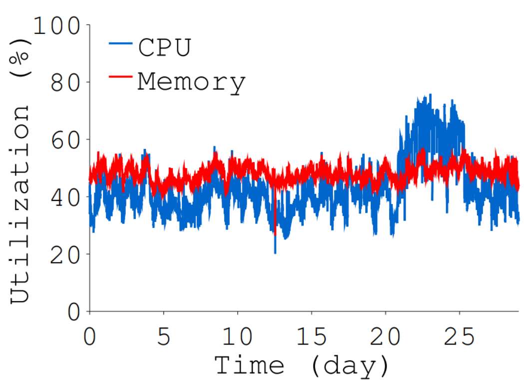
ODRP: On-Demand Remote Paging with Programmable RDMA

Zixuan Wang, Xingda Wei, Jinyu Gu, Hongrui Xie,
Rong Chen, Haibo Chen

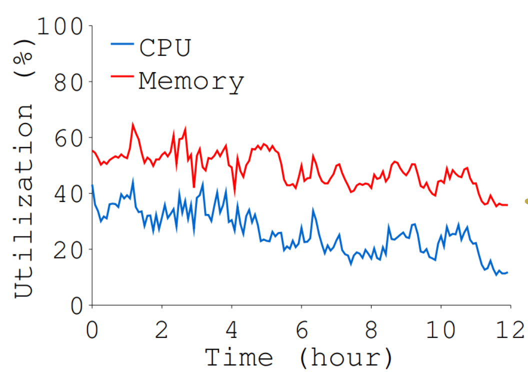
Institute of Parallel and Distributed Systems (IPADS)
Shanghai Jiao Tong University

Low Memory Utilization in Datacenters

- **Over-provisioning Memory**
  - The memory demand of memory-intensive applications ***varies*** over time.
  - Datacenters allocate memory based on their ***peak usage*** to meet their SLOs.



(a) Google Cluster

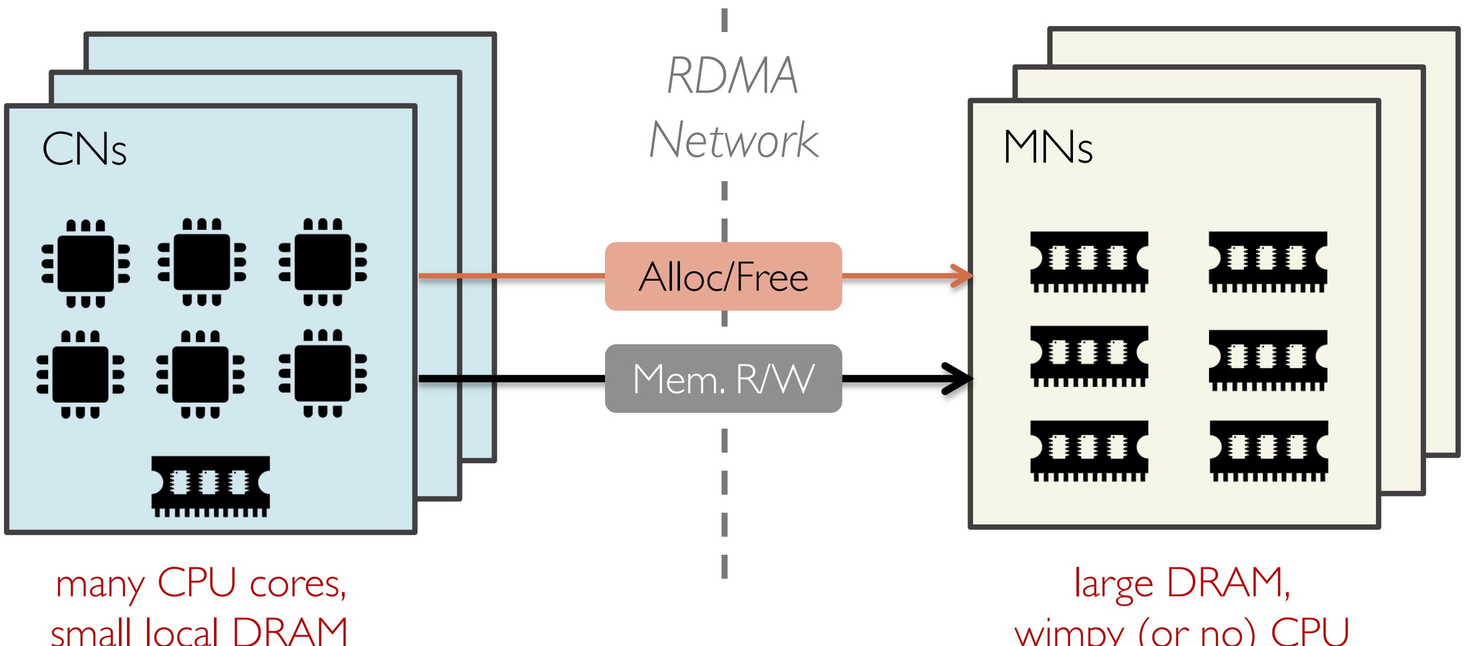


(b) Alibaba Cluster

Low average memory utilization!

Disaggregated Memory

- **Disaggregated Memory Architecture**
  - Physically separate CPU and memory resources into ***network-attached*** pools, namely compute nodes (CNs) and memory nodes (MNs).

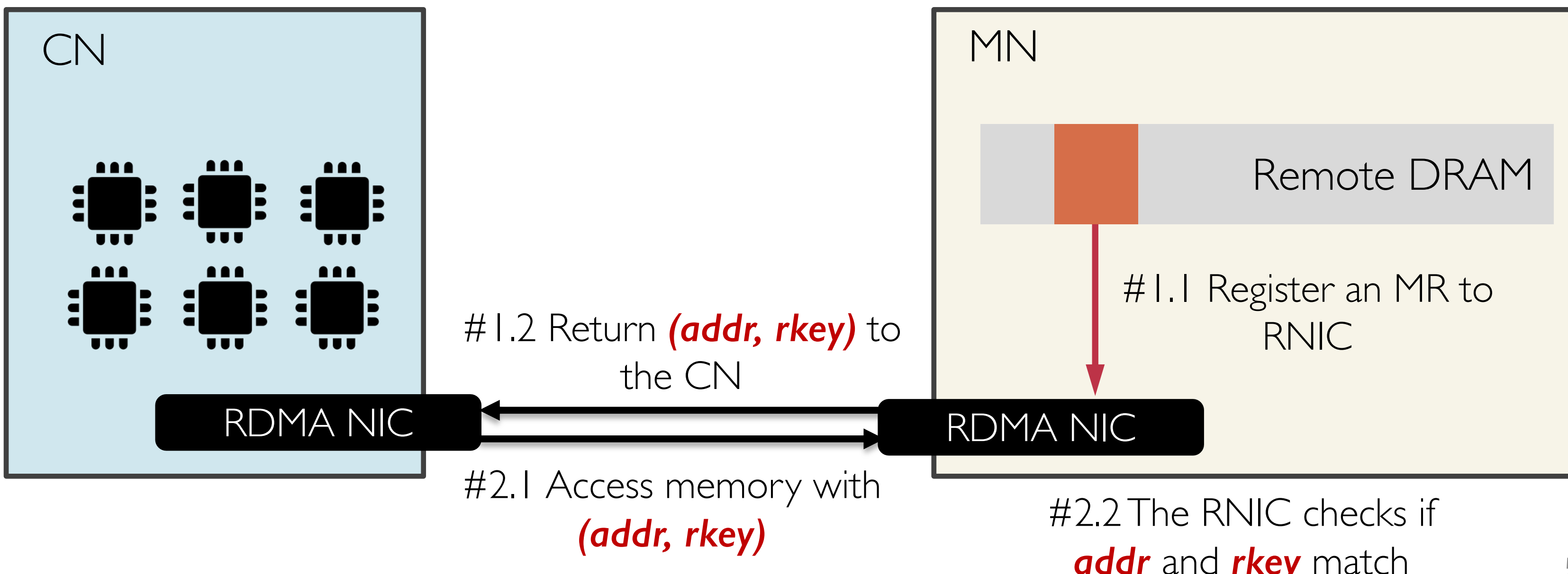


Disaggregated Memory Management

- **Memory Management in DM relies on Memory Region (MR).**
  - Requirement #1: ***Allocate/Free*** memory chunks to CNs.
  - Requirement #2: Ensure ***Memory Isolation*** between CNs.

Disaggregated Memory Management

- **Memory Management in DM relies on Memory Region (MR).**
  - Requirement #1: ***Allocate/Free*** memory chunks to CNs.
  - Requirement #2: Ensure ***Memory Isolation*** between CNs.

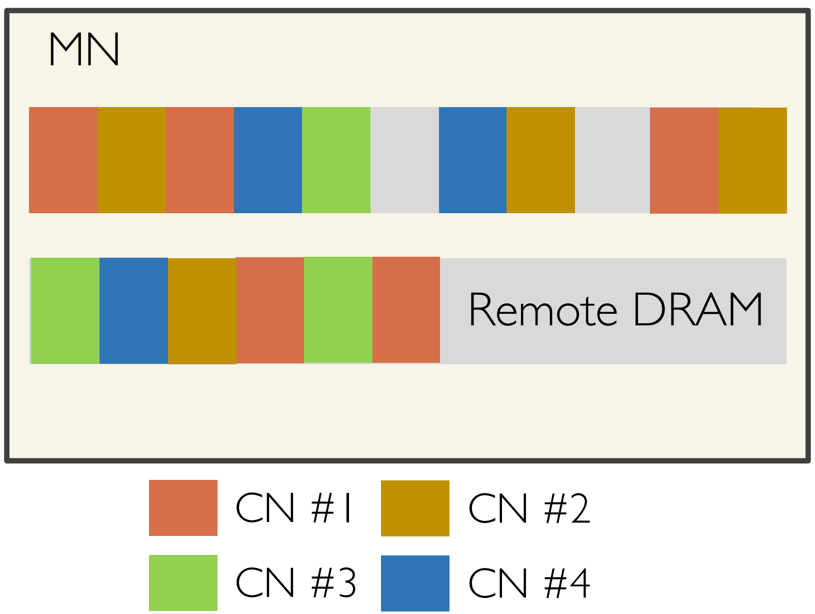


Existing Memory Management Approaches

- **#1 Fine-grained memory management with MR**
  - MNs register small MRs (e.g., 1 MB) and allocate them to CNs dynamically.

Existing Memory Management Approaches

- **#1 Fine-grained memory management with MR**
  - MNs register small MRs (e.g., 1 MB) and allocate them to CNs dynamically.



Pro:* *High memory utilization* *due to fine allocation granularity.

Limitation:* *Poor Performance

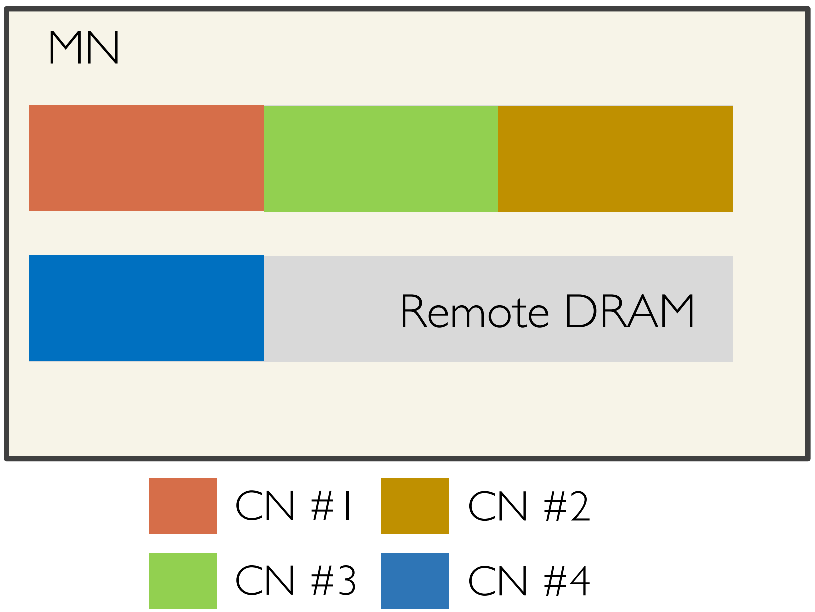
- ***frequent allocation requests***
- ***time-consuming MR registration***
- ***wimpy MN CPU***

Existing Memory Management Approaches

- **#2 Static memory management with MR**
  - MNs register large MRs (≥ 1 GB) and allocate them to CNs statically.

Existing Memory Management Approaches

- **#2 Static memory management with MR**
  - MNs register large MRs (≥ 1 GB) and allocate them to CNs statically.



***Pro:** High performance due to few (or no) allocation requests during runtime*

***Limitation:** Poor memory utilization due to severe internal fragmentation.*

Existing Memory Management Approaches

- **Existing approaches cannot achieve *high-performance, high memory utilization, and no MN CPU usage* at the same time.**

Goals

#1 Fine-grained memory management for ***high memory utilization***

#2 Fast data-path memory allocation for ***high performance***

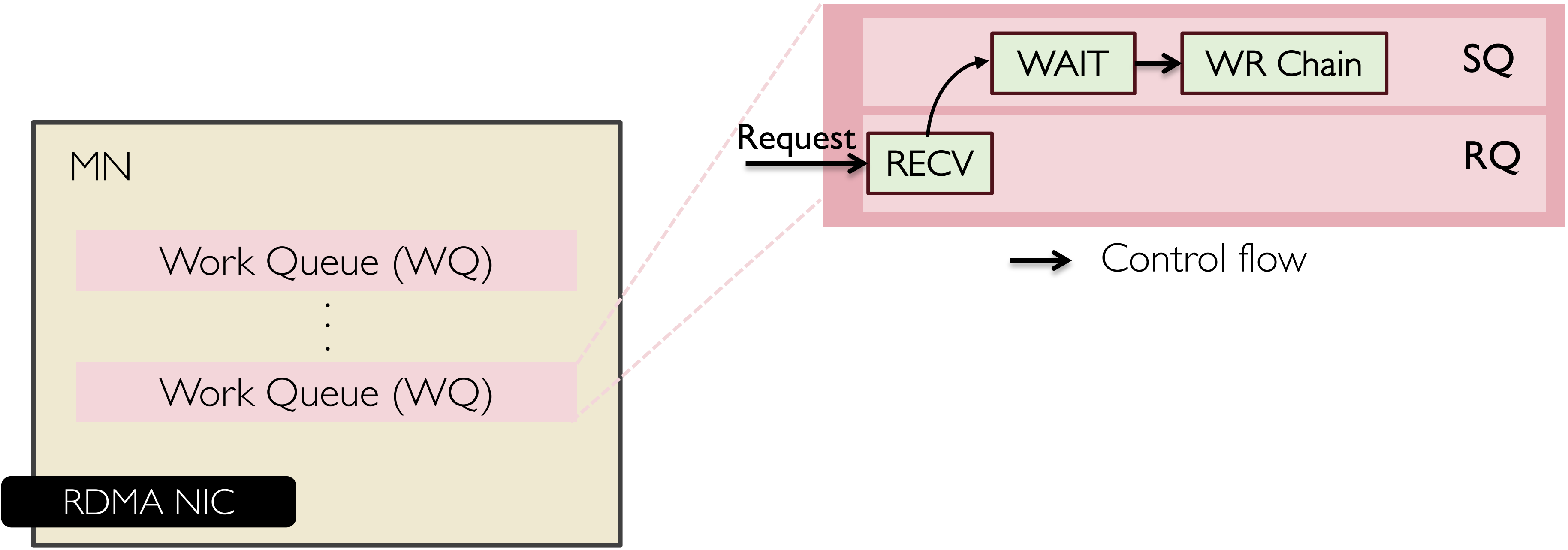
#3 No MN CPU usage to realize ***full disaggregation***

Opportunity – RNIC Offloading

- **Feature #1: Event-based work request triggering.**

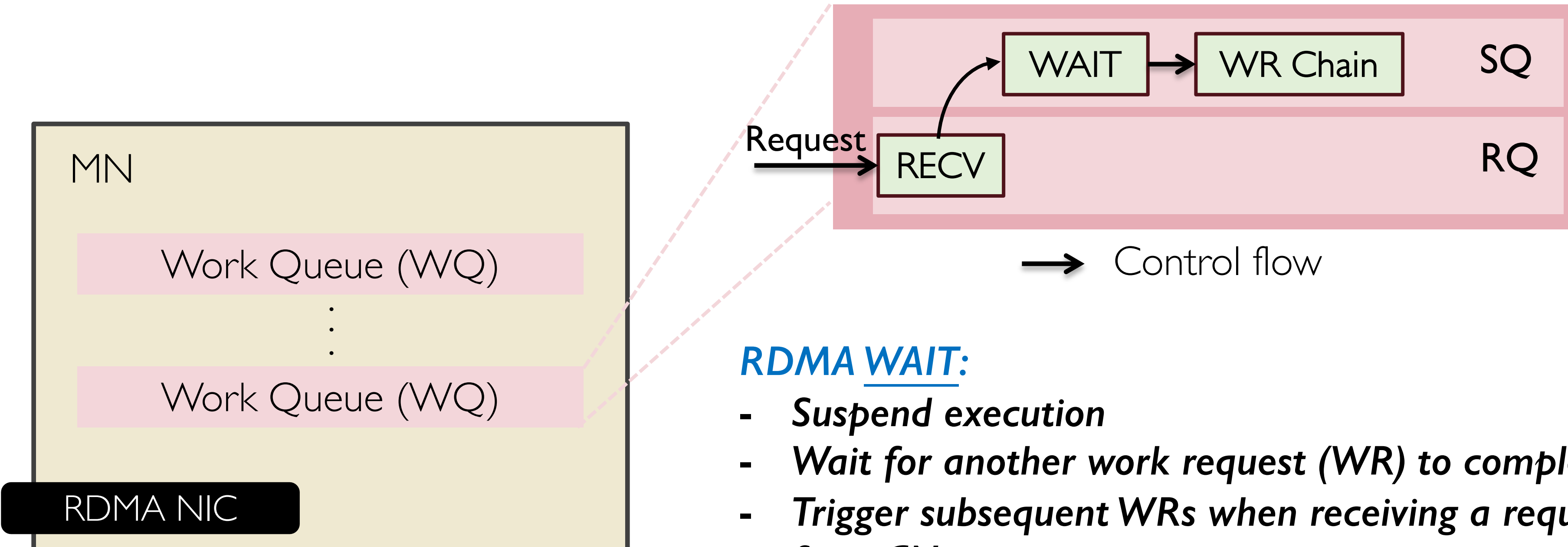
Opportunity – RNIC Offloading

- **Feature #1: Event-based work request triggering.**
  - Commodity RNICs support a special verb called ***WAIT***.



Opportunity – RNIC Offloading

- **Feature #1: Event-based work request triggering.**
  - Commodity RNICs support a special verb called ***WAIT***.



RDMA WAIT:

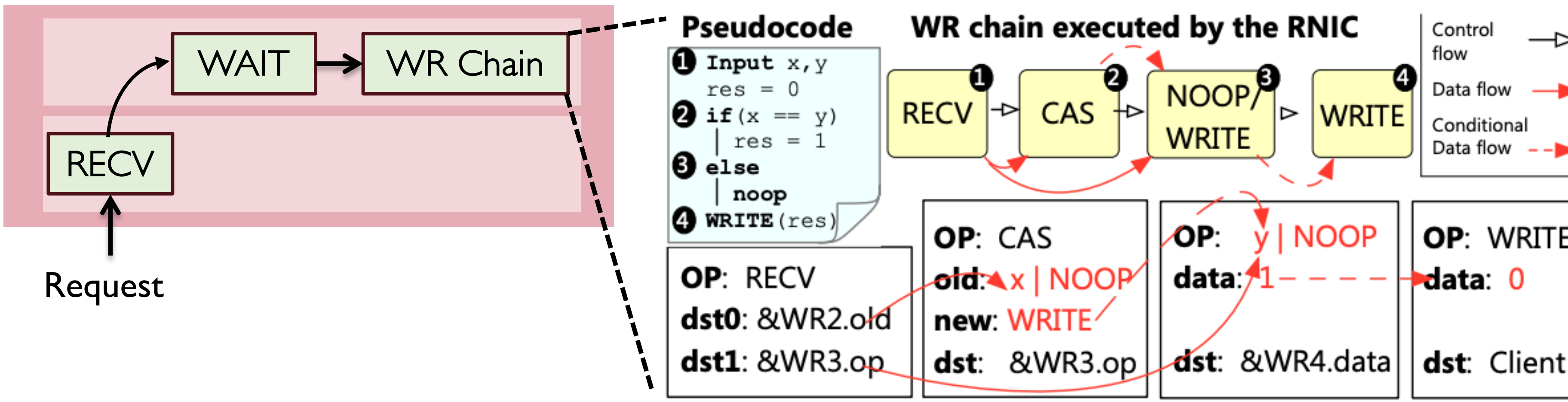
- ***Suspend execution***
- ***Wait for another work request (WR) to complete***
- ***Trigger subsequent WRs when receiving a request from CNs***

Opportunity – RNIC Offloading

- **Feature #2: Chain basic WRs to express complex logic.**
  – Previous WRs can modify the arguments of subsequent WRs.

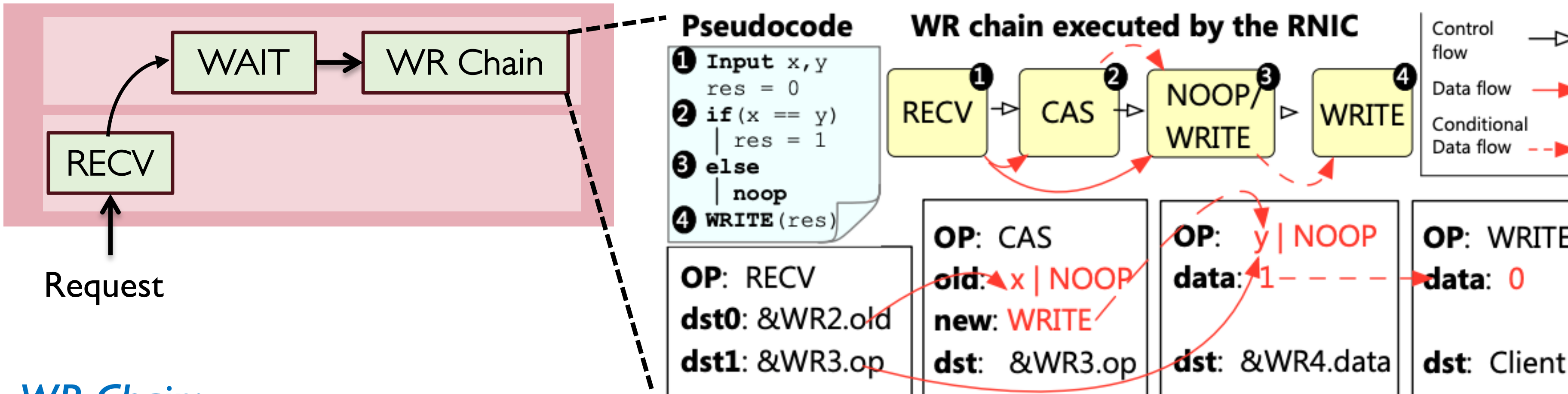
Opportunity – RNIC Offloading

- **Feature #2: Chain basic WRs to express complex logic.**
  - Previous WRs can modify the arguments of subsequent WRs.



Opportunity – RNIC Offloading

- **Feature #2: Chain basic WRs to express complex logic.**
  - Previous WRs can modify the arguments of subsequent WRs.



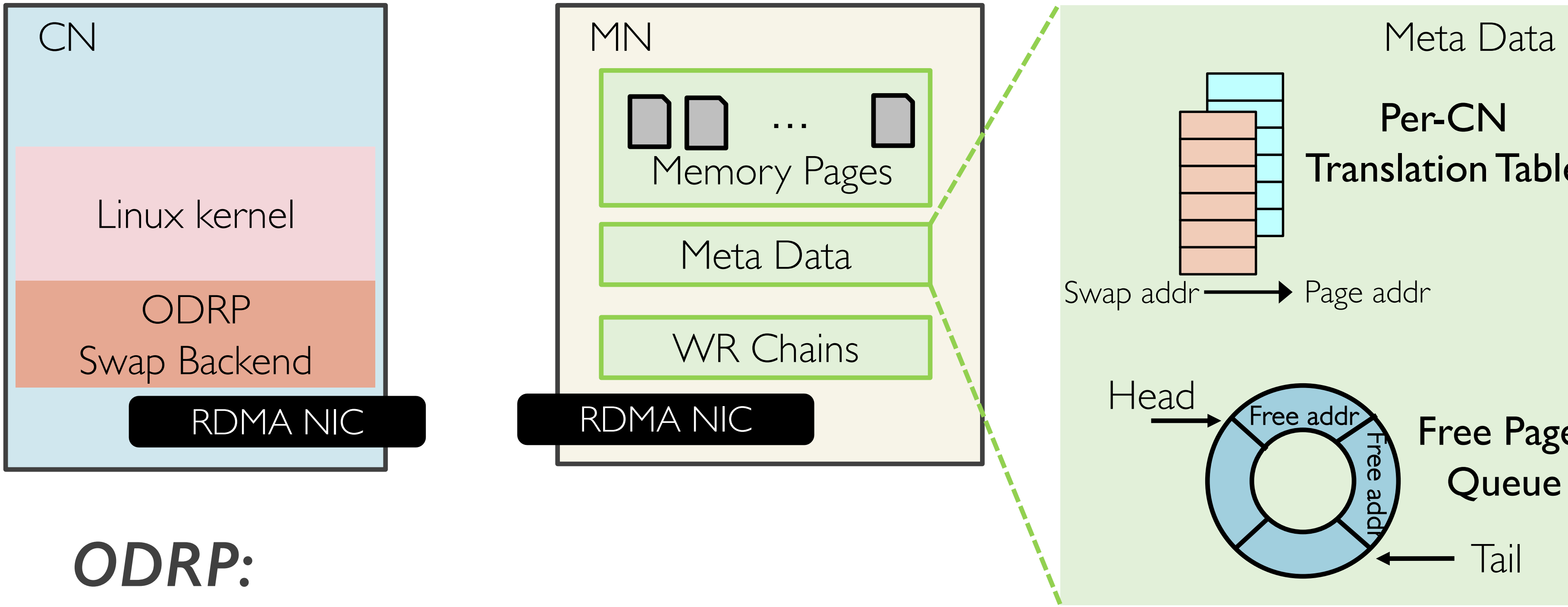
WR Chain:

- *Chain basic WRs (e.g., READ/WRITE) together*
- *Enable one-sided operations with richer semantics*

Opportunity – RNIC Offloading

Offload memory management logic to MN's RNIC.
CNs can *efficiently* allocate memory in the data path *on demand*!

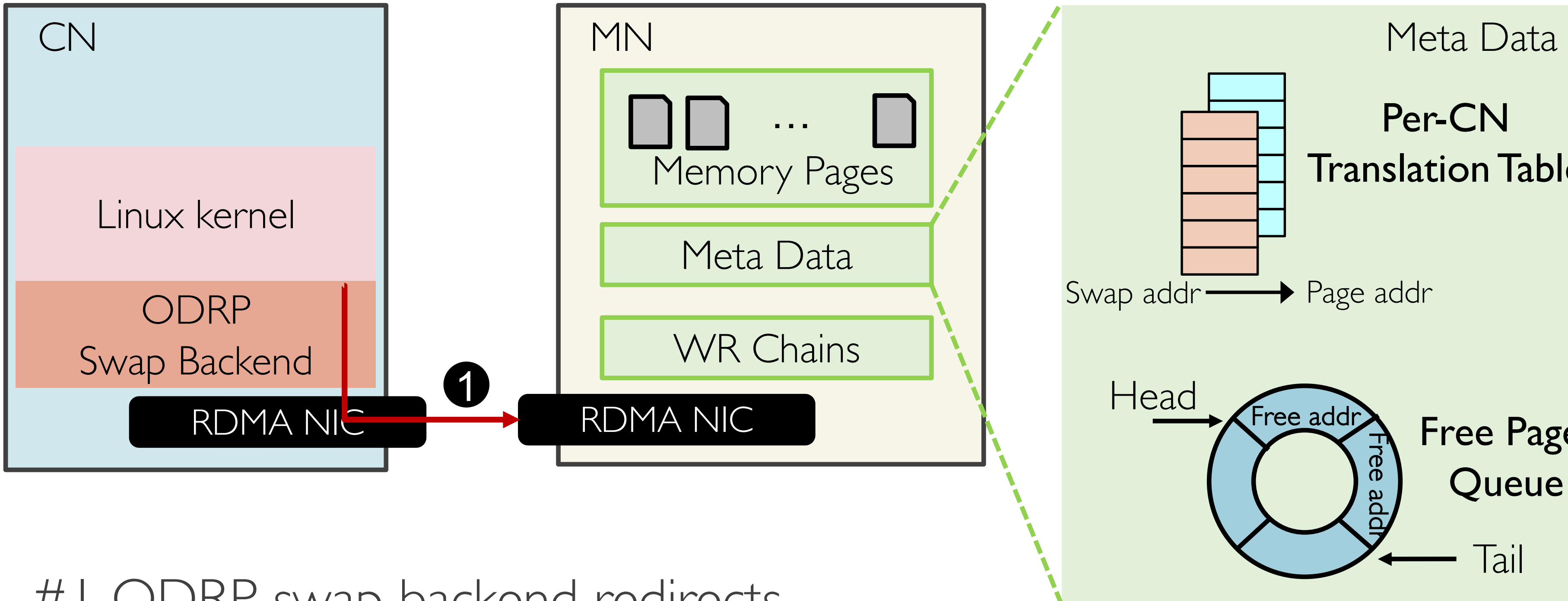
Design – Overview



ODRP:

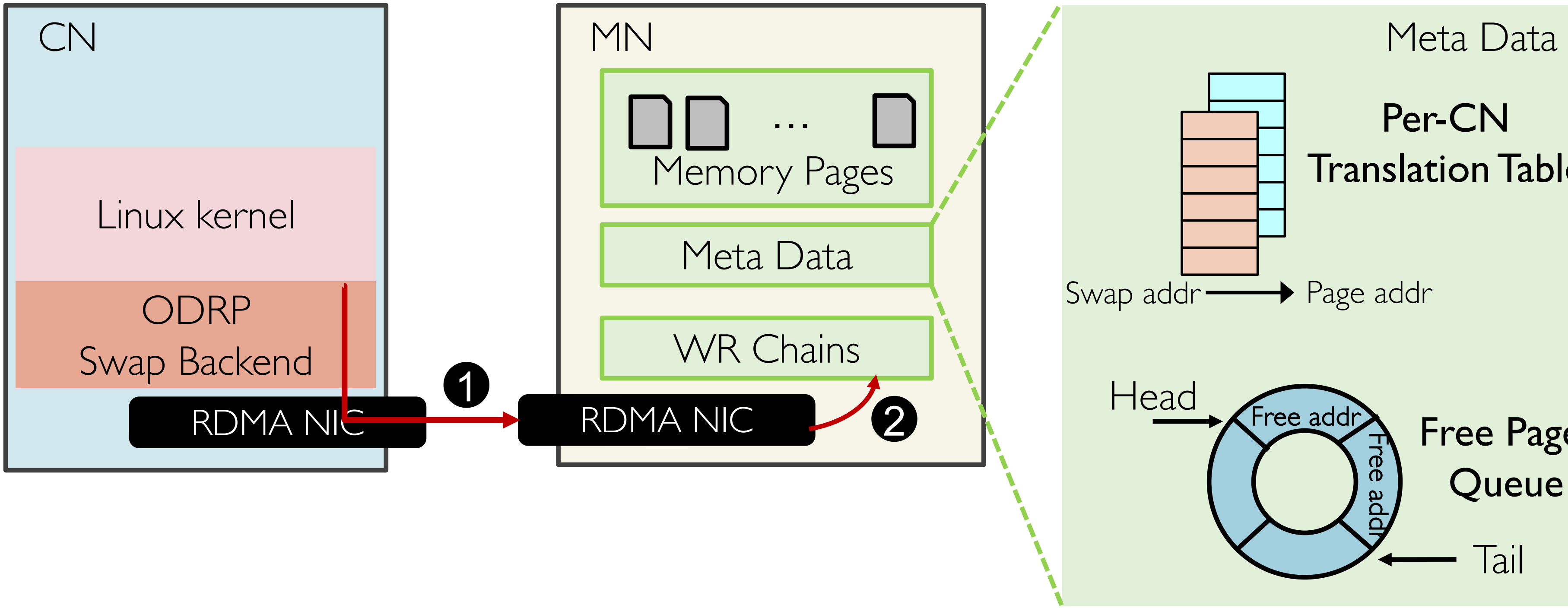
- *a swap-based system*
- *offload memory management logic to RNIC*

Design – Workflow



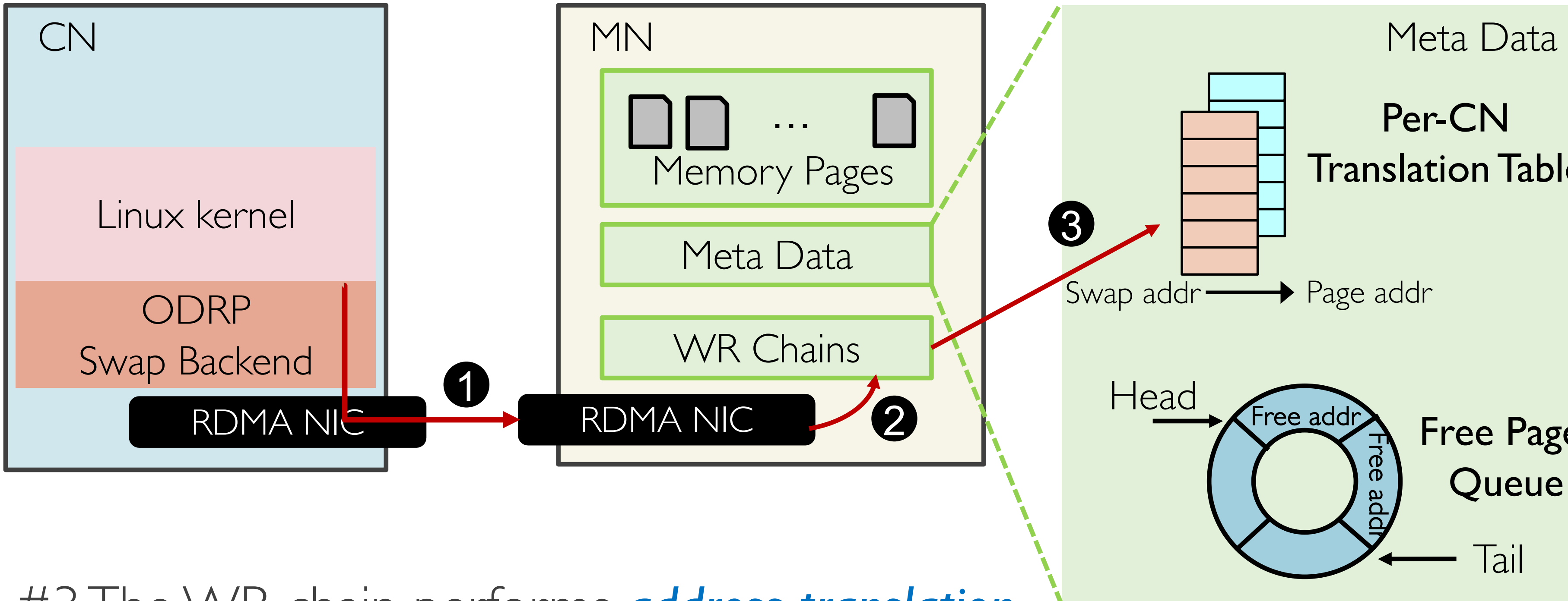
#1 ODRP swap backend redirects CNs' swap requests to the MN with RDMA SEND.

Design – Workflow



#2 The MN's RNIC then fetches and executes a WR chain.

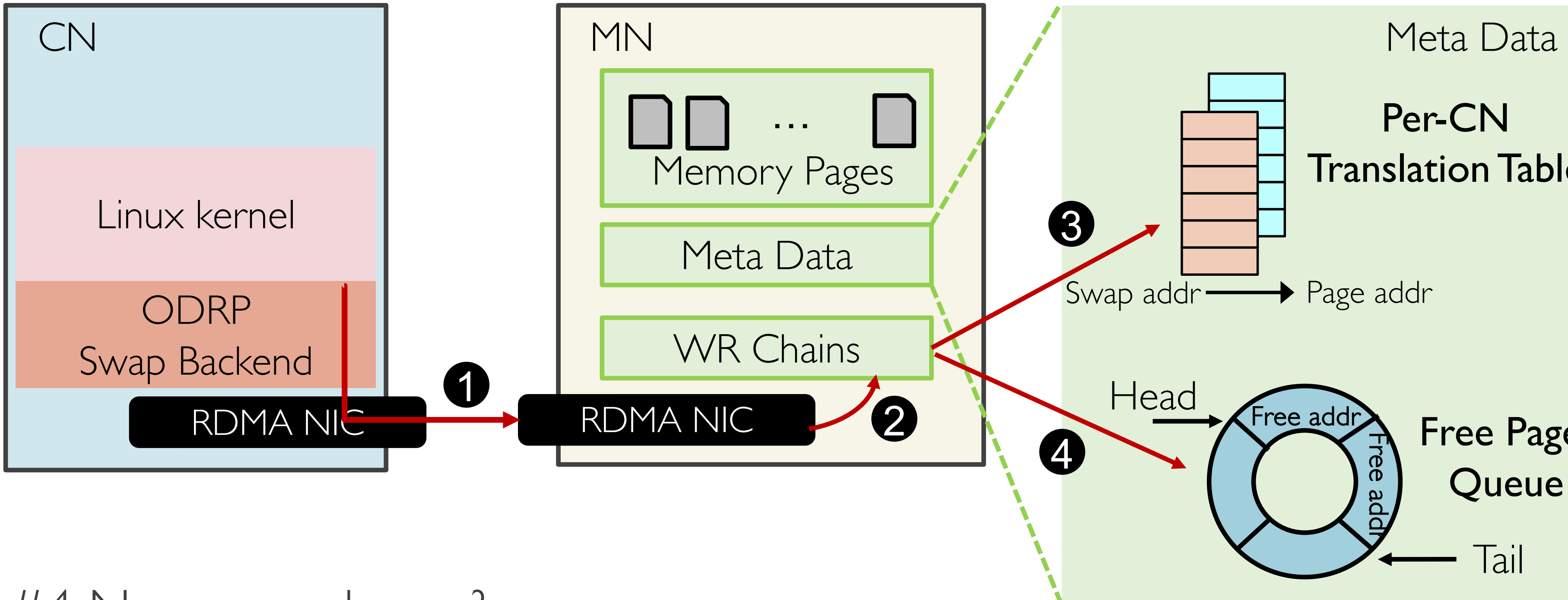
Design – Workflow



#3 The WR chain performs ***address translation***.

i.e. RDMA READ the translation table entry

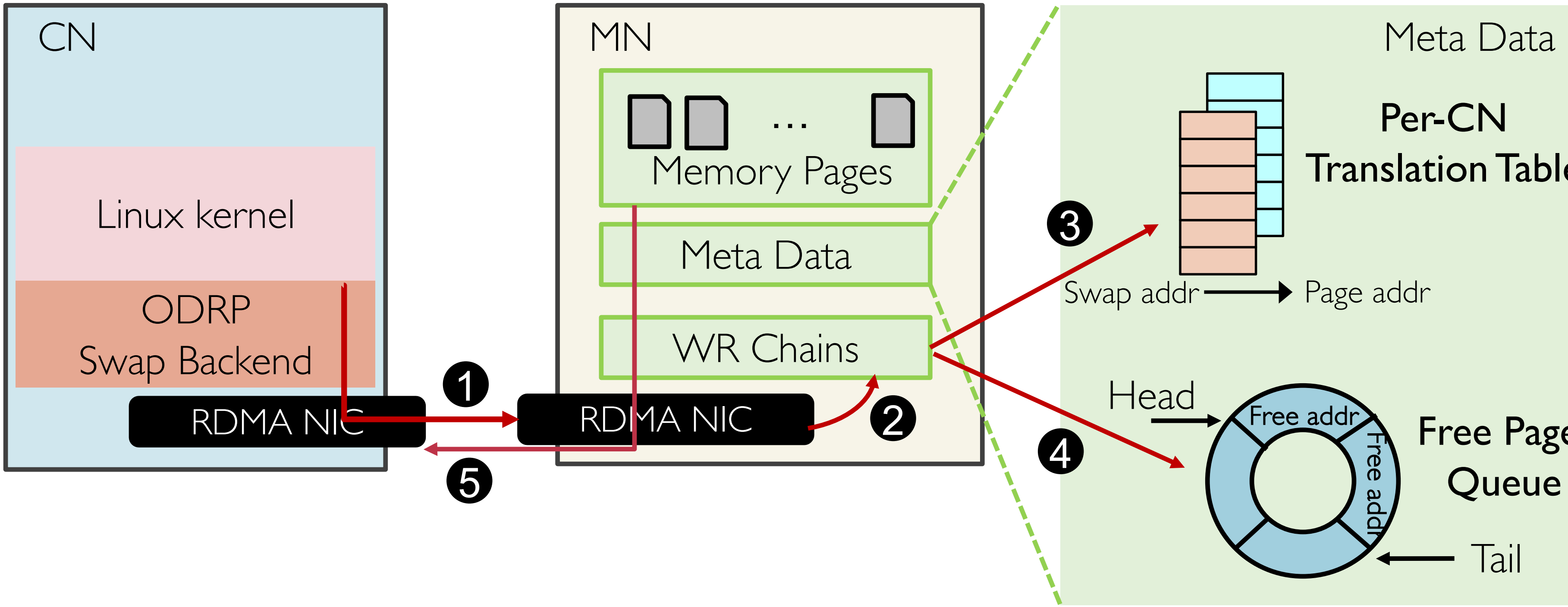
Design – Workflow



#4 No mapped page?

- *allocate* a page from Free Page Queue (i.e., RDMA FAA on Head)
- *update* the CN's translation table (i.e., RDMA WRITE the table entry)

Design – Workflow



#5 The WR chain reads/writes the memory page and returns the result.

Efficiency Challenge

- **C#1: Efficiency**
  - Complex logic requires ***longer WR chain → slower execution***.
  - How to minimize the number of WRs per chain?

Technique: CN-Assisted Principle

- **Shift part of the computation to CNs**
  - Observation #1: the base address of a CN's translation table can be shared with the CN securely.
  - Compute nodes directly provide the translation table entry address.

  - Observation #2: The CN knows whether a swap address is mapped.
  - The CN triggers different WR chains based on whether the swap address is mapped.
  - Avoid complex and time-consuming page fault detection in the WR chain.

Functionality Challenge

- **C#1: Efficiency**
  - Complex logic requires ***longer WR chain → slower execution***.
  - How to minimize the number of WRs per chain?

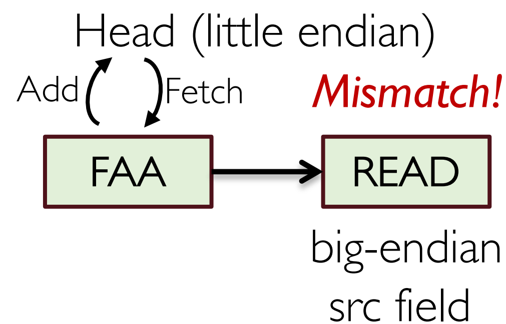
- **C#2: Functionality**

Functionality Challenge

- **Lack of modulo support**
  - Ring buffer necessitate modulo operation (%).
  - RDMA WRs *lack support for modulo operation* (%).

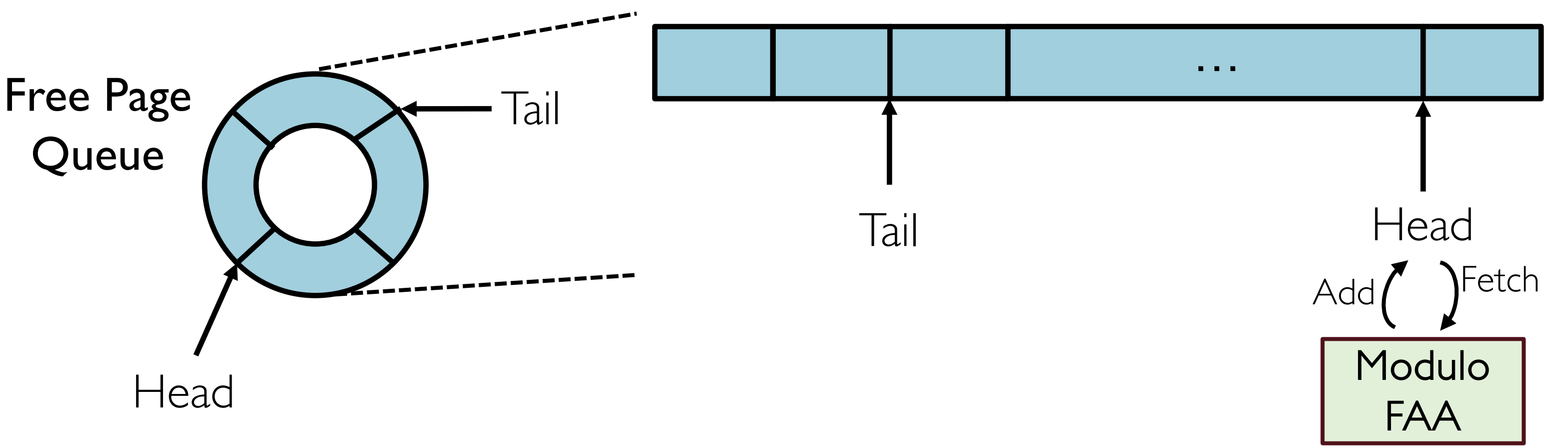
Functionality Challenge

- **Lack of modulo support**
  - Ring buffer necessitate modulo operation (%).
  - RDMA WRs *lack support for modulo operation* (%).
- **Mismatch in endianness**
  - FAA operates on little-endian values.
  - READ assumes a big-endian *src* field.



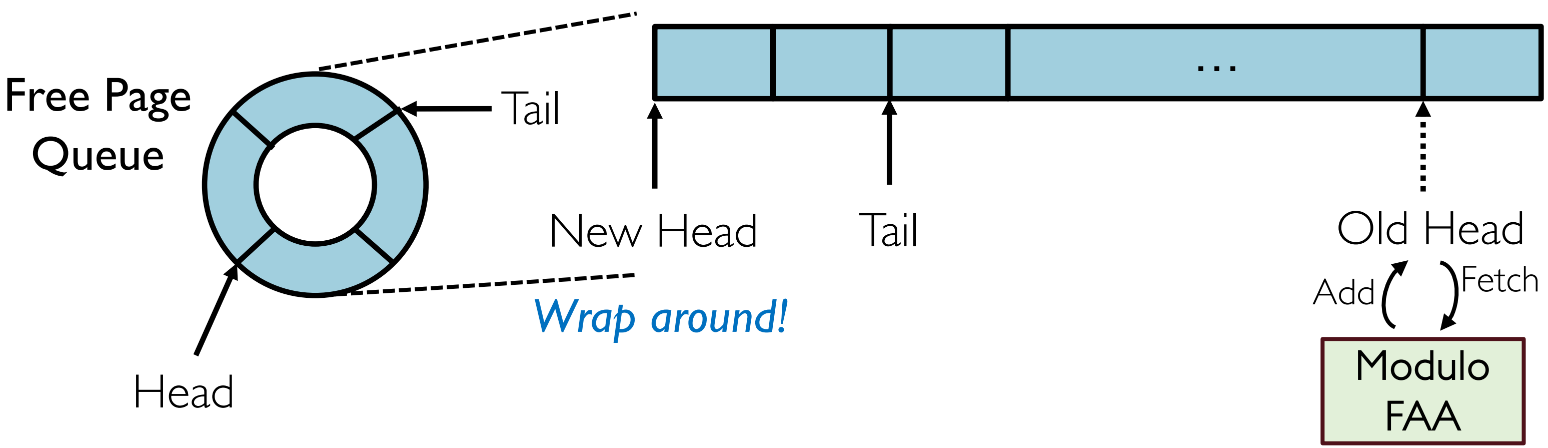
Technique: Meta WR

- **Meta WR #1: Fetch and Add with modulo support**
  - Observation: RNICs support an advanced WR – Masked Fetch and Add.
  - Mask the upper bits of the value to achieve ModuloFAA.



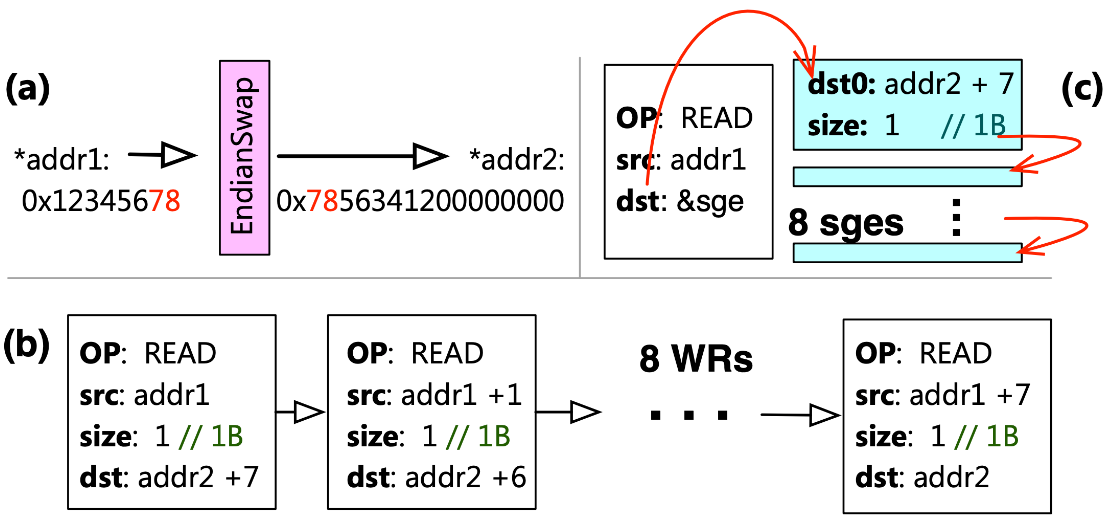
Technique: Meta WR

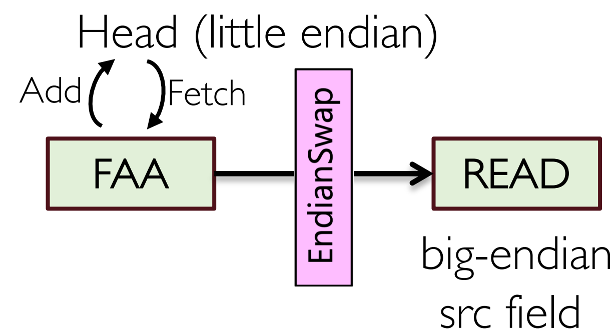
- **Meta WR #1: Fetch and Add with modulo support**
  - Observation: RNICs support an advanced WR – Masked Fetch and Add.
  - Mask the upper bits of the value to achieve ModuloFAA.



Technique: Meta WR

- **Meta WR #2: EndianSwap**
  - **Observation:** RNICs support scatter-gather I/O.
  - One RDMA READ to convert endianness.



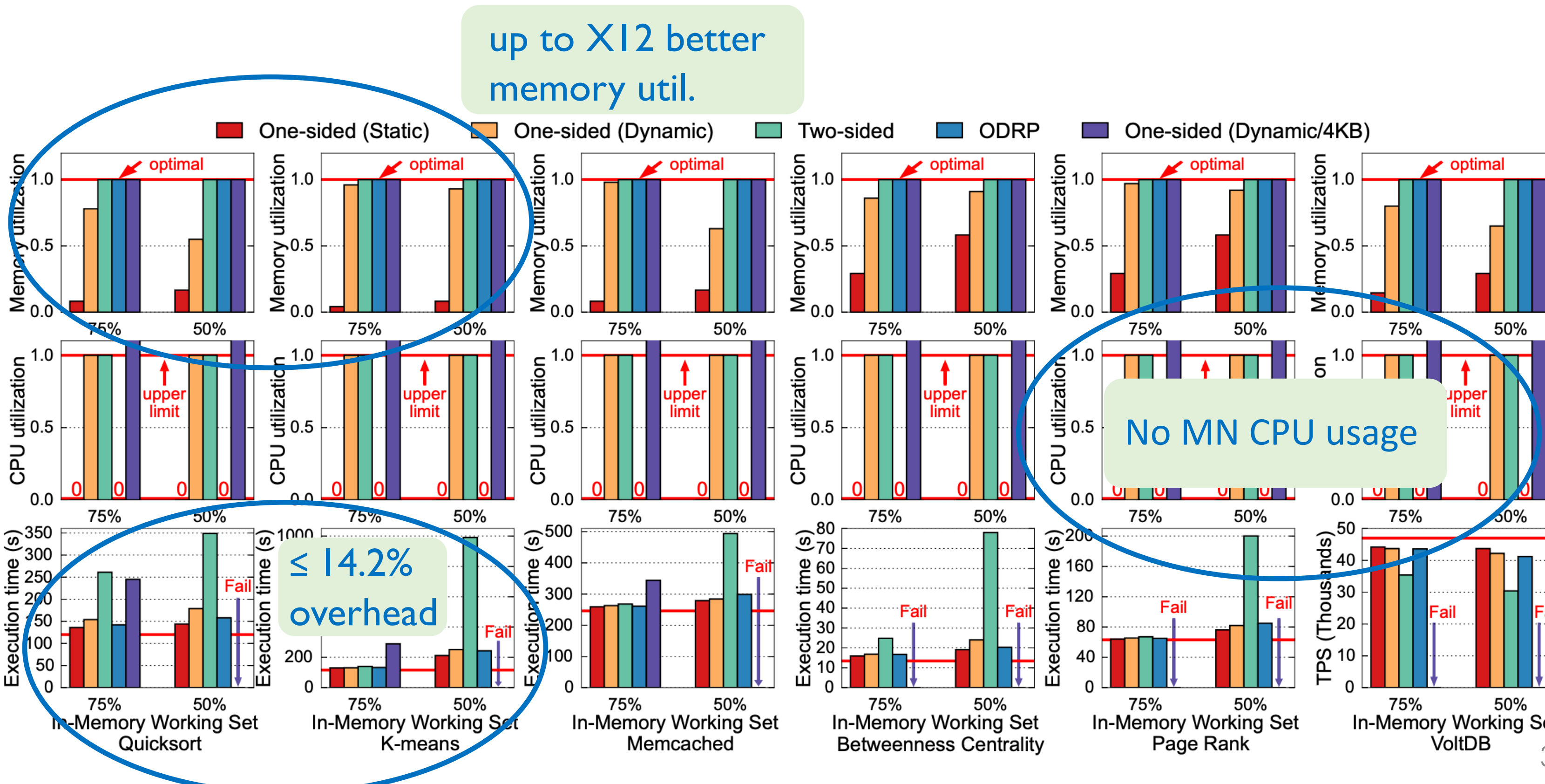


Experimental Setup

- **Harward Setup**
  - **CPU:** 12-Core Intel Xeon E5-2650 CPU
  - **DRAM:** 128 GB DDR4 RAM
  - **RNIC:** 100 Gbps Mellanox ConnectX-5 RNIC
  - **Cluster:** 8 CNs (12 GB swap space), 1 MN (only use one CPU core)
- **Other baselines**
  - **One-sided(Static):** pre-registering MR with the size of the CN's swap space
  - **One-sided(Dynamic):** registering and allocating 1MB MR on demand
  - **Two-sided:** using RPC in the data path

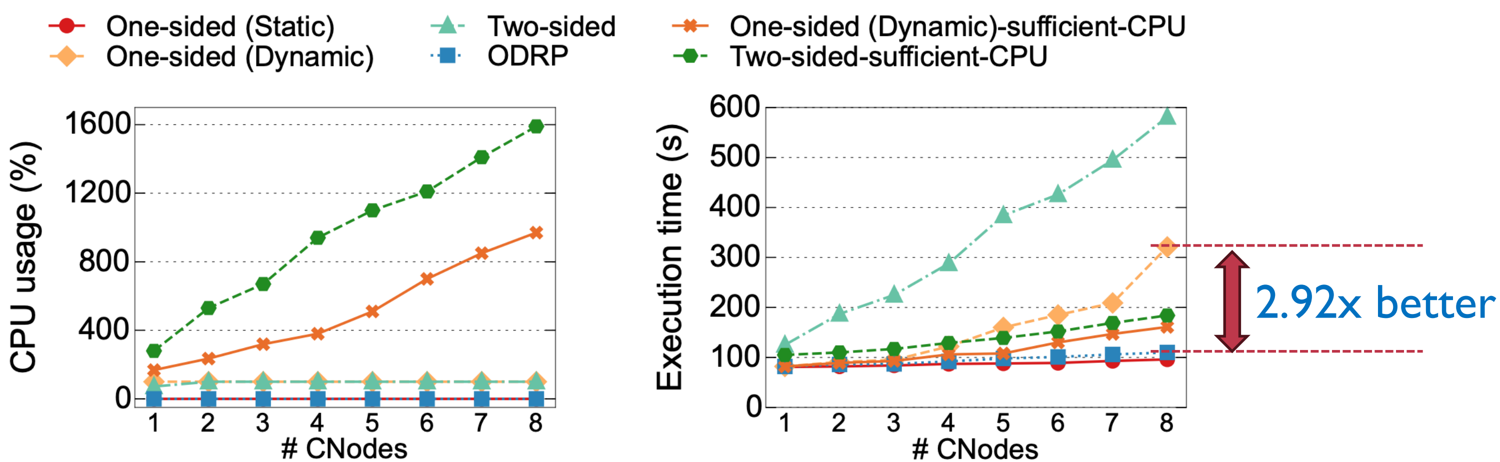
Application Benchmark

- Run real-world workloads on 8 CNs.



Scalability

- Can ODRP scale as the number of CNs increases?



ODRP can prevent the weak MN CPU from becoming a bottleneck.

- less than 14.6% performance overhead compared to One-sided(Static)
- 2.92x better performance than one-sided(Dynamic)

Conclusion

- Current DM systems cannot achieve high memory utilization, zero MN CPU usage, and high performance at the same time.
- We propose ODRP, the first system that leverages RNIC offloading to achieve
  - ideal memory utilization
  - zero MN CPU usage (i.e., true disaggregation)
  - high performance
- We introduce two software techniques to address the efficiency and functionality challenges of RNIC offloading.
- ODRP significantly improves memory utilization with less than 14.6% performance overhead in real-world workloads.

Thanks!

Q&A