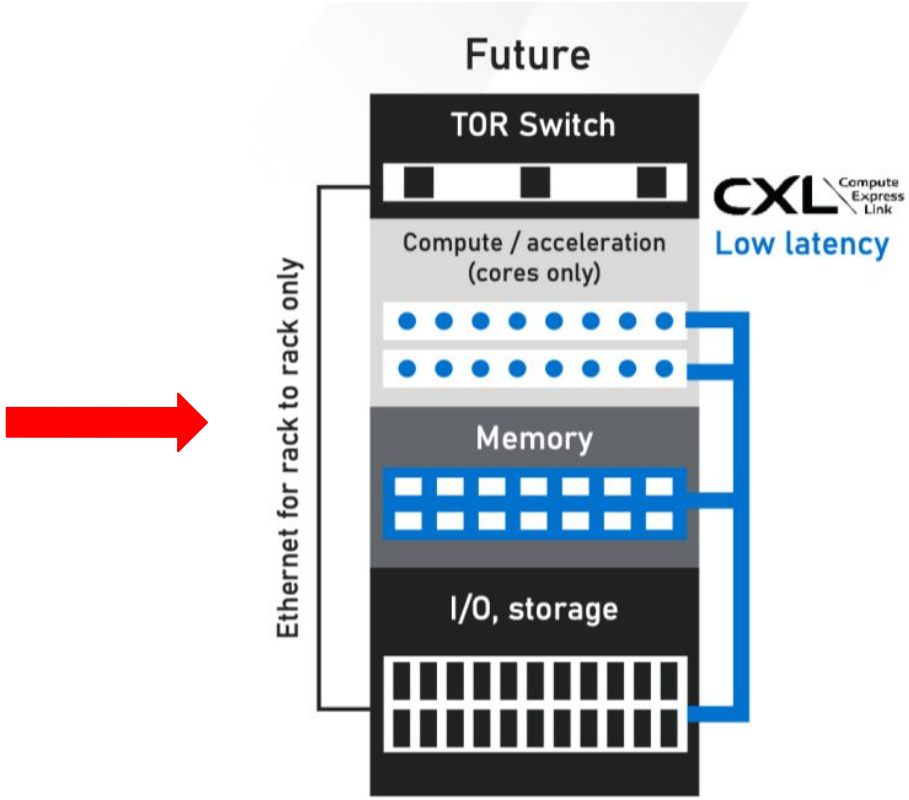
CXL benchmarking problem

- Determination allocation policy and memory access pattern(s) capable of improving an application performance
- Determination memory (CXL) configuration and migration policy capable of decreasing a total latency of particular memory access pattern
- Emulation of CXL infrastructure with the goal of determination an efficient configuration
- Detection of CXL infrastructure bottlenecks
- Continuous detection/management of CXL infrastructure latency/performance degradation
- Elaboration of CXL infrastructure architecture capable of decreasing TCO cost

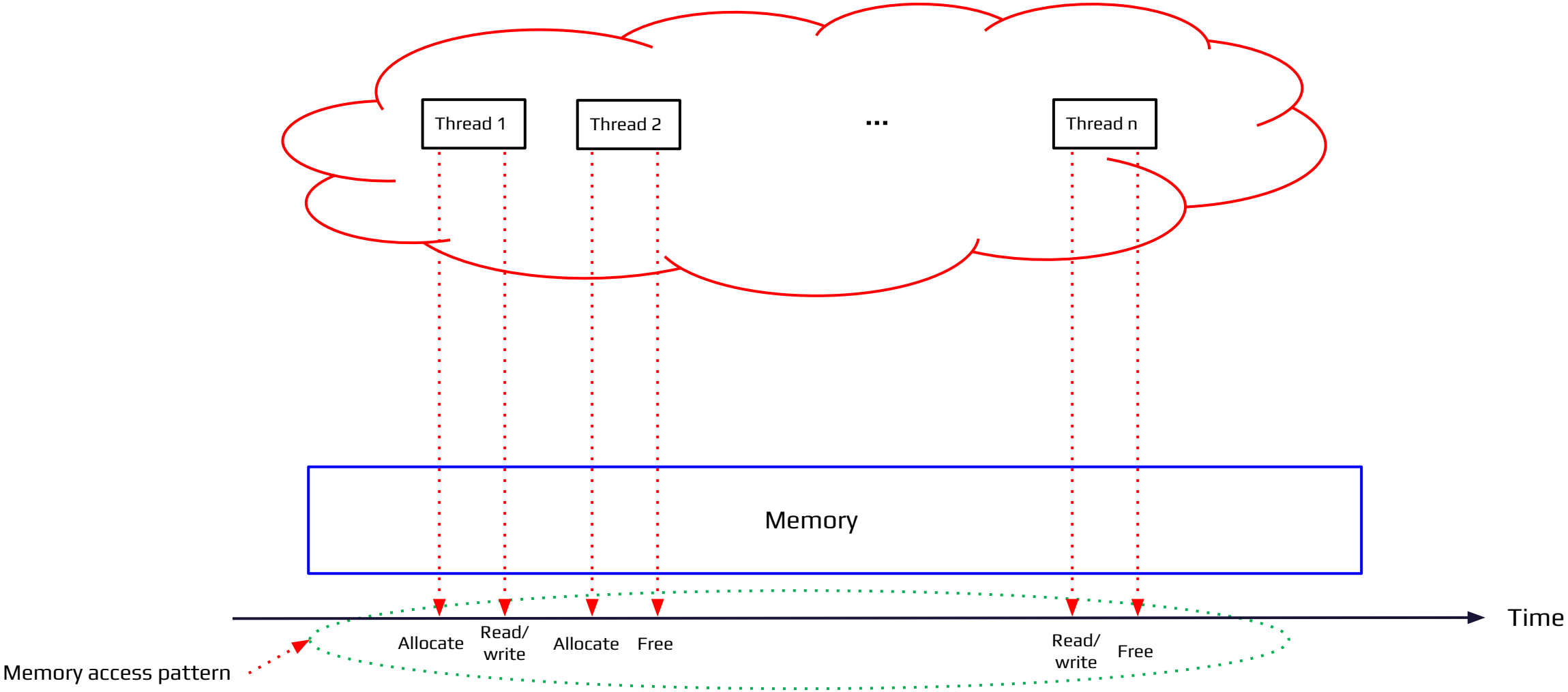
Potential target use-cases

- Huge relational and NoSQL databases
- In-memory database
- Social networks
- AI/ML workloads

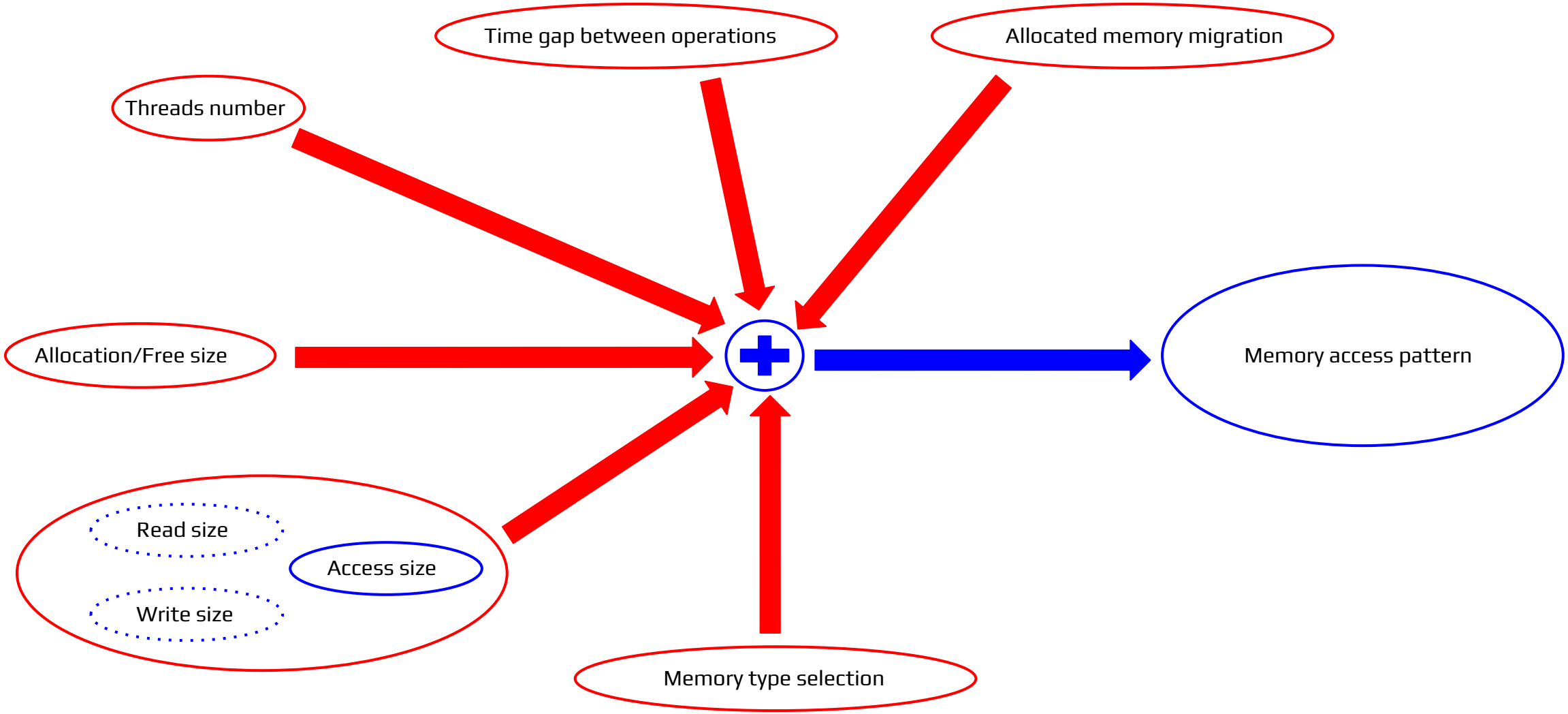




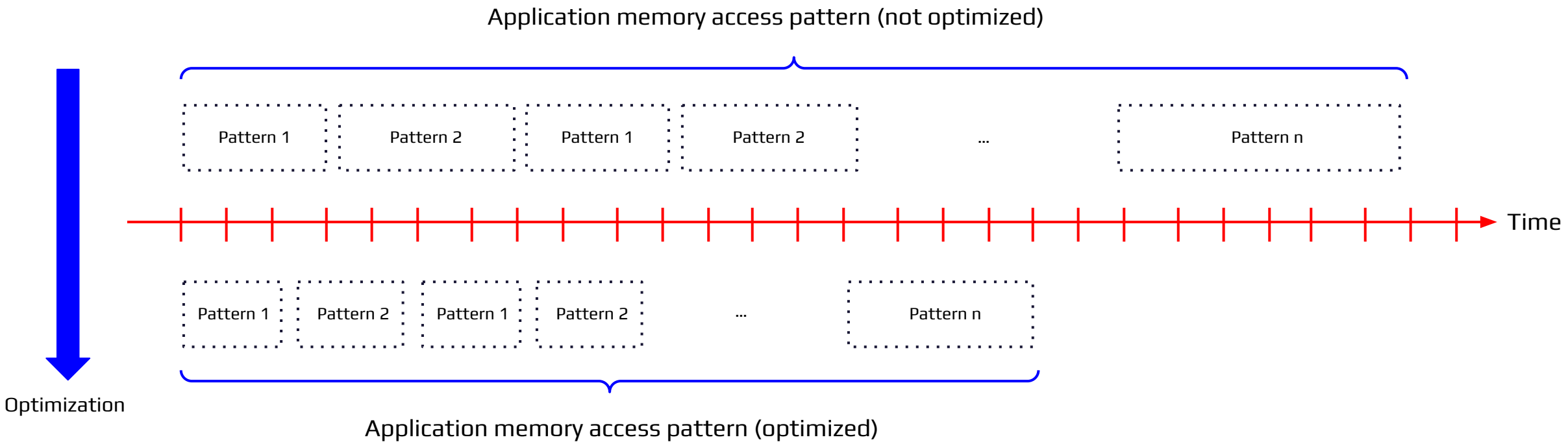
Target use-cases emulation problem



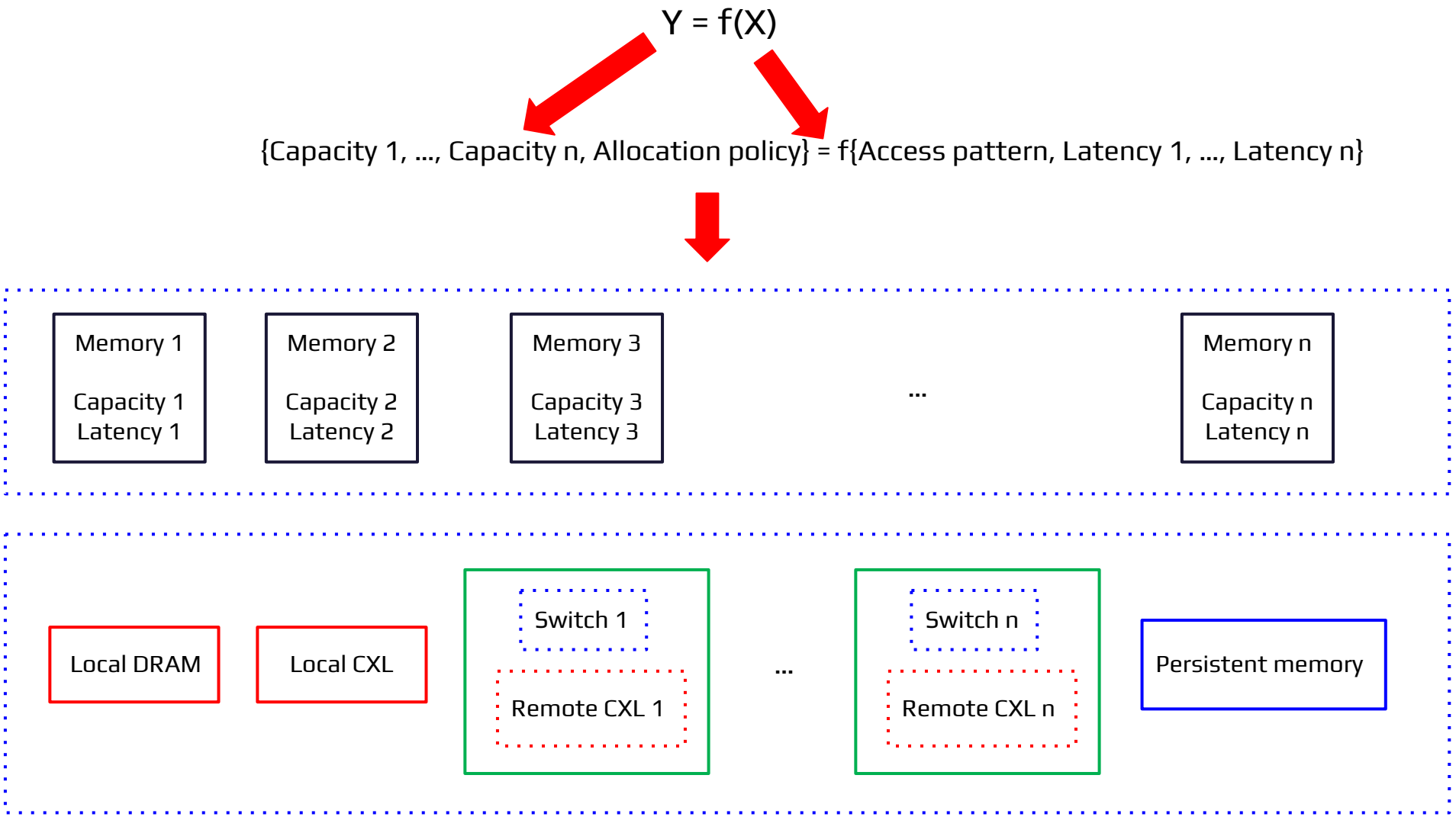
Memory access patterns emulation



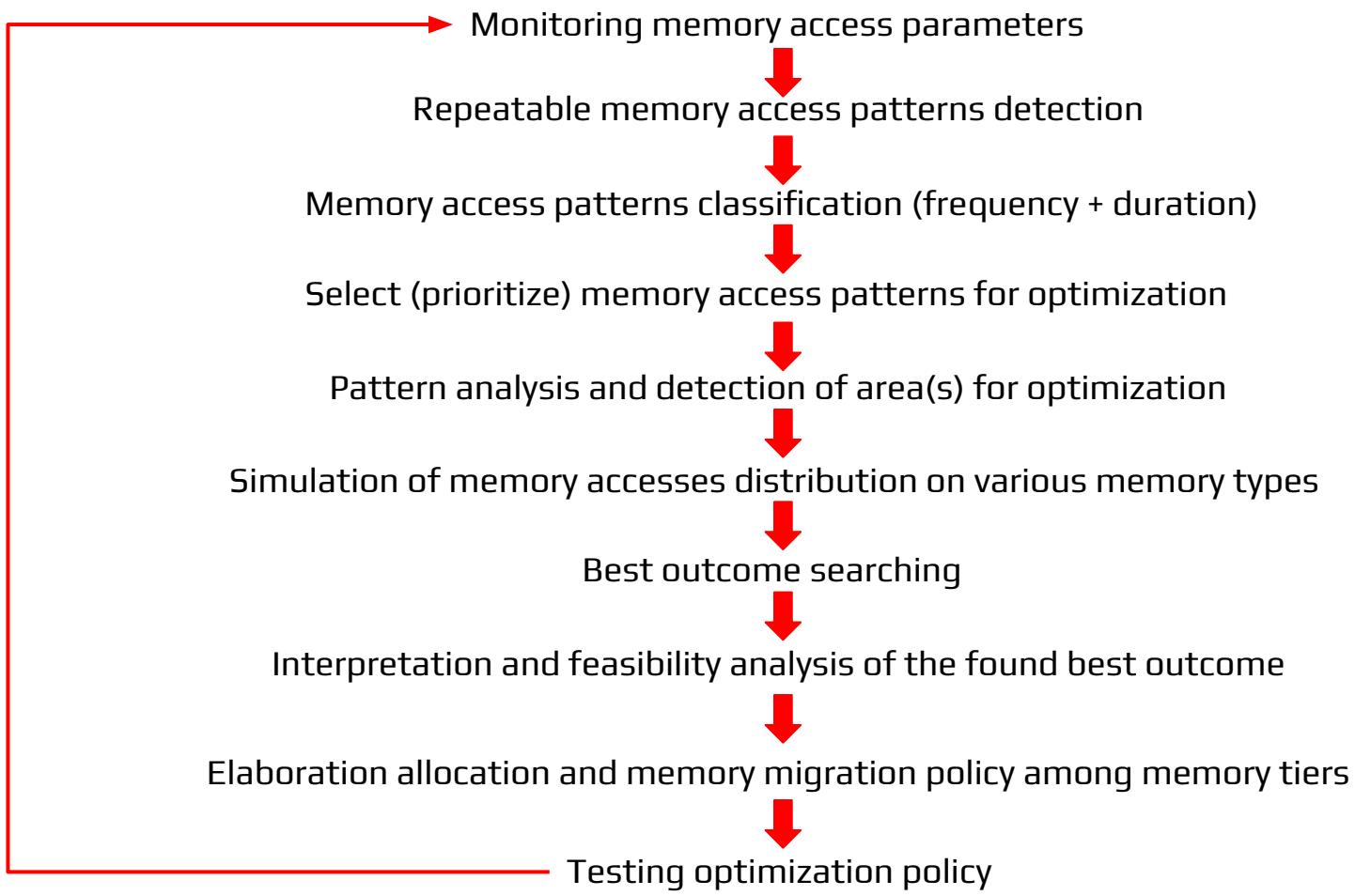
Optimization problem



Optimization problem

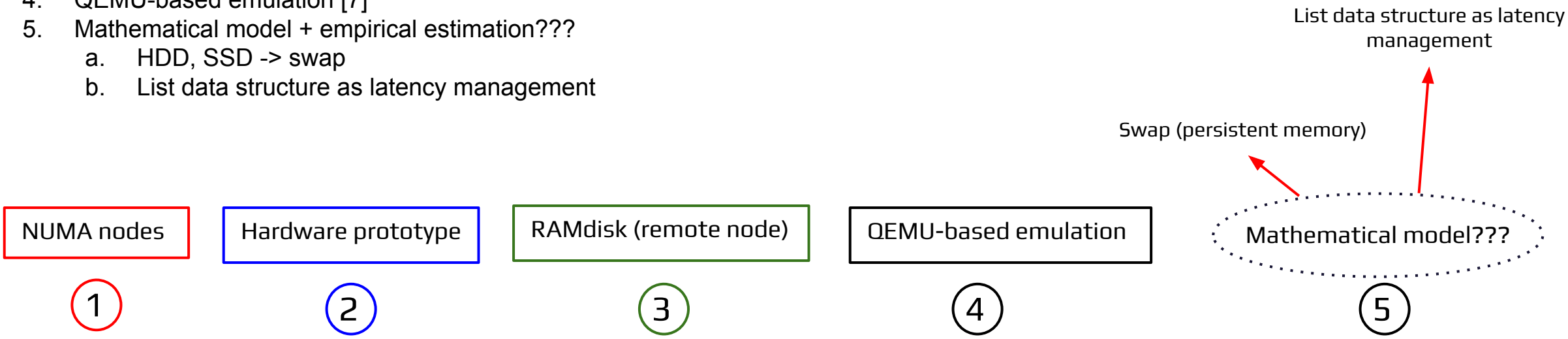


Optimization workflow



CXL memory types emulation

1. NUMA nodes [1, 6]
2. Hardware prototype [2, 3, 4]
3. RAMdisk on remote nodes [5]
4. QEMU-based emulation [7]
5. Mathematical model + empirical estimation???
   a. HDD, SSD -> swap
   b. List data structure as latency management



1. Li, Huaicheng, et al. ”Pond: CXL-based memory pooling systems for cloud platforms.” Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2. 2023.
2. Gouk, D., Lee, S., Kwon, M. & Jung, M. Direct Access, High-Performance Memory Disaggregation with DirectCXL. 2022 USENIX Annual Technical Conference (USENIX ATC 22). pp. 287-294 (2022,7)
3. M. Jung, ”Hello bytes, bye blocks: PCIe storage meets compute express link for memory expansion (CXL-SSD)”. In Proceedings of the 14th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage ’22). Association for Computing Machinery, New York, NY, USA, 45–51.
4. M. Ahn et al., “Enabling CXL memory expansion for in-memory database management systems,” Data Management on New Hardware, 2022.
5. M. D. Flouris, E. P. Markatos. ”The Network RamDisk: Using remote memory on heterogeneous NOWs”. Cluster Computing 2, 4 (1999), 281–293.
6. Wahlgren, J., Gokhale, M. & Peng, I. Evaluating Emerging CXL-enabled Memory Pooling for HPC Systems. ArXiv Preprint ArXiv:2211.02682. (2022)
7. Raja Gond and Purushottam Kulkarni, emucxl: an emulation framework for CXL-based disaggregated memory applications. ArXiv Preprint arXiv:2404.08311 (2024)

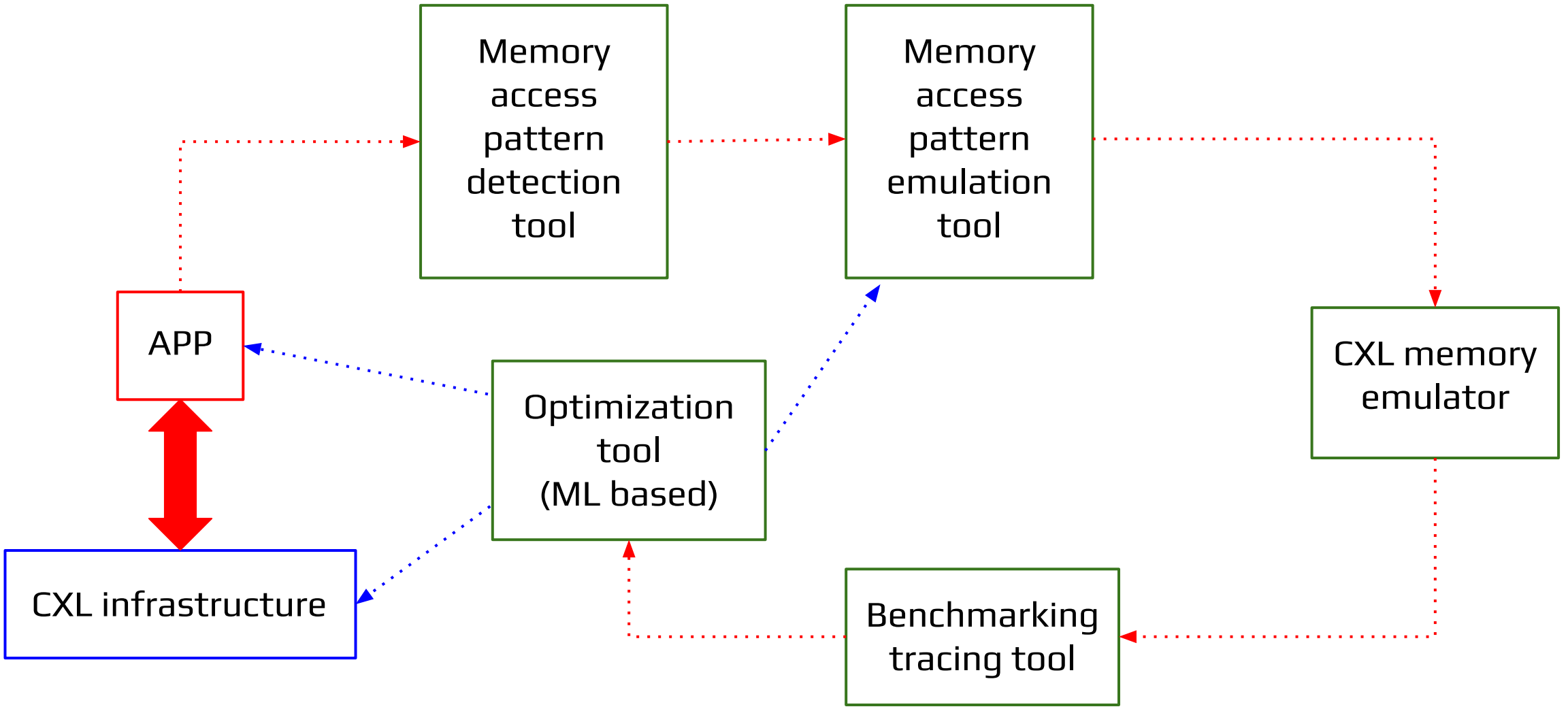
Potential memory type selection policies

- **Priority-based allocation policy**
  - higher priority in local DRAM, lower priority in CXL memory
- **Size-based allocation policy**
  - smaller sizes in local DRAM, bigger sizes in CXL memory
- **Pre-fetch based allocation policy**
  - pre-fetching from persistent memory into CXL memory
- **Pre-allocation based policy**
  - pre-allocation big chunks of memory (memory pool) in CXL memory
- **Pre-mapping based policy**
  - page faults elimination by pre-mapping physical memory pages for virtual address space
- **Lifetime based allocation policy**
  - short-lived memory chunks in local DRAM, long-lived memory chunks in CXL memory
  - application-based hints on memory chunks' lifetime
  - lifetime-based memory heaps (short-lived, long-lived memory pools)

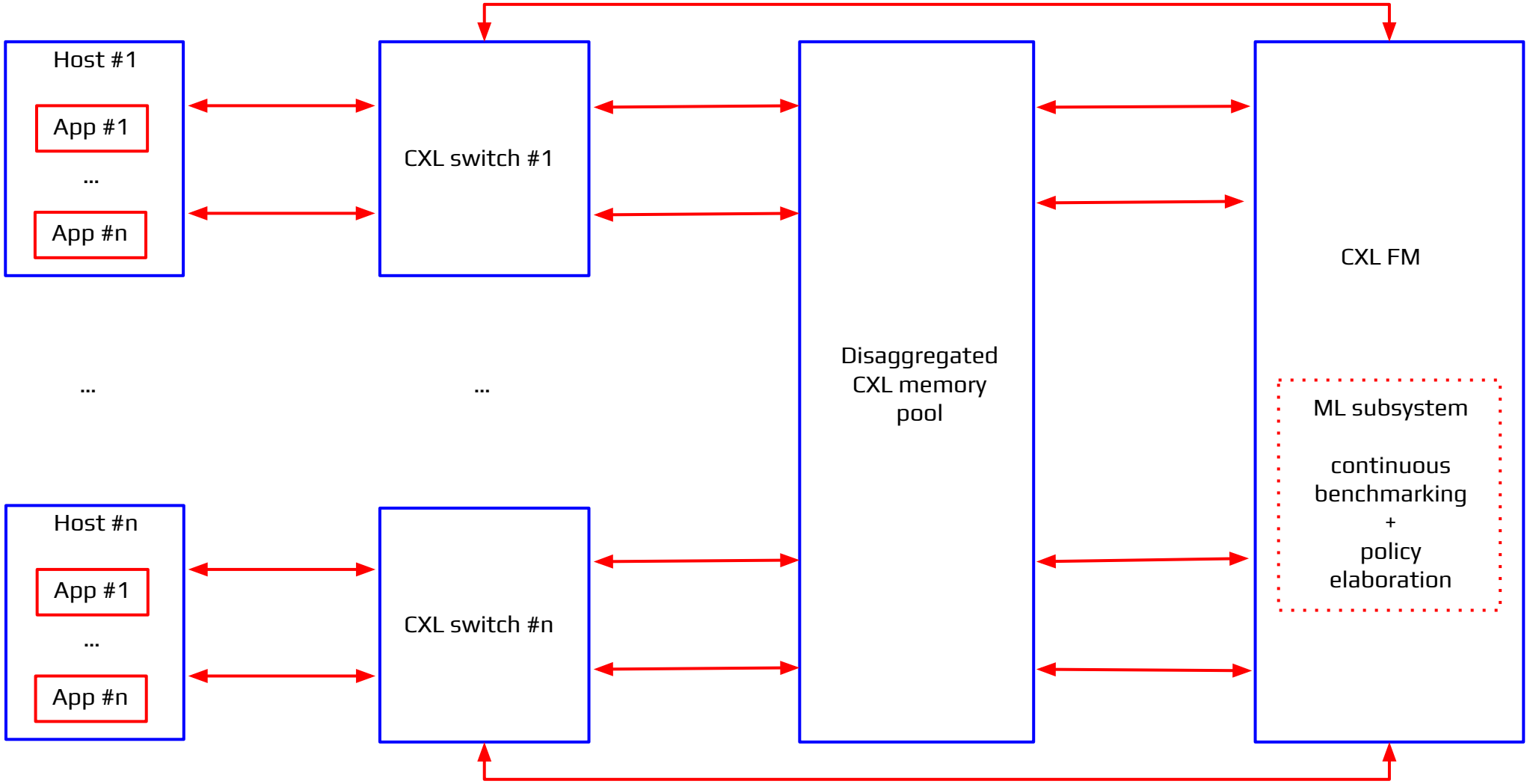
Potential allocated data migration policies

- **Dynamic lifetime-based policy**
  - initial allocation from short-lived memory pool (local DRAM)
  - growing lifetime initiates migration in longer-lived memory pools (CXL memory)
- **Swap-based migration policy**
  - swapping from local DRAM into CXL memory
- **File system like interaction between local DRAM and CXL memory**
  - pre-fetching portion of CXL memory content into local DRAM
  - hardware-based management???

CXL benchmarking framework



CXL Fabric Manager (FM) as benchmarking subsystem

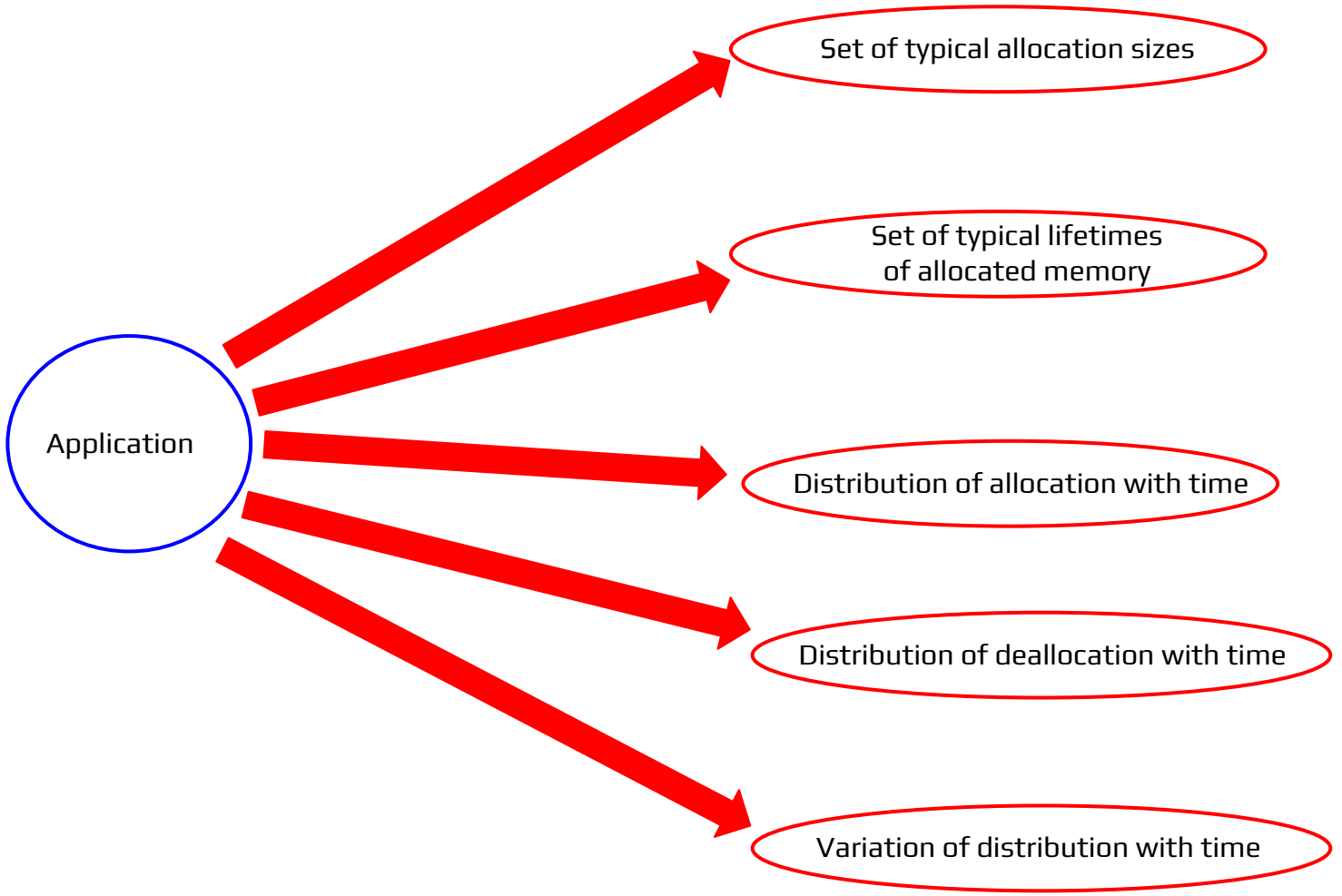


Open questions

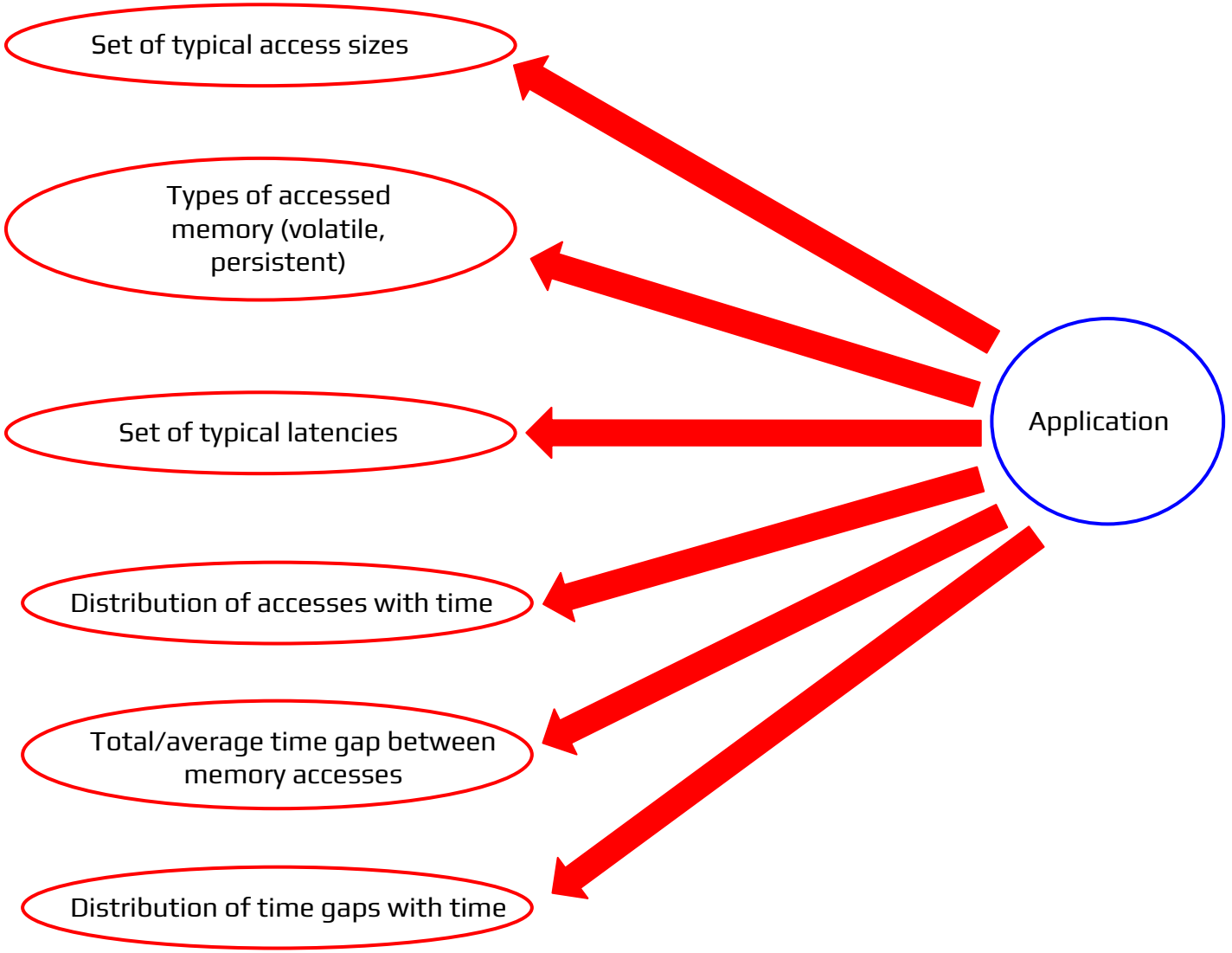
- How to emulate memory latency?
- How to emulate application behavior?
- How to estimate application performance?
- How to treat benchmarking results?
  - How good is “good” numbers?
  - How bad is “bad” numbers?
- Isolated workload vs. real-life environment?
- How to identify and isolate code patterns determining application behavior sensitive to memory allocation types?
- How feasible is implementation of found best optimization outcomes?
- How useful can be an abstractness of mathematical model?


THANK YOU
QUESTIONS???

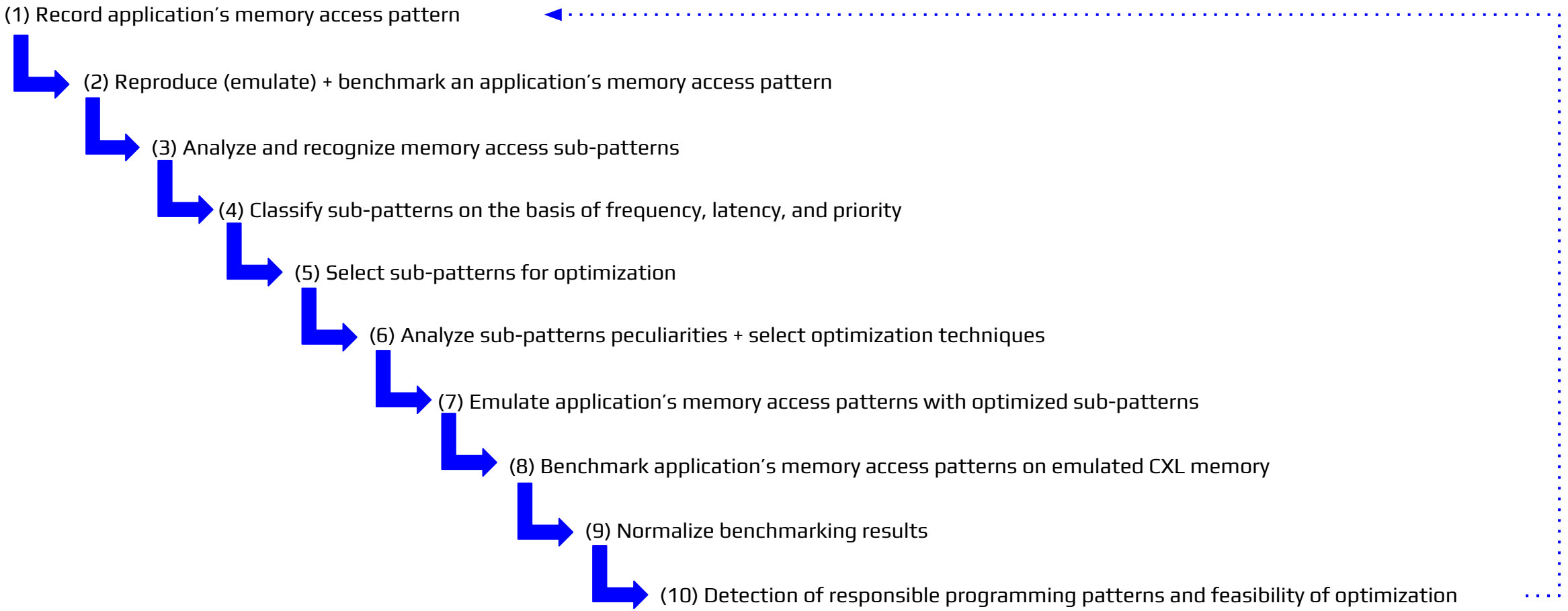
Allocate/free memory patterns



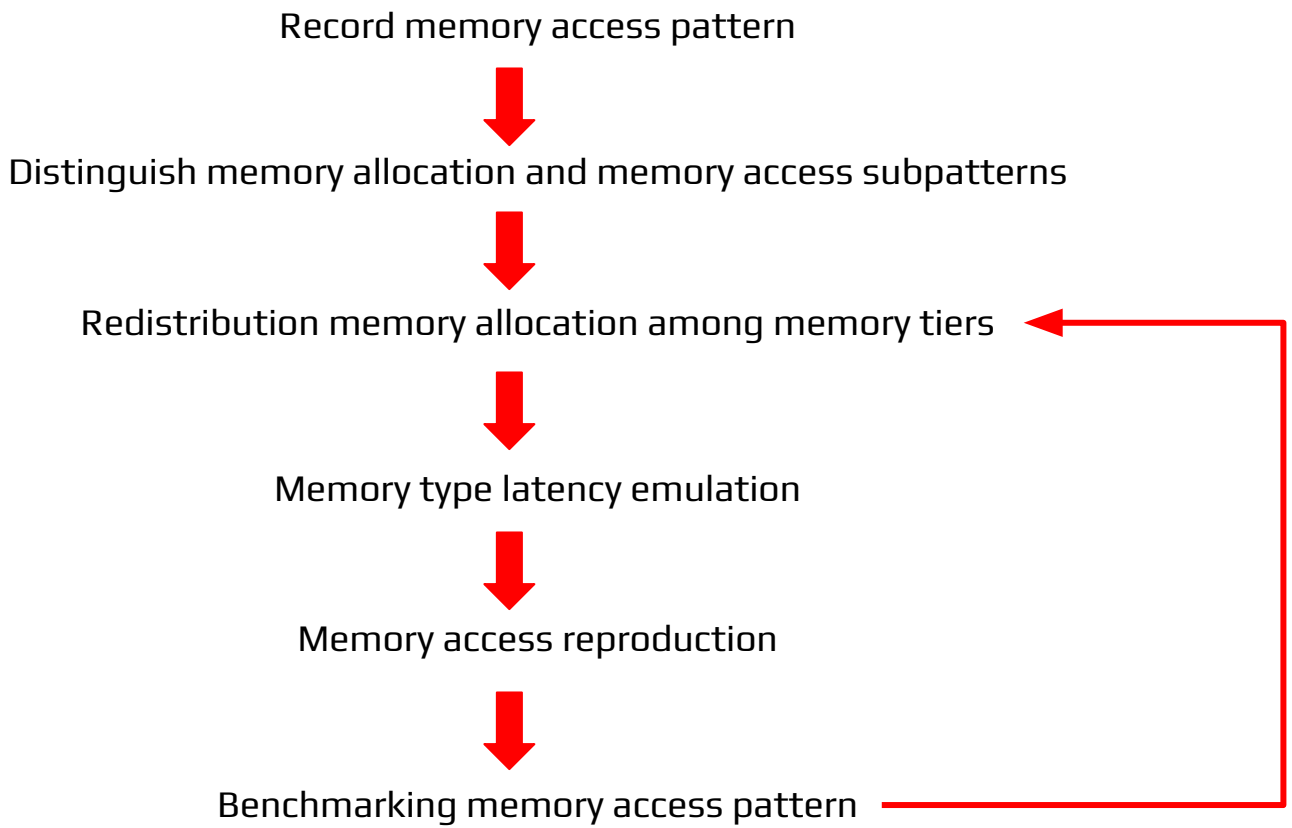
Read/write memory patterns



CXL benchmarking framework



Correct latency and performance estimation problem



How to emulate memory latency?

How to emulate application behavior?

How to estimate application performance?

How to treat benchmarking results?

Benchmarking result interpretation problem

- Measured numbers variation
- Latency emulation approach
- Measurement approach
- Approach of redistribution memory allocation among tiers
- Abstractness of mathematical model