Octopus: an RDMA-enabled Distributed Persistent Memory File System

Youyou Lu[1], Jiwu Shu[1], Youmin Chen[1], Tao Li[2]

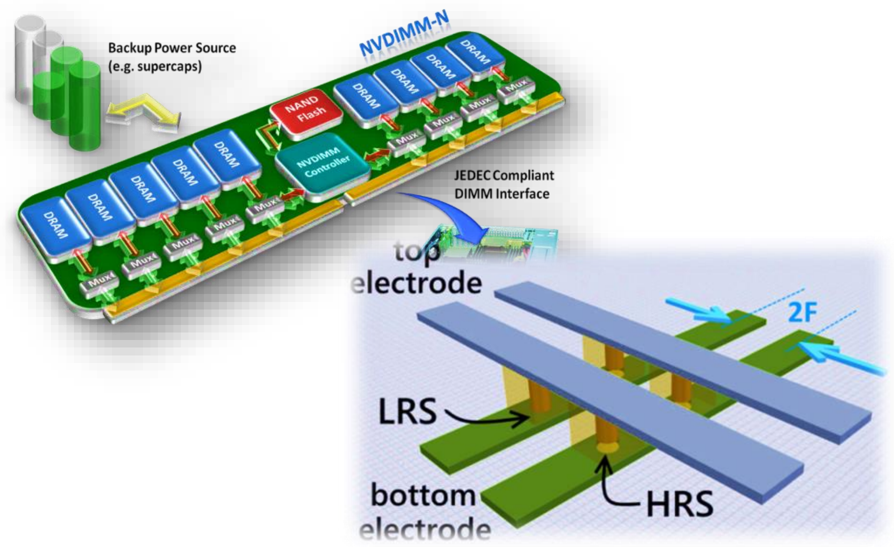
[1]Tsinghua University
[2]University of Florida

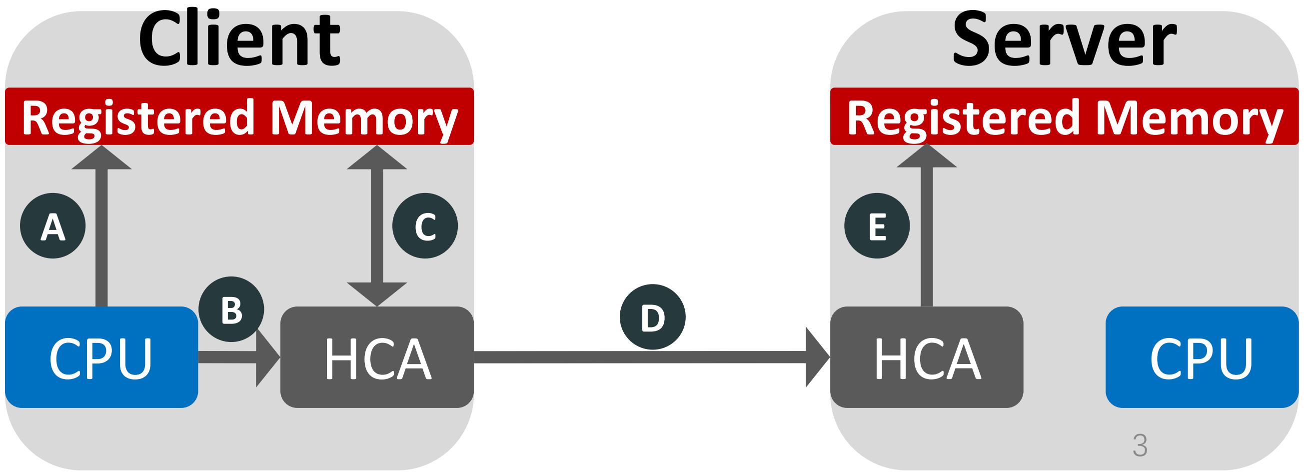
Outline

- Background and Motivation
- Octopus Design
- Evaluation
- Conclusion

NVMM & RDMA

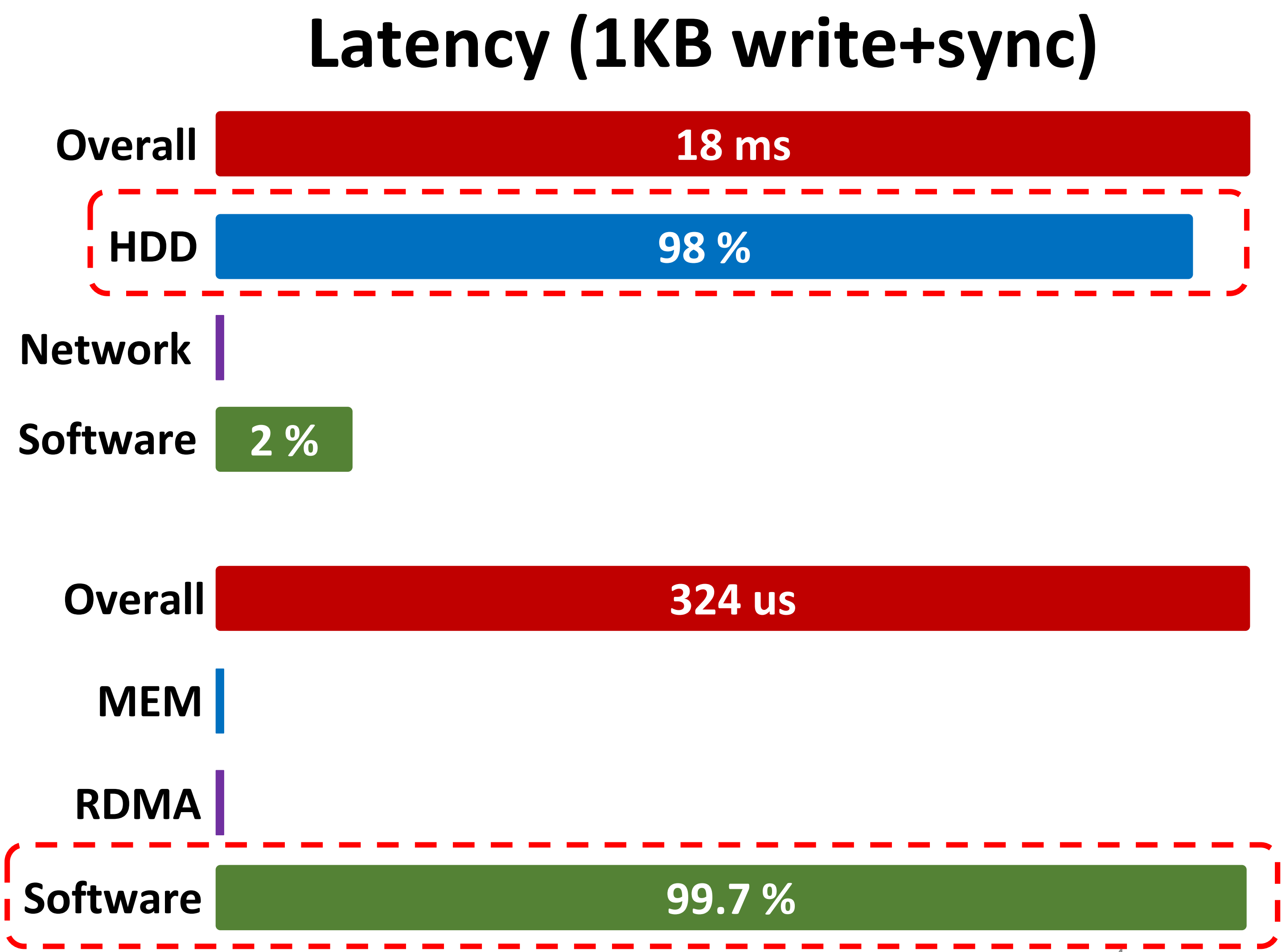
- **NVMM** (PCM, ReRAM, etc)
  - Data persistency
  - Byte-addressable
  - Low latency
- **RDMA**
  - Remote direct access
  - Bypass remote kernel
  - Low latency and high throughput





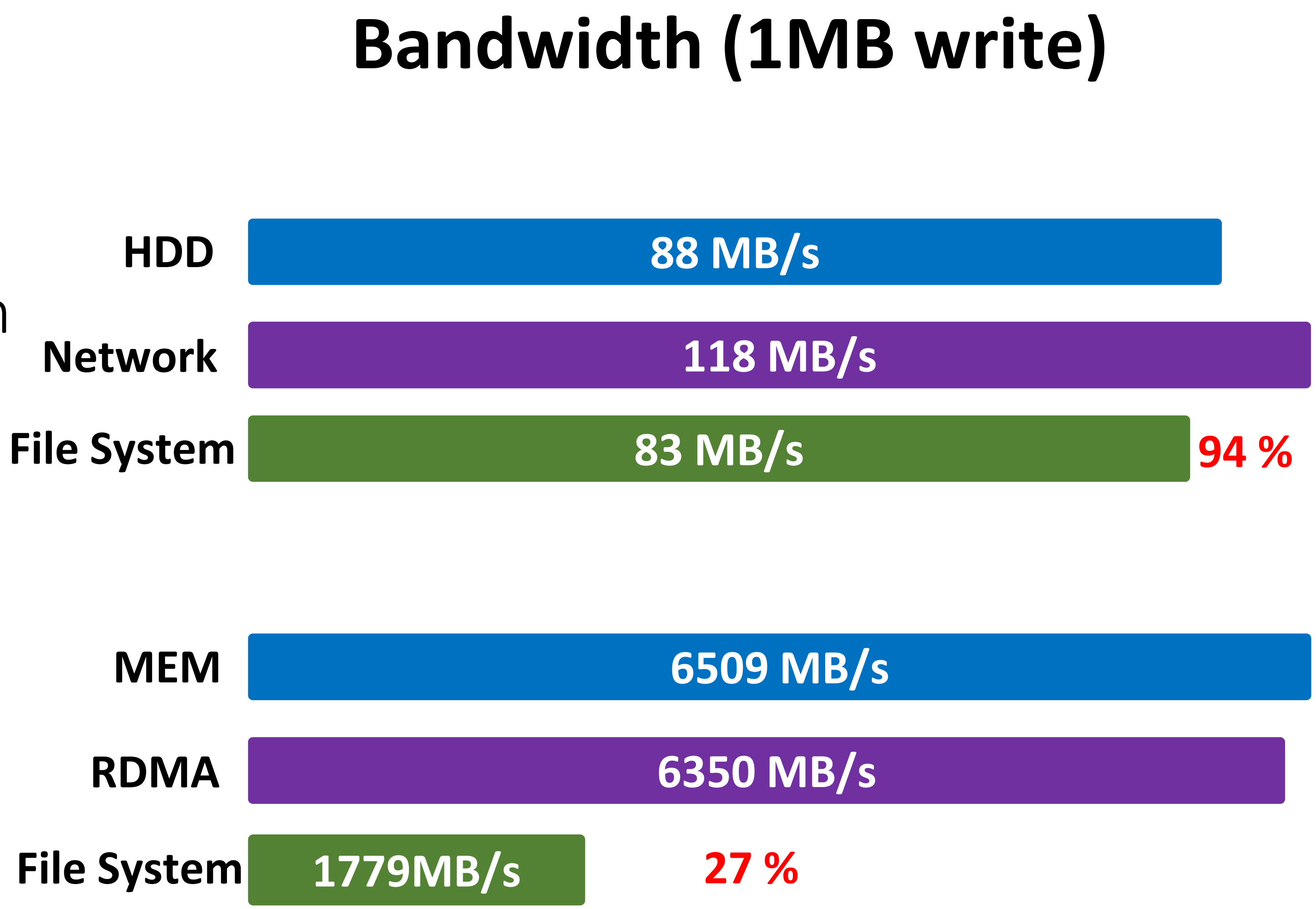
Modular-Designed Distributed File System

- DiskGluster
  - Disk for data storage
  - GigE for communication
- MemGluster
  - Memory for data storage
  - RDMA for communication



Modular-Designed Distributed File System

- DiskGluster
  - Disk for data storage
  - GigE for communication
- MemGluster
  - Memory for data storage
  - RDMA for communication



RDMA-enabled Distributed File System

- More than fast hardware
  - It is suboptimal to simply replace the network/storage module
- Opportunities and Challenges
  - NVM
    - Byte-addressability
    - Significant overhead of data copies

  - RDMA
    - Flexible programming verbs (message/memory semantics)
    - Imbalanced CPU processing capacity vs. network I/Os

RDMA-enabled Distributed File System

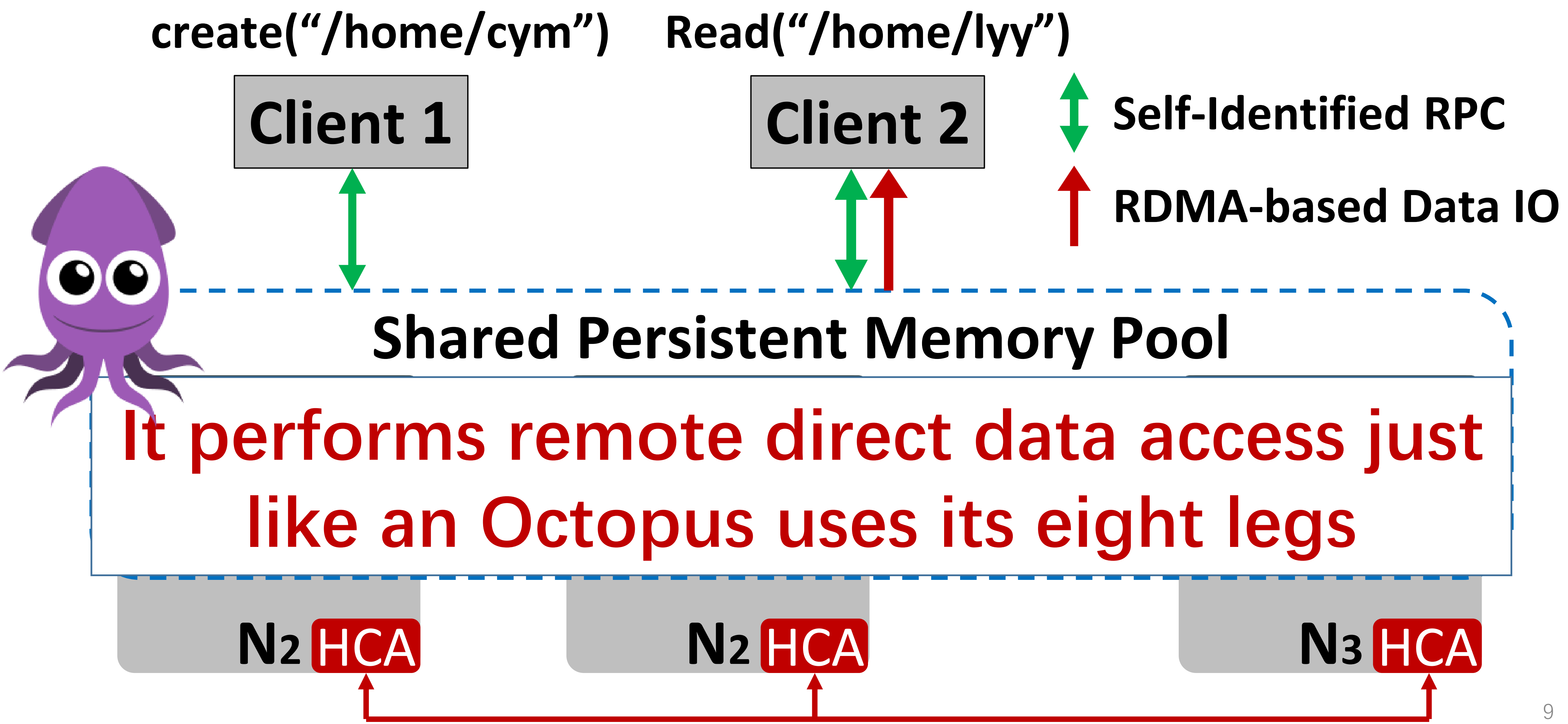
Opportunity		Approaches
Byte-addressability of NVM One-sided RDMA verbs CPU is the new bottleneck	→	Shared data managements New data flow strategies
Flexible RDMA verbs	→	Efficient RPC design
RDMA Atomics	→	Concurrent control

- It is necessary to rethink the design of DFS over NVM & RDMA

Outline

- Background and Motivation
- Octopus Design
  - Shared Persistent Memory Pool -> Reduce data copies
  - Self-Identified Metadata RPC -> Reduce response latency
  - Client-Active Data I/O -> Rebalance CPU/network overhead
  - Collect-Dispatch Transaction -> Efficient concurrent control
- Evaluation
- Conclusion

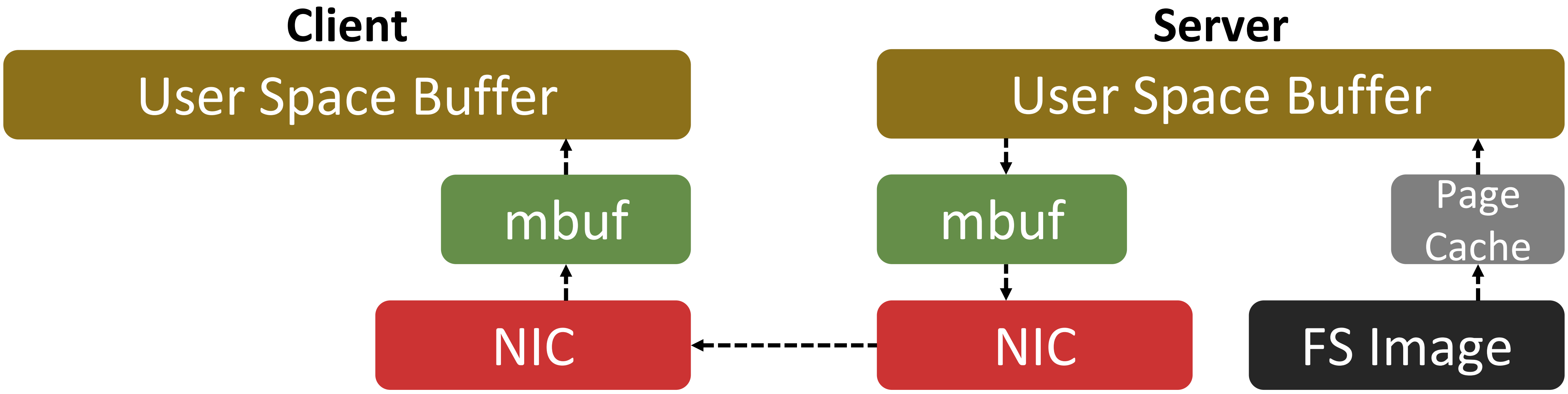
Octopus Architecture



1. Shared Persistent Memory Pool

- Existing DFSs
  - Redundant data copy

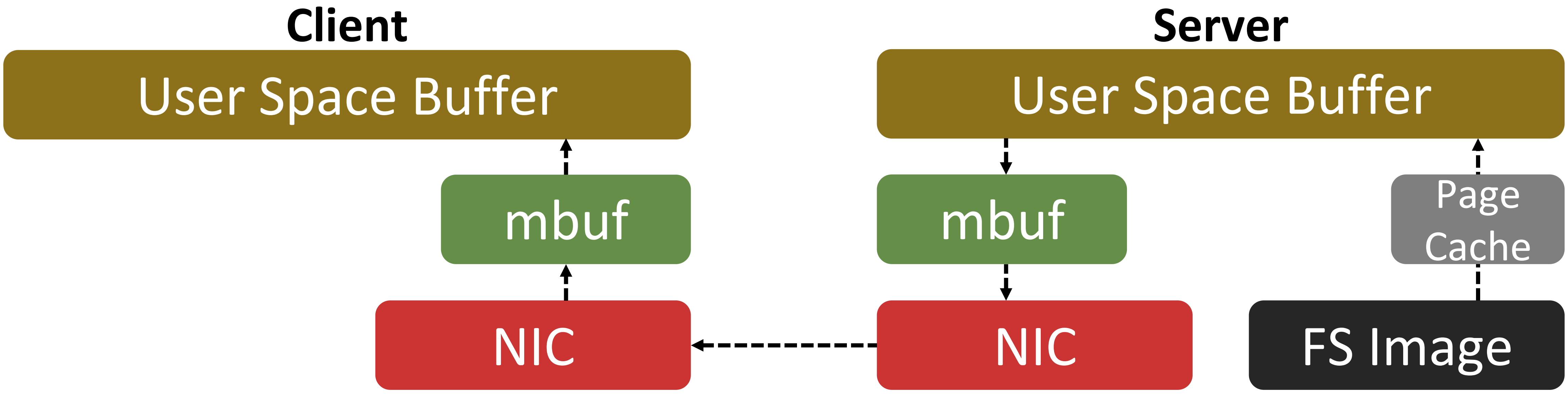
GlusterFS **7 copy**



1. Shared Persistent Memory Pool

- Existing DFSs
  - Redundant data copy

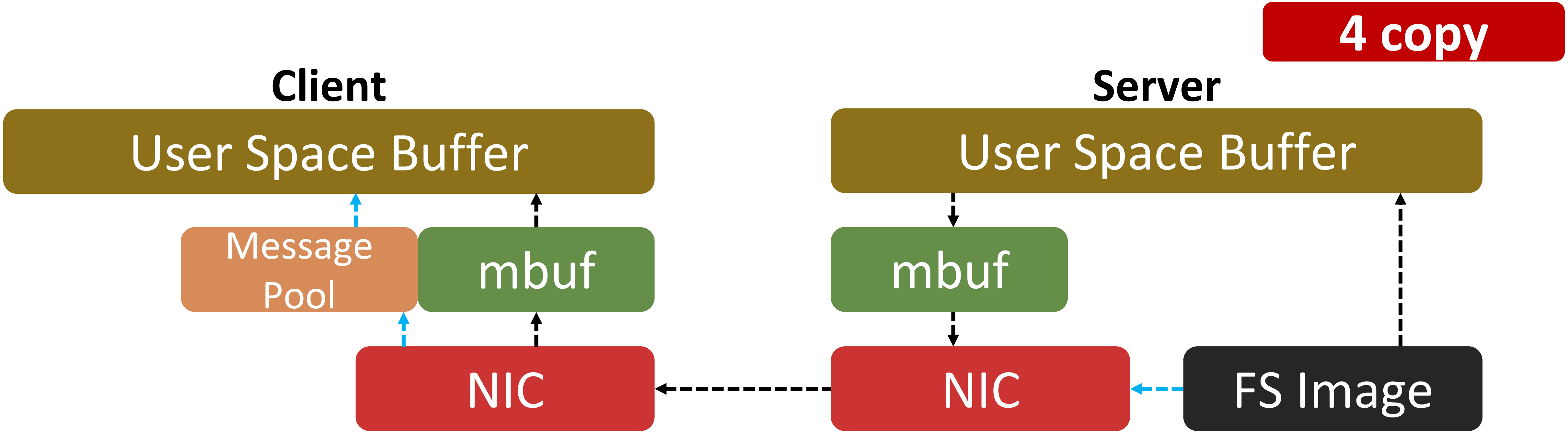
GlusterFS + DAX **6 copy**



1. Shared Persistent Memory Pool

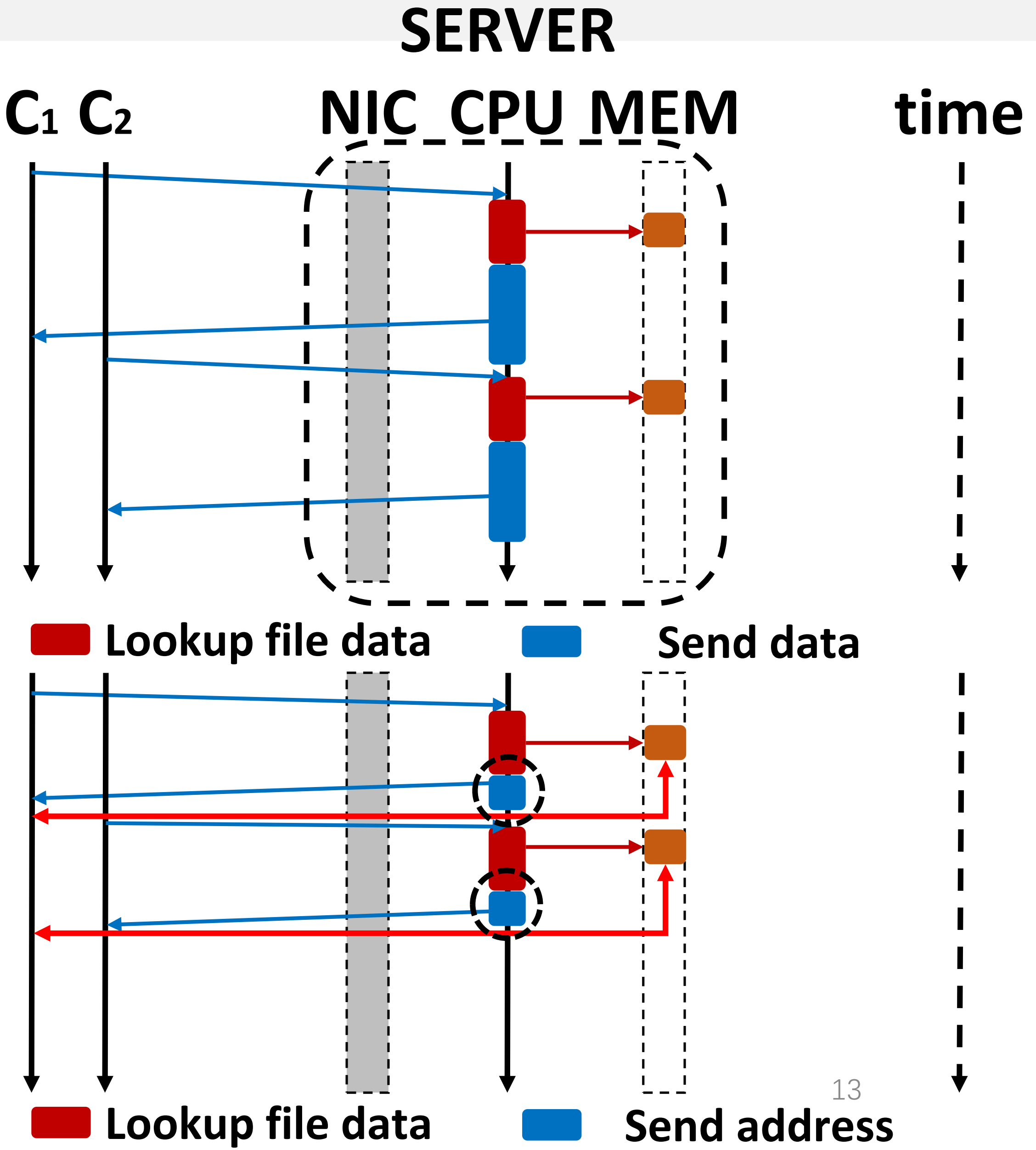
- Existing DFSs
  - Redundant data copy

- **Octopus with SPMP**
  - Introduces the *shared persistent memory pool*
  - Global view of data layout



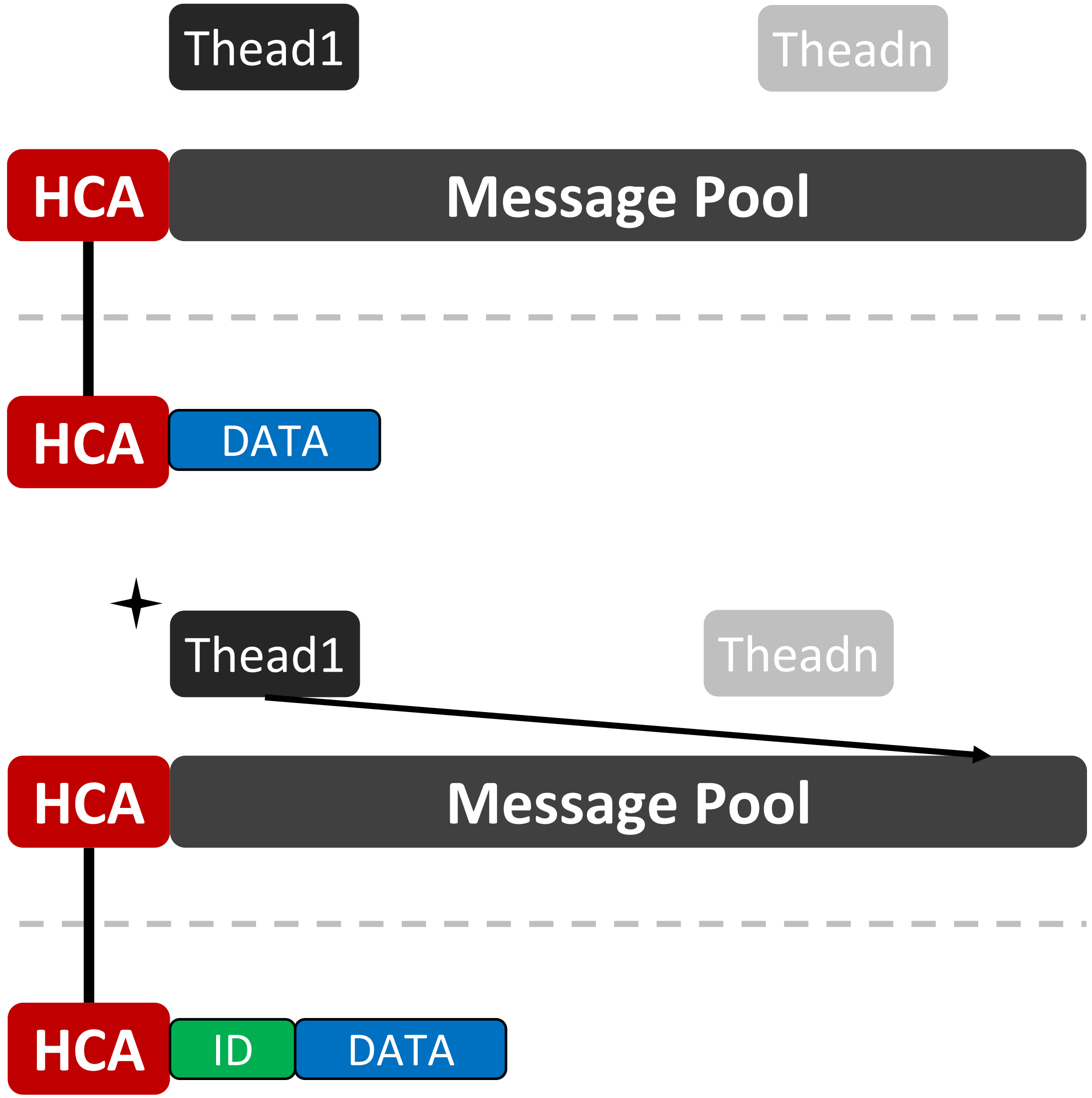
2. Client-Active Data I/O

- Server-Active
  - Server threads process the data I/O
  - Works well for slow Ethernet
  - CPUs can easily become the bottleneck with fast hardware
- Client-Active
  - Let clients read/write data directly from/to the SPMP



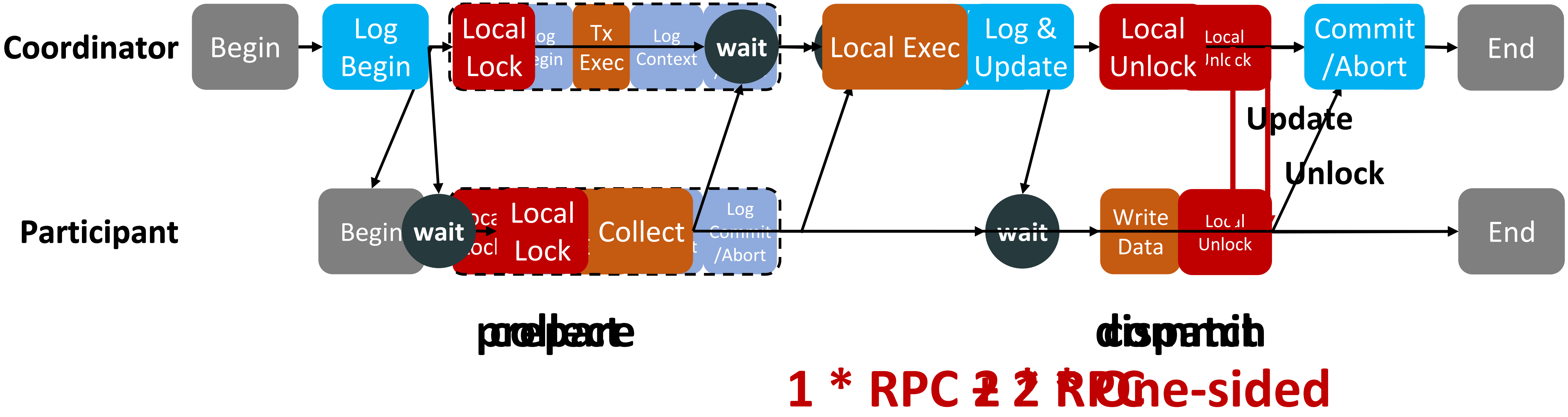
3. Self-Identified Metadata RPC

- Message-based RPC
  - easy to implement, lower throughput
  - DaRPC[SoCC'14], FaSST[OSDI'16]
- Memory-based RPC
  - CPU cores scan the message buffer
  - FaRM[NSDI'14]
- Using rdma_write_with_imm?
  - Scan by polling
  - Imm data for self-identification



4. Collect-Dispatch Distributed Transaction

- mkdir, mknod operations need distributed transactions
- Collect-Dispatch Transaction / Two-Phase Locking & Commit
  - Distributed logging / Local logging with remote in-place update
  - Distributed coordination / Two-phased lock



Outline

- Background and Motivation
- Octopus Design
- Evaluation
- Conclusion

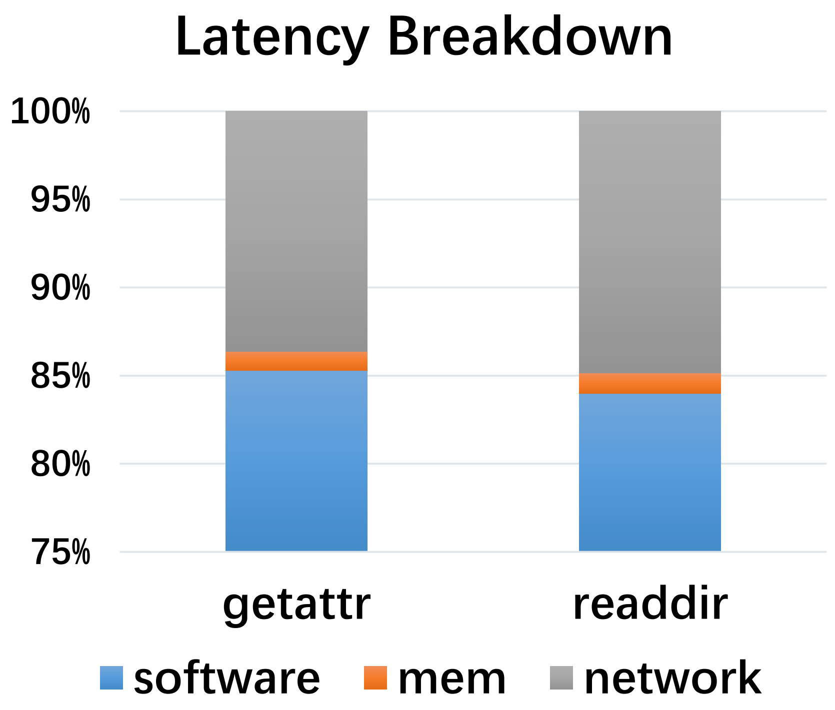
Evaluation Setup

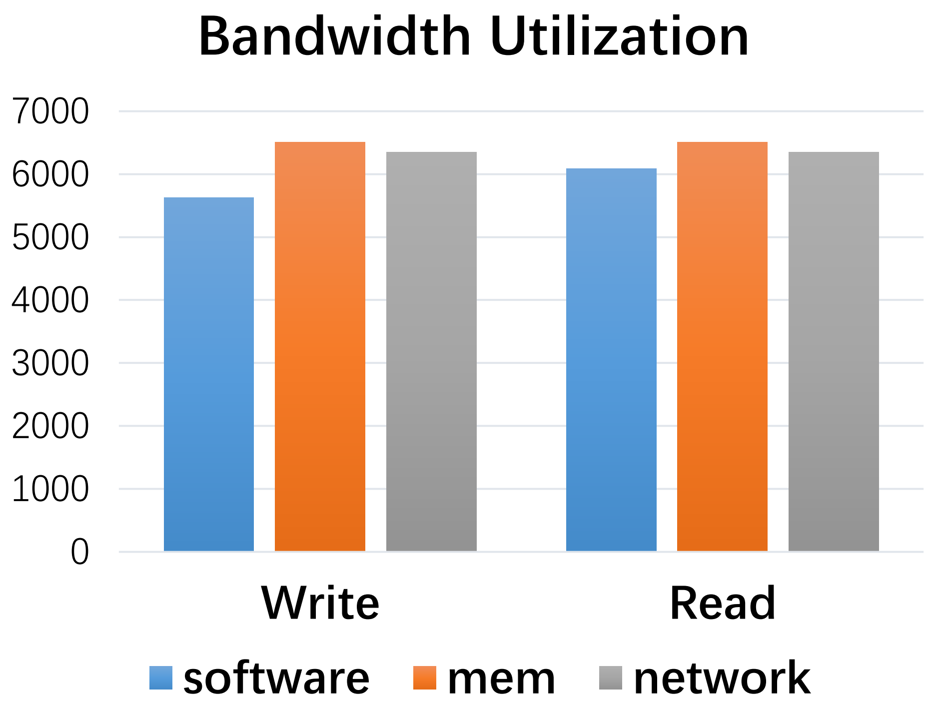
- Evaluation Platform

Cluster	CPU	Memory	ConnectX-3 FDR	Number
A	E5-2680 * 2	384 GB	Yes	* 5
B	E5-2620	16 GB	Yes	* 7

  - Connected with Mellanox SX1012 switch
- Evaluated Distributed File Systems
  - memGluster, runs on memory, with RDMA connection
  - NVFS[OSU], Crail[IBM], optimized to run on RDMA
  - memHDFS, Alluxio, for big data comparison

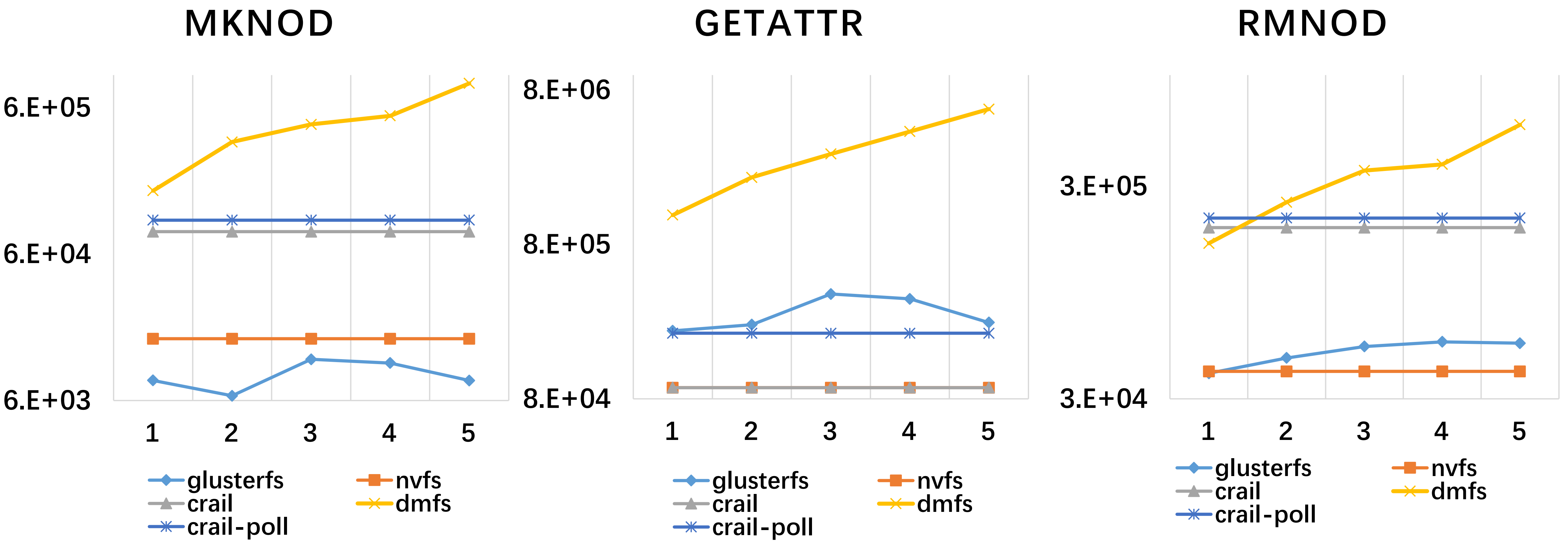
Overall Efficiency





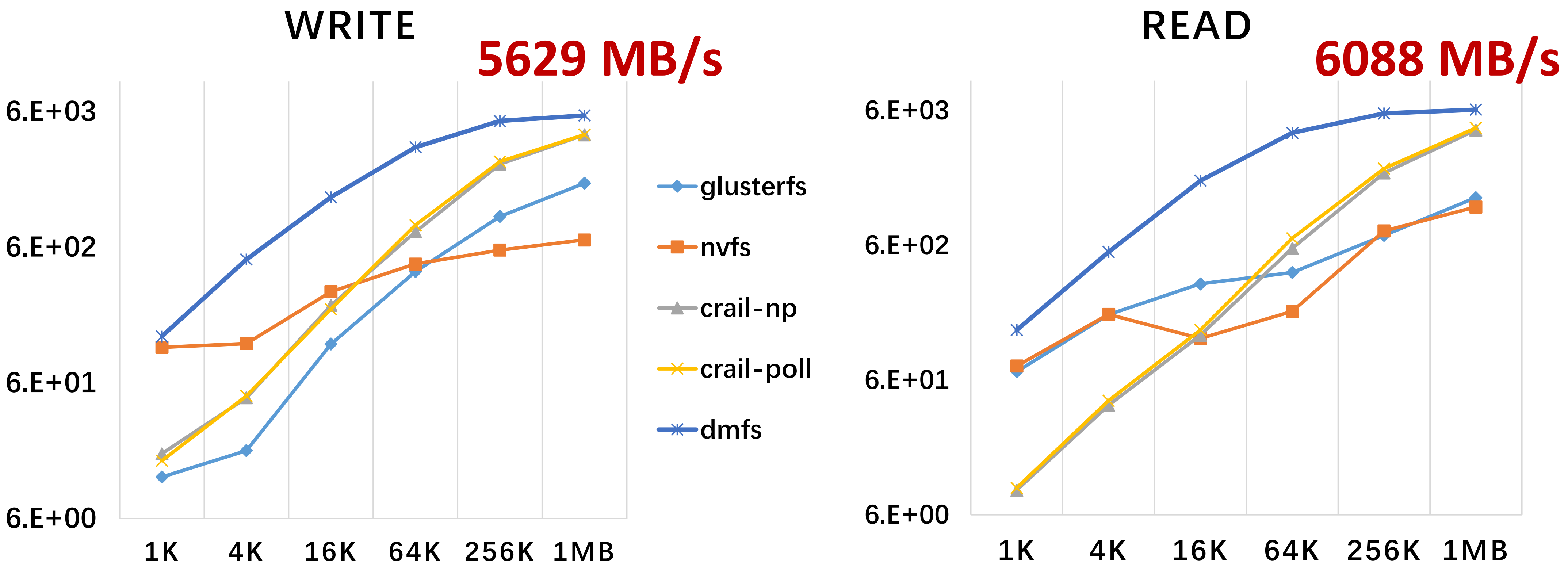
- Software latency is reduced rom **326 us** to **6 us**
- Achieves read/write bandwidth that approaches the raw storage and network bandwidth

Metadata Operation Performance



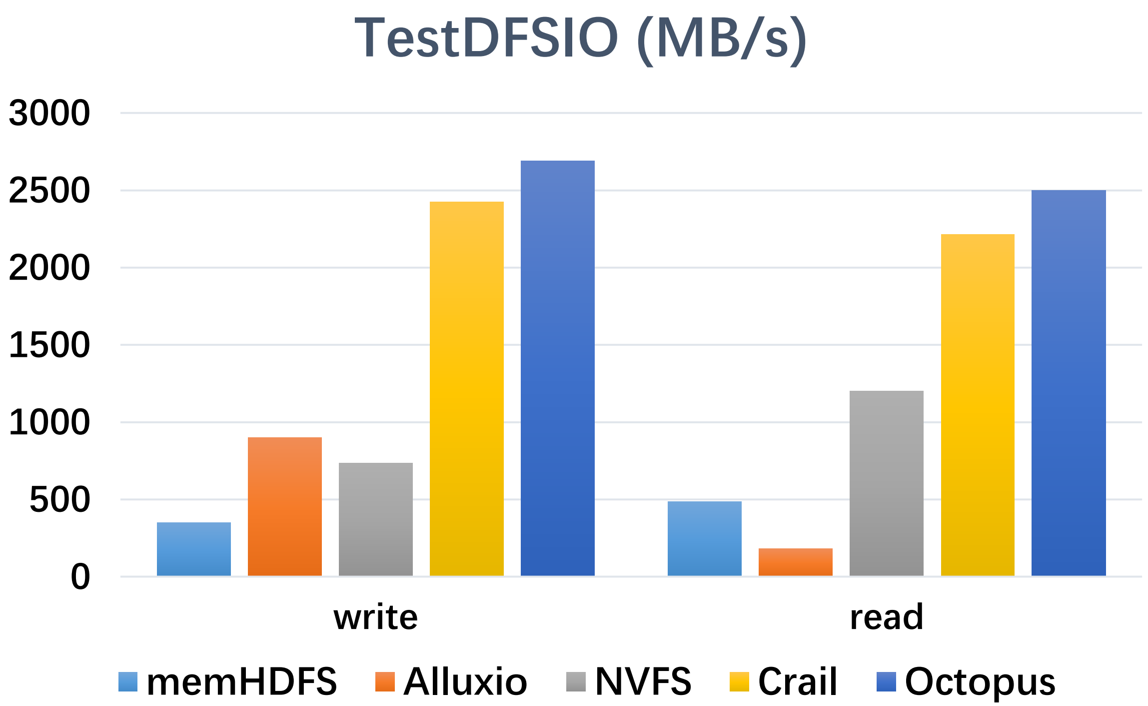
- Octopus provides metadata IOPS in the order of $10^5 \sim 10^6$
- Octopus can scales linearly

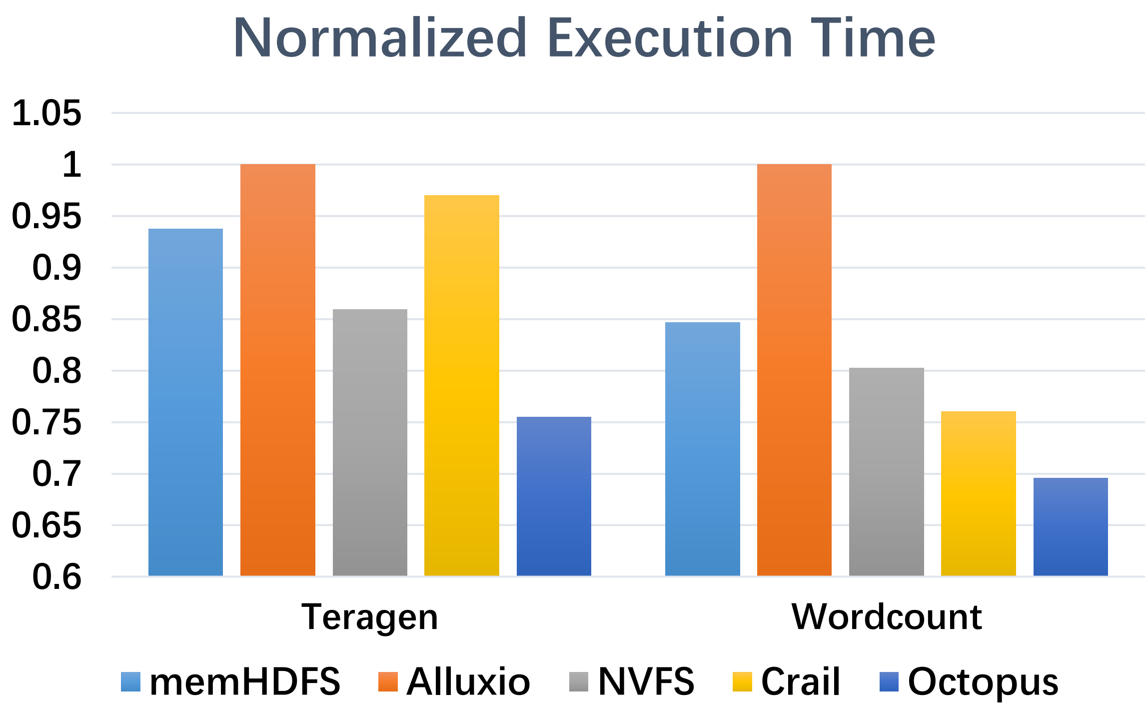
Read/Write Performance



- Octopus can easily reach the maximum bandwidth of hardware with a single client
- Octopus can achieve the same bandwidth as Crail even add an extra data copy [not shown]

Big Data Evaluation





- Octopus can also provide better performance for big data applications than existing file systems.

Outline

- Background and Motivation
- Octopus Design
- Evaluation
- Conclusion

Conclusion

- It is necessary to rethink the DFS designs over emerging H/Ws
- Octopus's internal mechanisms
  - Simplifies data management layer by **reducing data copies**
  - **Rebalances network and server loads** with Client-Active I/O
  - Redesigns the **metadata RPC** and **distributed transaction** with RDMA primitives
- Evaluations show that Octopus significantly outperforms existing file systems

Q&A

Thanks