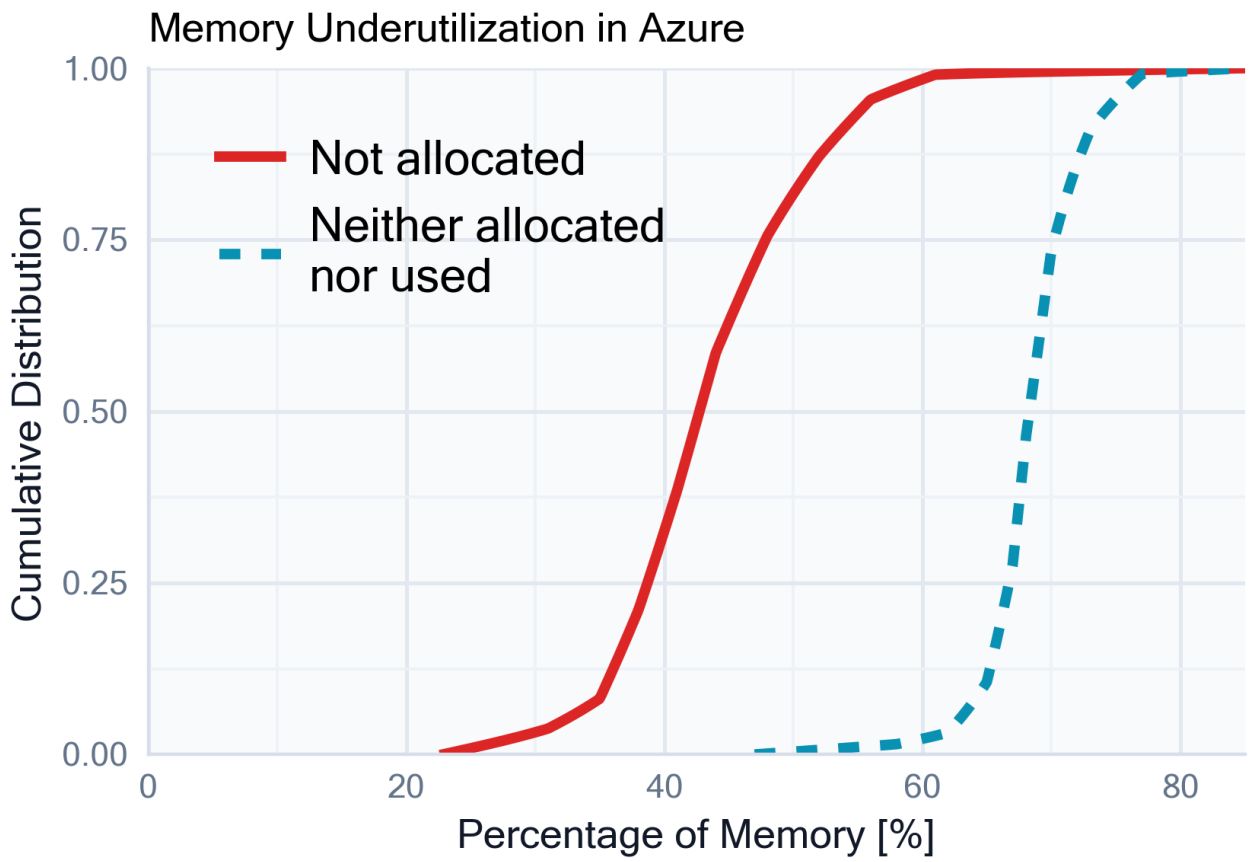
PD3
Prefetching Data with DPUs for Disaggregated Memory

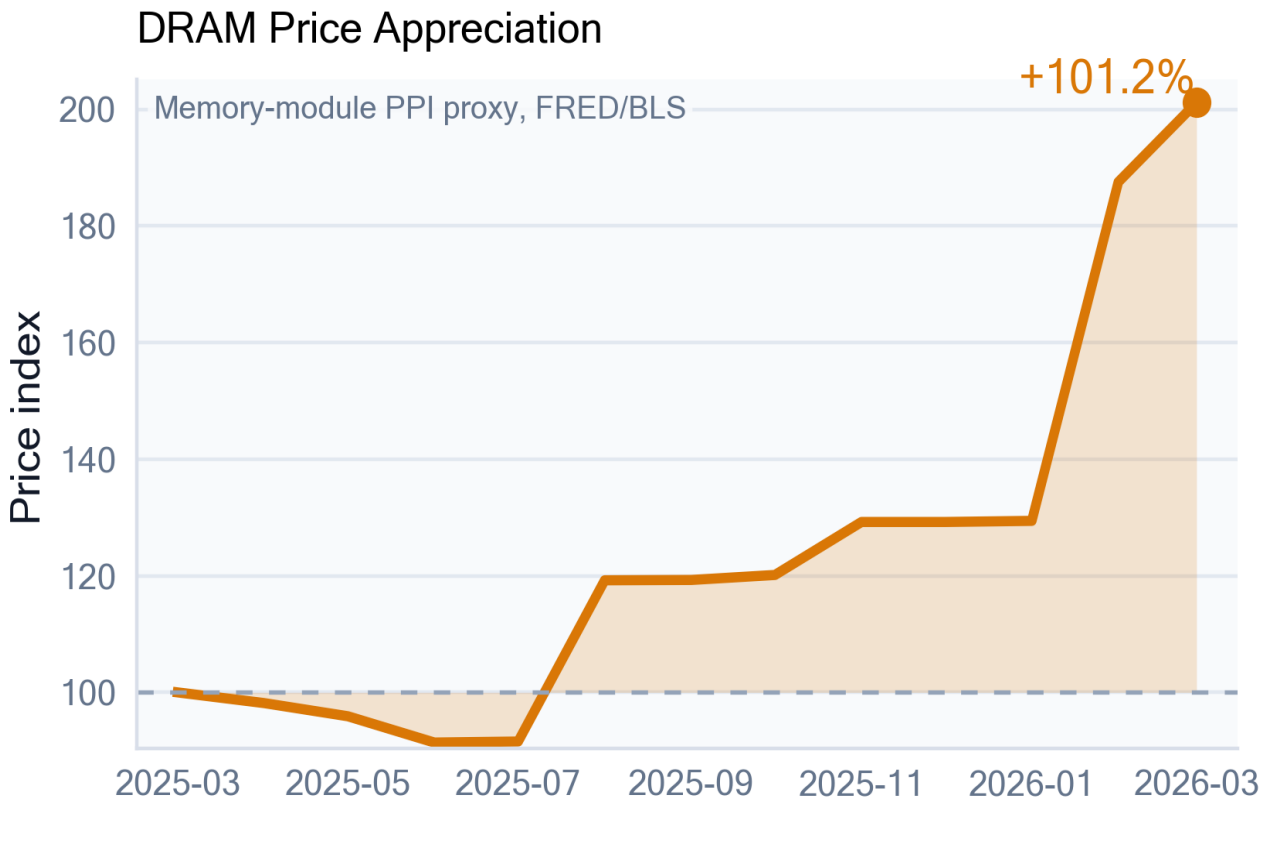
Sidharth Sankhe, Felix Zhang, Umayrah Chonee, Sherman Lim,
Jiasheng Hu, Jialin Li, Qizhen Zhang

Memory-intensive Applications

Memory Utilization in Data Centers

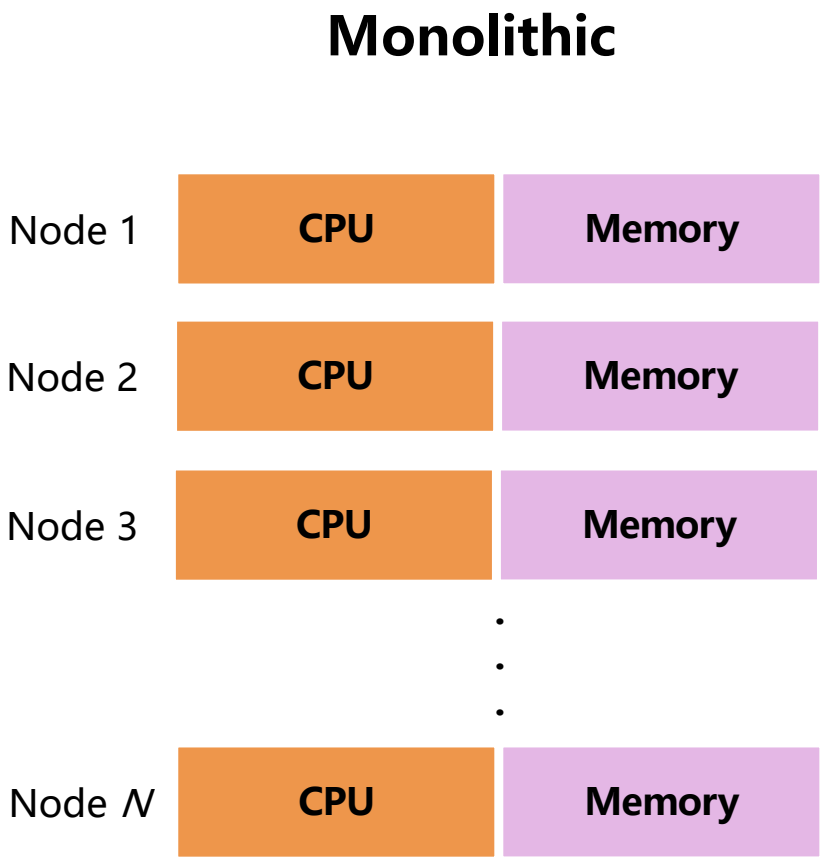


CompuCache [CIDR '22]

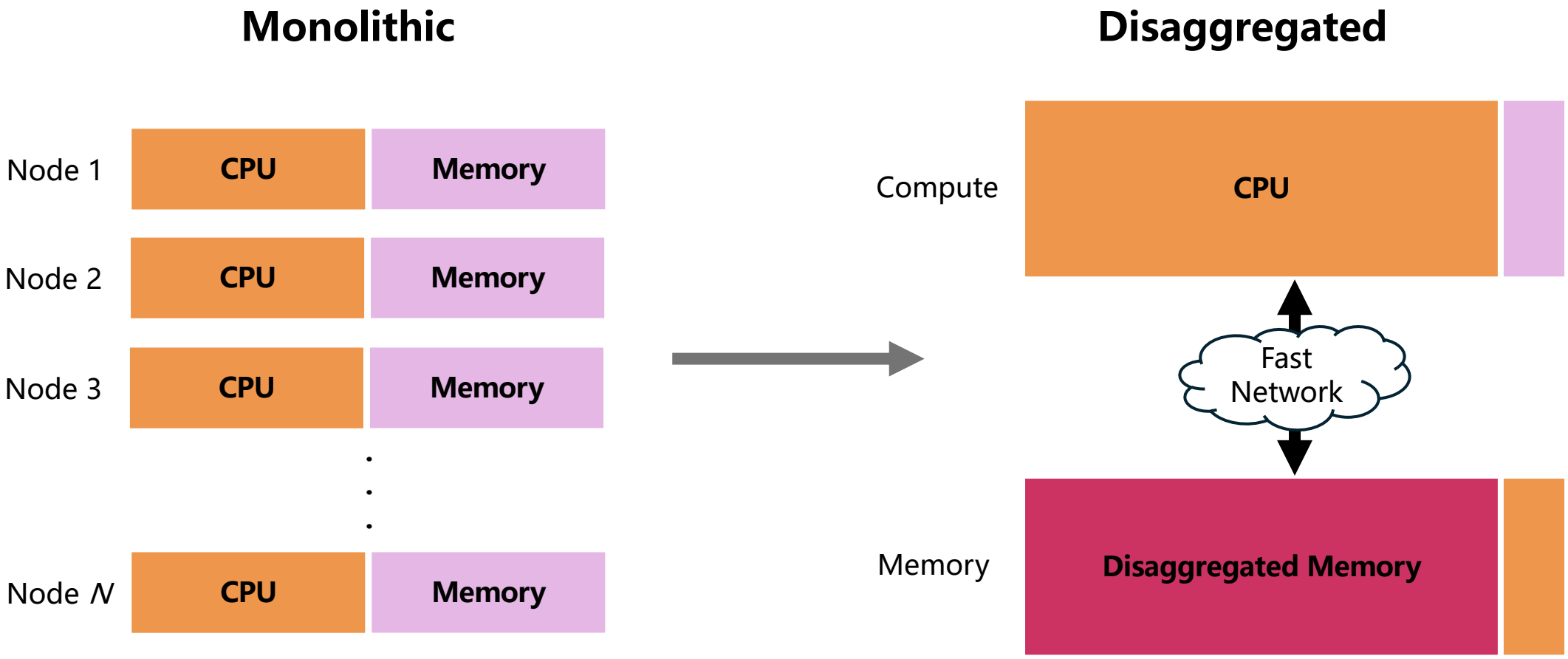


Source: FRED WPU117861, generated 2026-05-01

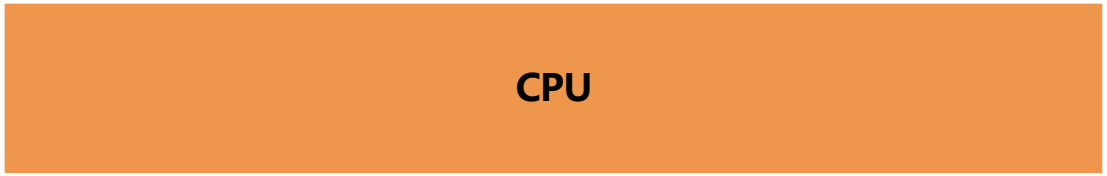
Memory Disaggregation



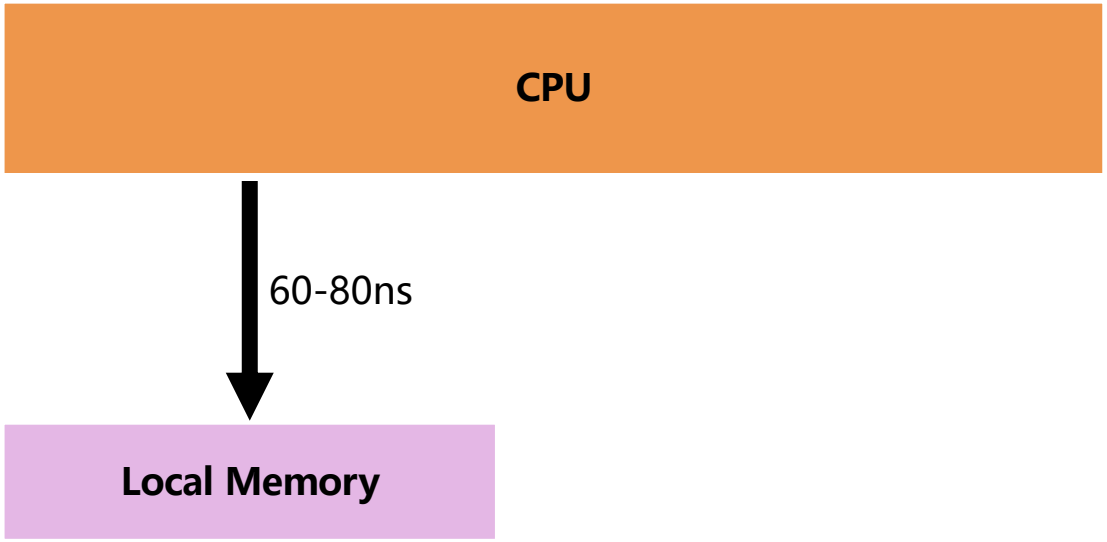
Memory Disaggregation



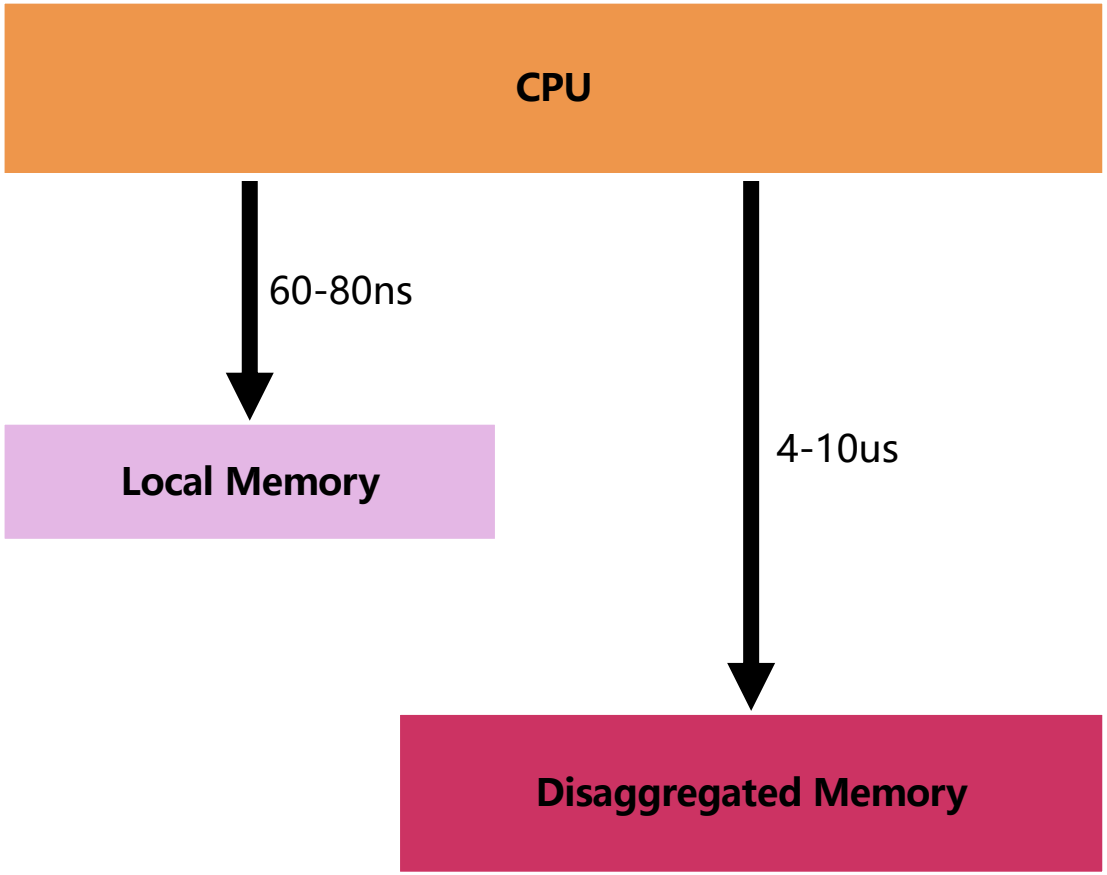
Performance Degradation



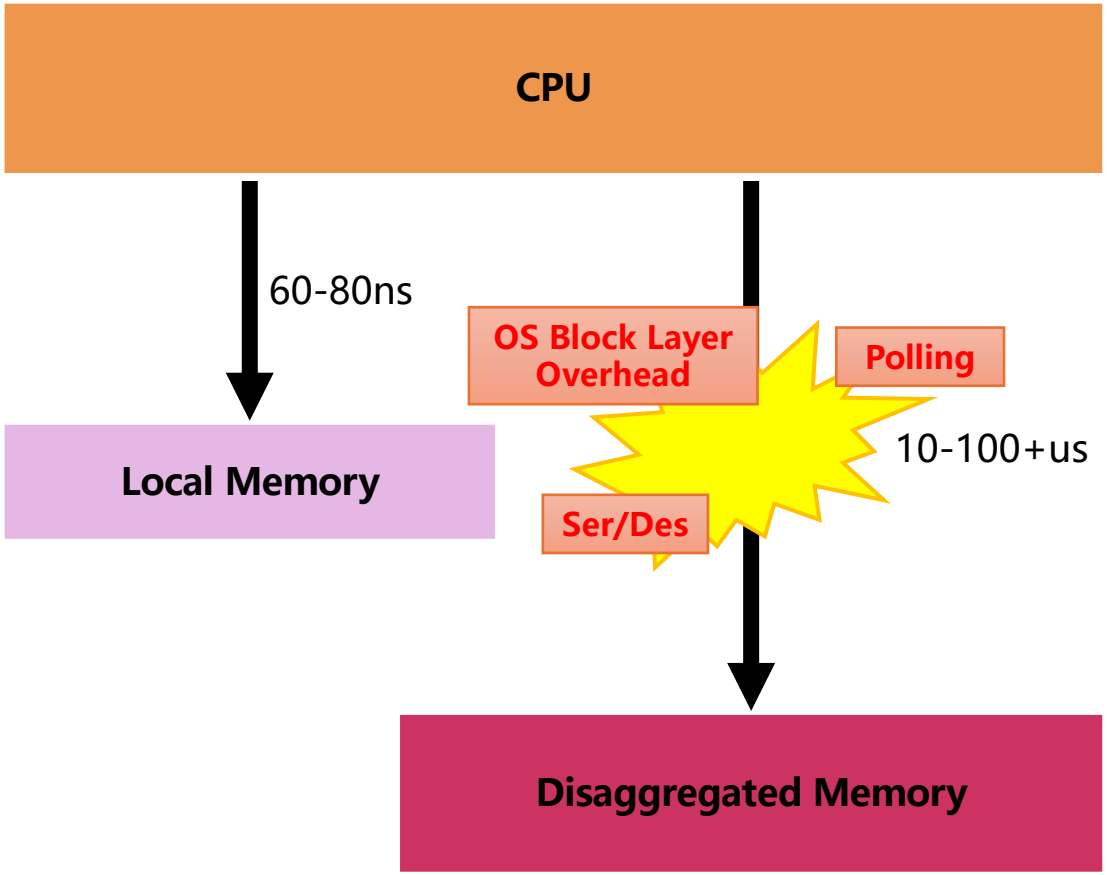
Performance Degradation



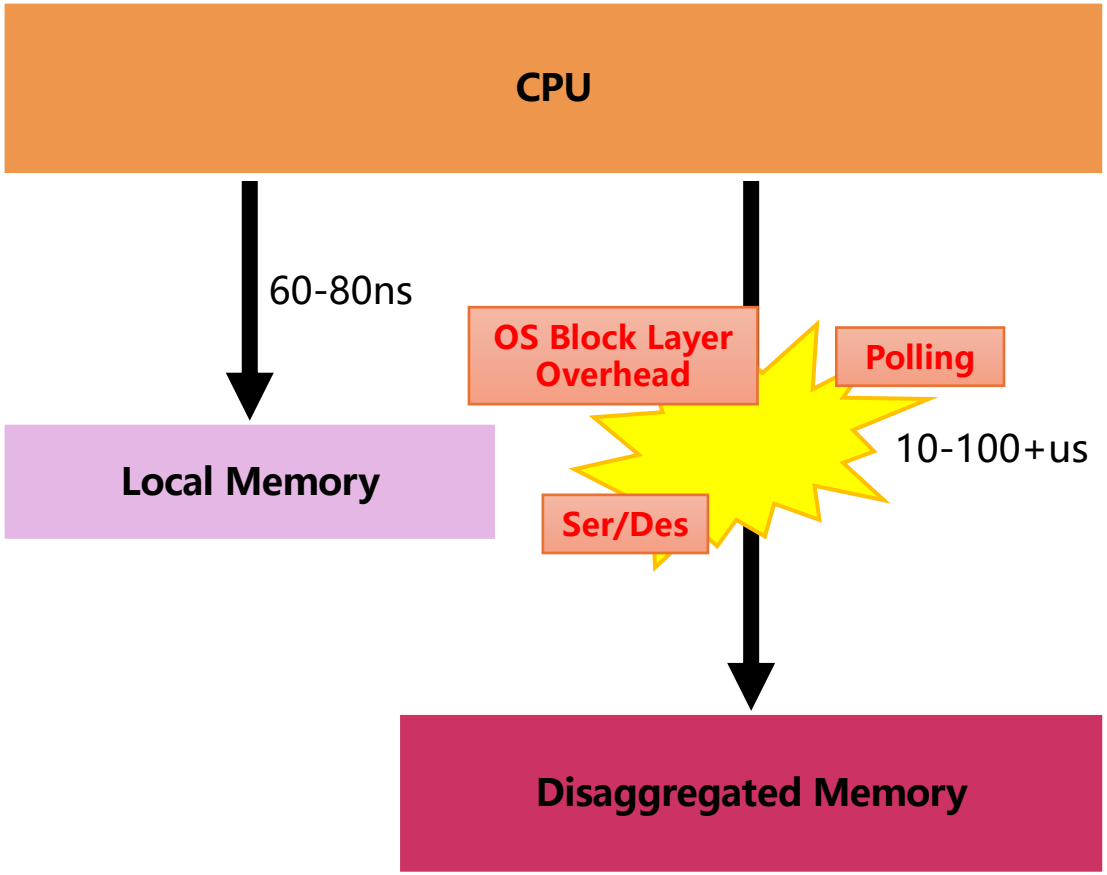
Performance Degradation



Performance Degradation

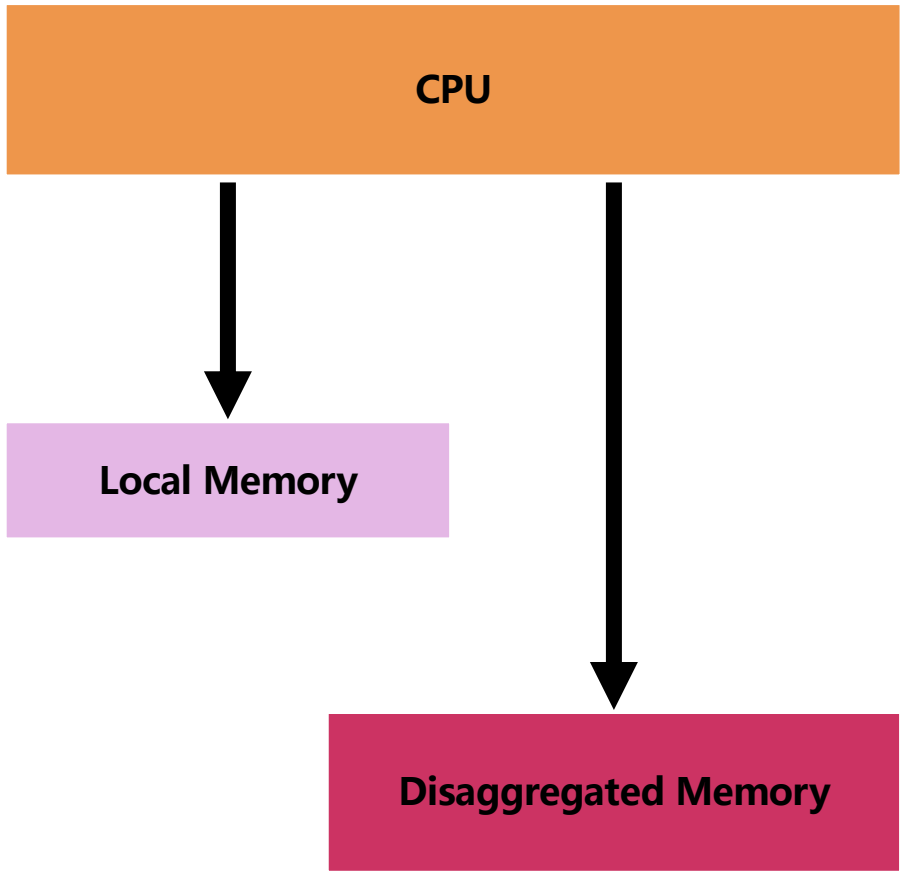


Performance Degradation

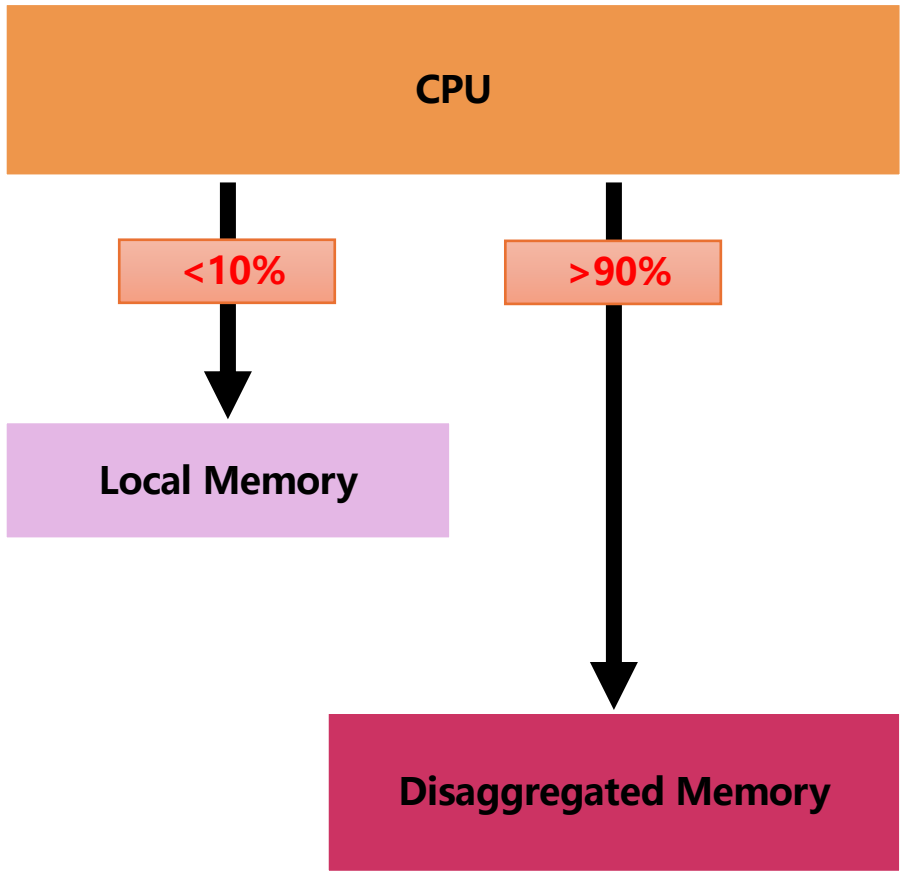


- **OS:** Infiniswap [NSDI '17], Leap [ATC '20], FastSwap [EuroSys '20], TELEPORT [SIGMOD '22]
- **Runtime:** AIFM [OSDI '20], Redy [VLDB '22], Beehive [NSDI '25], Eden [NSDI '25]
- **Data structures:** Sherman [SIGMOD '22], FUSEE [FAST '23]
- And many more…

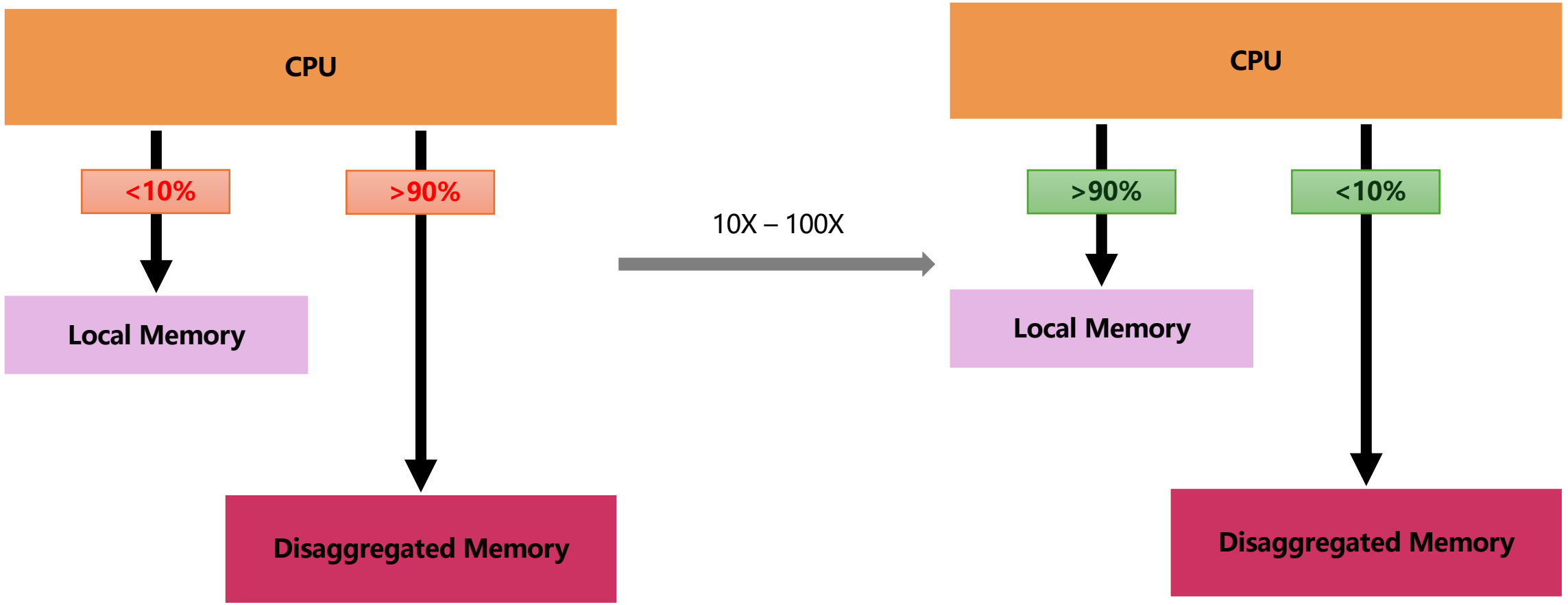
Cache Misses



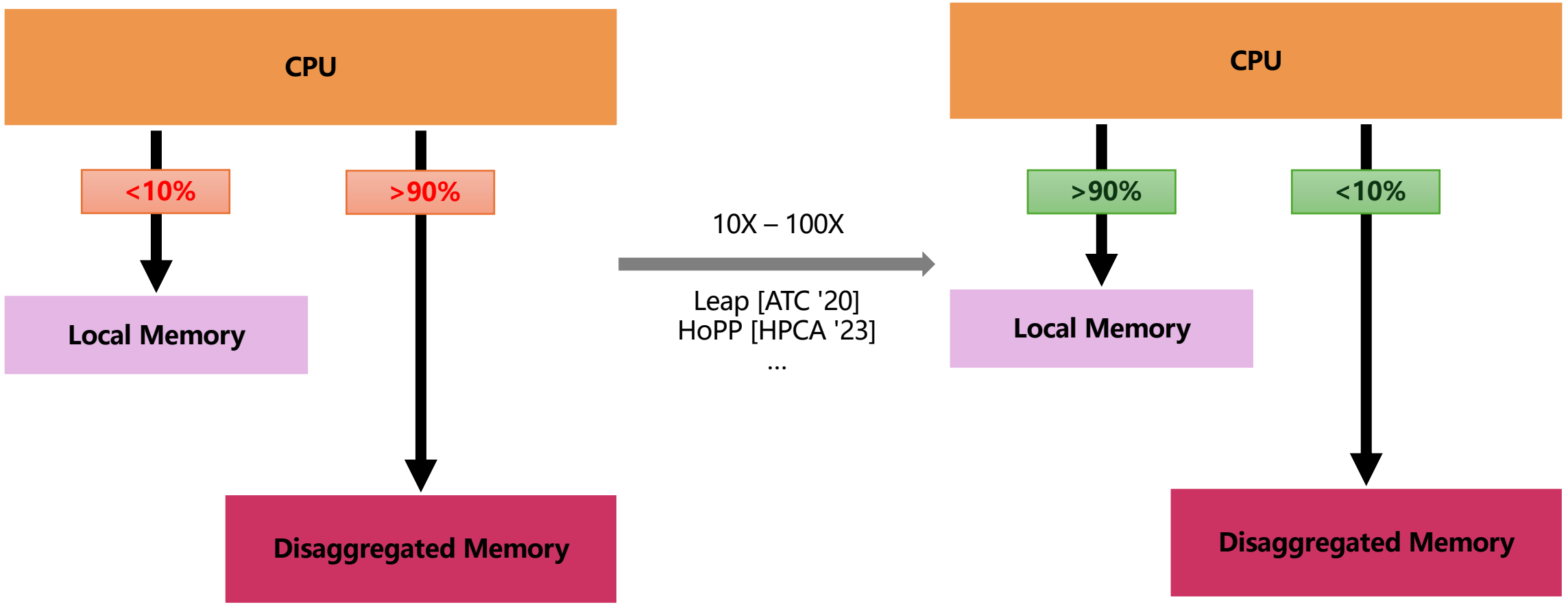
Cache Misses



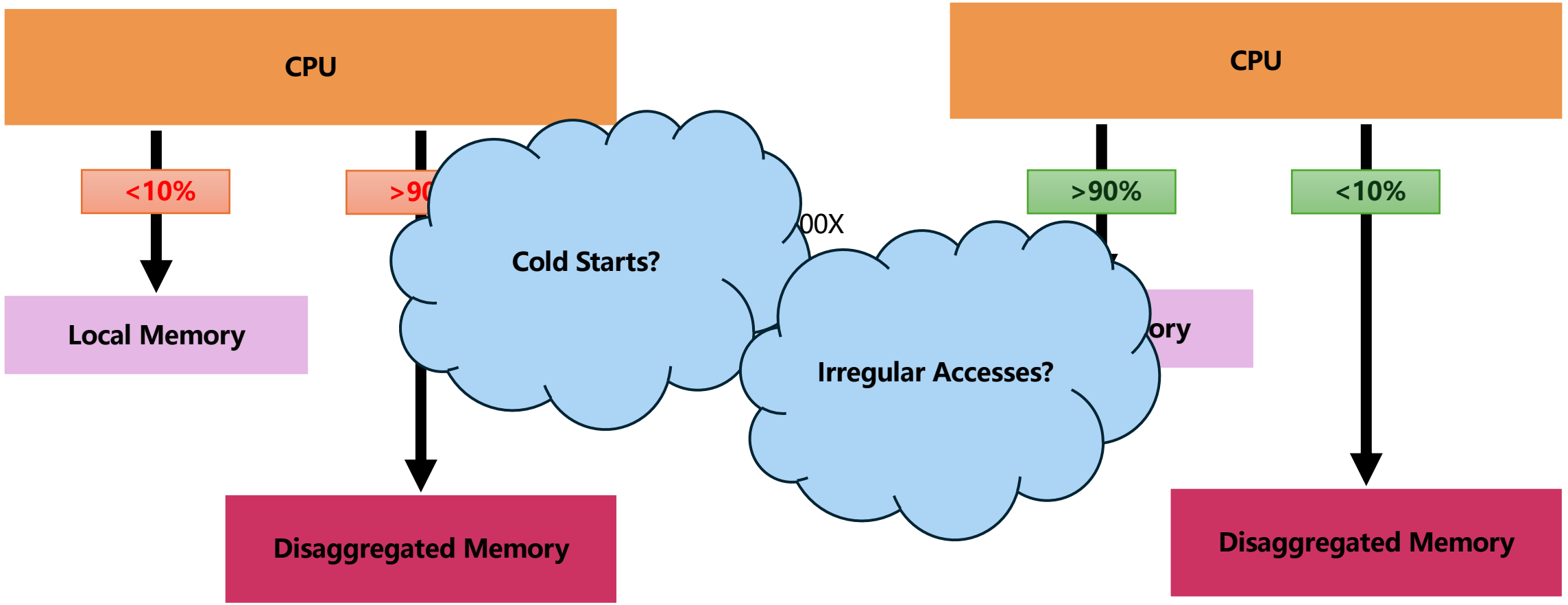
Cache Misses



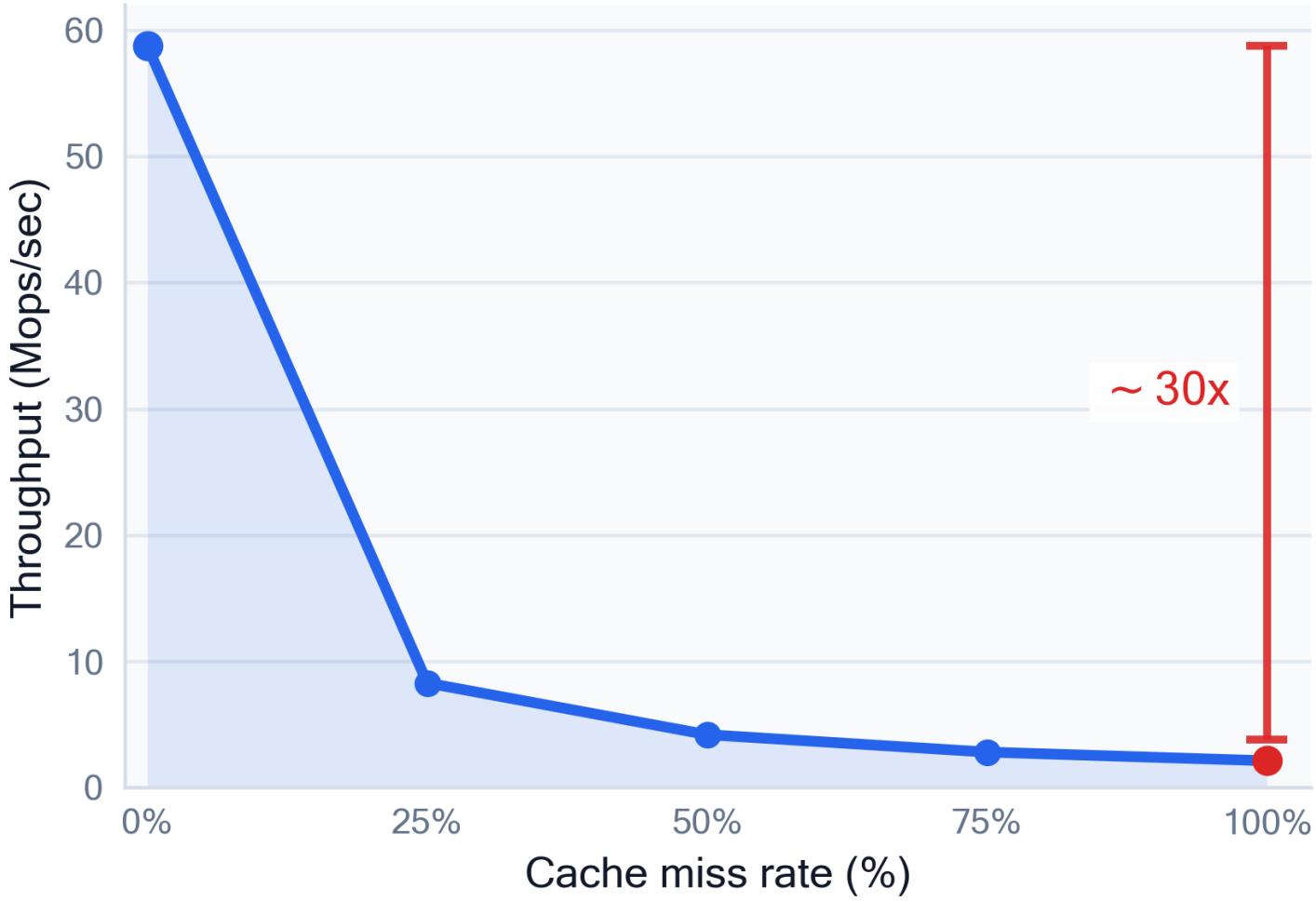
Cache Misses



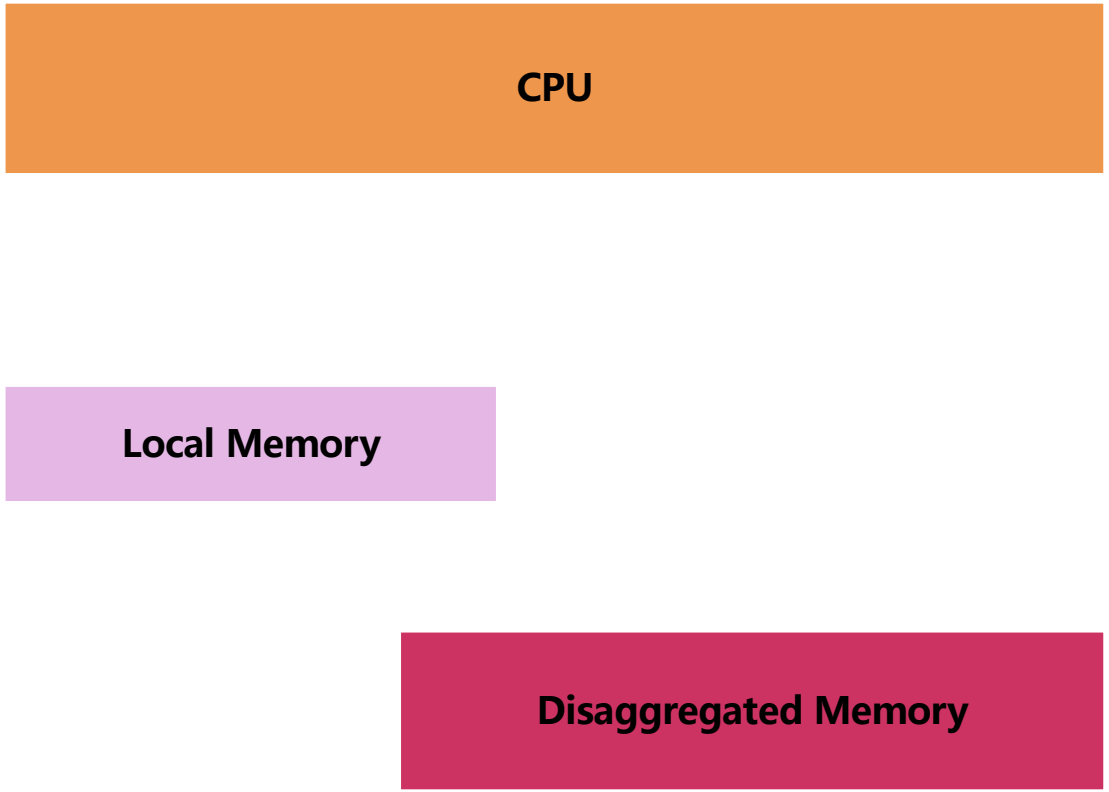
Cache Misses



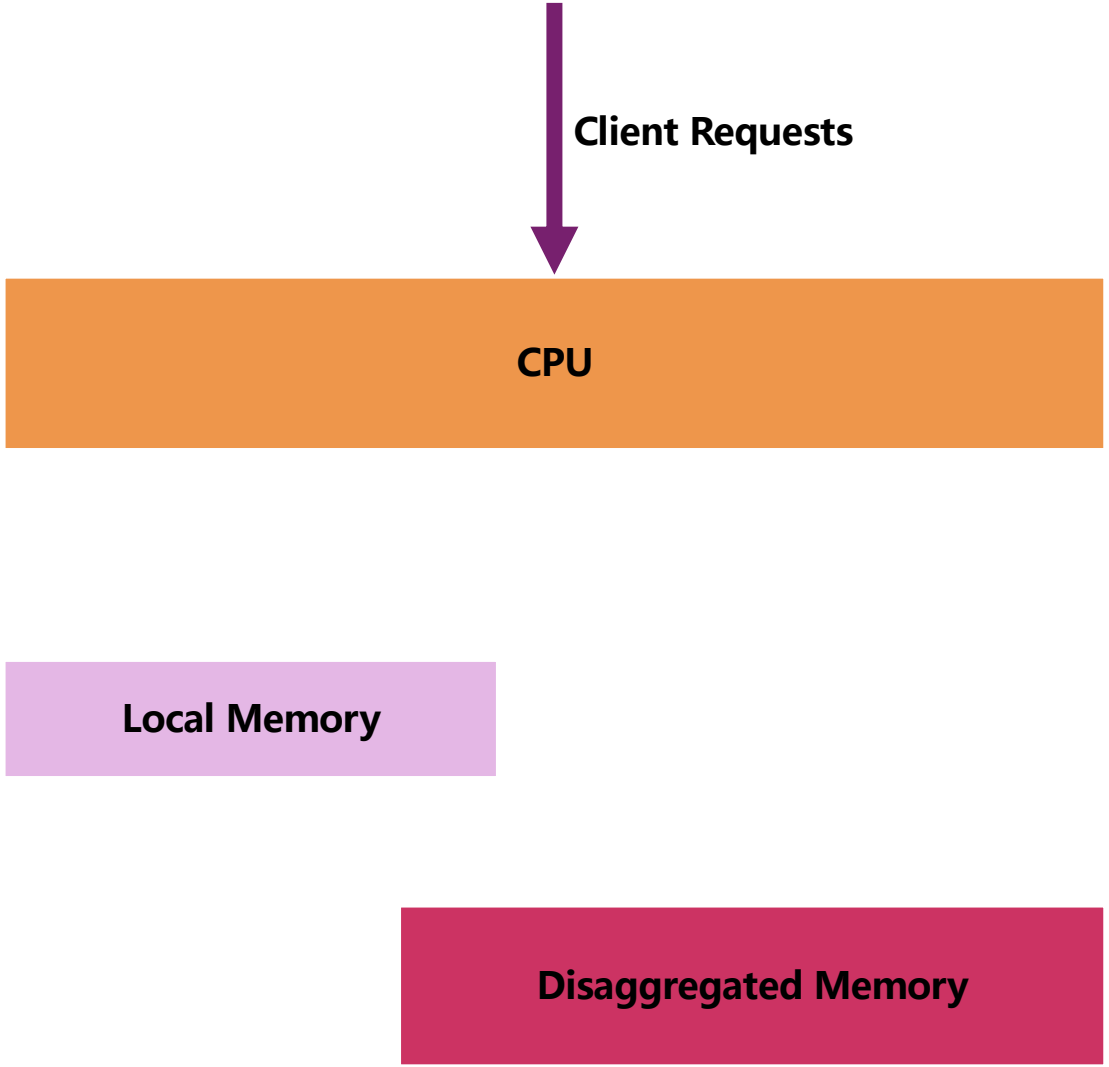
Cache Misses (HashMap)



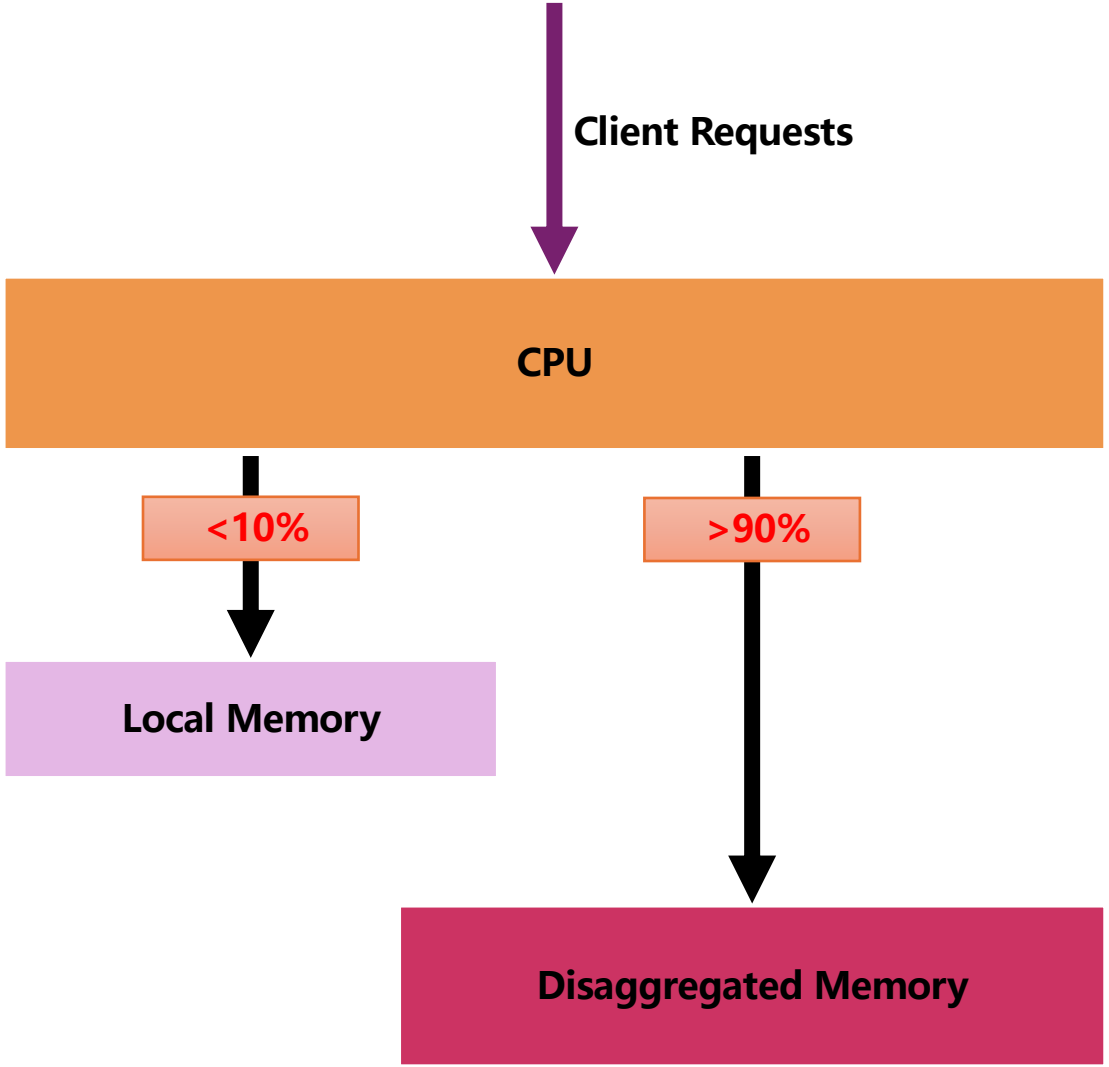
Our Proposal



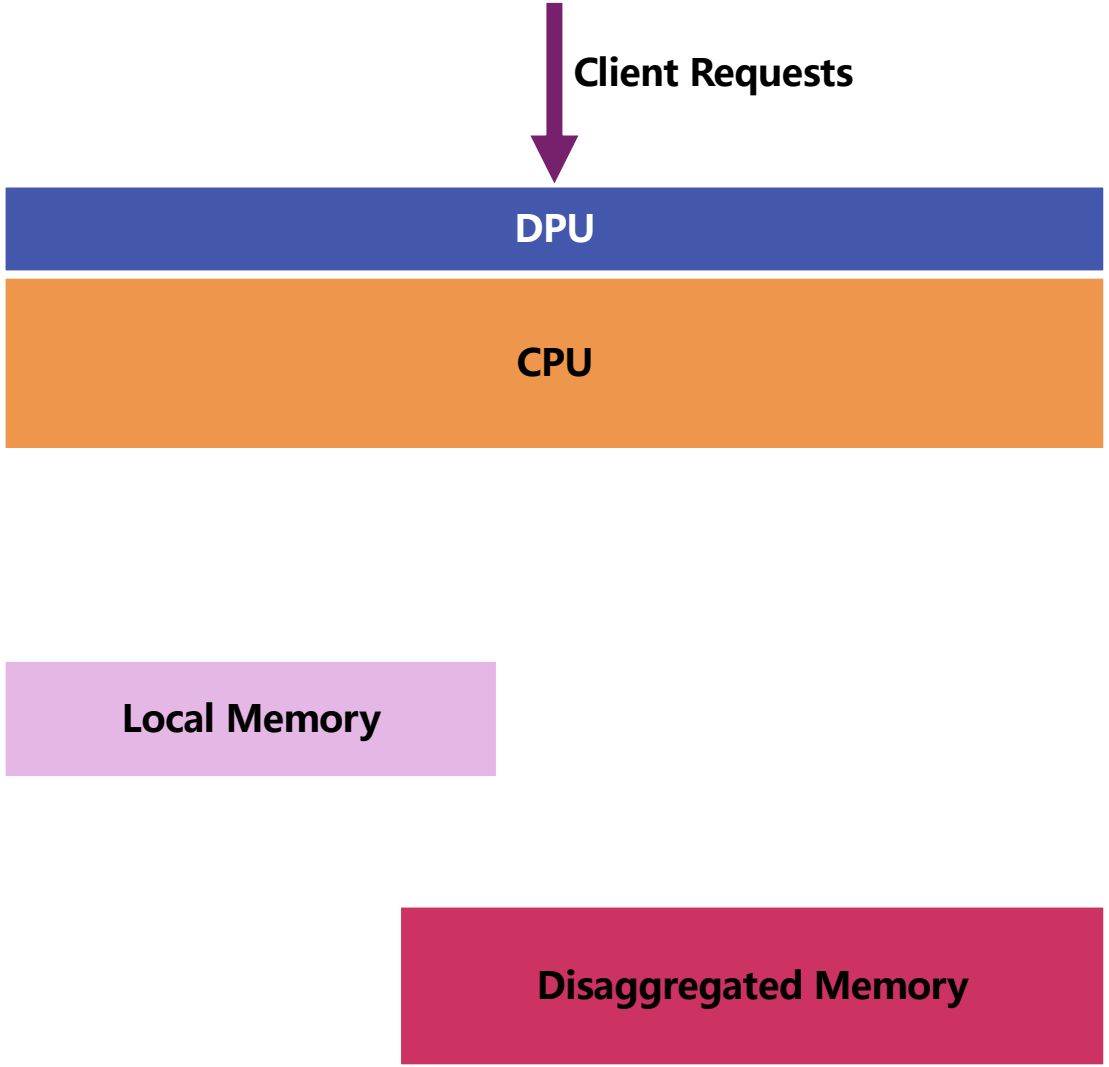
Our Proposal



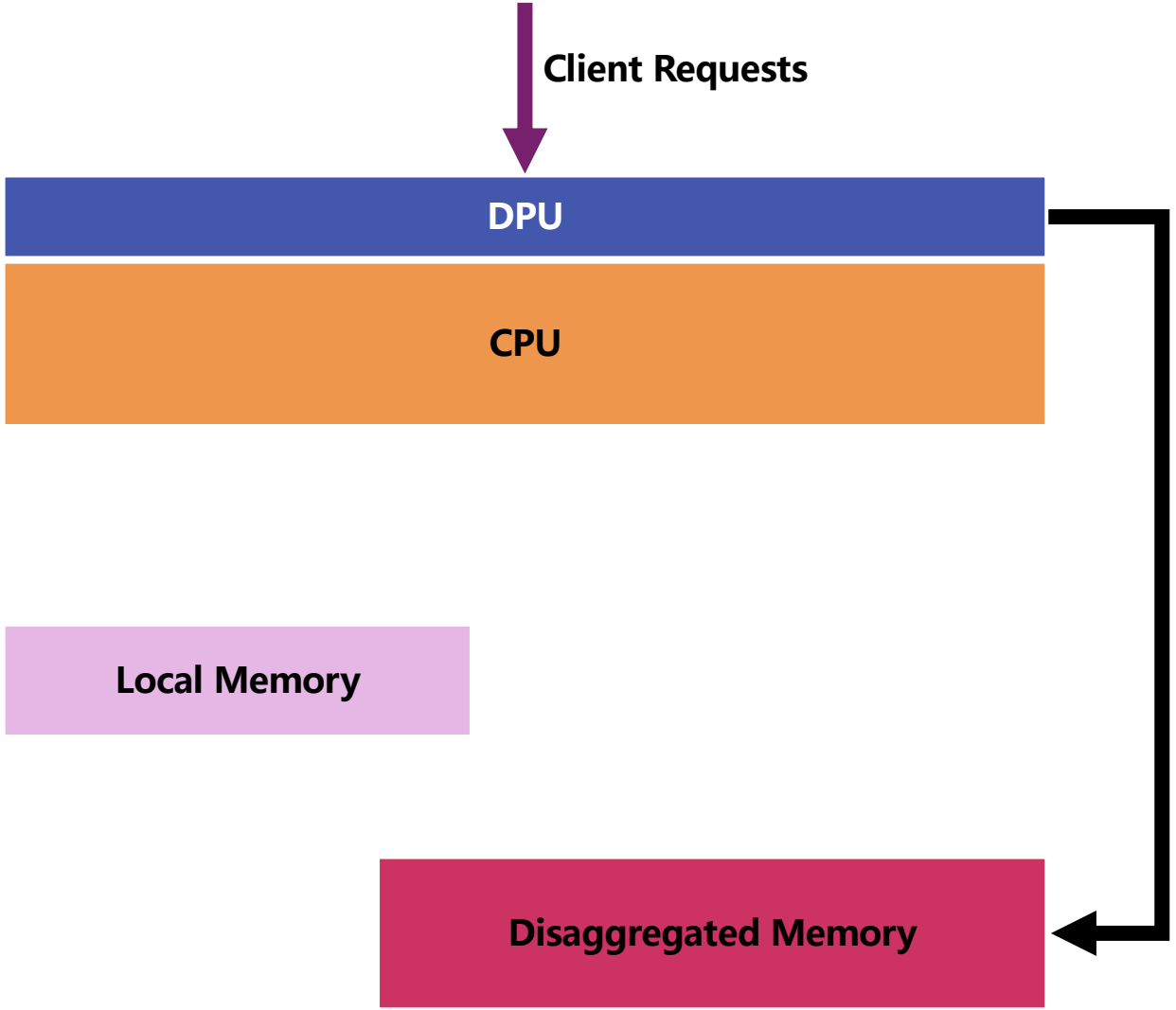
Our Proposal



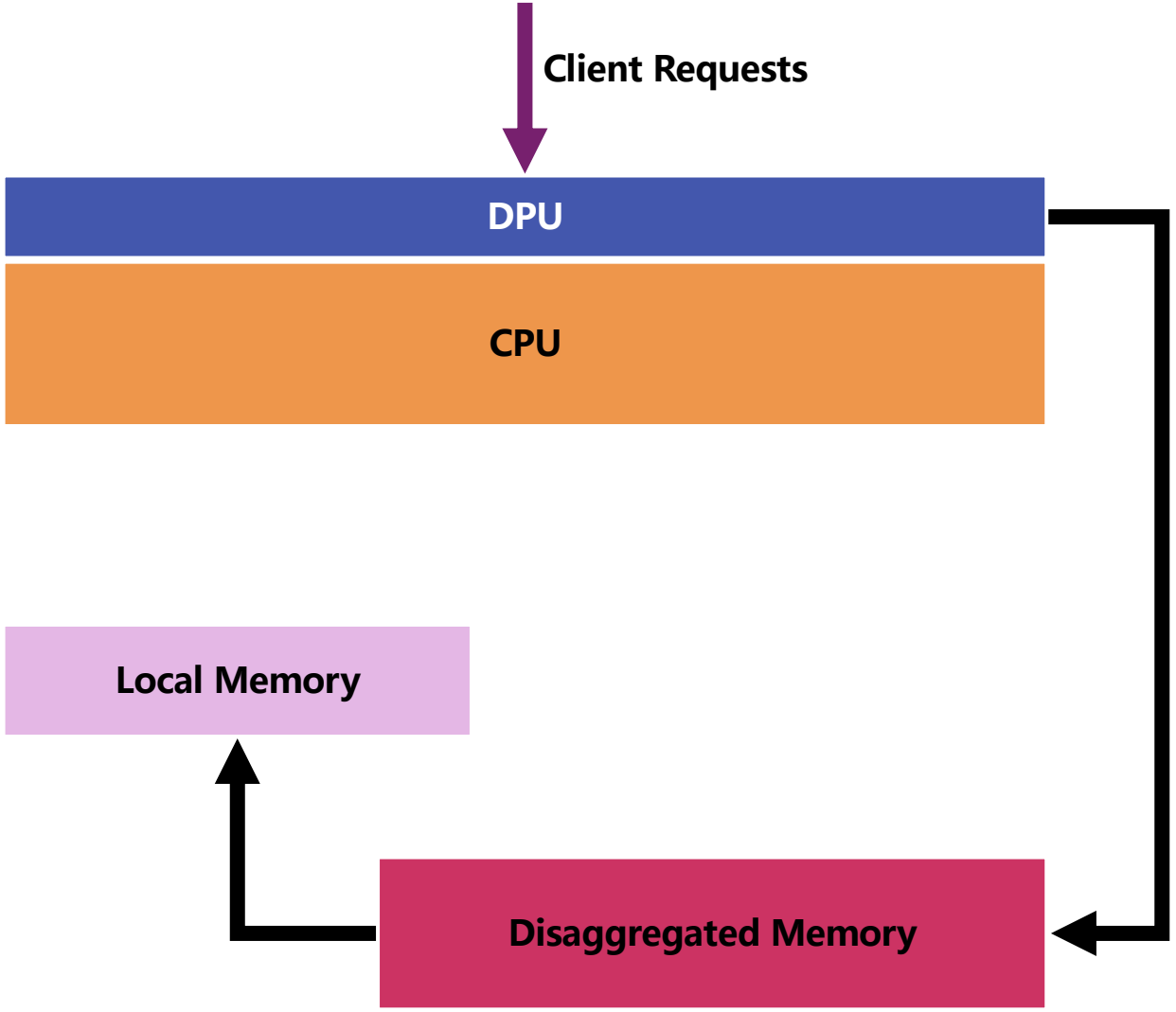
Our Proposal



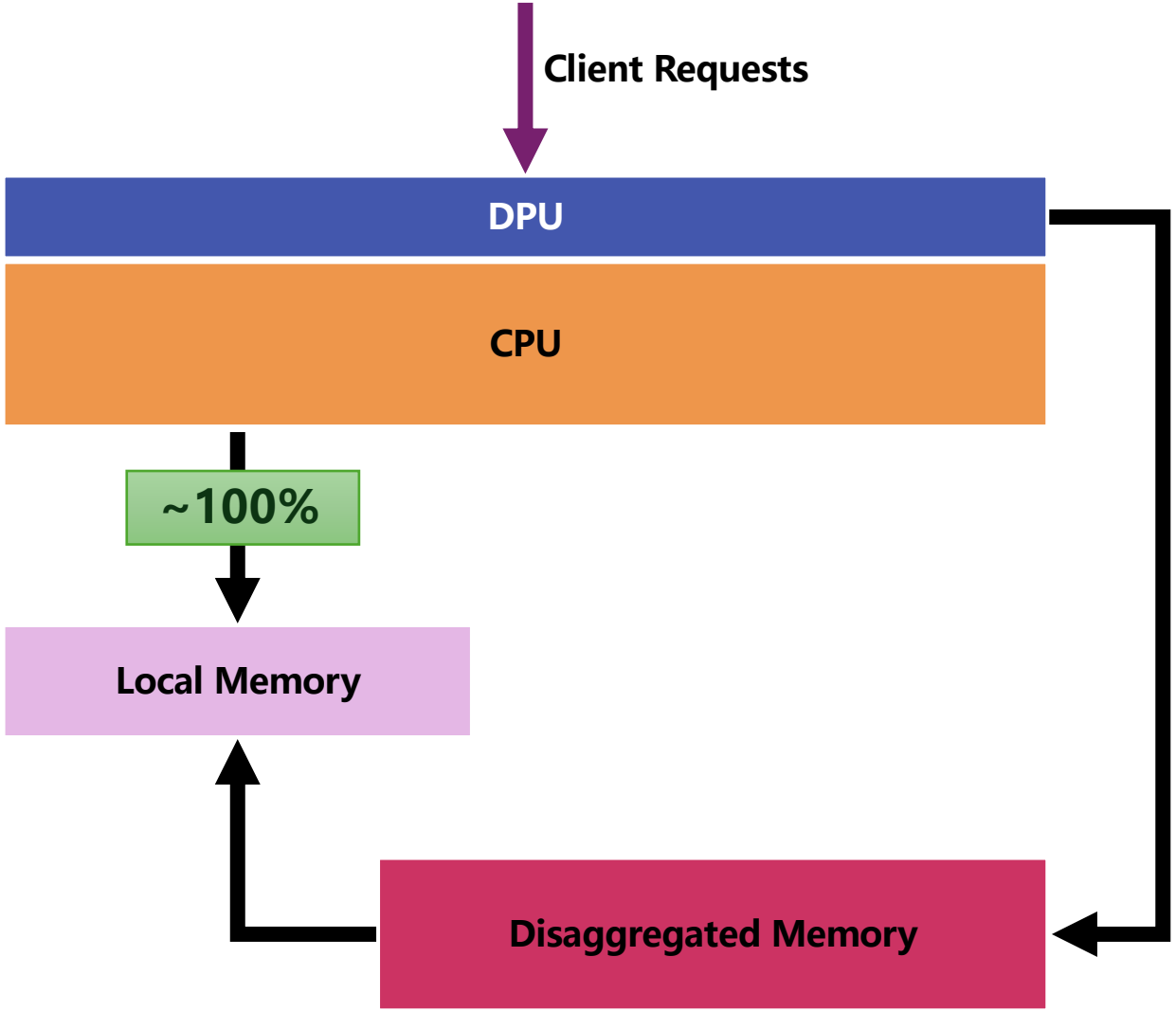
Our Proposal



Our Proposal



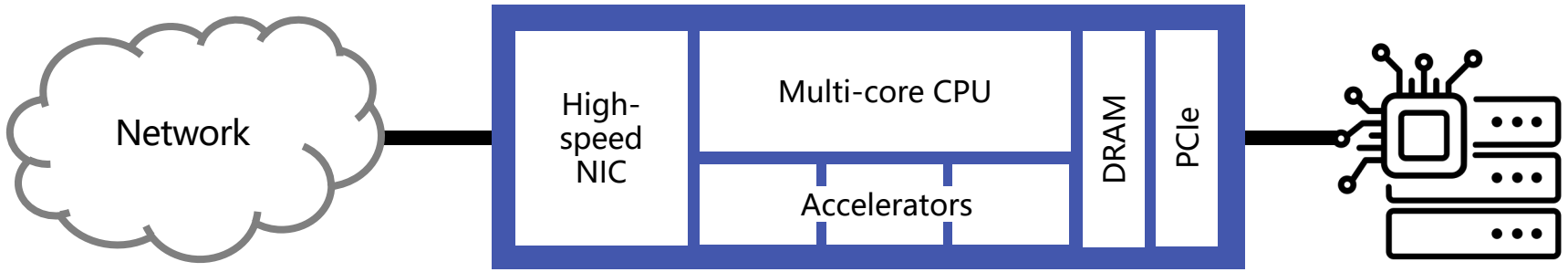
Our Proposal



Data Processing Units

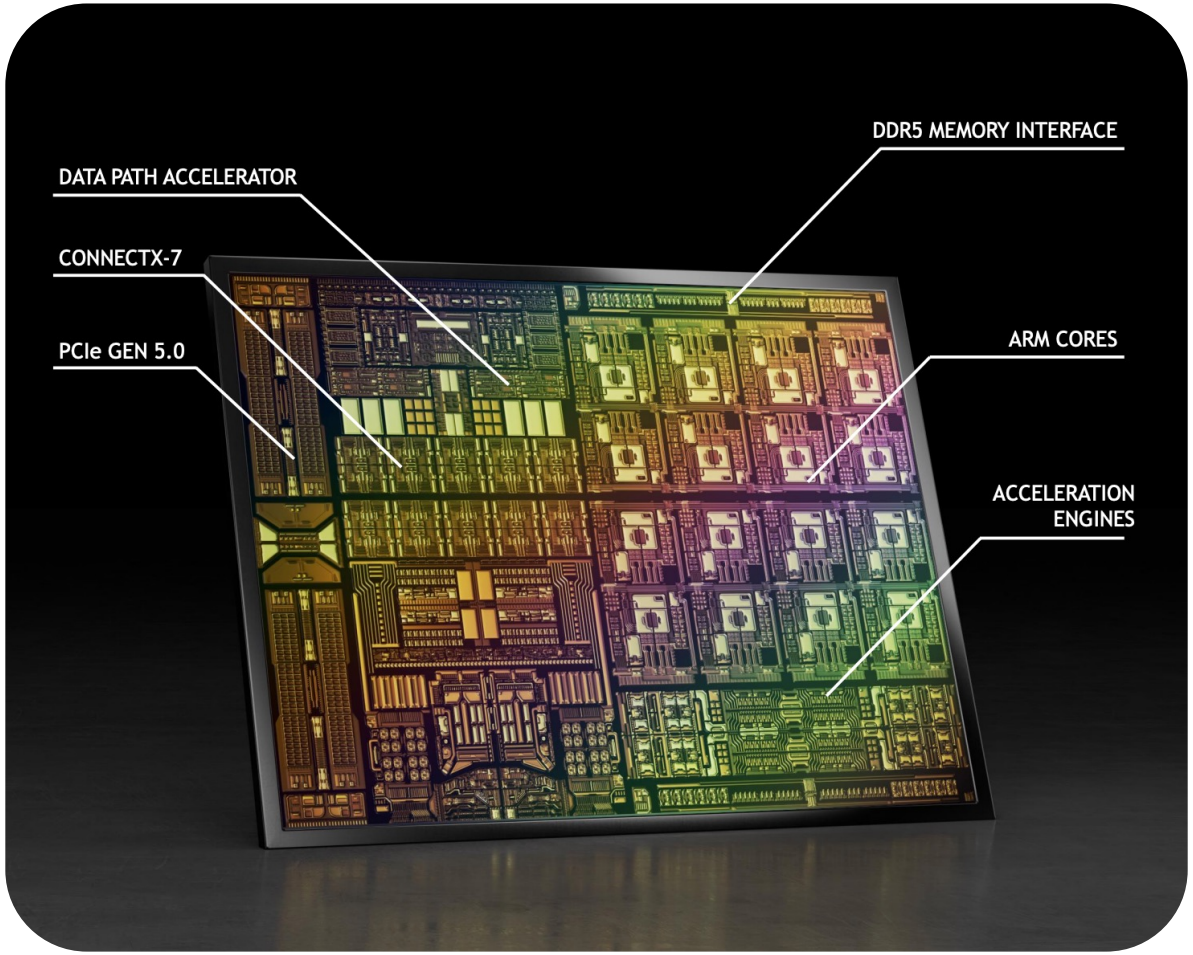
System-on-a-chip (SoC) SmartNIC

- **Multi-core CPU**: energy-efficient cores
- **High-speed network interface**: 100+ Gbps
- **Onboard memory**: 8-32GB
- **Data Accelerators**: DMA and more



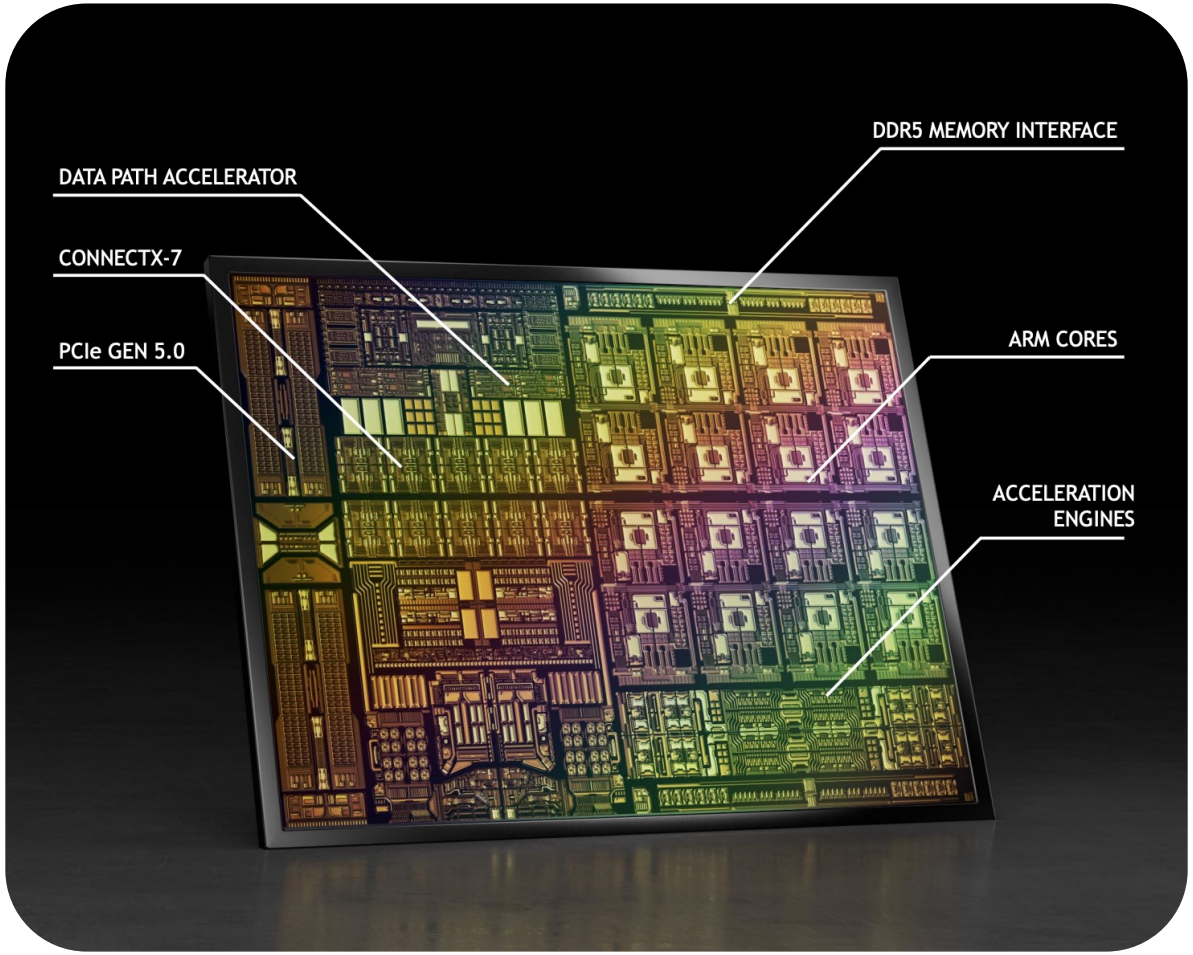
DPU Opportunities

- CPU cores tightly coupled with the NIC subsystem
- Highly Programmable
- Efficient access to hardware accelerators

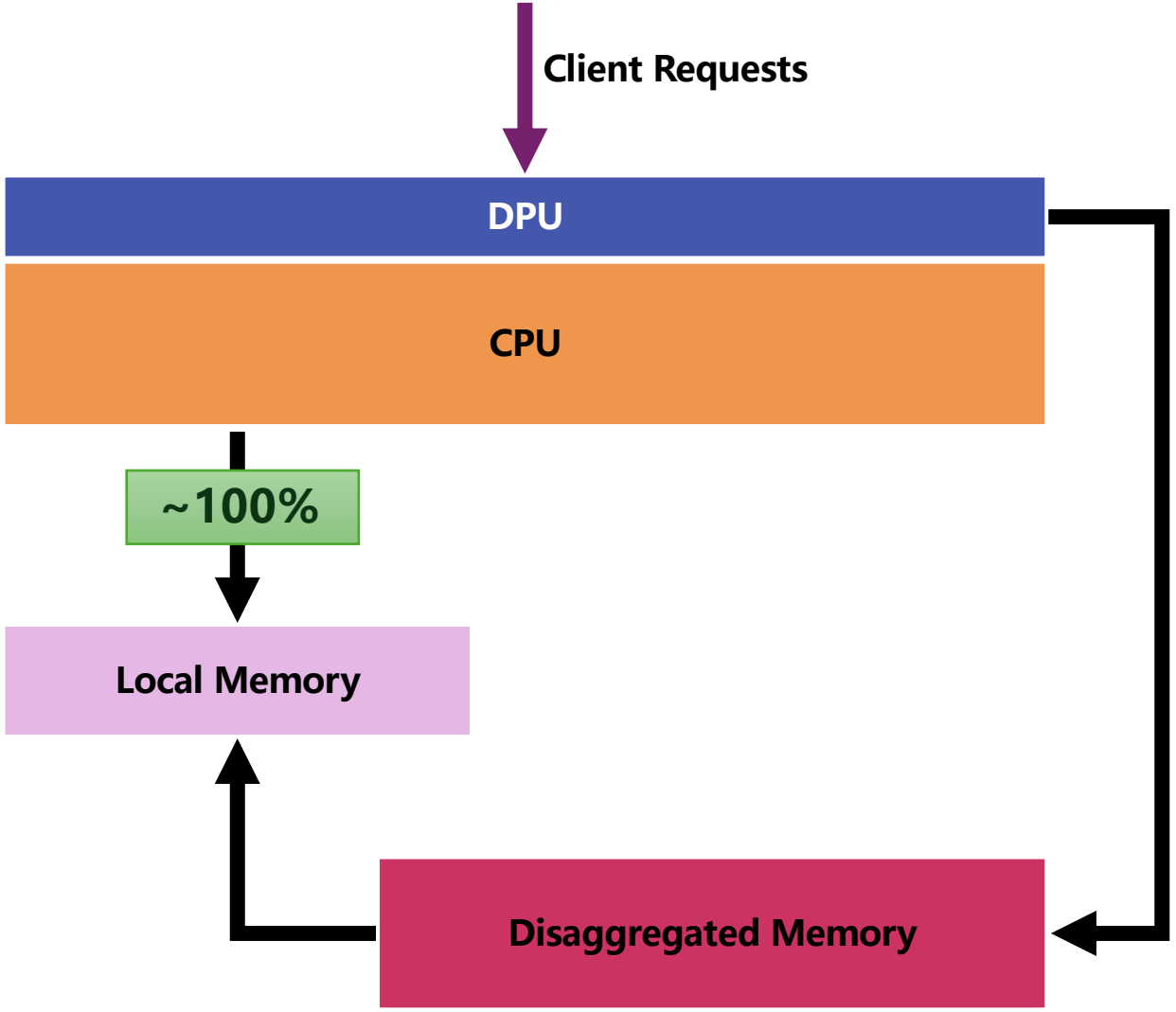


DPU Weaknesses

- Wimpy CPU cores
- Lower memory capacity
- Lower memory bandwidth

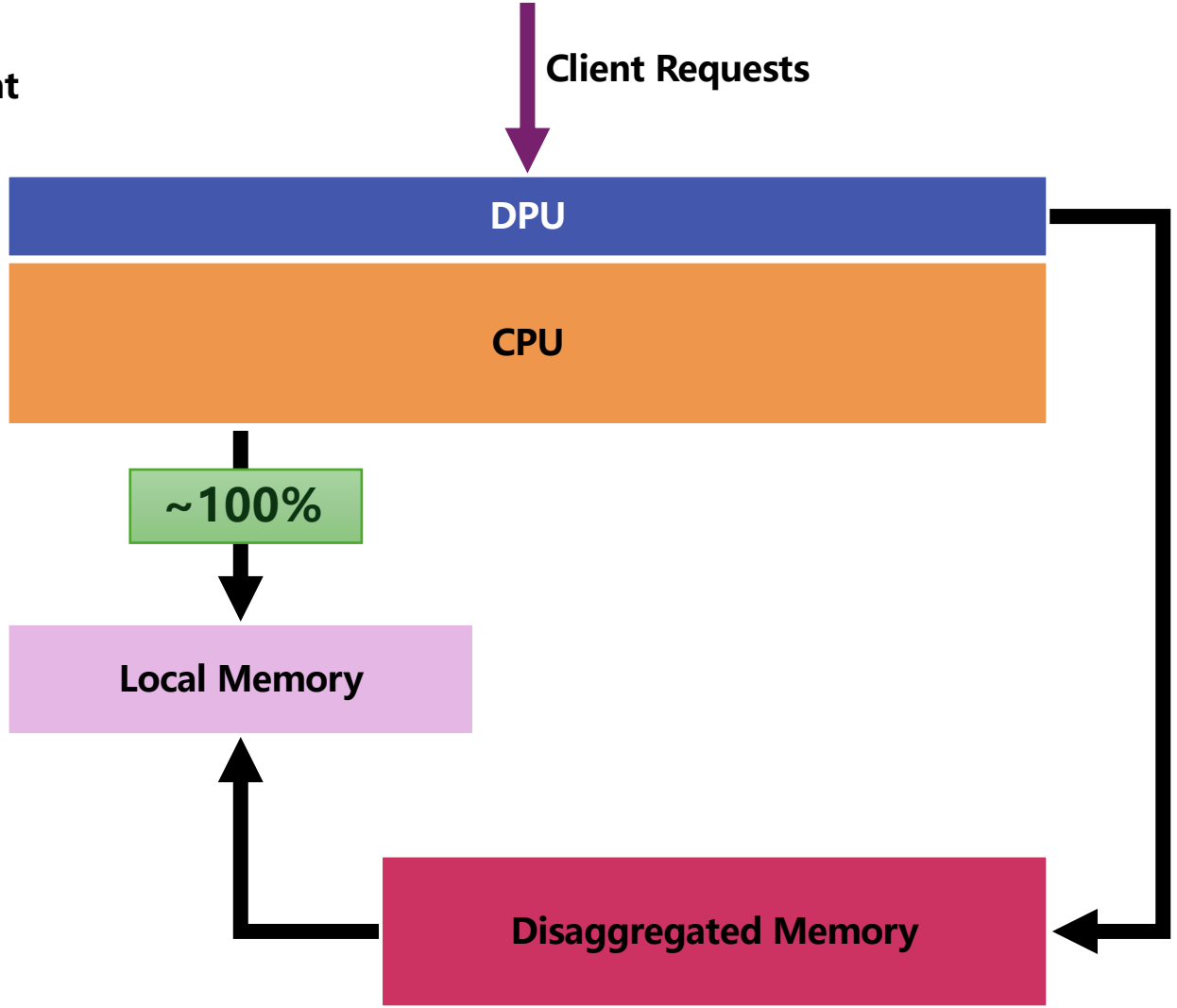


Challenges


Client Requests
DPU
CPU
~100%
Local Memory
Disaggregated Memory

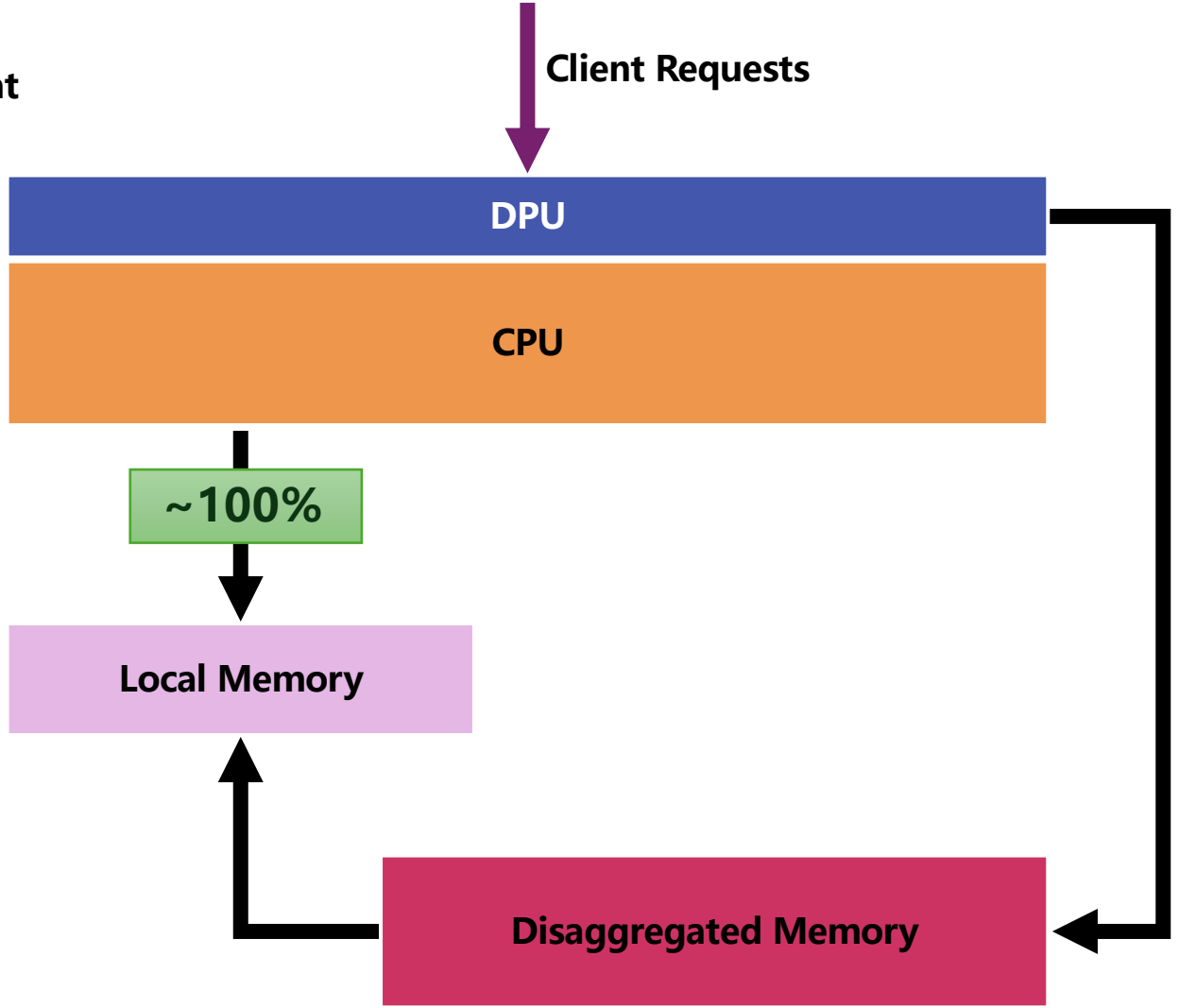
Challenges

How do we determine what to prefetch?



Challenges

How do we determine what to prefetch?



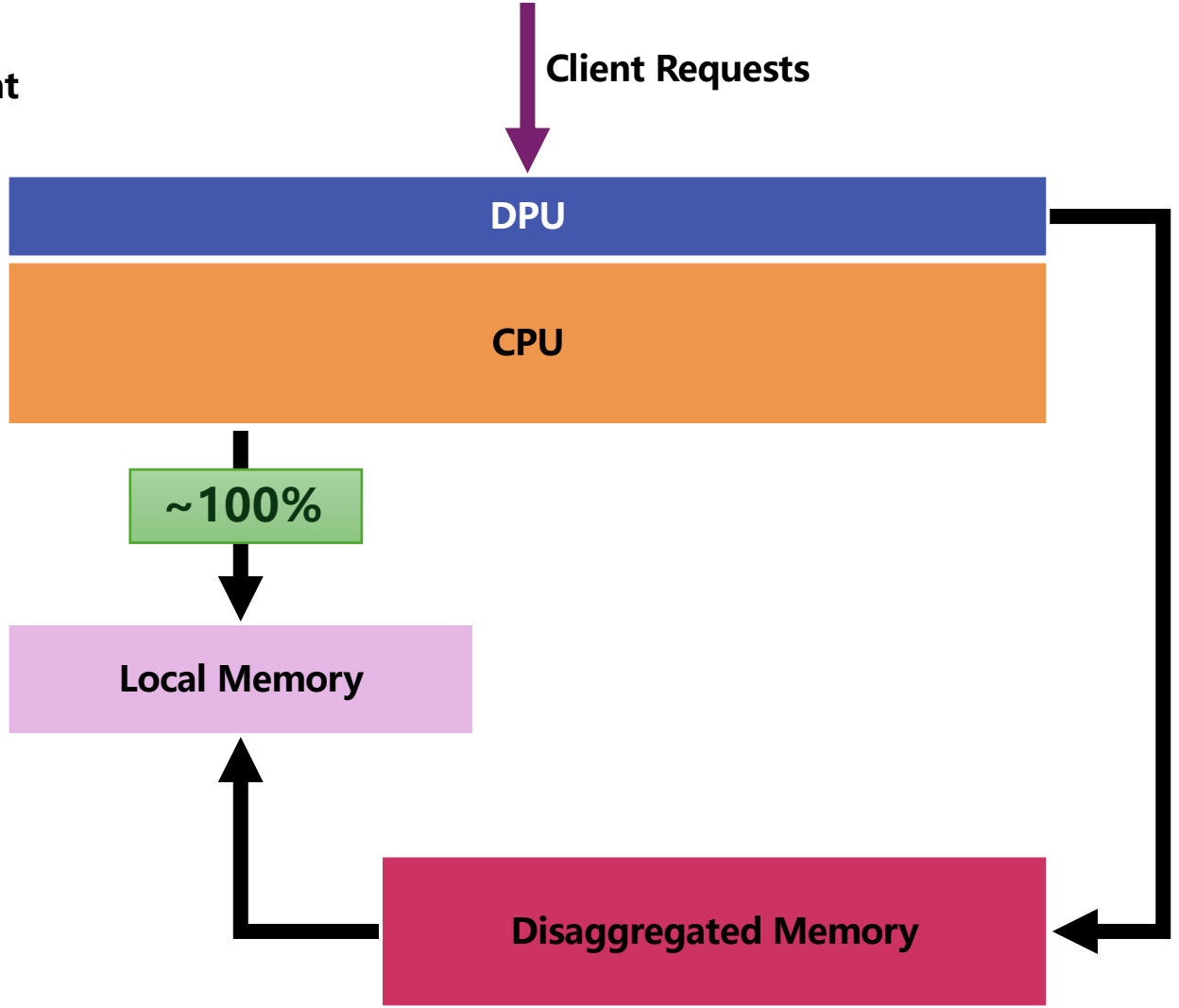
How do we prefetch efficiently?

Challenges

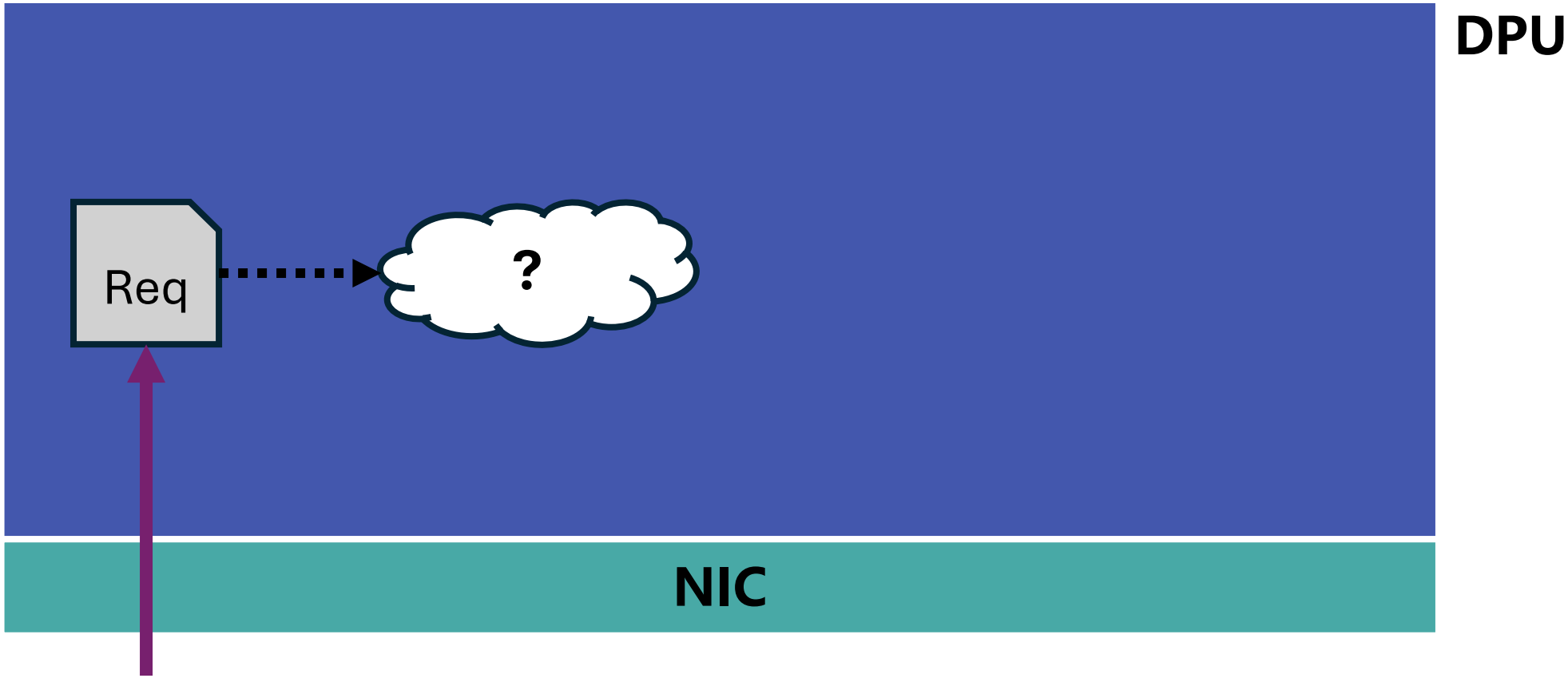
How do we determine what to prefetch?

How do we serve the data items for fast application access?

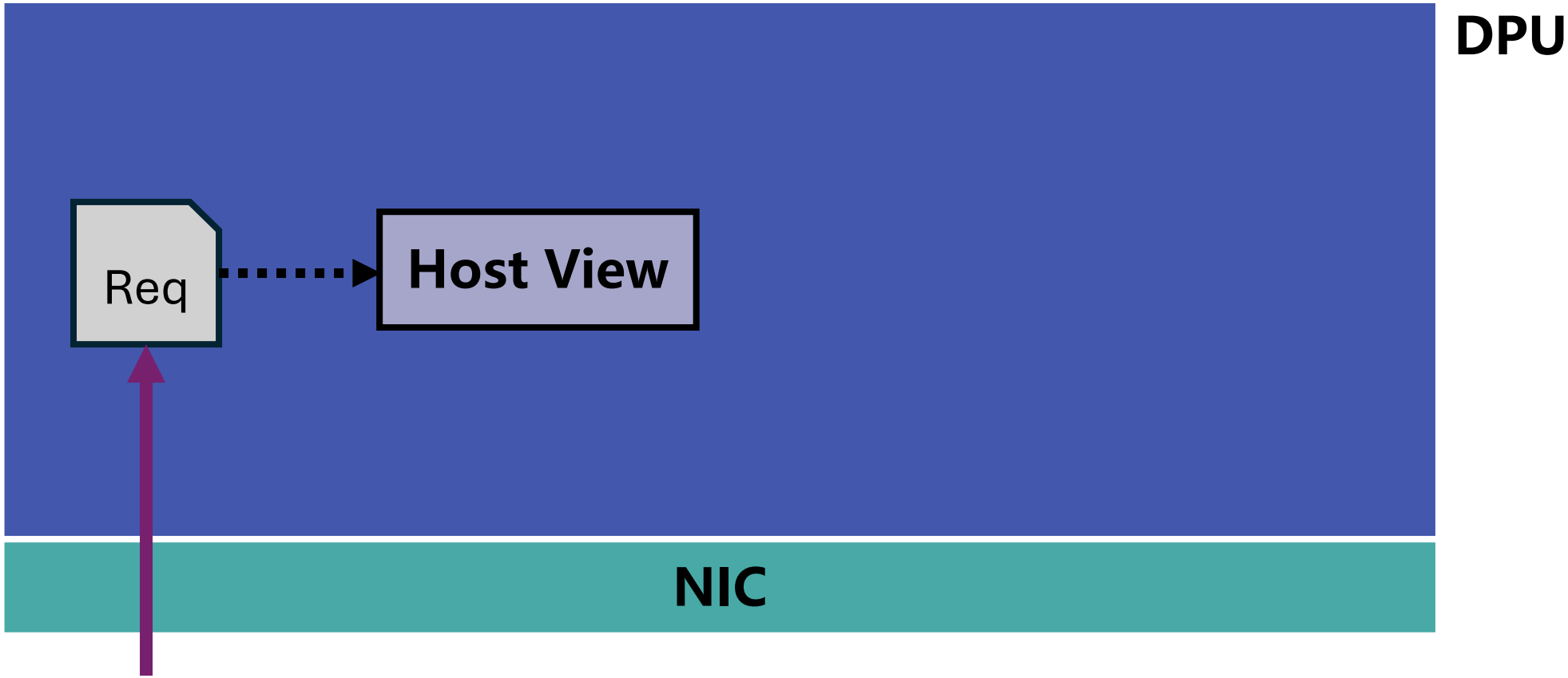
How do we prefetch efficiently?



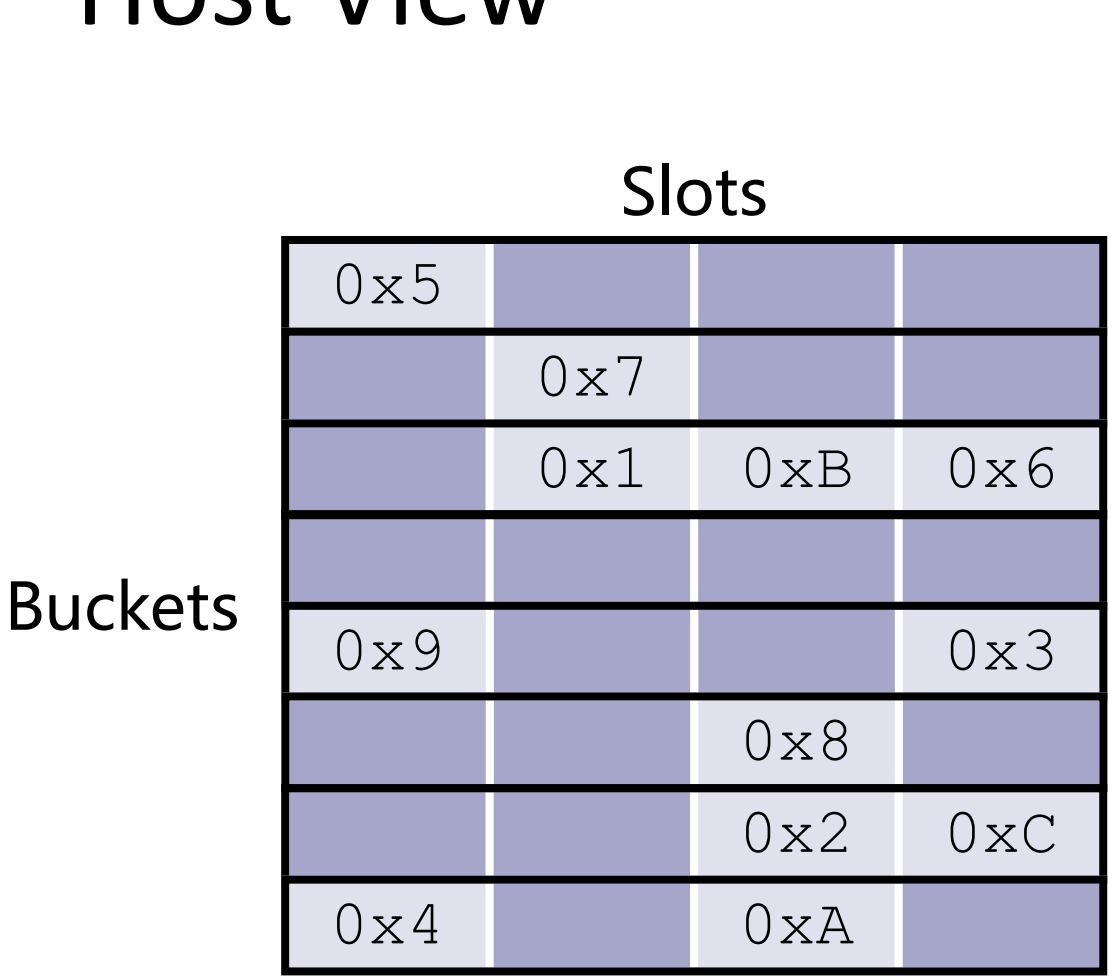
Determine Cache Miss



Determine Cache Miss

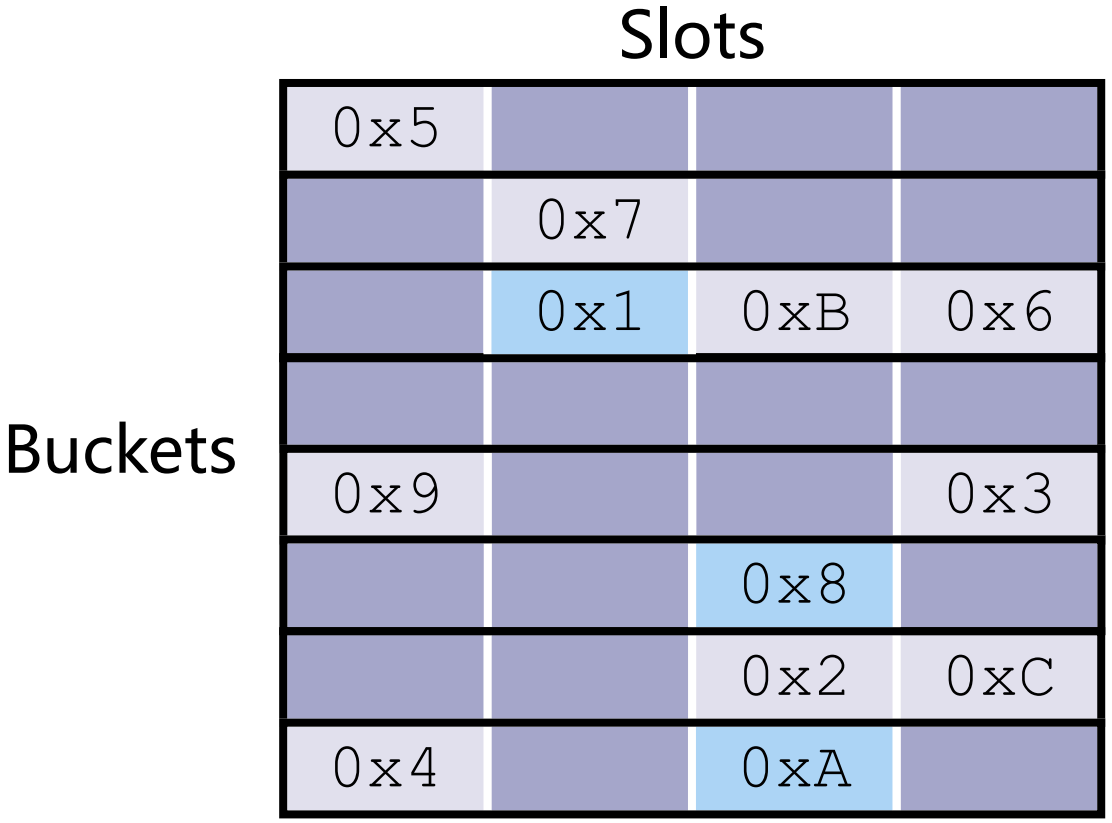


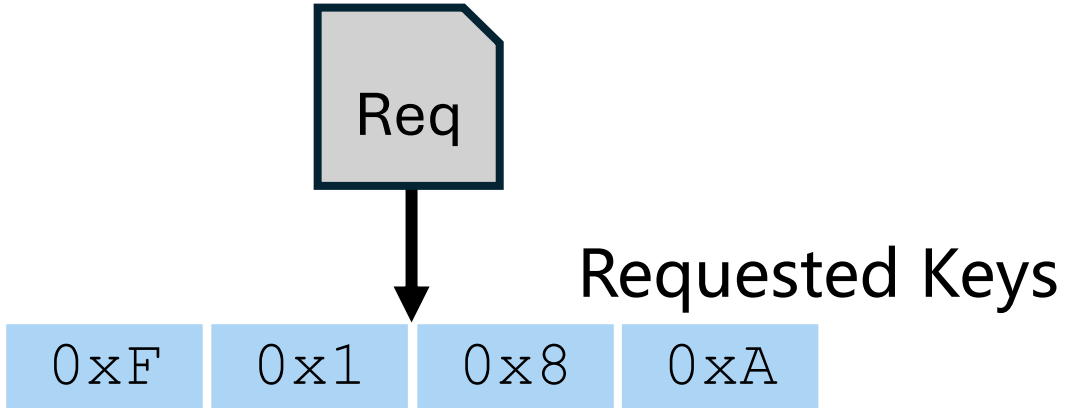
Host View



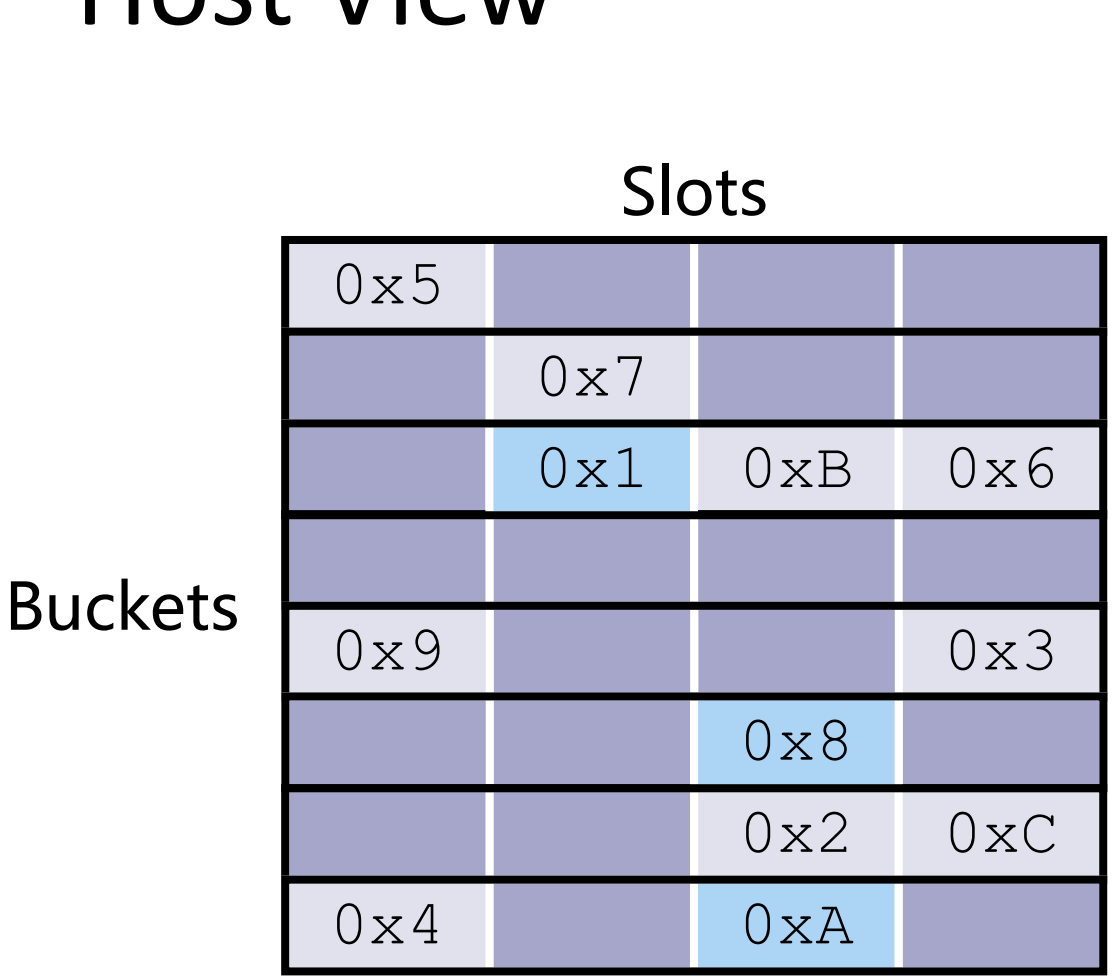
Req

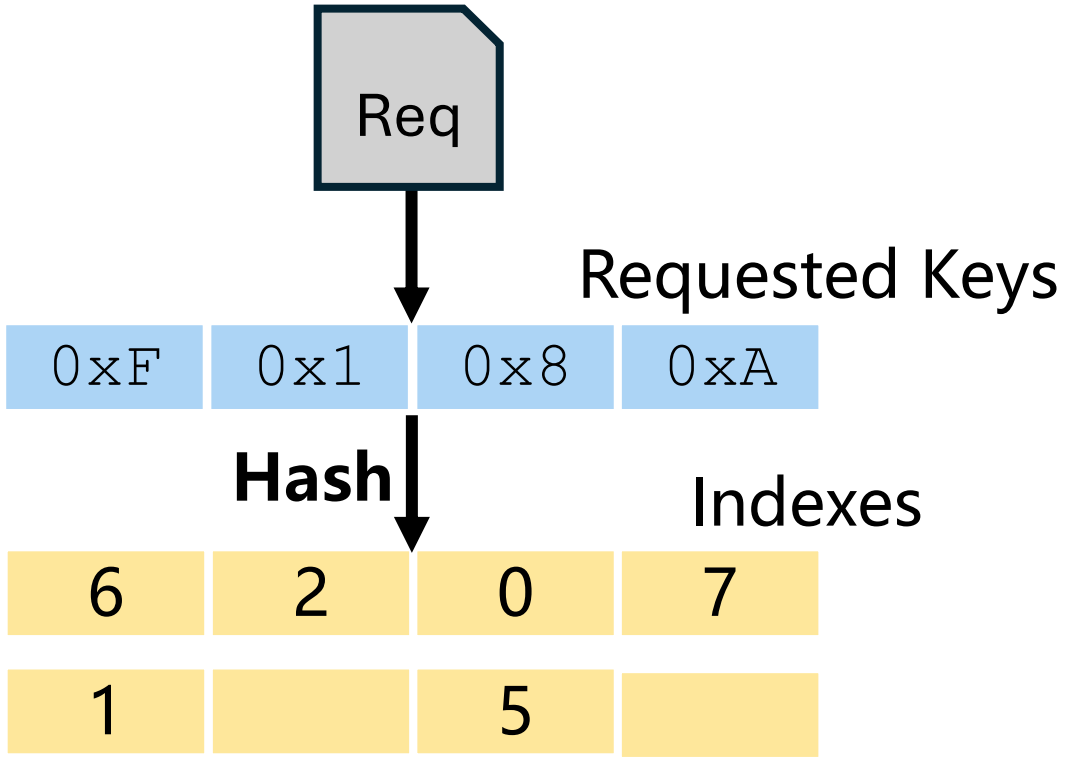
Host View


Slots
Buckets
0x5
0x7
0x1
0xB
0x6
0x9
0x3
0x8
0x2
0xC
0x4
0xA

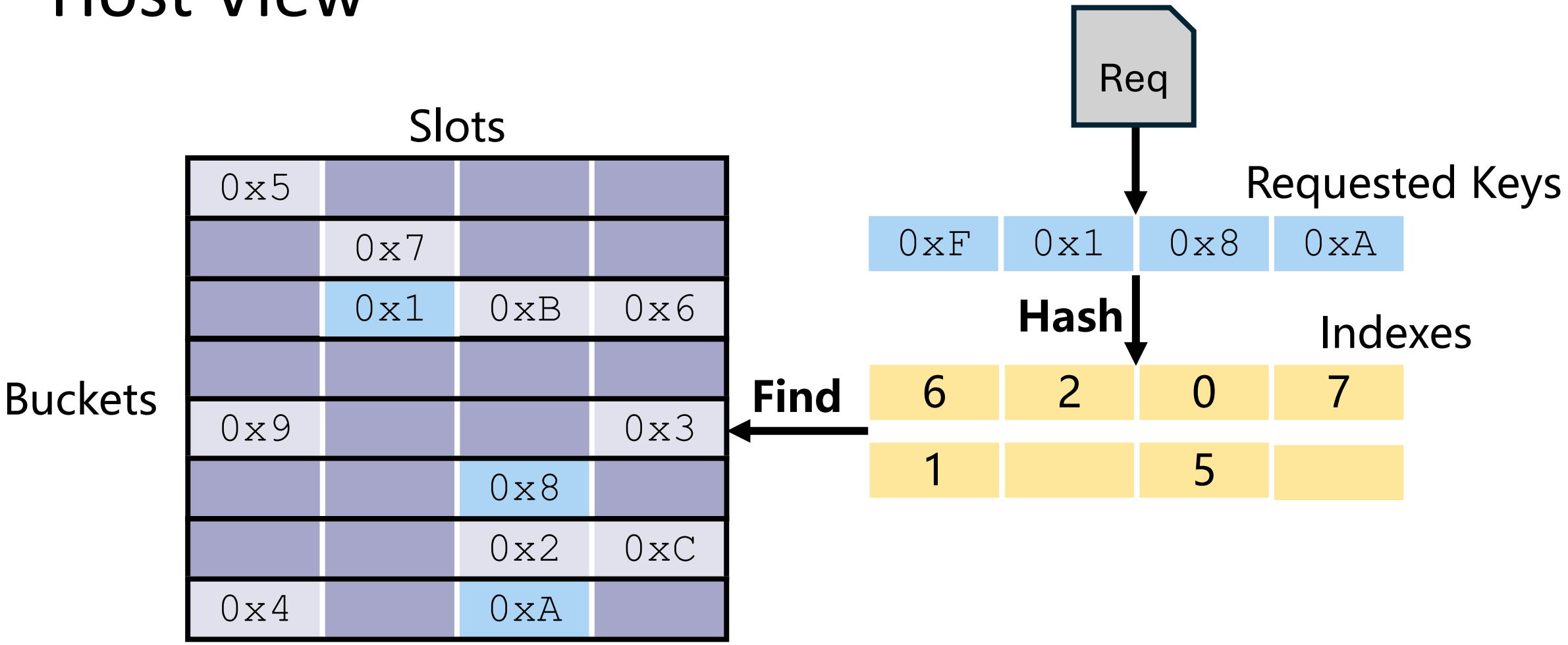

Req
Requested Keys
0xF
0x1
0x8
0xA

Host View

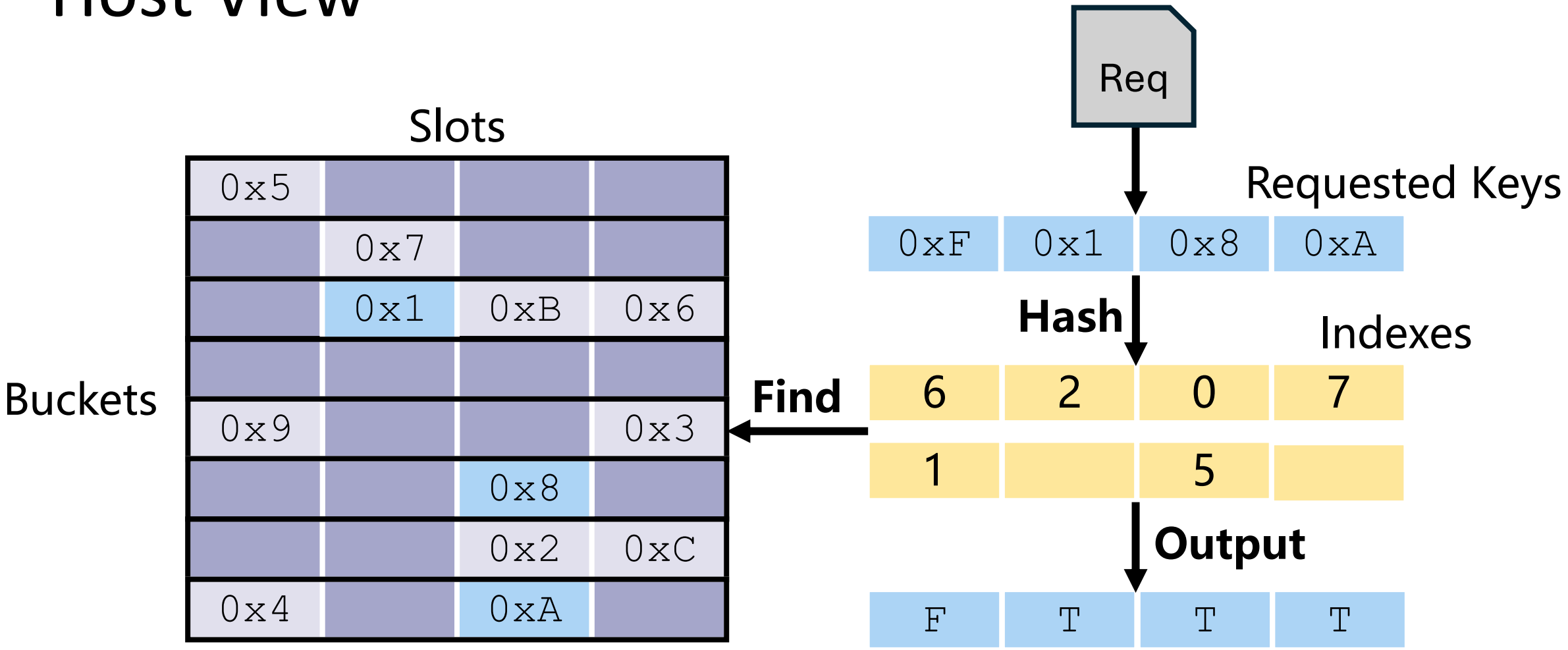

Slots
Buckets
0x5
0x7
0x1
0xB
0x6
0x9
0x3
0x8
0x2
0xC
0x4
0xA


Req
Requested Keys
0xF
0x1
0x8
0xA
Hash
Indexes
6
2
0
7
1
5

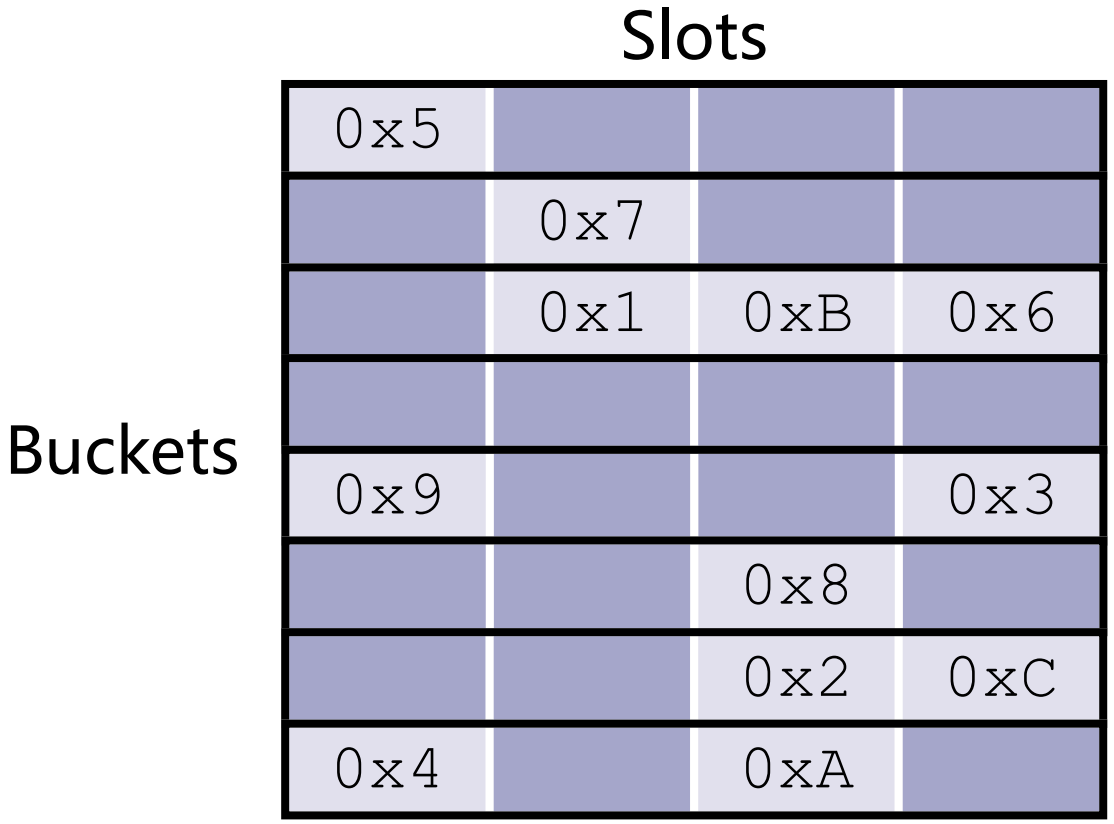
Host View

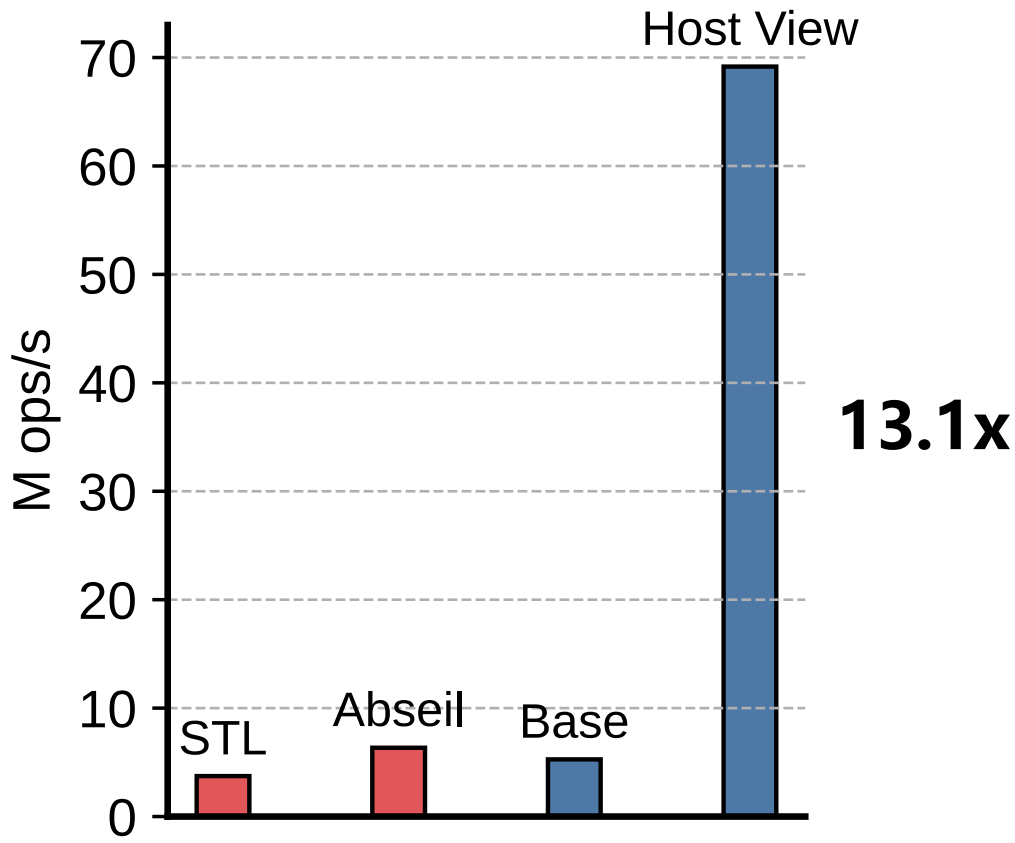

Req
Requested Keys
0xF
0x1
0x8
0xA
Hash
Indexes
6
2
0
7
1
5
Find
Slots
Buckets
0x5
0x7
0x1
0xB
0x6
0x9
0x3
0x8
0x2
0xC
0x4
0xA

Host View

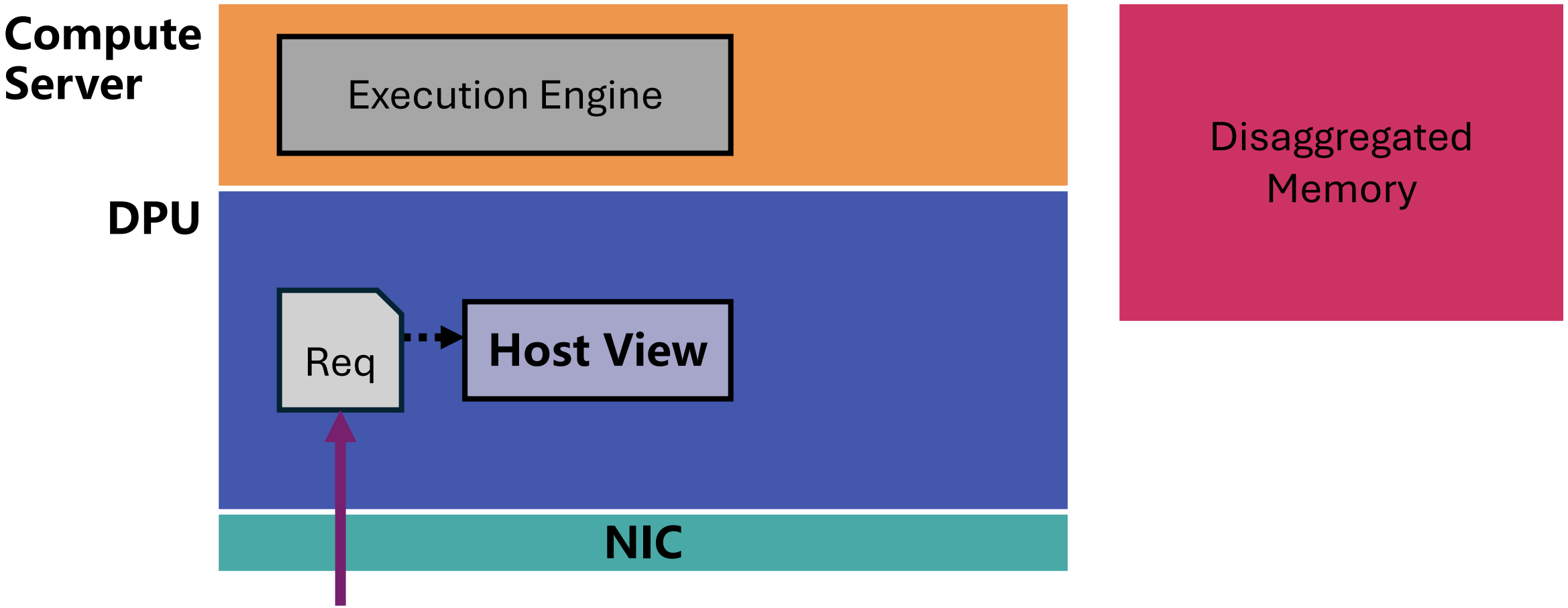

Slots
Buckets
0x5
0x7
0x1
0xB
0x6
0x9
0x3
0x8
0x2
0xC
0x4
0xA
Find
Req
Requested Keys
0xF
0x1
0x8
0xA
Hash
Indexes
6
2
0
7
1
5
Output
F
T
T
T

Host View

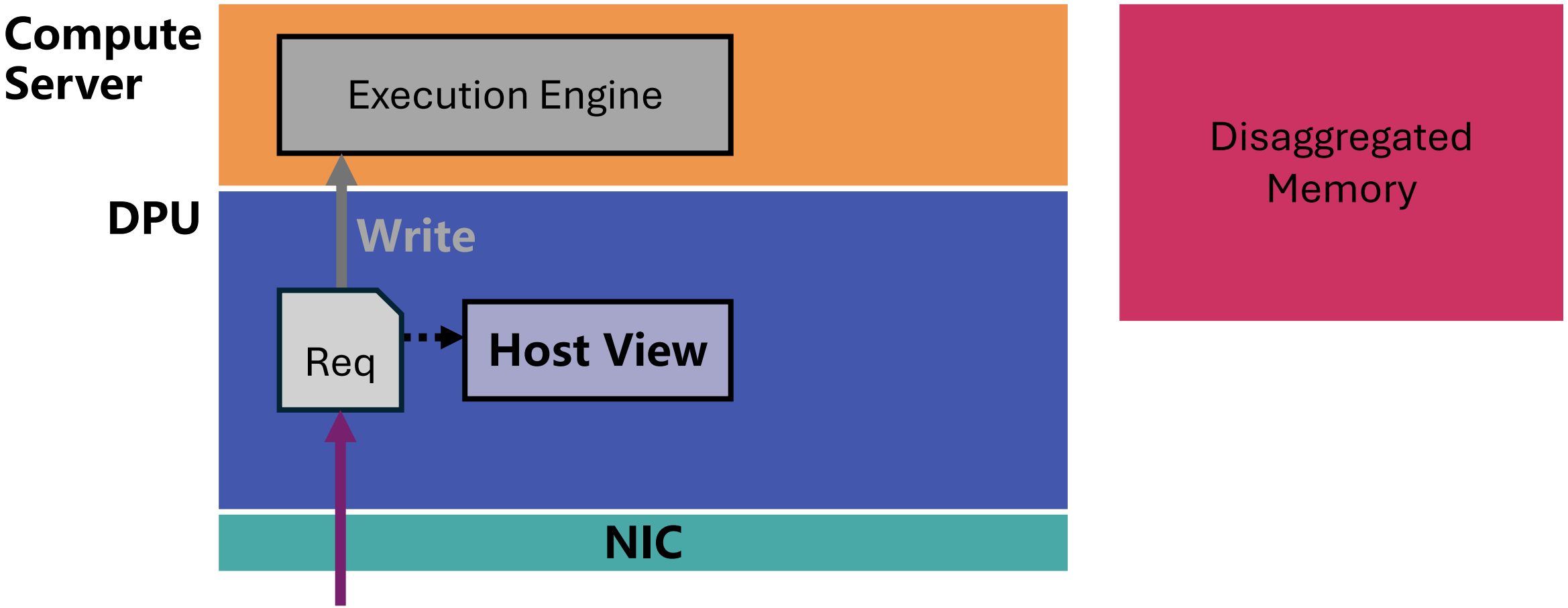

Slots
Buckets
0x5
0x7
0x1
0xB
0x6
0x9
0x3
0x8
0x2
0xC
0x4
0xA


Host View
M ops/s
70
60
50
40
30
20
10
0
STL
Abseil
Base
13.1x

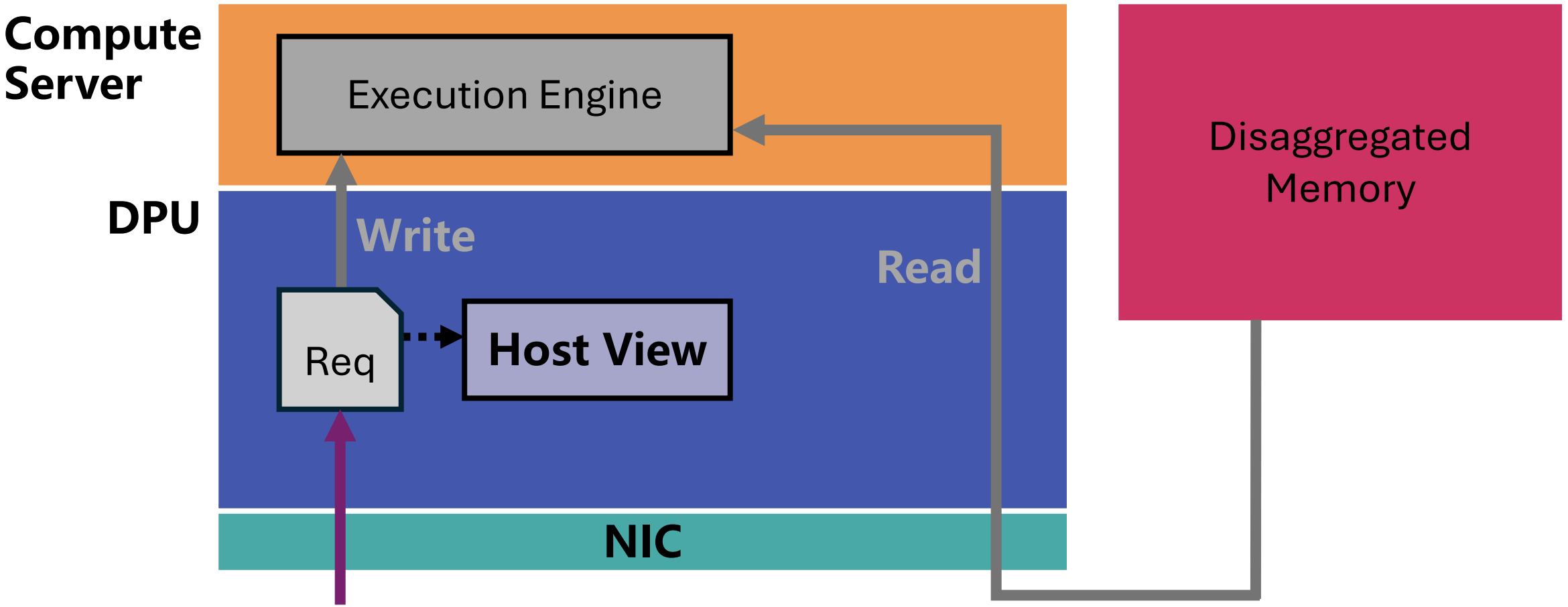
Consistency Considerations



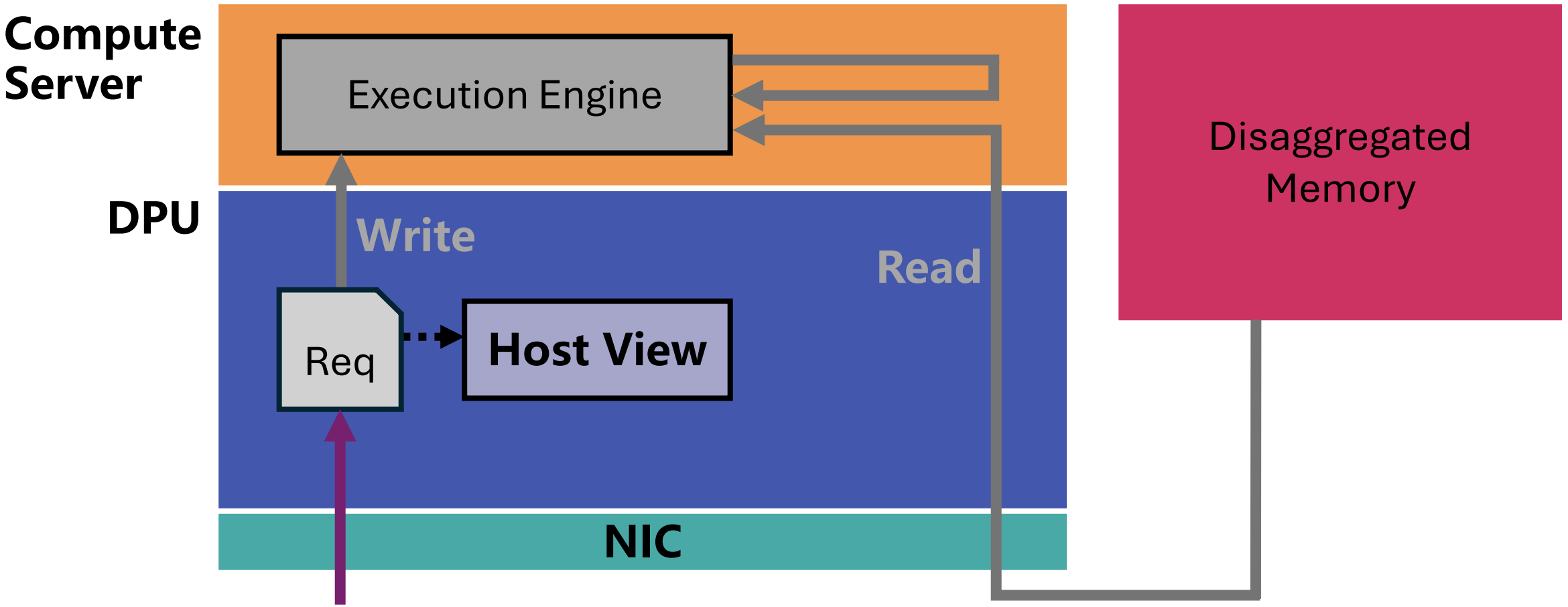
Consistency Considerations



Consistency Considerations

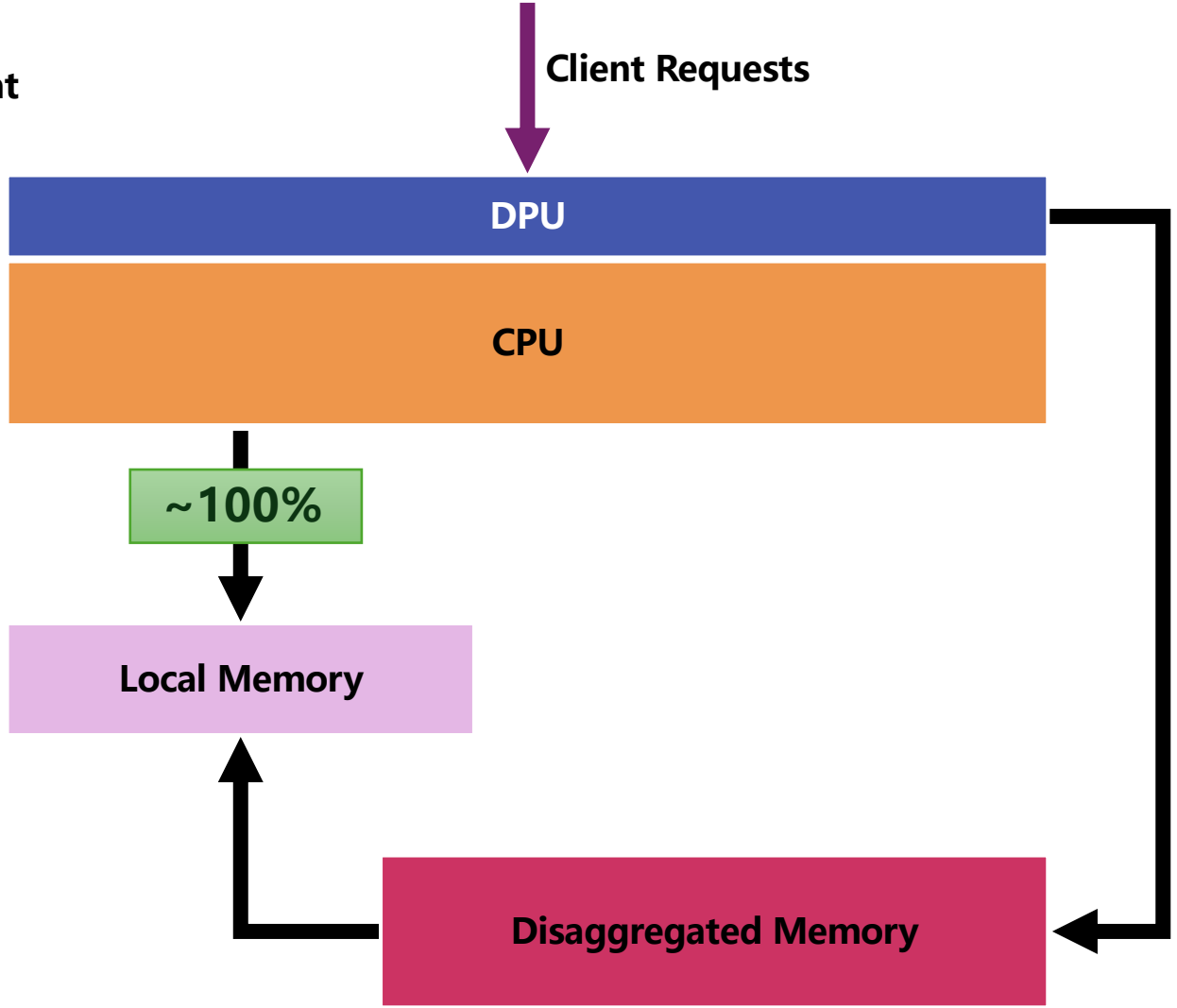


Consistency Considerations



Challenges

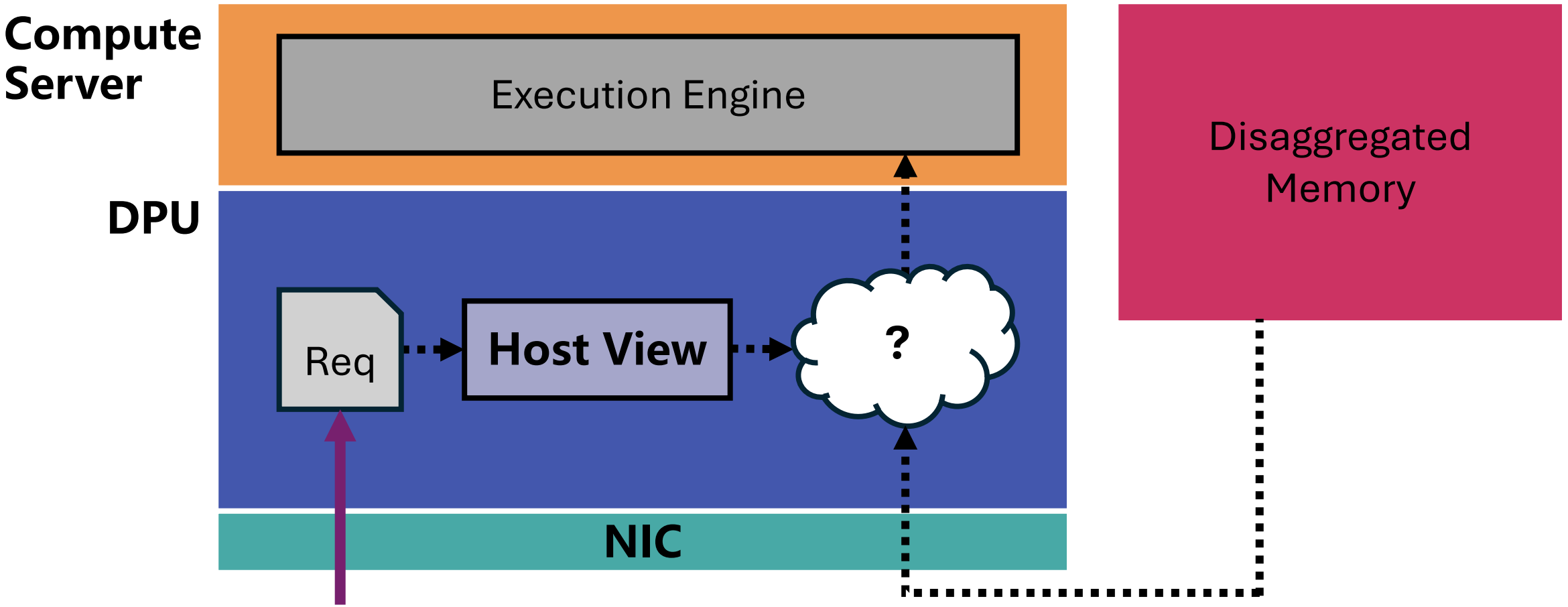
How do we determine what to prefetch?



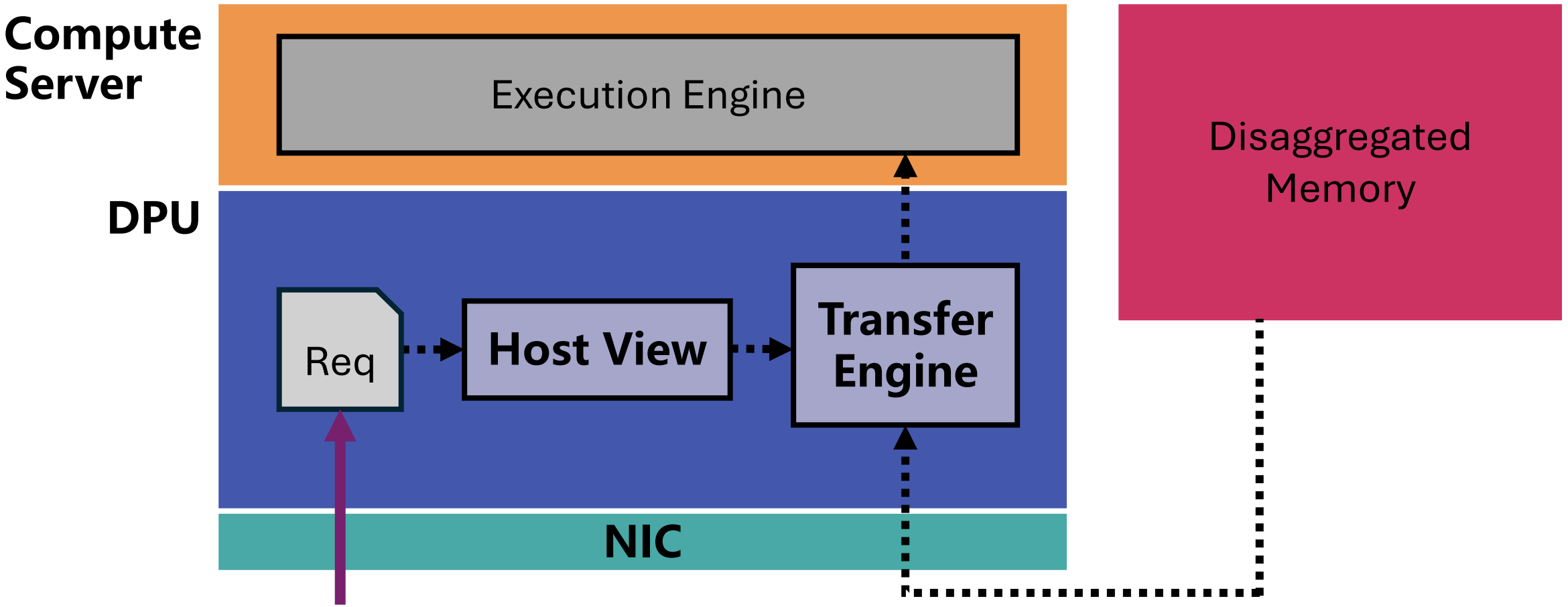
How do we serve the data items for fast application access?

How do we prefetch efficiently?

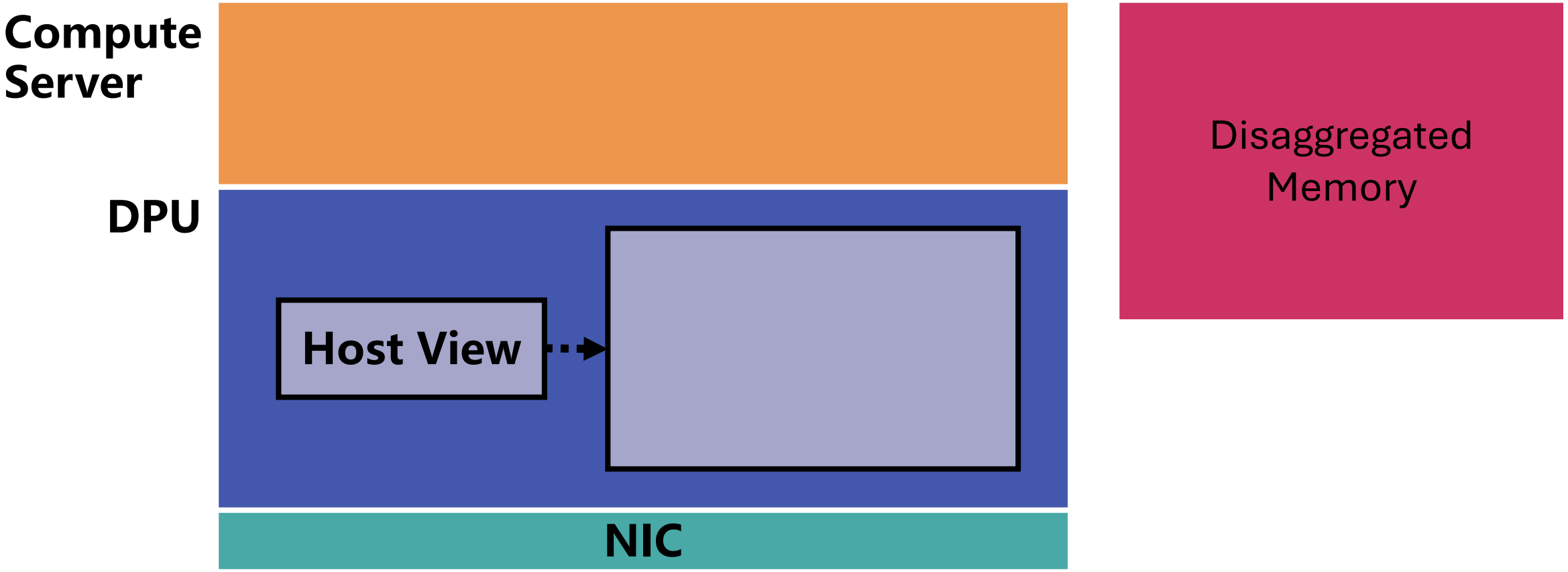
Prefetch Items



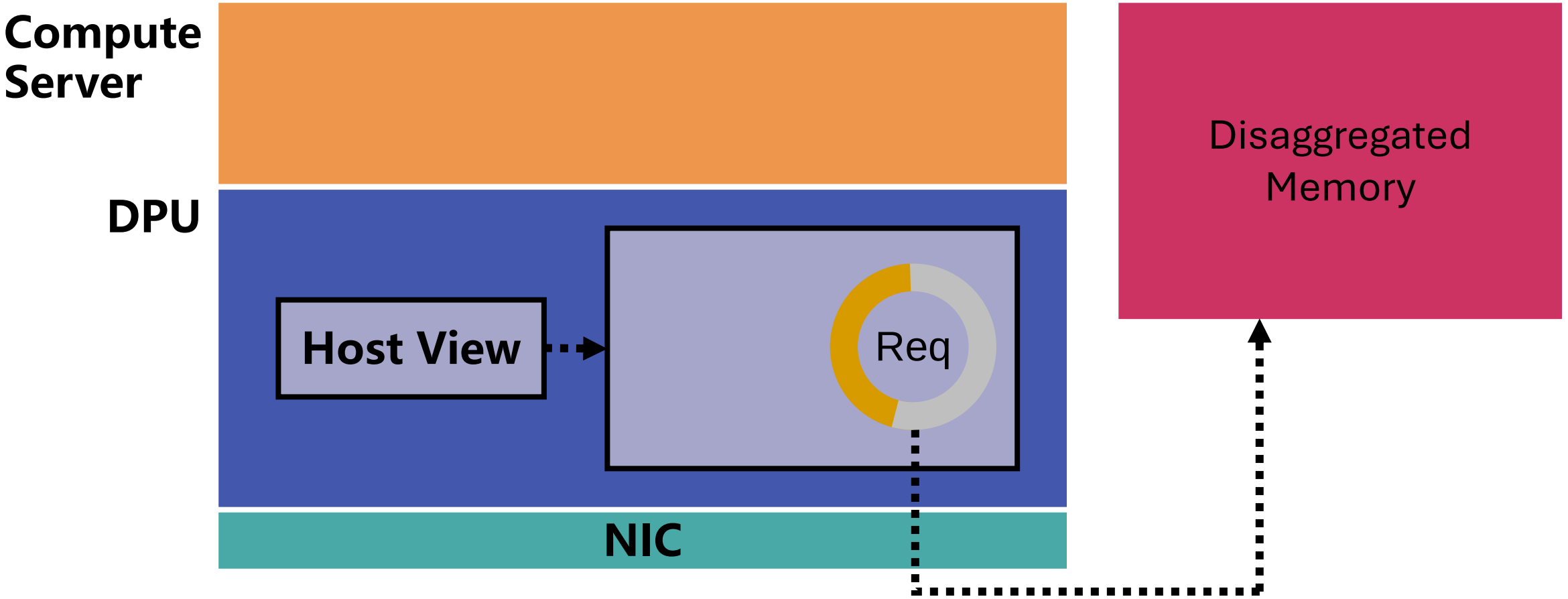
Prefetch Items



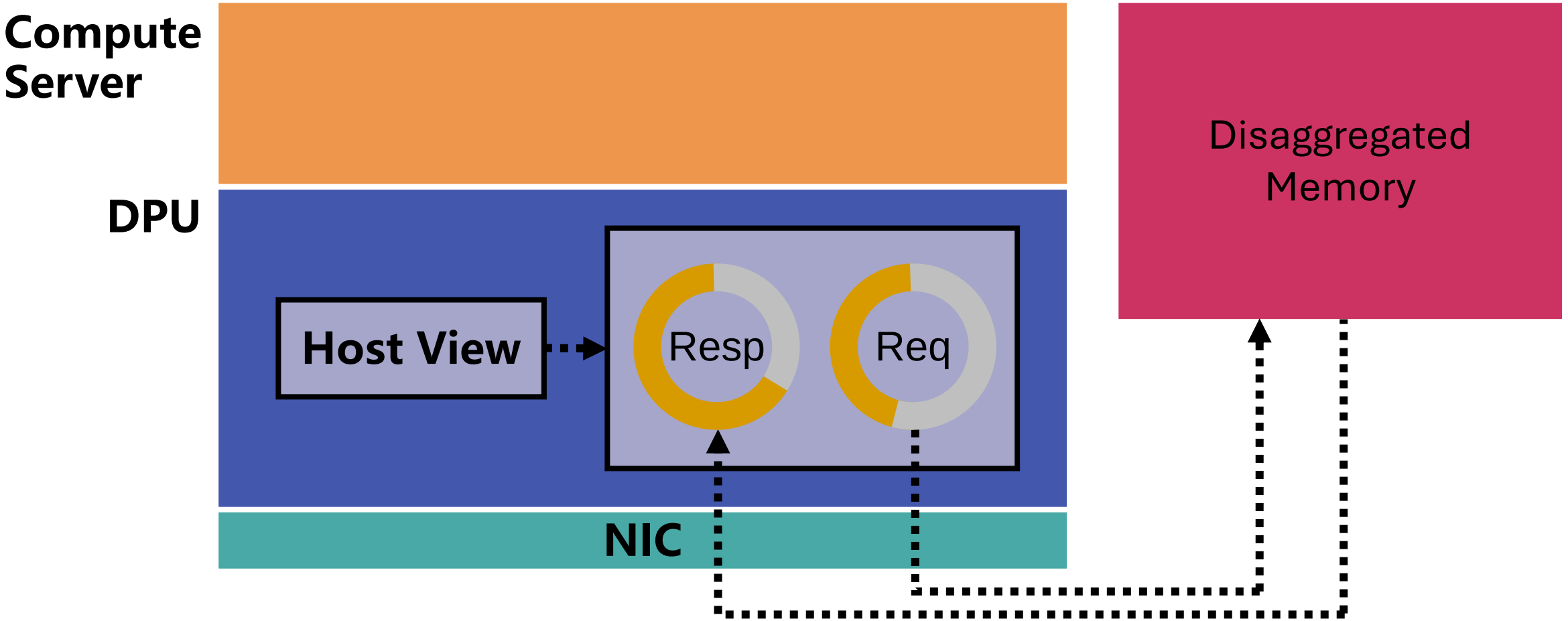
Transfer Engine



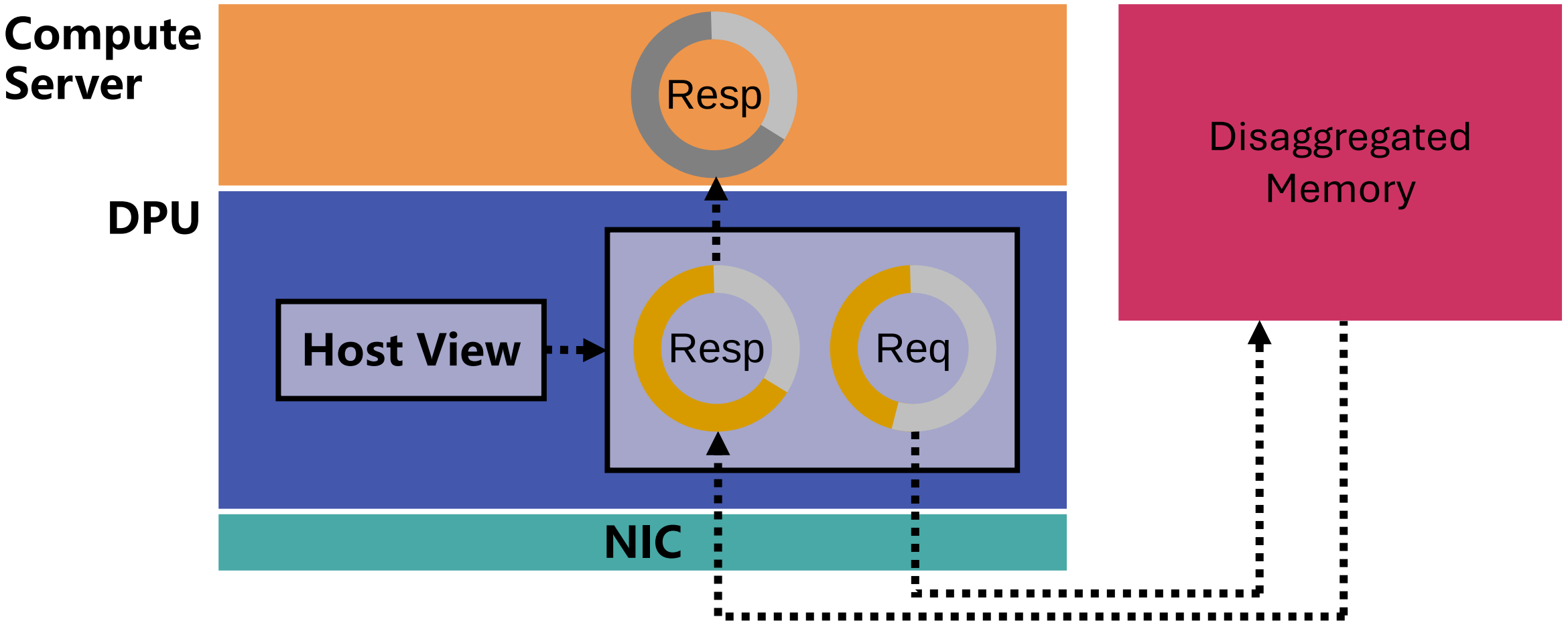
Transfer Engine



Transfer Engine



Transfer Engine

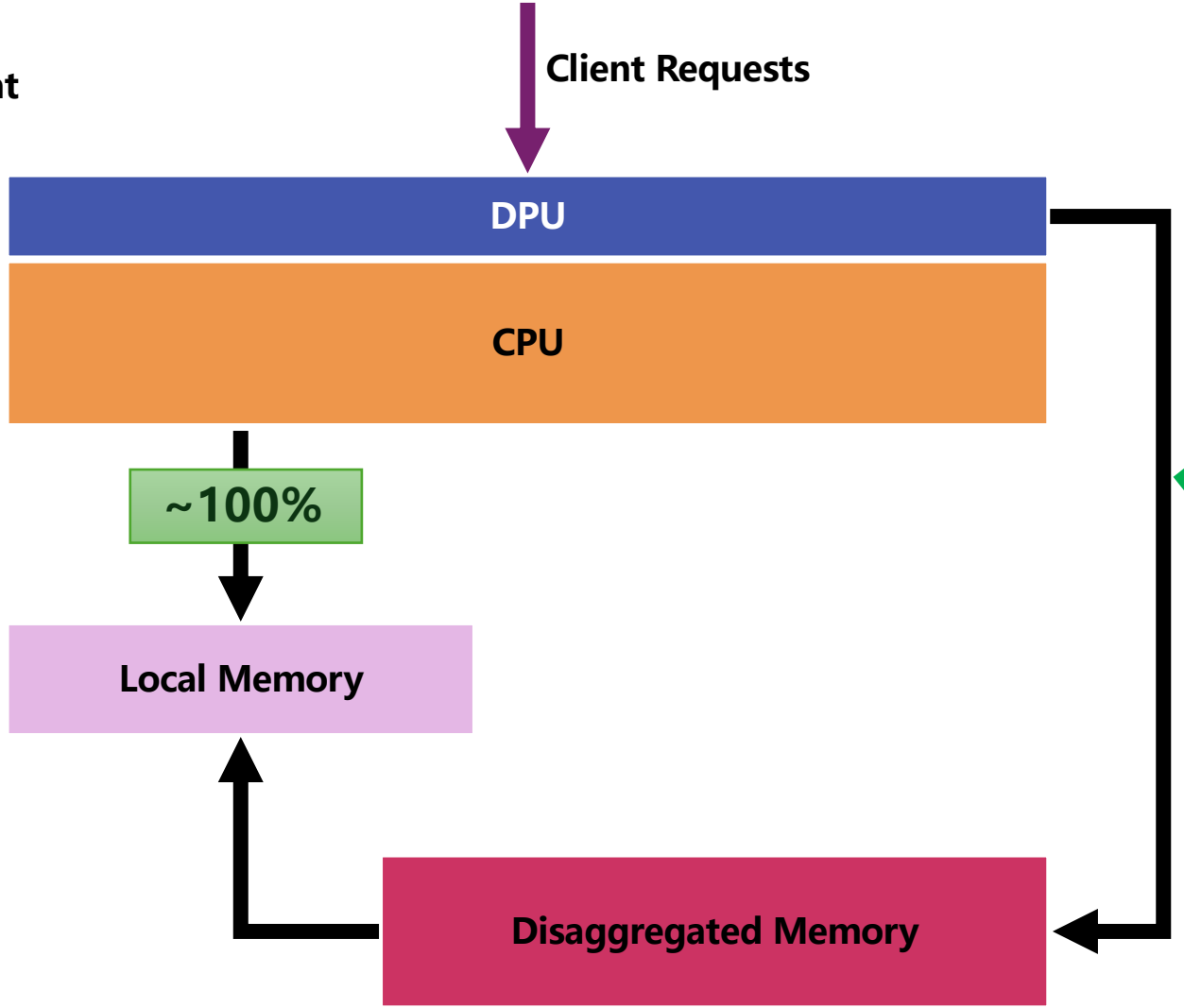


Challenges

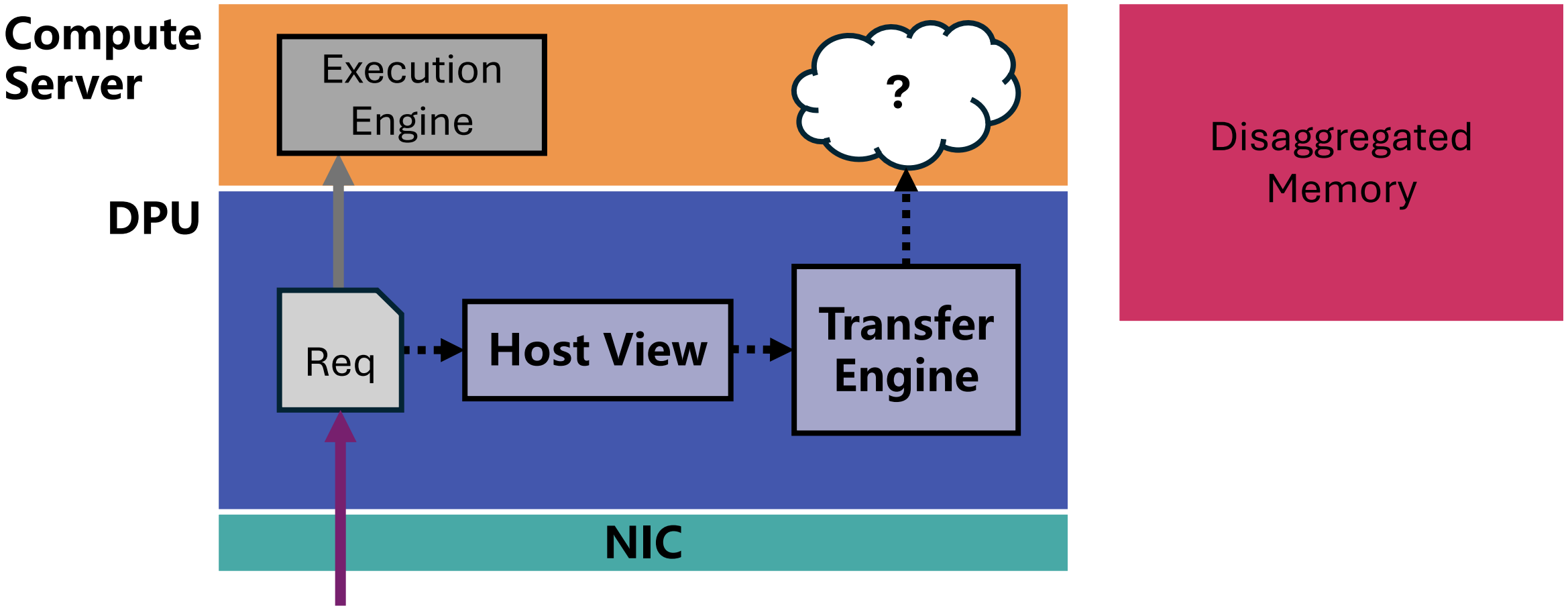
How do we determine what to prefetch?

How do we serve the data items for fast application access?

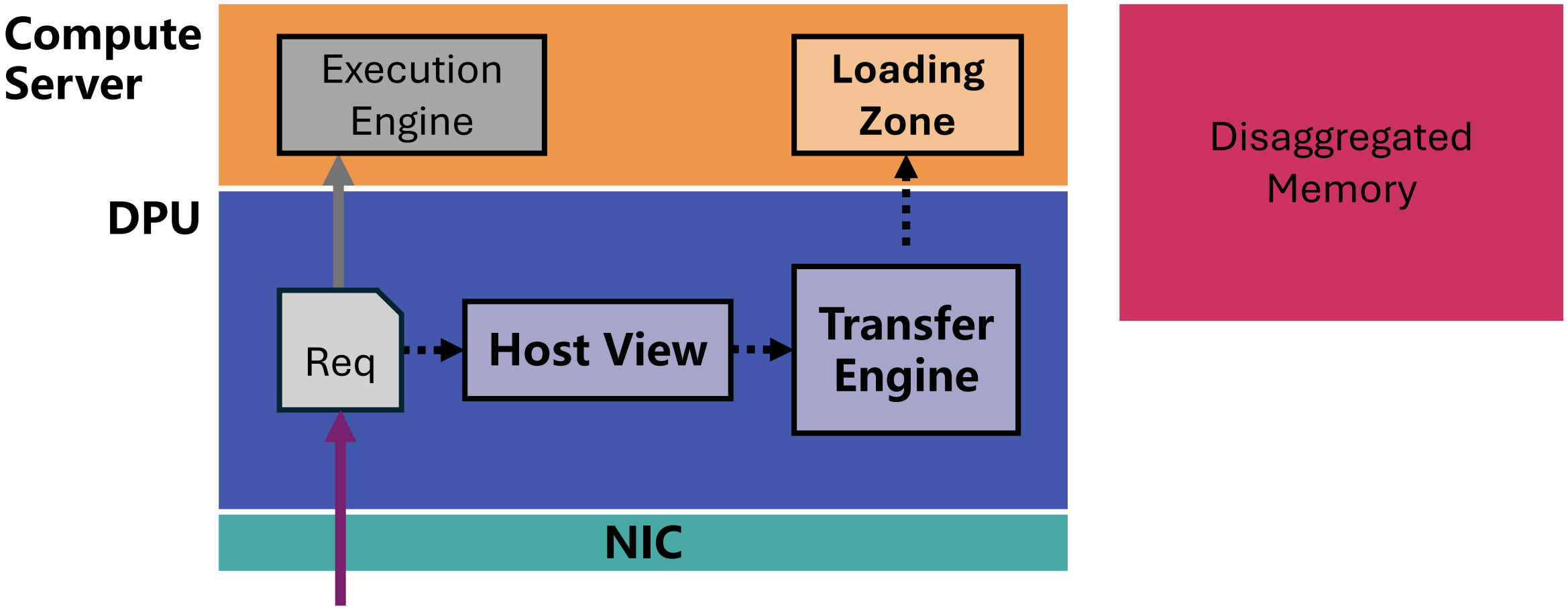
How do we prefetch efficiently?



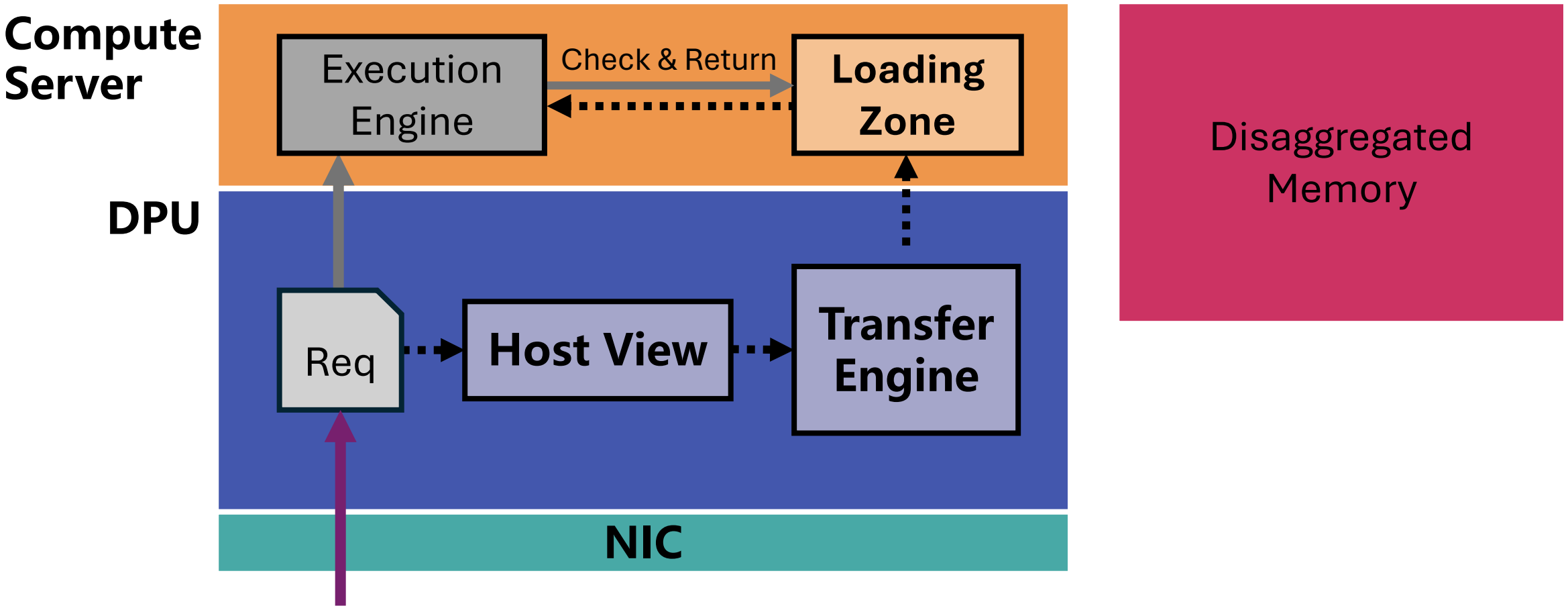
Serve Values



Serve Values



Serve Values

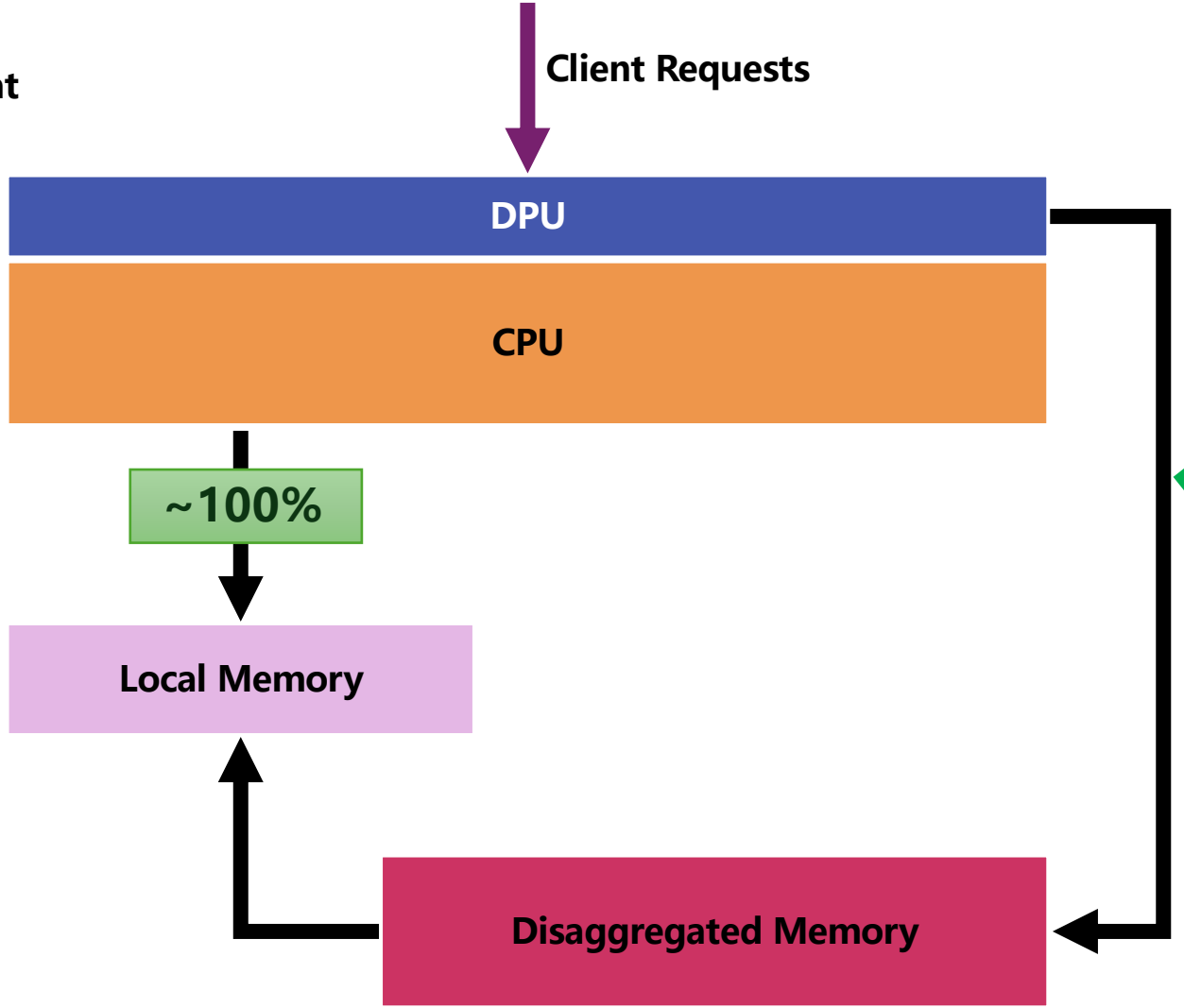


Challenges

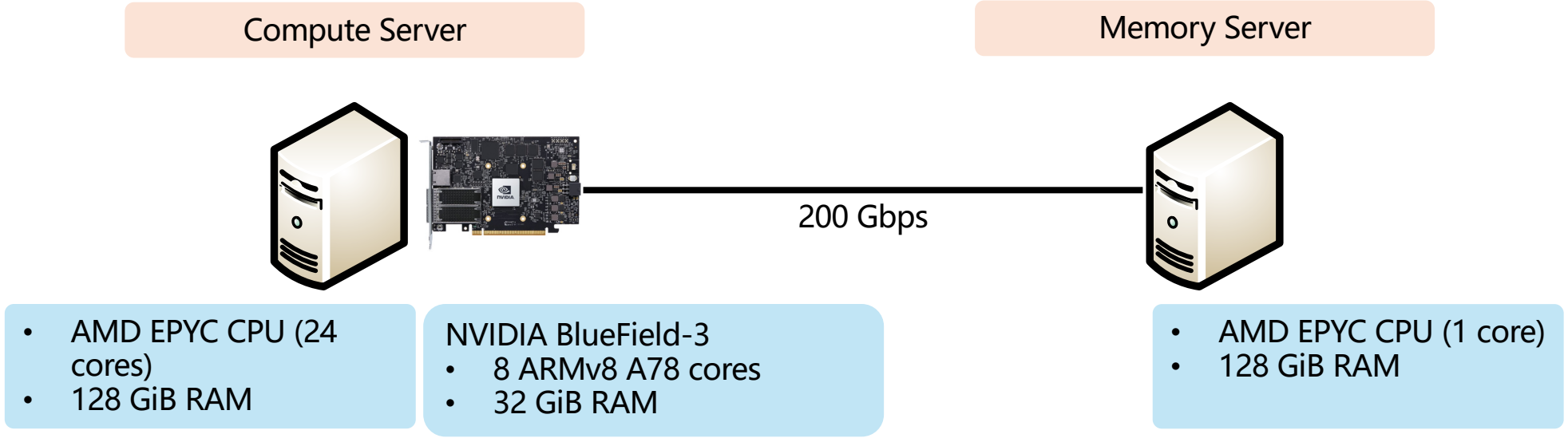
How do we determine what to prefetch?

How do we serve the data items for fast application access?

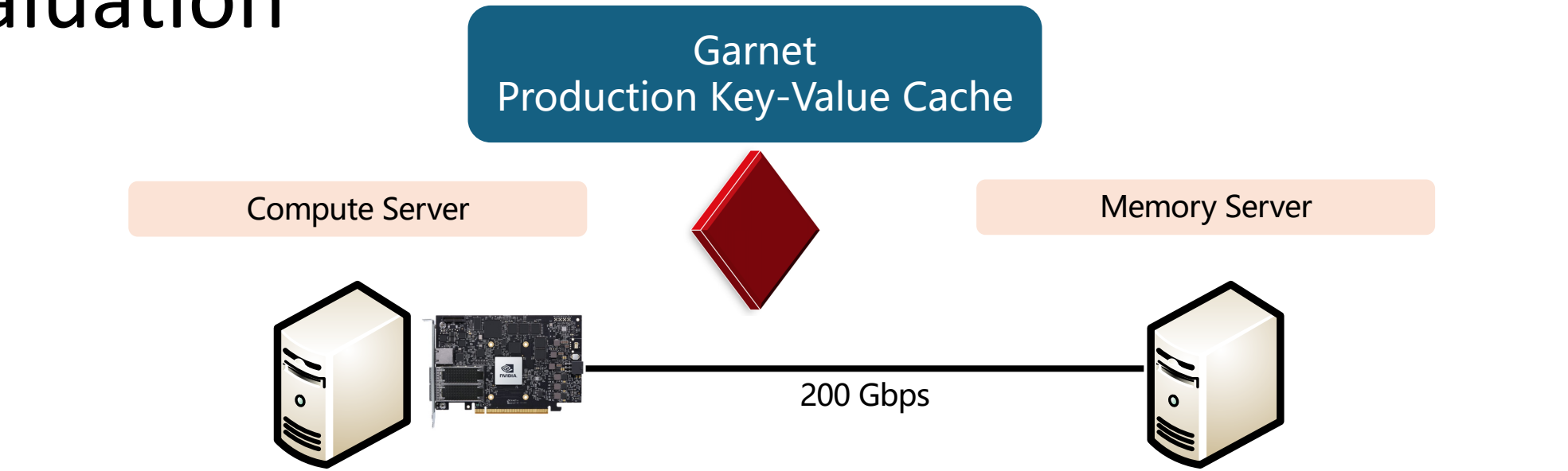
How do we prefetch efficiently?



Evaluation


Compute Server
Memory Server
200 Gbps
• AMD EPYC CPU (24 cores)
• 128 GiB RAM
NVIDIA BlueField-3
• 8 ARMv8 A78 cores
• 32 GiB RAM
• AMD EPYC CPU (1 core)
• 128 GiB RAM

Evaluation



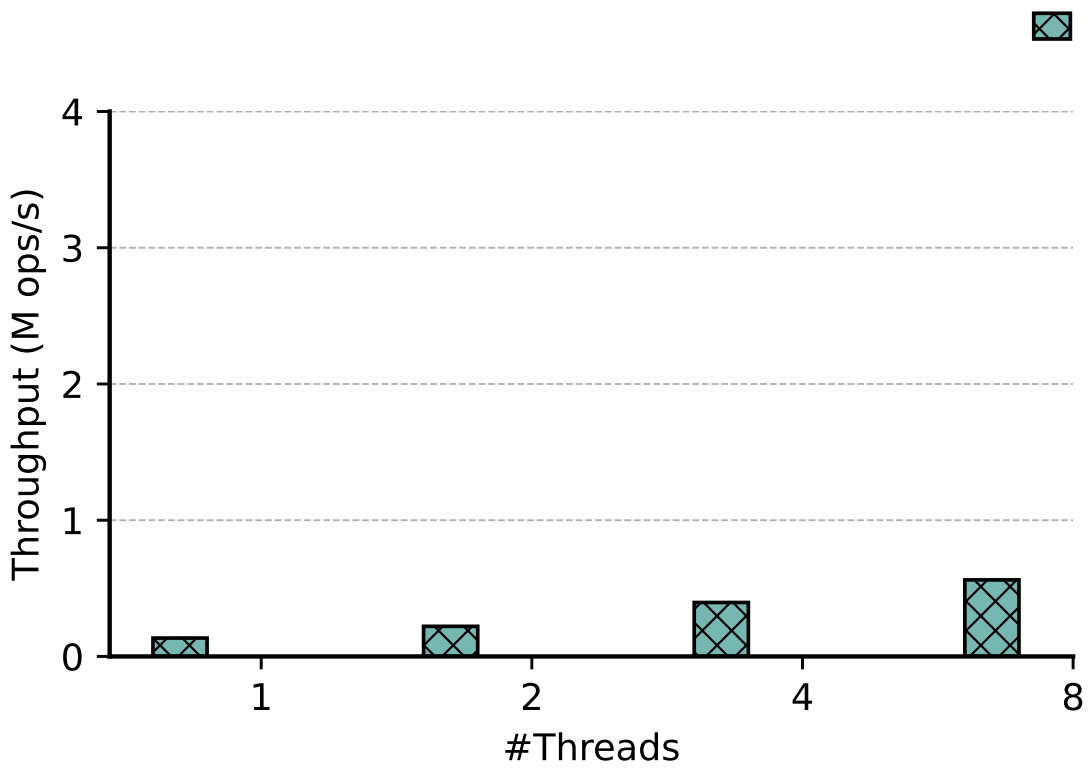
- AMD EPYC CPU (24 cores)
- 128 GiB RAM

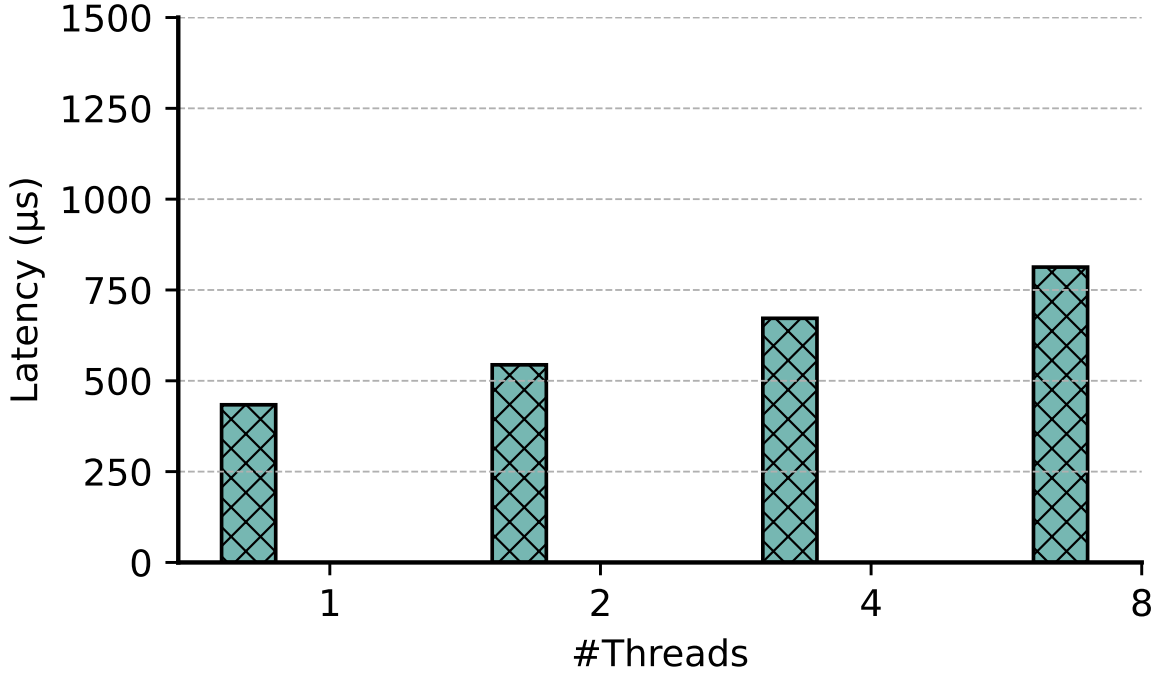
NVIDIA BlueField-3
- 8 ARMv8 A78 cores
- 32 GiB RAM

Utilization
- 4 CPU cores
- 1 GB RAM
  (100 million keys)

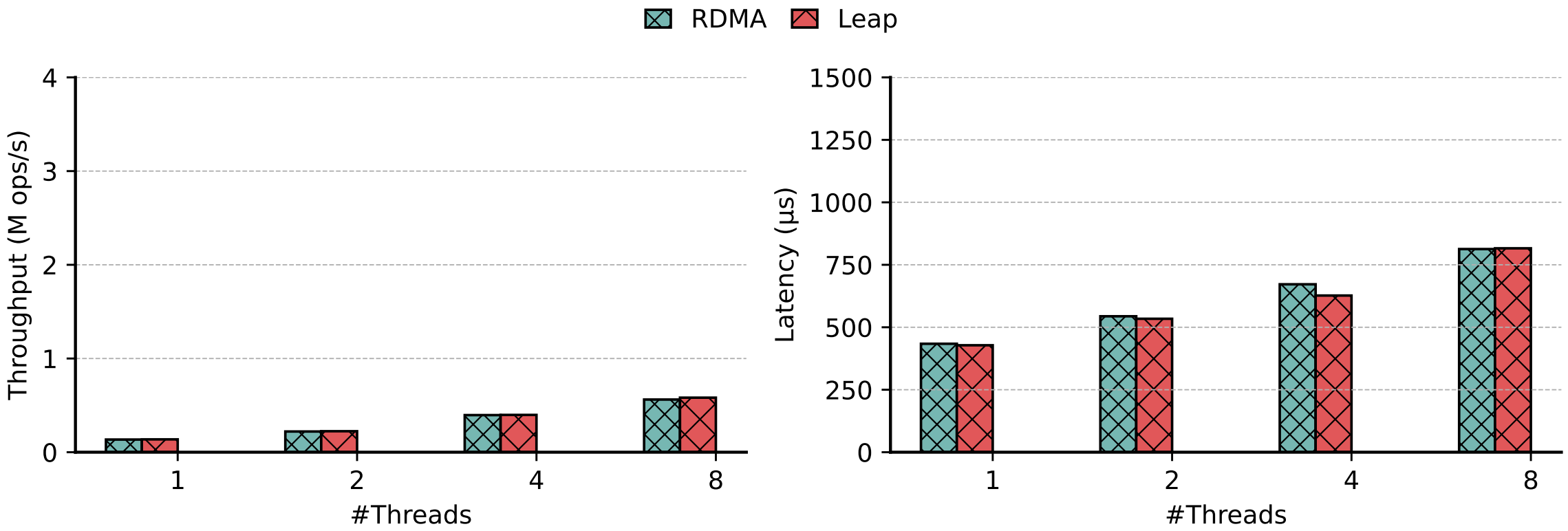
- AMD EPYC CPU (1 core)
- 128 GiB RAM

End-to-End Performance

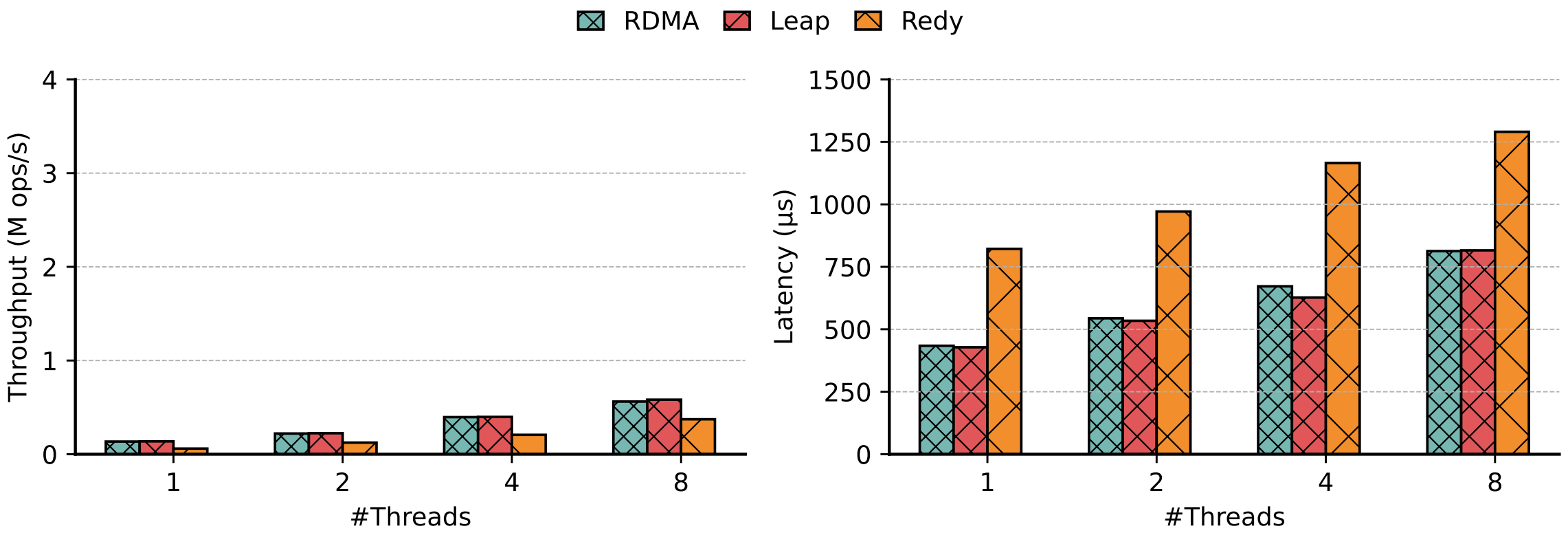




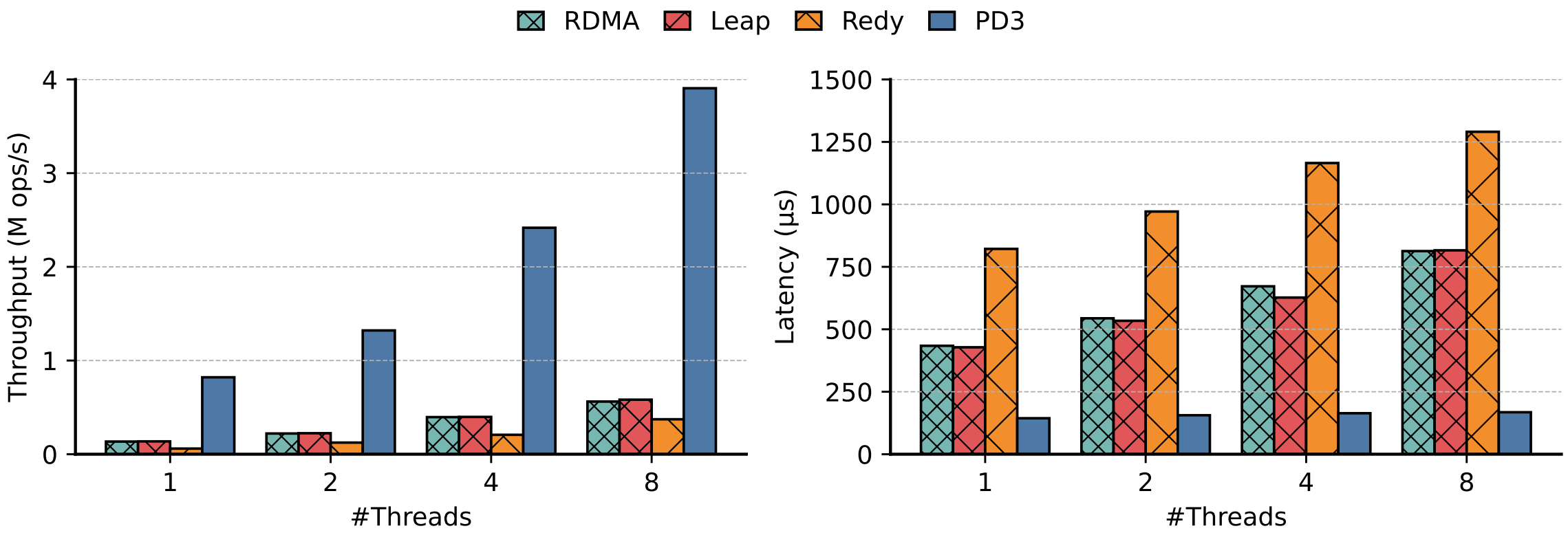
End-to-End Performance



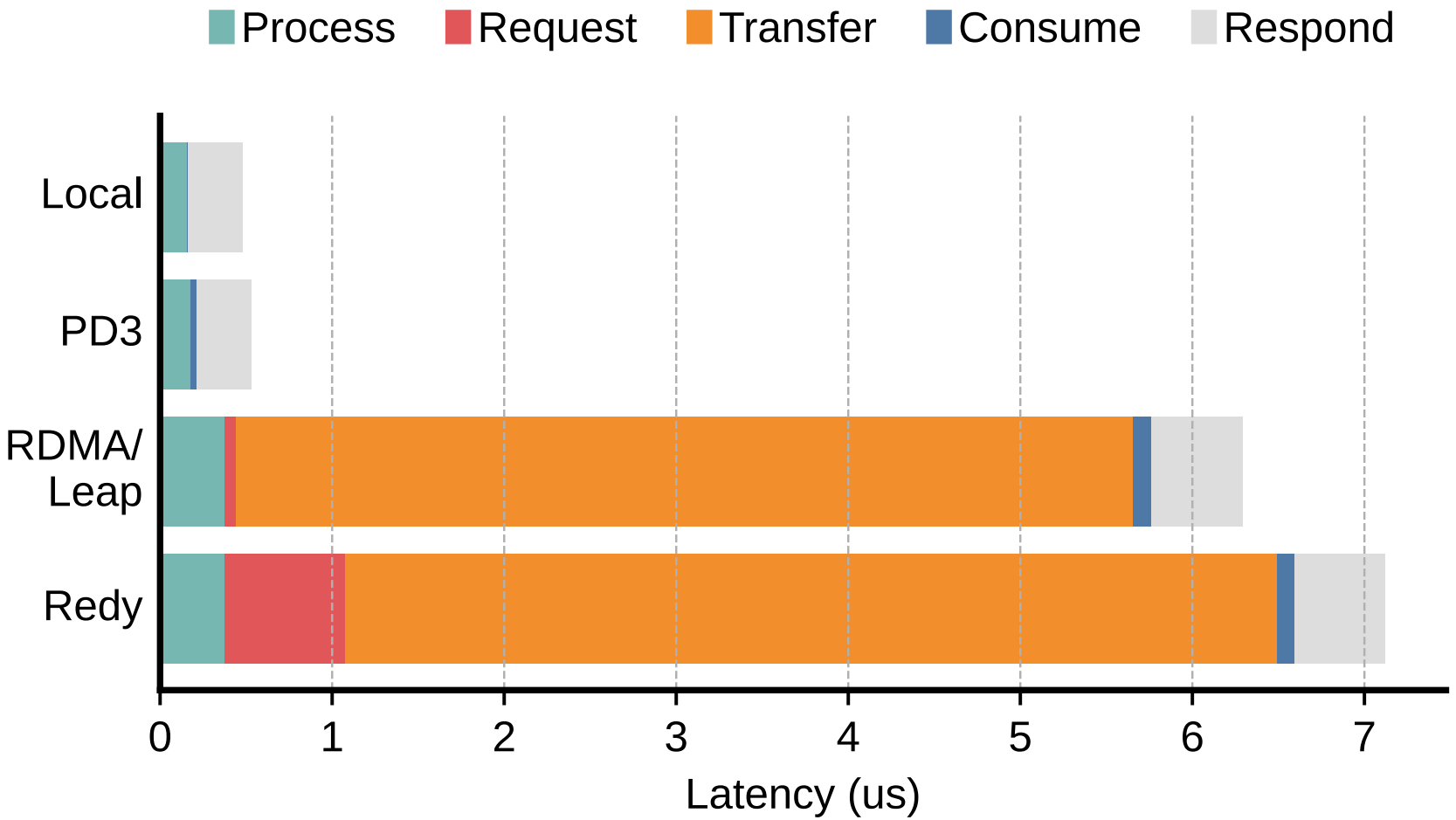
End-to-End Performance



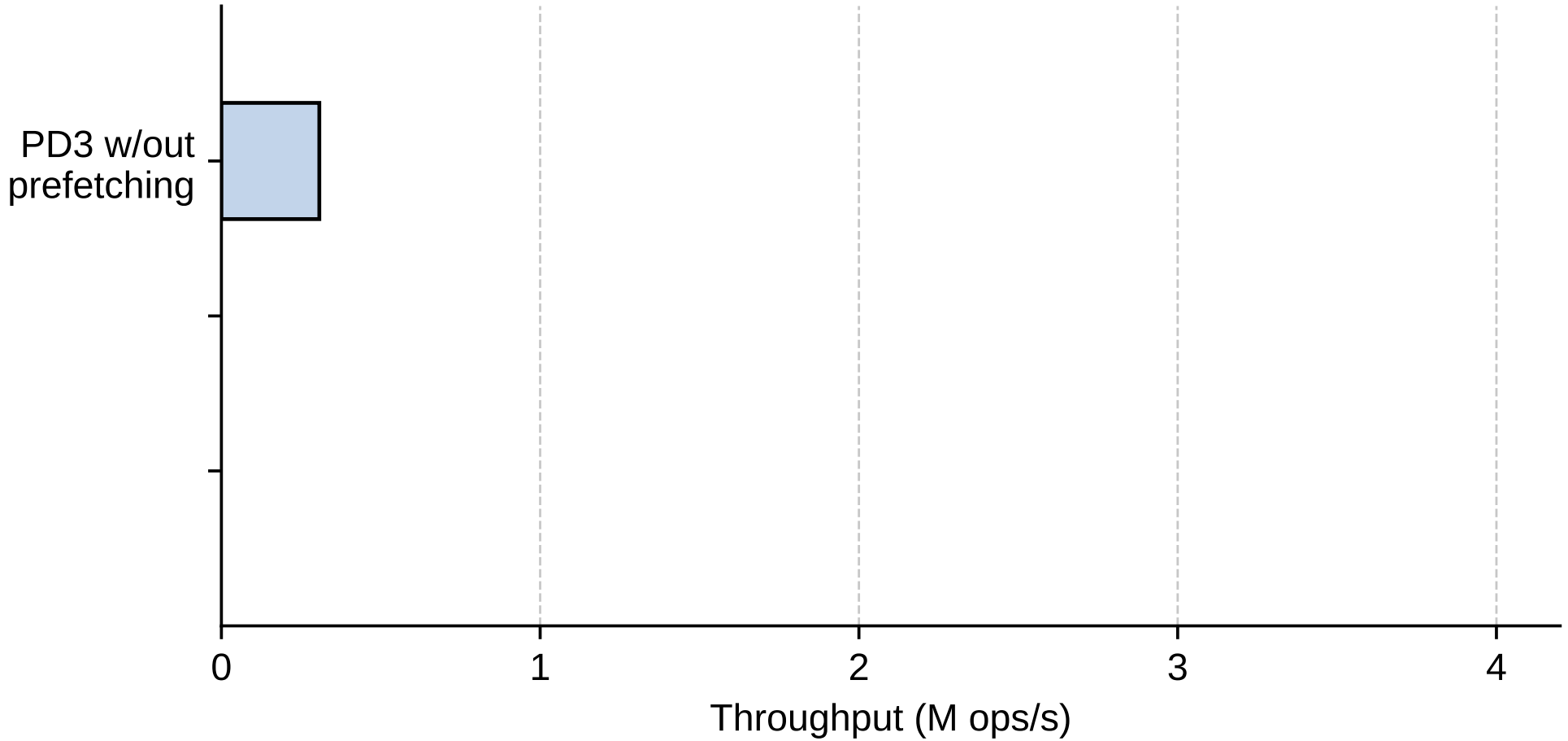
End-to-End Performance



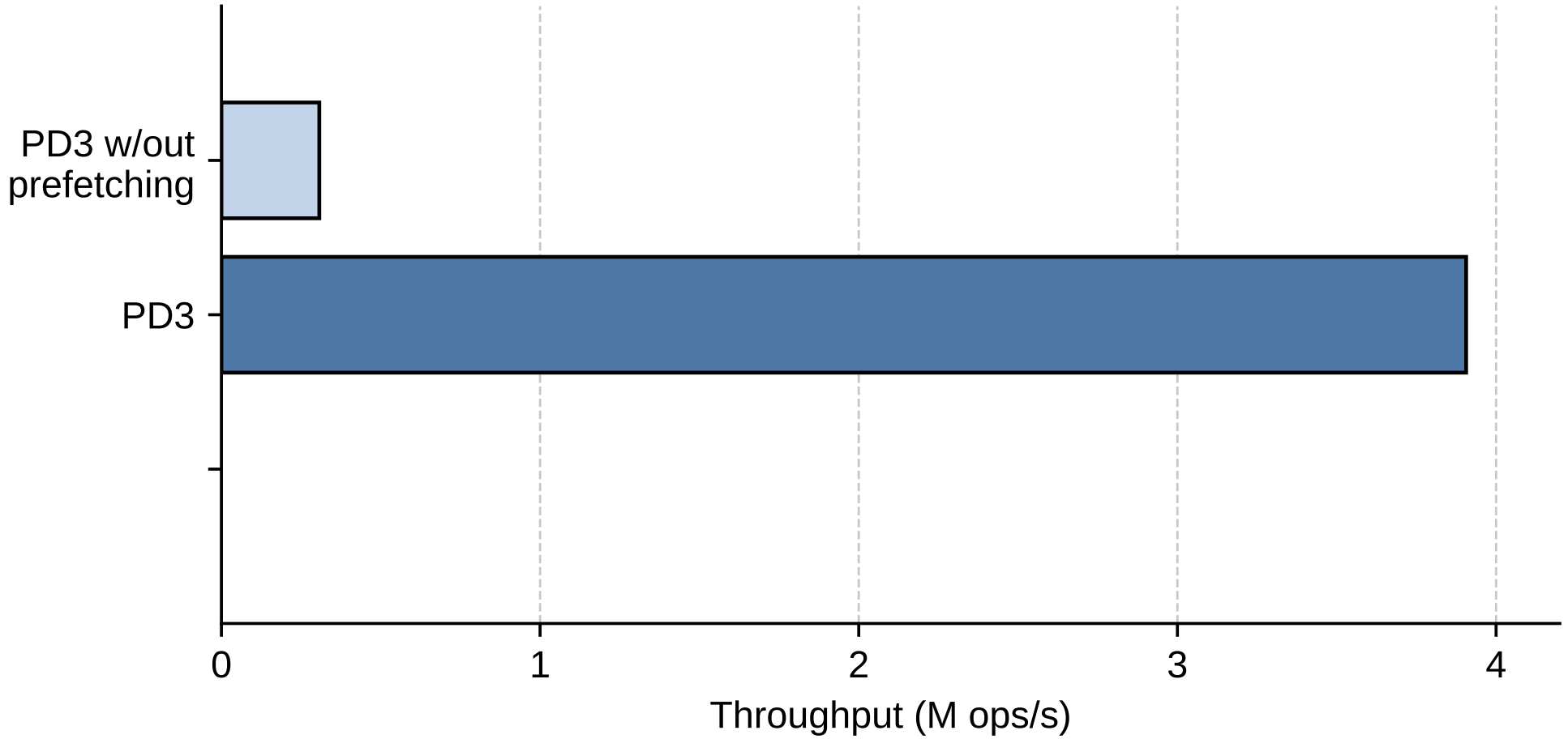
Execution Breakdown



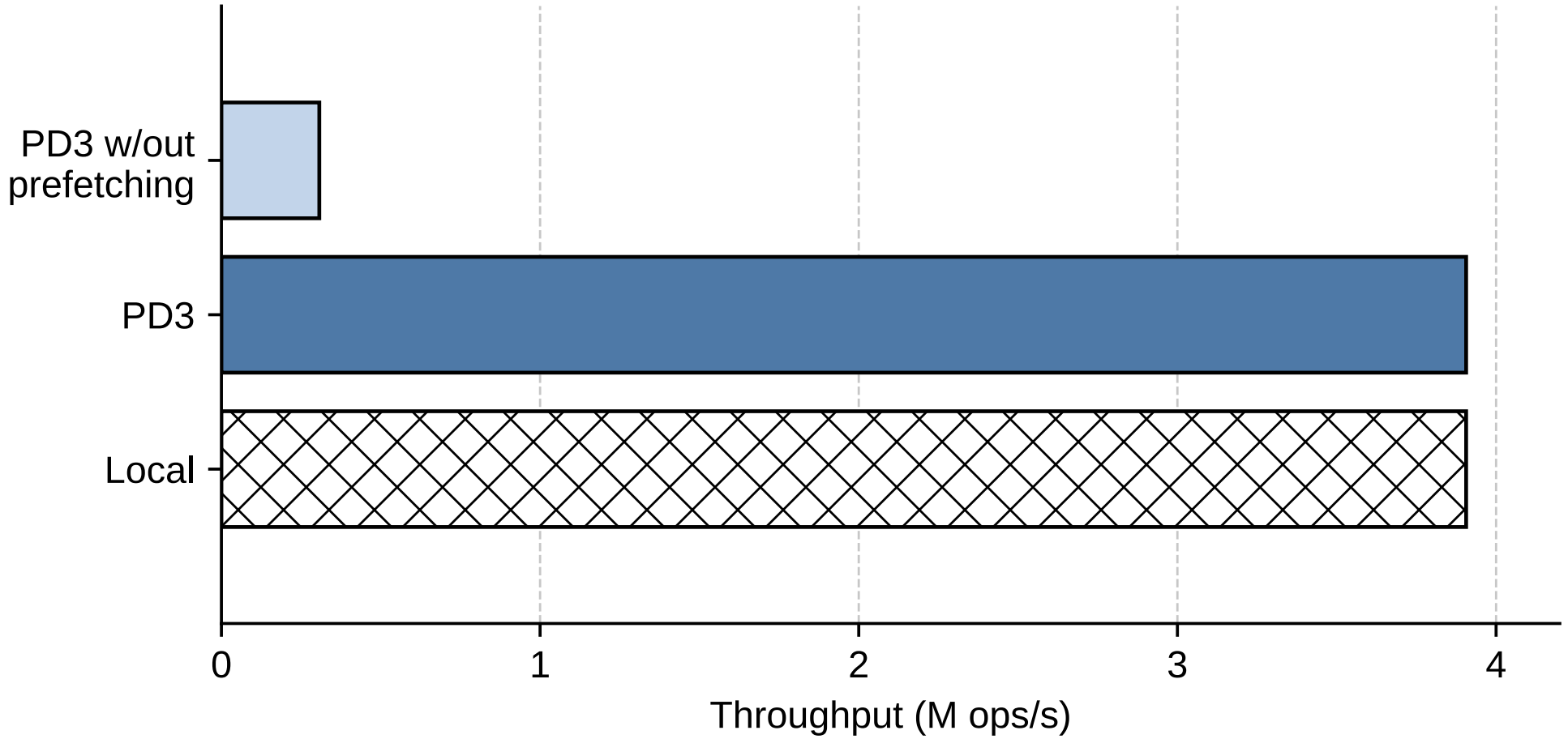
Prefetching Impact



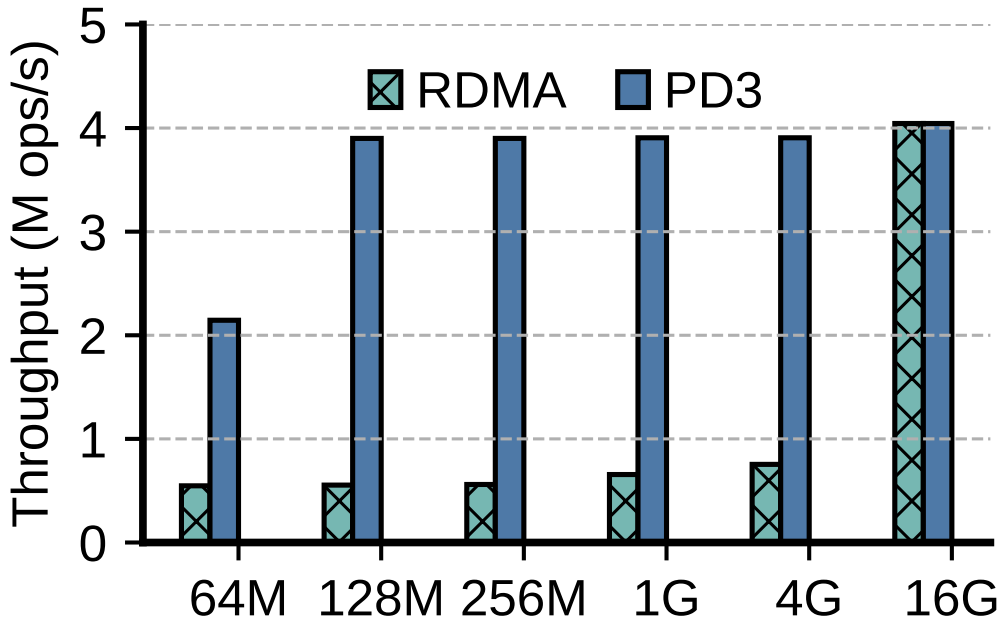
Prefetching Impact

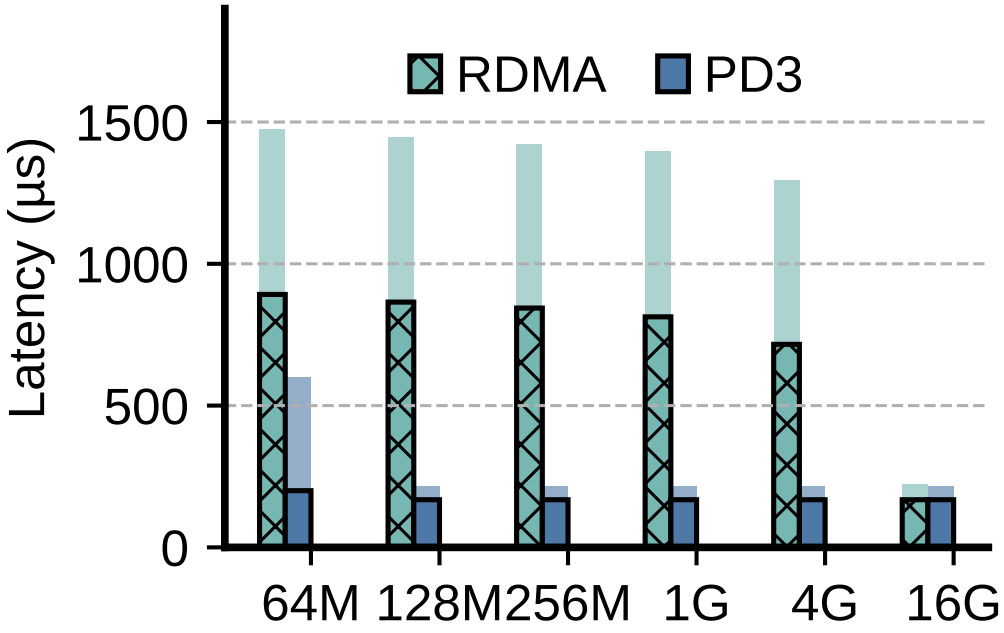


Prefetching Impact



Compute-local Memory Size





Summary

- Memory disaggregation incurs significant performance penalties due to cache misses
- By leveraging the capability of DPUs, PD3 determines cache misses for incoming requests and prefetches the required data
- Using PD3, disaggregated applications reach performance parity with their monolithic counterparts

Thank You

https://github.com/fardatalab/PD3

FDL
University of Toronto

Computer Science
UNIVERSITY OF TORONTO

NUS | Computing
National University of Singapore