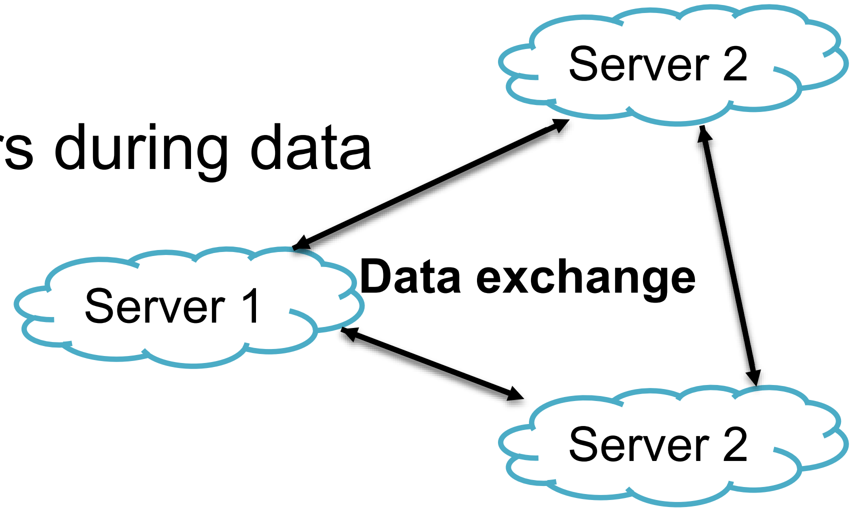
Zero-Change Object Transmission for Distributed Big Data Analytics

Mingyu Wu, Institute of Parallel and Distributed Systems, SEIEE, Shanghai Jiao Tong University and Shanghai AI Laboratory; Shuaiwei Wang, Institute of Parallel and Distributed Systems, SEIEE, Shanghai Jiao Tong University; Haibo Chen and Binyu Zang, Institute of Parallel and Distributed Systems, SEIEE, Shanghai Jiao Tong University and Engineering Research Center for Domain-specific Operating Systems, Ministry of Education, China

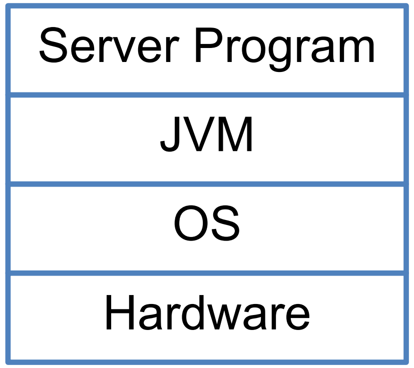
ATC'22

Background

- **Distributed big data**
  - Data is distributed in different server
  - Data needs to be exchanged between servers during data analysis
  - Data is generally read-only



- **Server runtime environment**
  - JVM (java virtual machine)
  - Data is exchanged between different JVM instances



Background

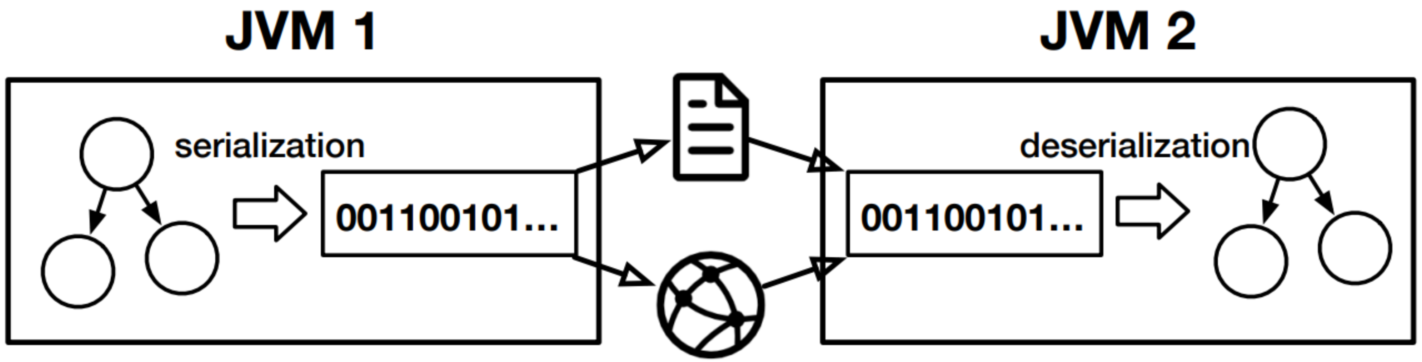
- **How to exchange: Serialization and Deserialization**
  - Data is stored in JVM's memory in the form of objects
  - Serialization: Object -> binary array (standardized method)
  - Deserialization: Binary array -> Object (standardized method)

Object

Class info
data filed

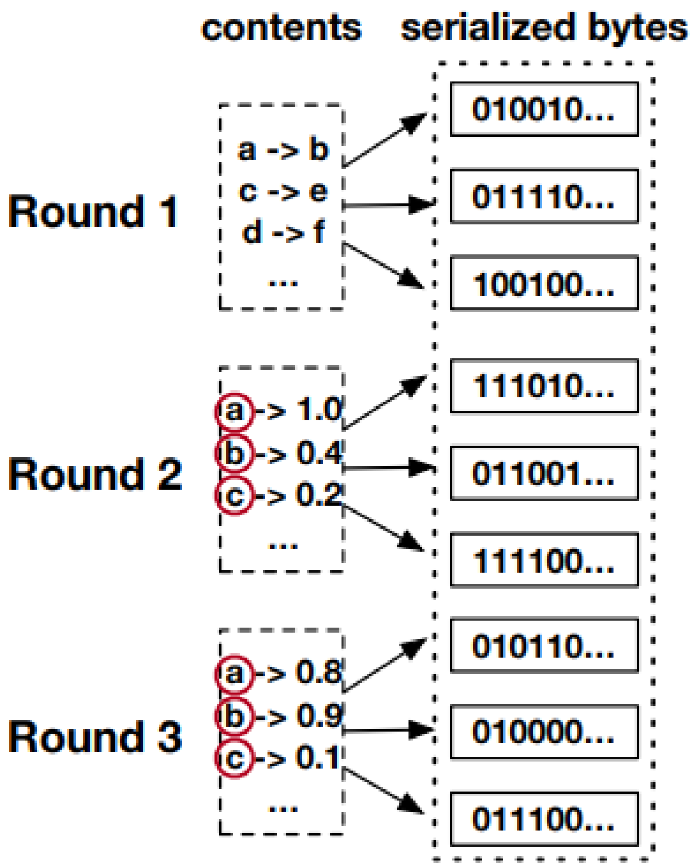
Binary array

metadata
Binary data



Problem

- **CPU overhead**
  - Object ←[CPU]→ Binary array
- **Memory overhead**
  - Additional memory overhead during (de)serialization
  - Data redundancy due to standardized methods (extra metadata)
- **Overhead from repeated data transmission (in specific scenario)**
  - iterative algorithm
  - …



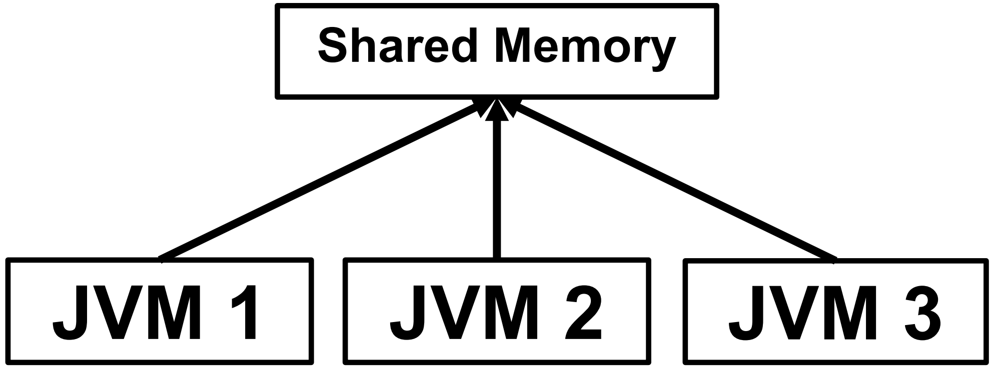
Idea & Challenge

- **IDEA**

  Remove (De)serialization

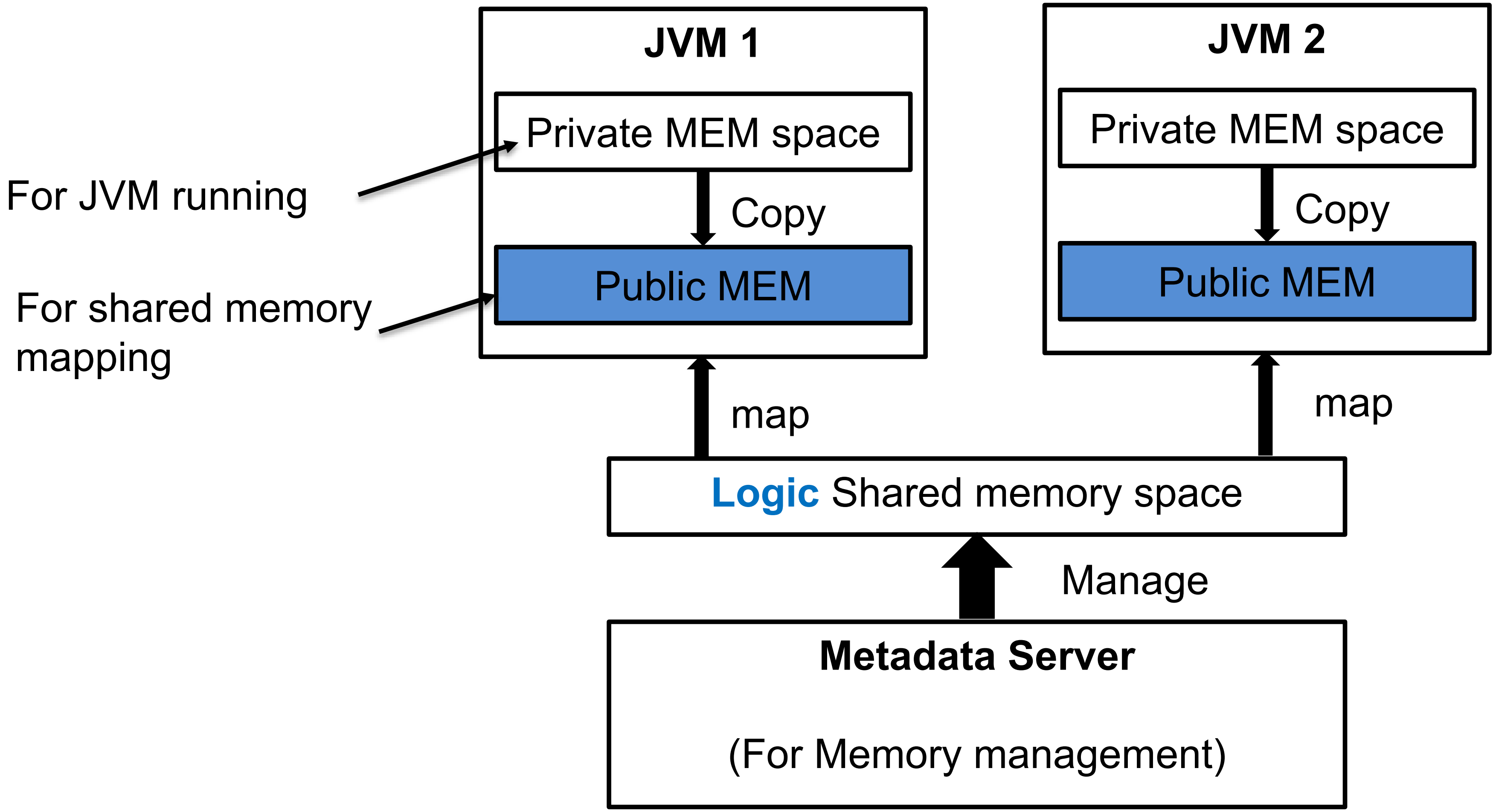
  Use distributed shared memory for data exchange

- **Challenge**
  - Architecture design
  - Data exchange protocol
  - Shared memory management
  - Remove redundant data transmission



Architecture Design

- **Architecture**



Lazy data exchange protocol

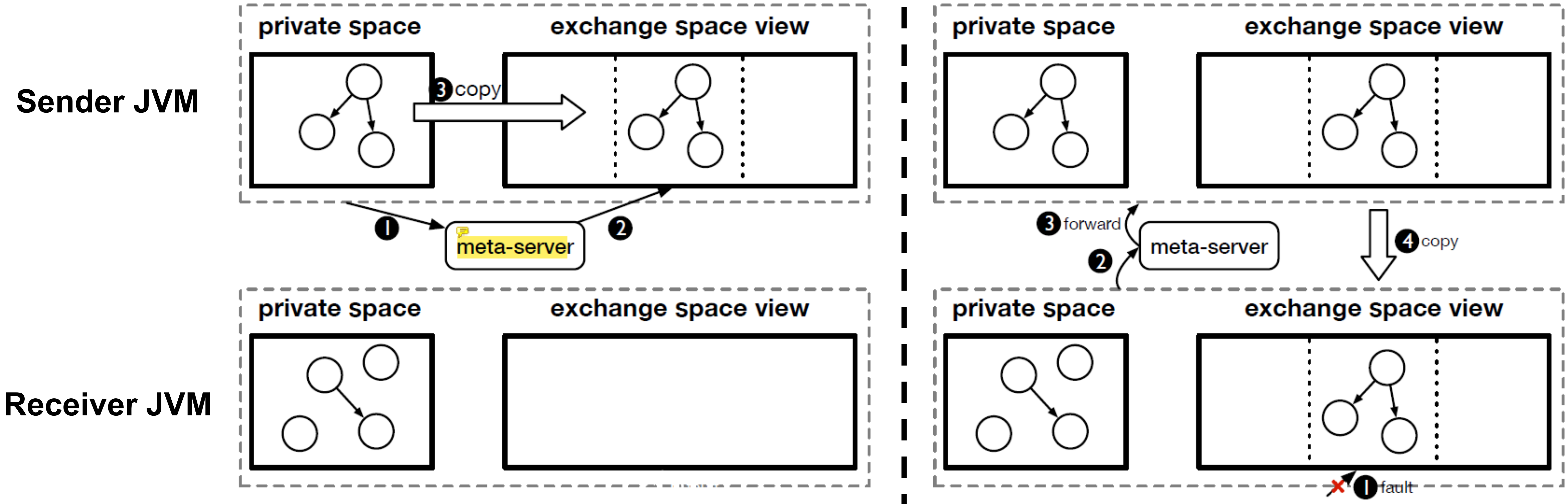
- **Sender**
  - Acquire memory from shared memory space
  - Copy data to public MEM in specific address (still in local memory)

- **Receiver**
  - Read remote memory from shared memory space
  - Memory is in local -> Read Directly
  - Memory is in remote (mapped to another JVM) -> Build network connection

- **Problem & Guess : How to know the address of data**
  - There is additional network communication to exchange extra info

Data exchange protocol

➢ **Example**



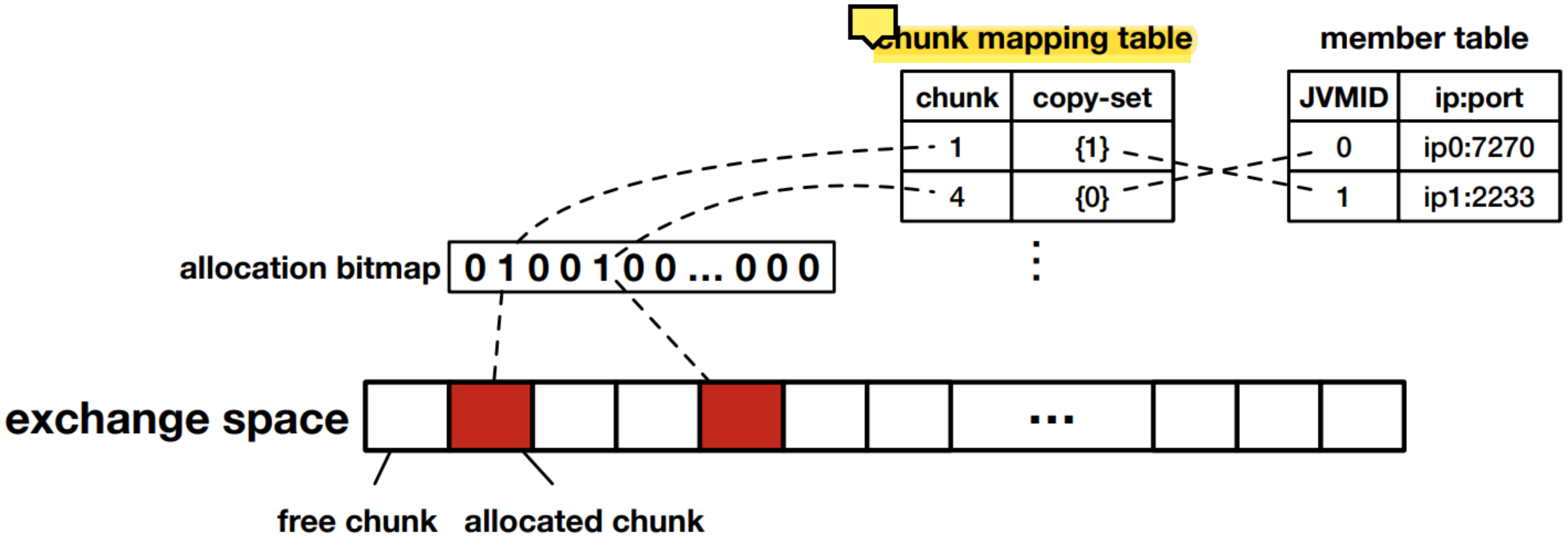
View of Sender

View of receiver

Shared Memory Management (Metadata server)

➢ **Logical shared memory layout**

- Allocation bitmap
- Member table (JVM info table)
- Chunk mapping table (chunk id → {JVMs})
  (check if a JVM has a copy of this memory chunk)



Shared Memory Management (JVM)

➢ **Acquire()->address**
- When: has no public memory to store data
- How: Metadata server: scan bitmap and allocate chunk

➢ **Get remote(address)**
- When: memory needed is not in local
- How: Metadata server find memory location and help build connection to copy memory chunk

➢ **Release(address)**
- When: memory chunk is not needed (or GC)
- How: Metadata server: reclaim memory

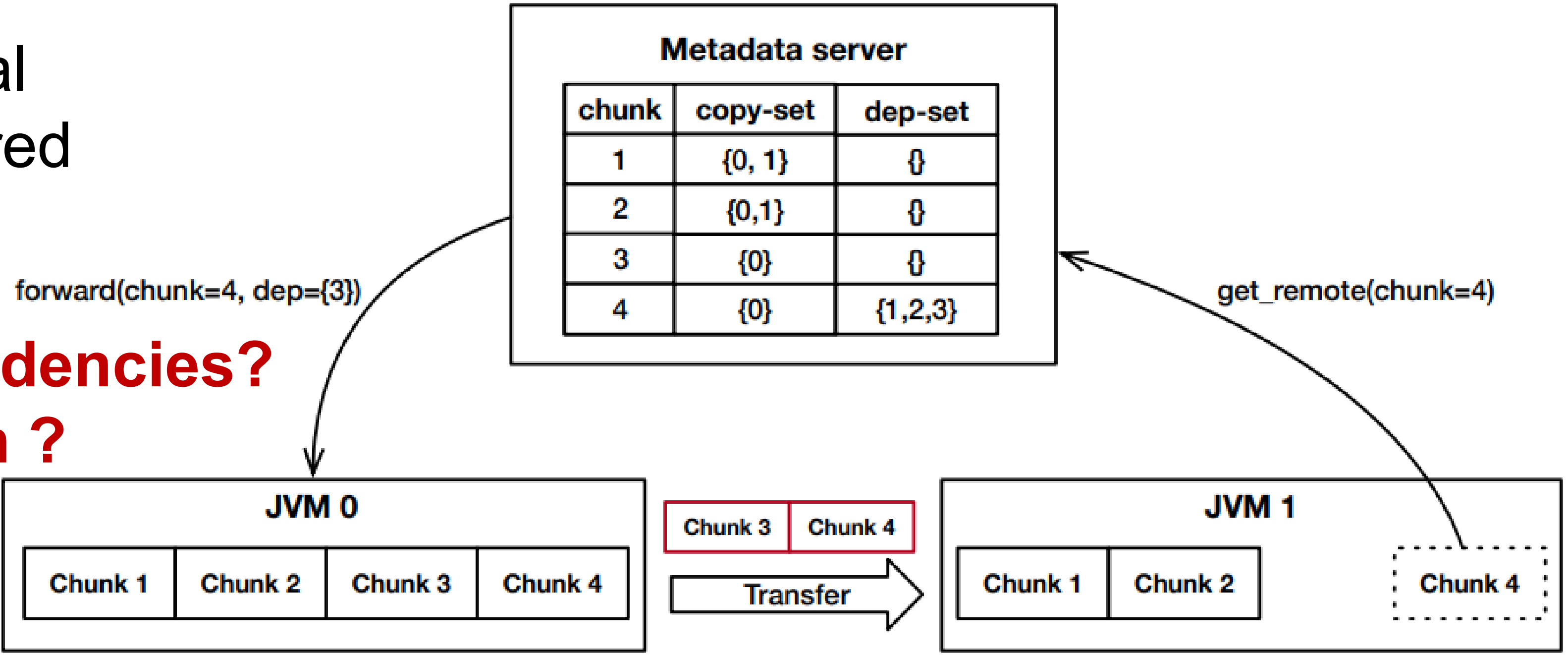
Question: How JVM manage their public memory? (no answer)

Data Duplication in Exchange(?)

- **Build reference between memory chunks**
  - B reference A : Some objects in A are the same as those in B

- **Only dependent chunks will be transferred**
  1. JVM: need chunk 4
  2. 4 -> {1,2,3}
  3. 1,2 are already in local
  4. Only 3 will be transferred

- **Problems**
  - **How to maintain dependencies?**
  - **How to save bandwidth ?**



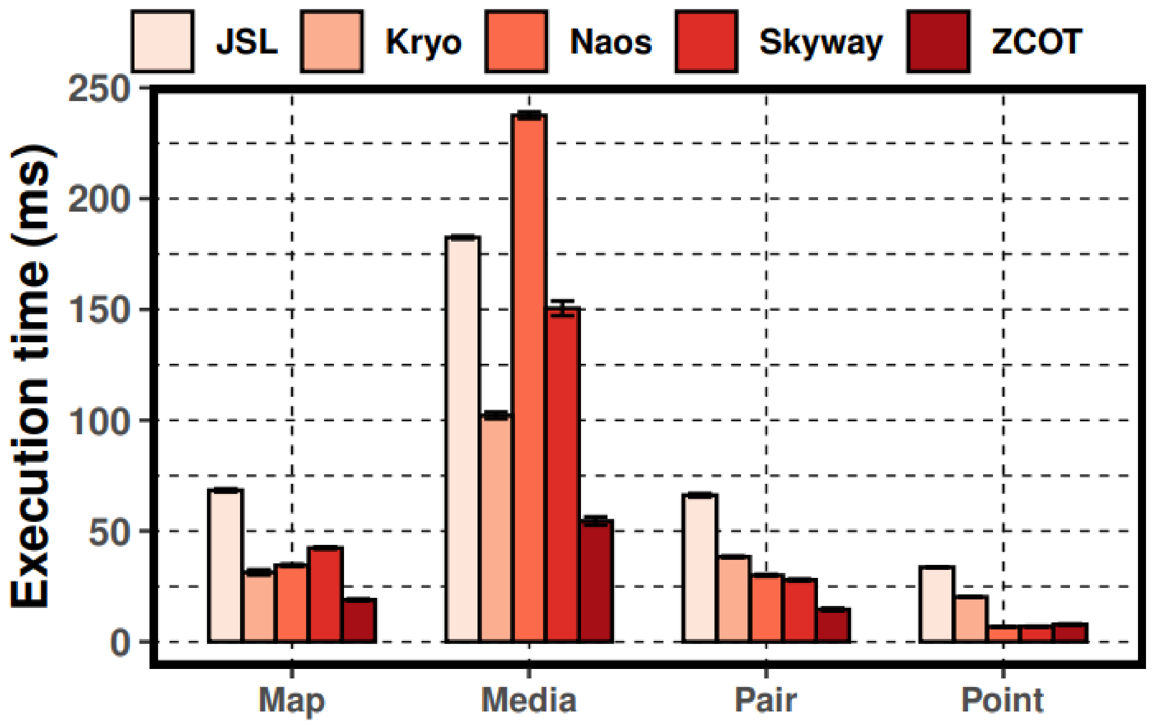
Evaluation

- **Hardware**
  - Cluster with 4 nodes + Xeon E5-2650 CPU * 2 + 128GB DRAM
- **Software platform**
  - JVM: Hotspot JVM 11
  - Microbenchmark
  - Spark (data analytics engine)
  - Flink (data processing engine)
- Compare objects
  - JSR → (standard method, baseline)
  - Kryo →(refined binary data format, smaller size)
  - Naos → (bases on RDMA, no memory copy/ (De)serialization)
  - Skyway → (no (De)serialization + extra metadata)
  - **ZCOT**

RDMA: remote direct memory access

Evaluation

- **Microbenchmark**
  Under two machines
- **Overall time**
  Media > Map > Pair > Point
  (different data complexities)
- **Performance**
  ZCOT > (JSL, Kryo, Naos, Skyway)

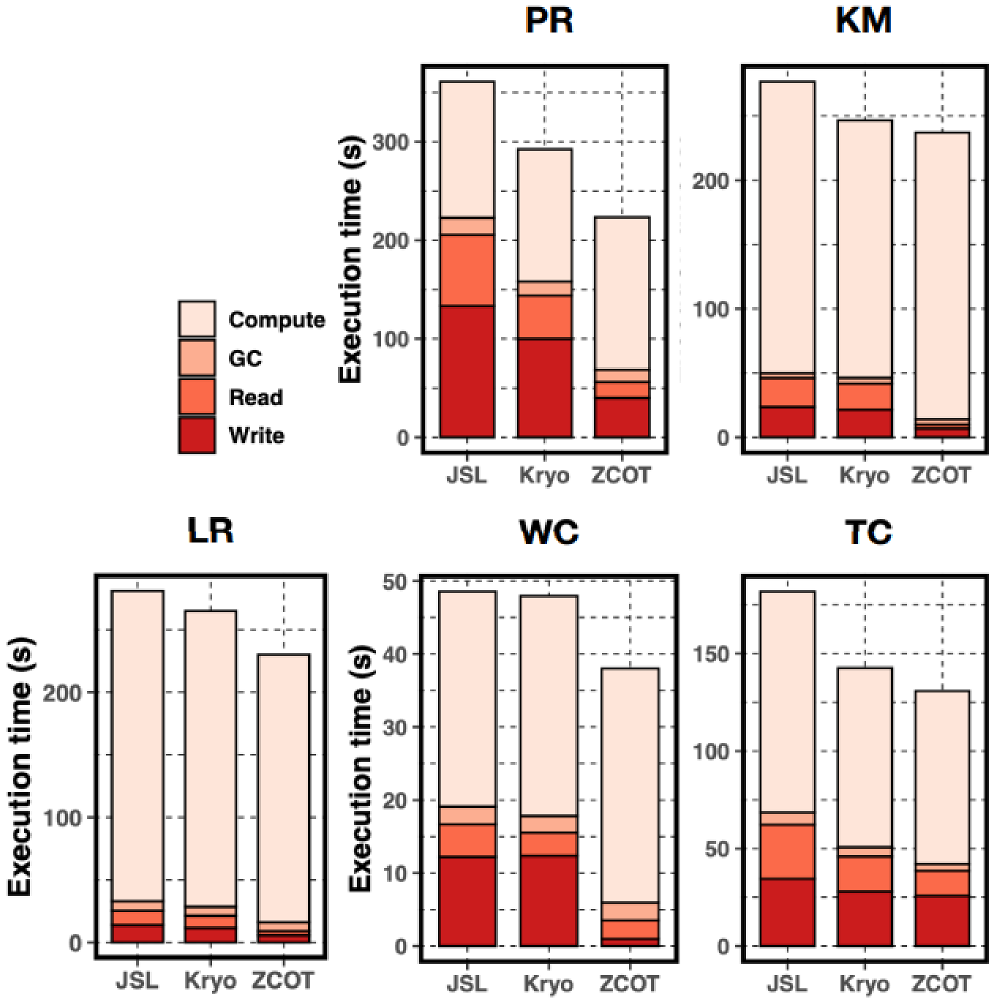


Evaluation

- **Spark**
- **Overall**
  ZCOT > Kryo > JSL
  (Lower read and write times)
- **Total data transferred**
  Data de-duplication works

	PR	WC	TC	KM	LR
dedup	15.25	4.13	5.03	5.37	5.55
no-dedup	31.64	5.50	10.88	5.86	6.04

Application	Dataset
PageRank (PR)	LiveJournal [4]
Word Count (WC)	LiveJournal
KMeans (KM)	USCensus1990 [10]
Transitive Closure (TC)	Blogs [1, 17]
Logistic Regression (LR)	SUSY [5]



Conclusion

Problem

(De)Serialization overhead In Big Data Analytics

Idea

Use **distributed shared memory** to remove serialization process

Design

- DSM design & management
- Data exchange protocol
- Remove duplicate data transmission