PID-Comm:

Fast and Flexible Collective Communication Framework for Commodity Processing-in-DIMM Devices [ISCA '24]

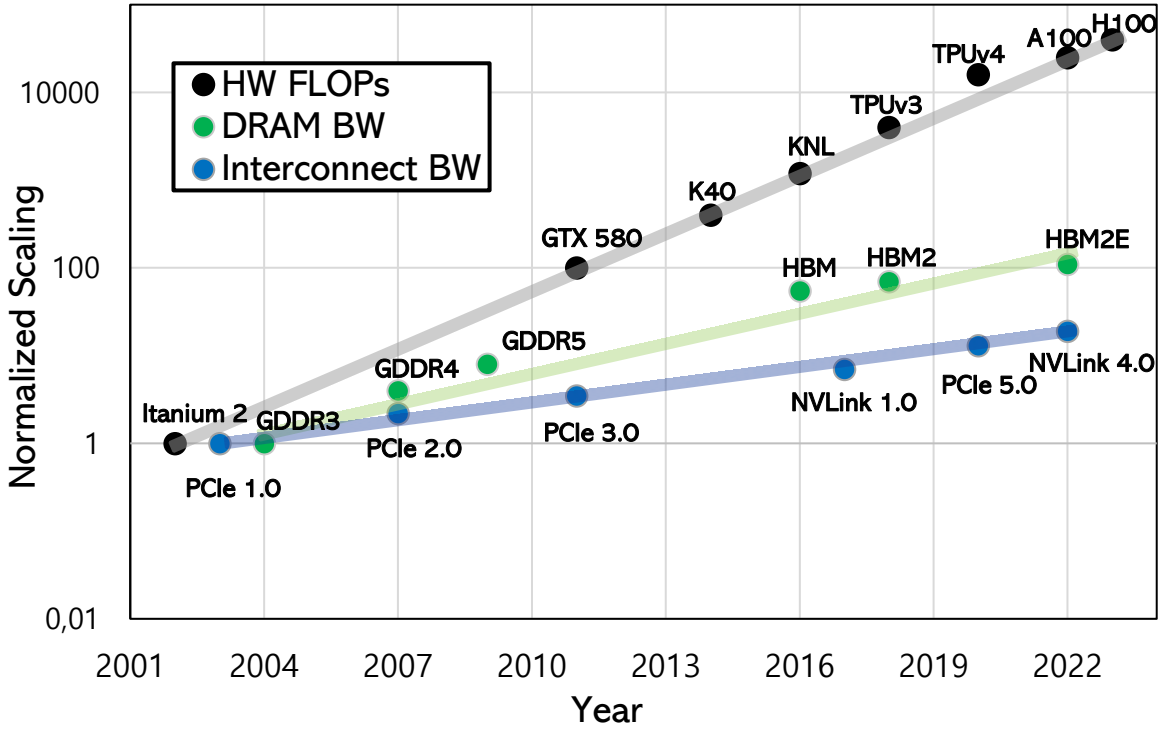
Si Ung Noh[1], **Junguk Hong**[1], Chaemin Lim[2], Seongyeon Park[1],
Jeehyun Kim[2], Hanjun Kim[3], Youngsok Kim[2] and Jinho Lee[1]

[1]: Department of Electrical and Computer Engineering, Seoul National University
[2]: Department of Computer Science, Yonsei University
[3]: School of Electrical and Electronic Engineering, Yonsei University

*The 2nd Minisymposium on Application and Benefits of UPMEM commercial Massively Parallel Processing-In-Memory Platform (**ABUMPIMP**)*

The Memory Wall

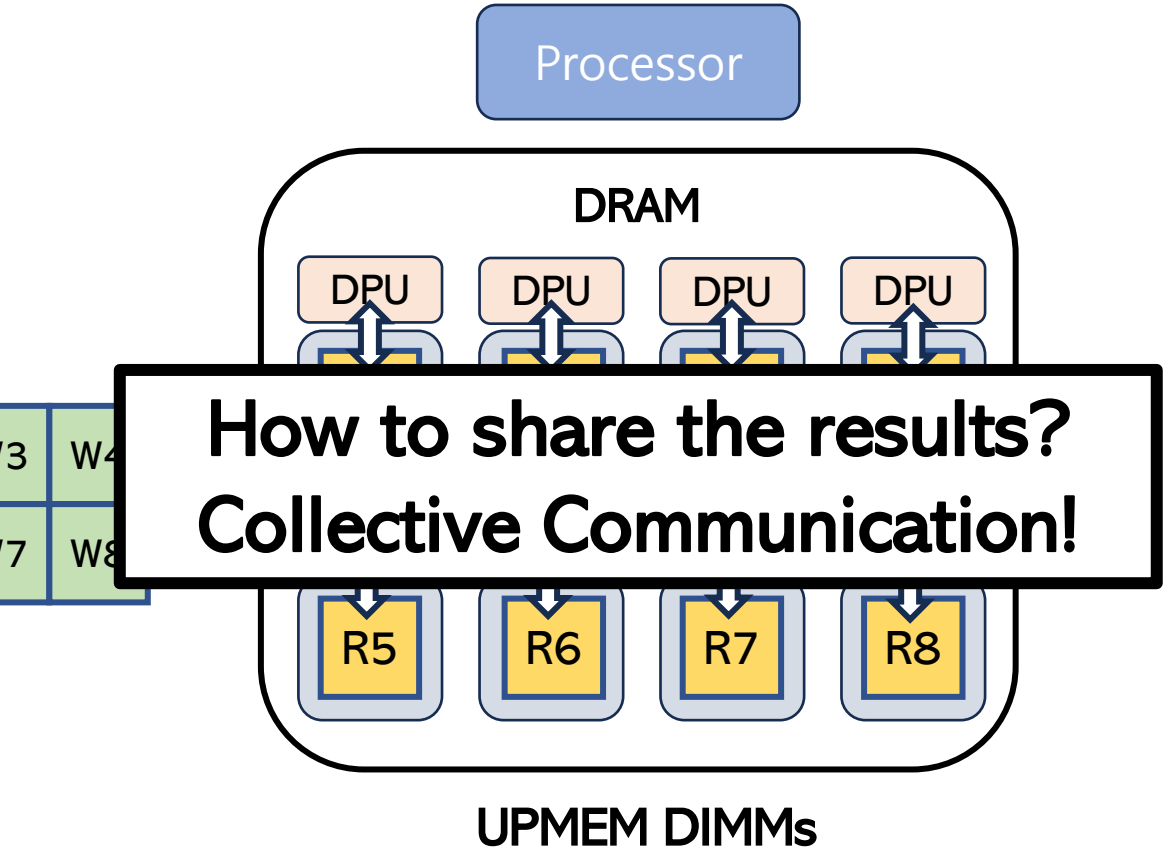
- Processor performance is outpacing memory interconnect bandwidth
- AI applications require high memory bandwidth
- Memory is becoming the dominant bottleneck



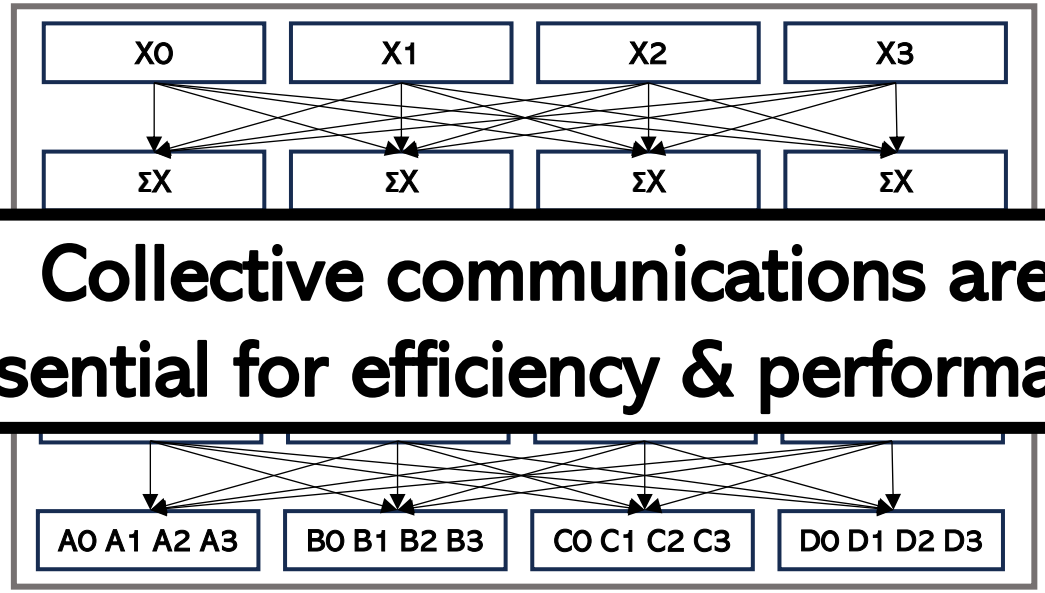
[+]Source: Amir Gholami, Zhewei Yao, Sehoon Kim, Coleman Hooper, Michael W. Mahoney, Kurt Keutzer. 2024 AI and Memory Wall. In IEEE Micro

Inter-DPU Communications

- Workload is distributed to each DPU (nodes) for computation
- For efficient sharing of intermediate results, collective communications are essential



How to share the results?
Collective Communication!



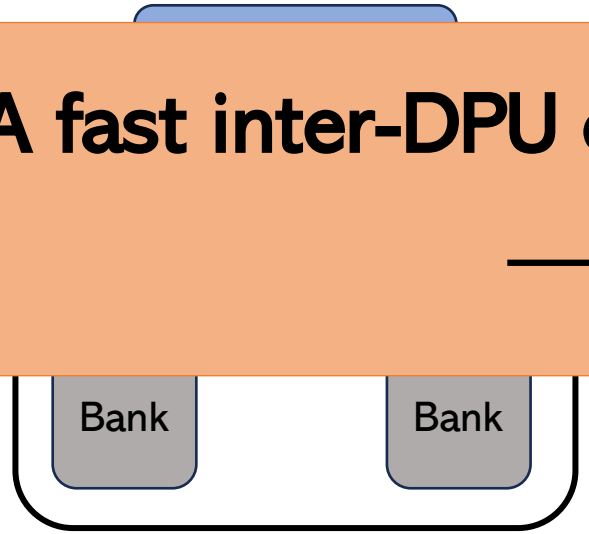
Collective communications are essential for efficiency & performance

Inter-DPU Communications

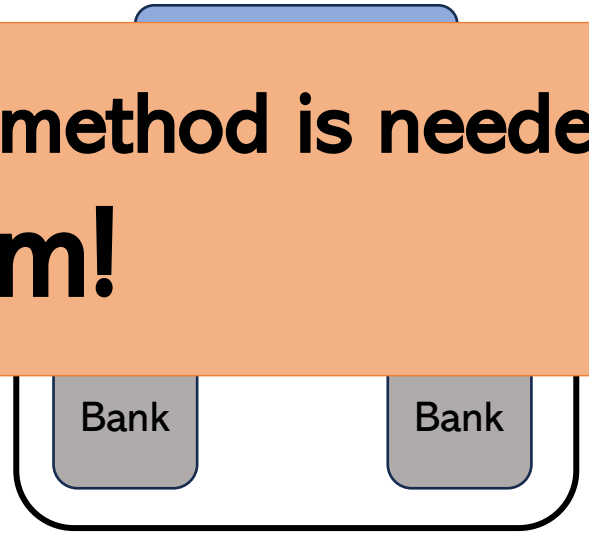
- No direct path between each DRAM processing elements (DPUs)
- Host processor becomes the medium for inter-DPU communication
- Inter-DPU communications are becoming the bottleneck of major applications

A fast inter-DPU communication method is needed

→ PID-Comm!



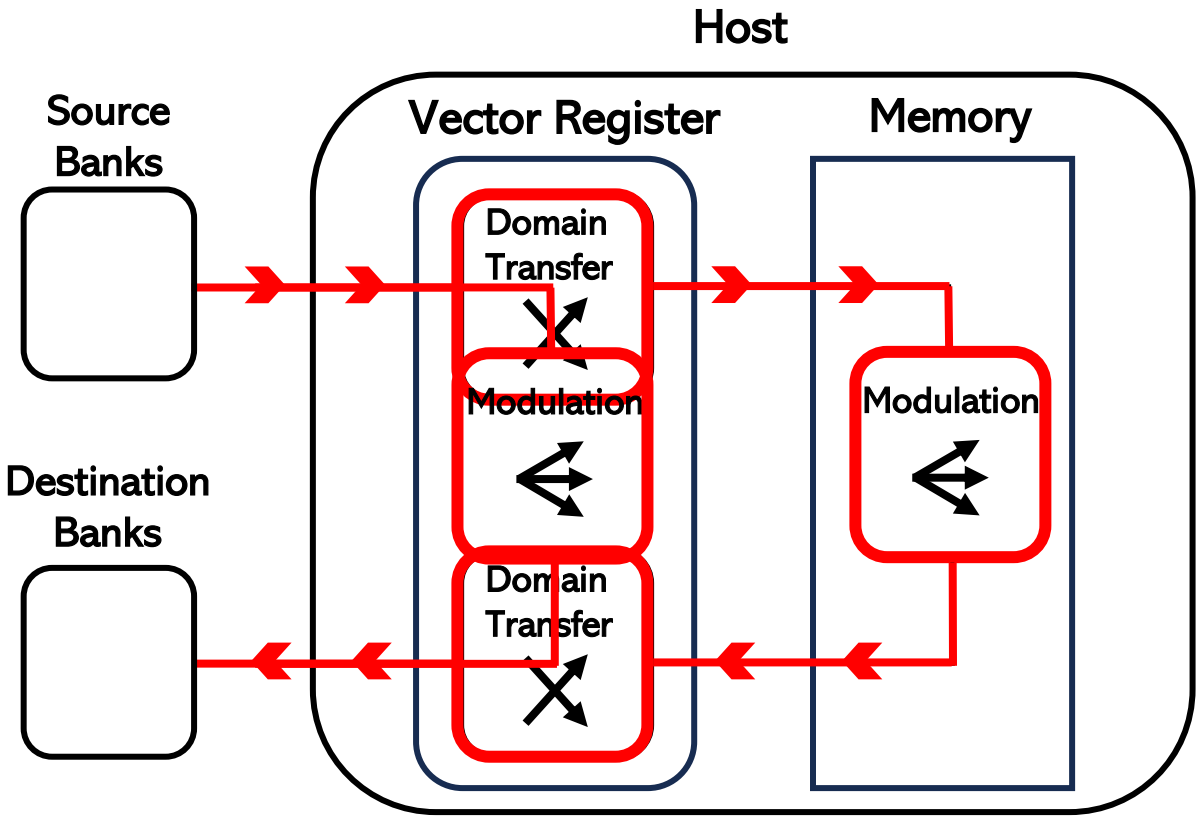
Lack of a direct path between DPUs



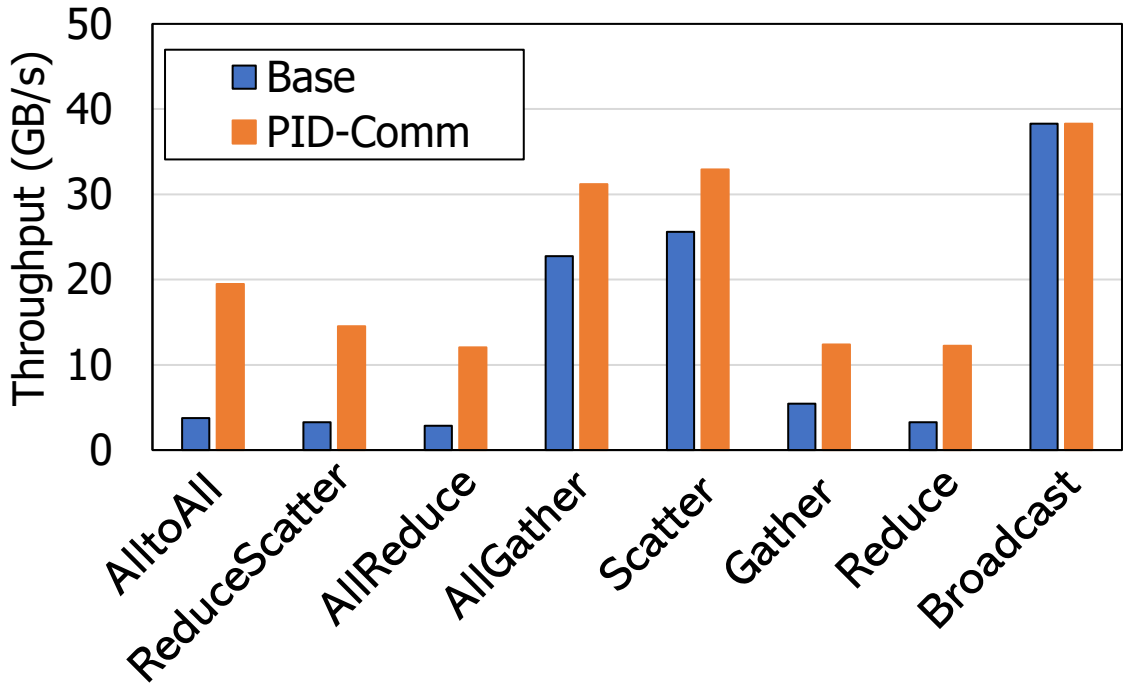
Existing path passes the host processor

PID-Comm

- Optimize inter-DPU collective communications
- Geomean speedup of 2.83x for the new collective primitives

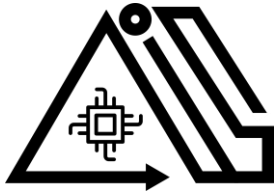


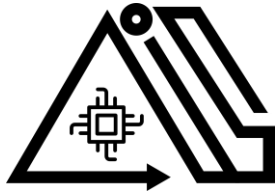
Optimization of inter-DPU communication



Speedup of PID-Comm collective primitives

Contents



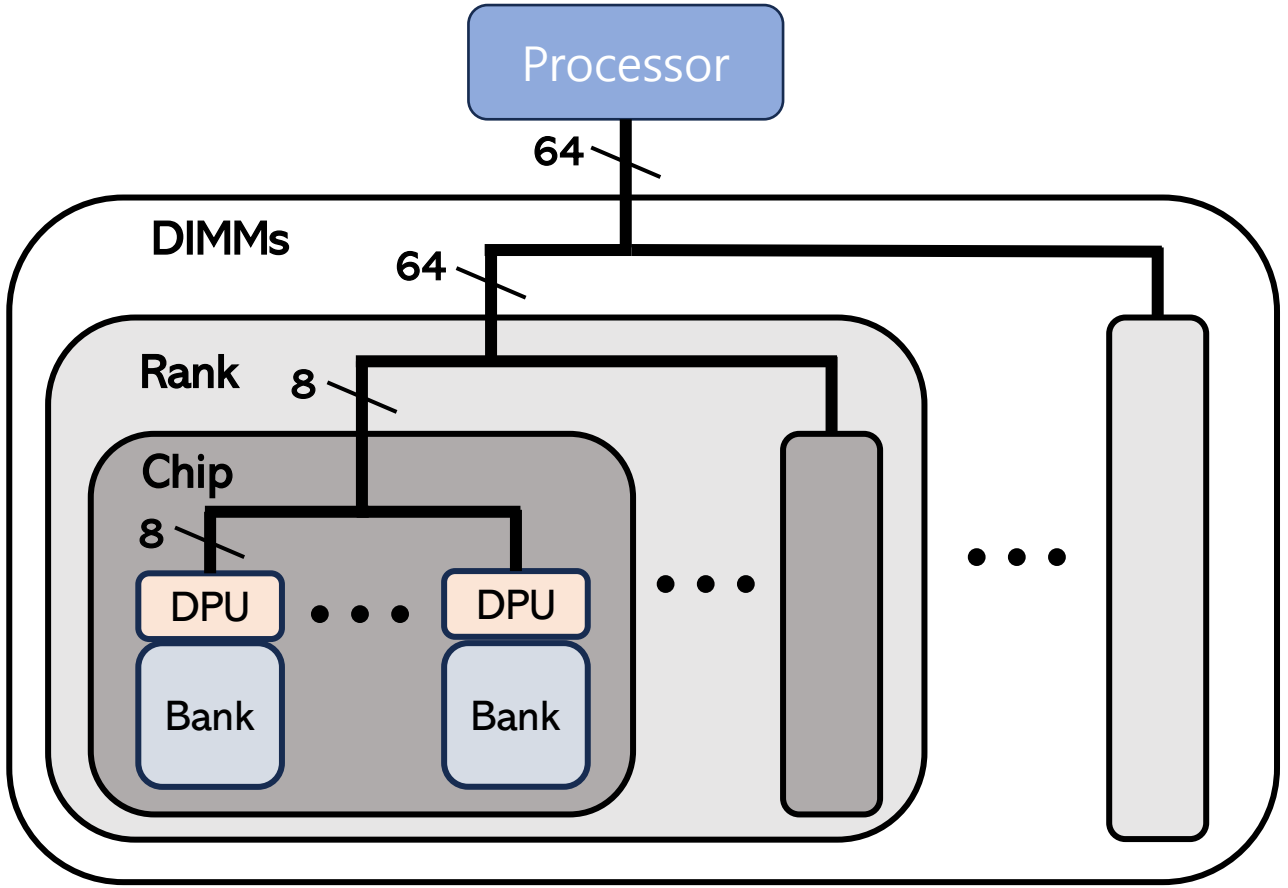


2. Background

- PIM-enabled DIMMs
- Entangled Groups and Domain Transfer
- Conventional inter-DPU communication

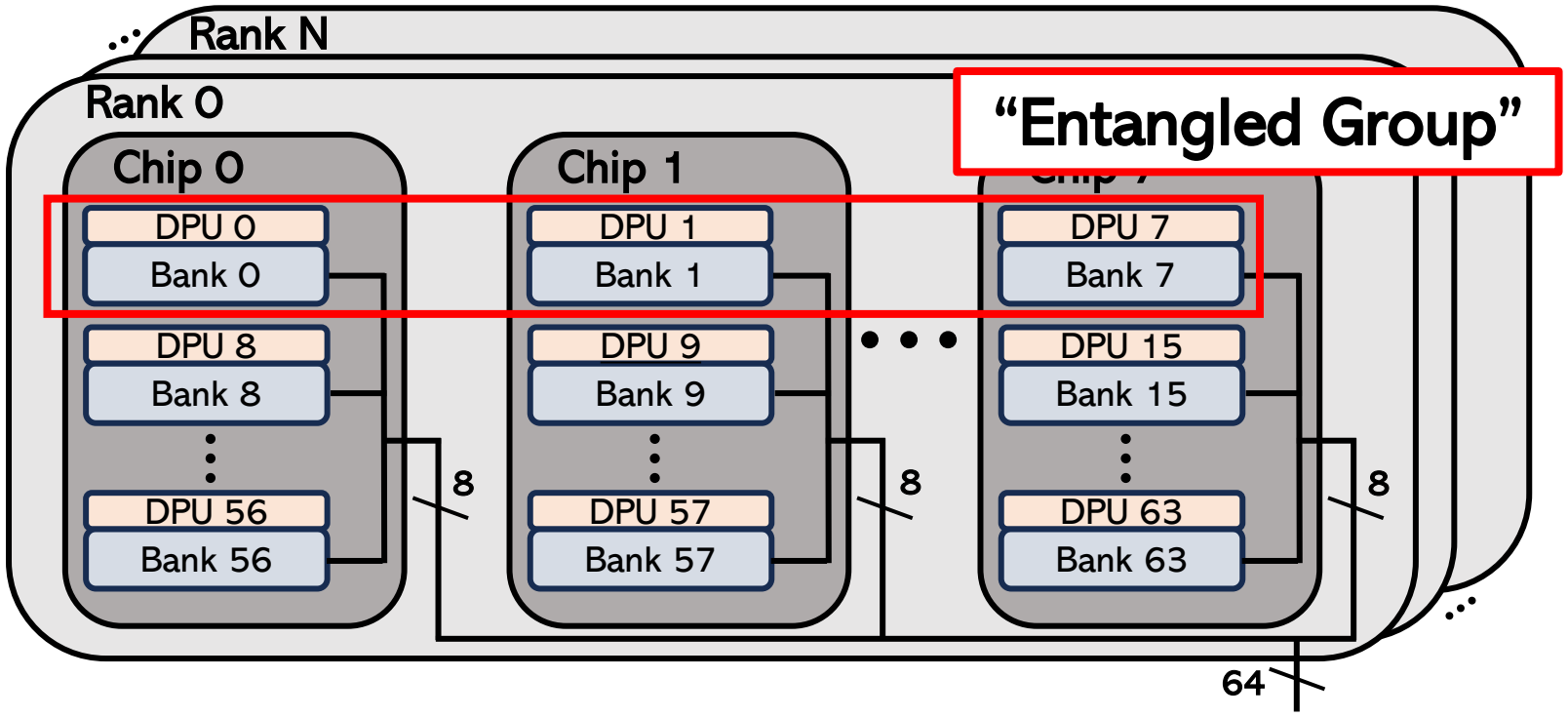
PIM-enabled DIMMs

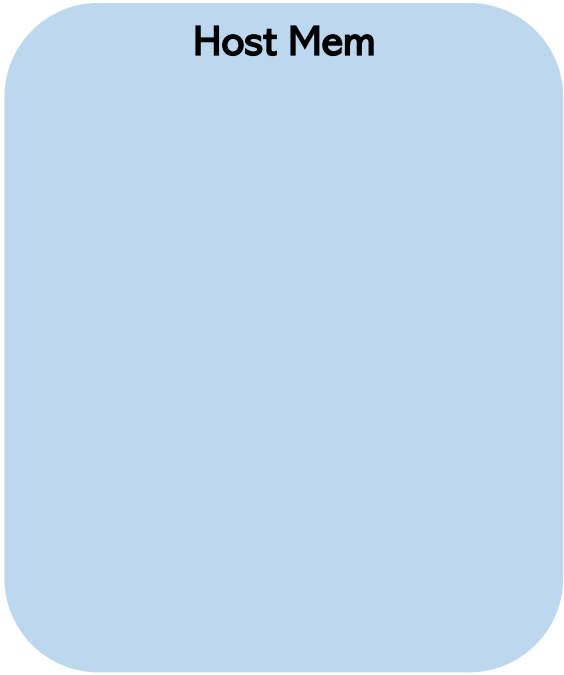
- Refers to PIM implemented on dual in-line memory modules (UPMEM DIMMs)
- Shares the same hierarchy as DDR4 DRAMs



PIM-enabled DIMMs

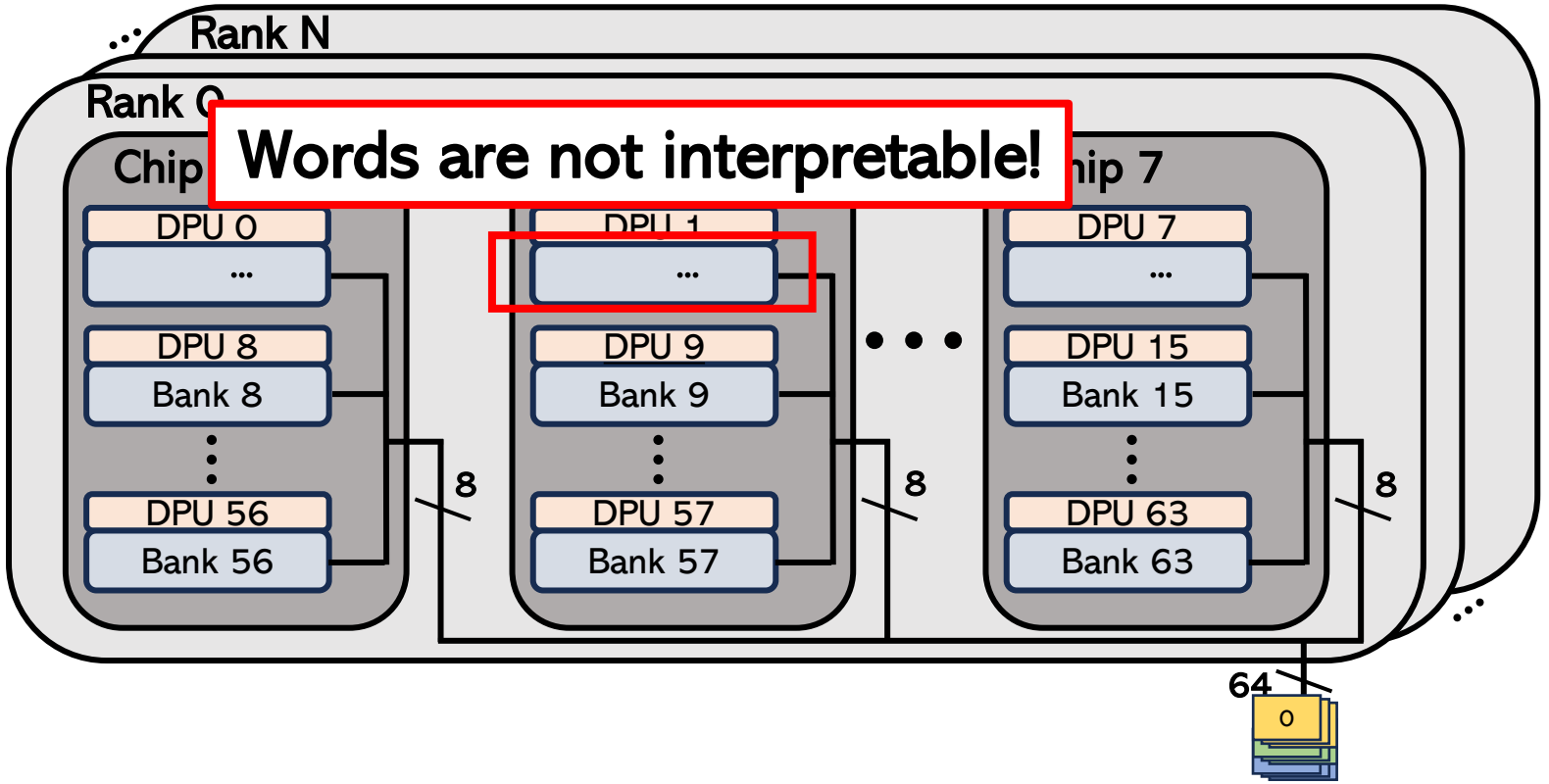
- 8-bit bus per chip to form 64-bit channel bus
- The same bank of each chip in a rank are accessed at once
- We name the group of banks accessed together as “Entangled Group”

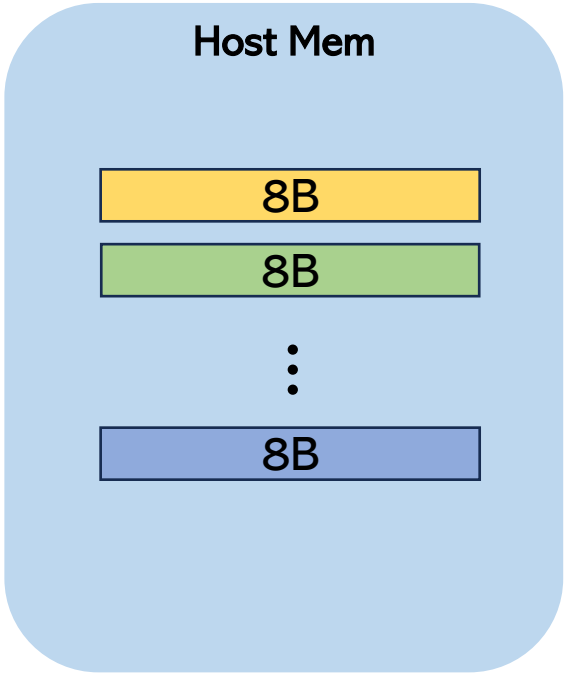




PIM-enabled DIMMs

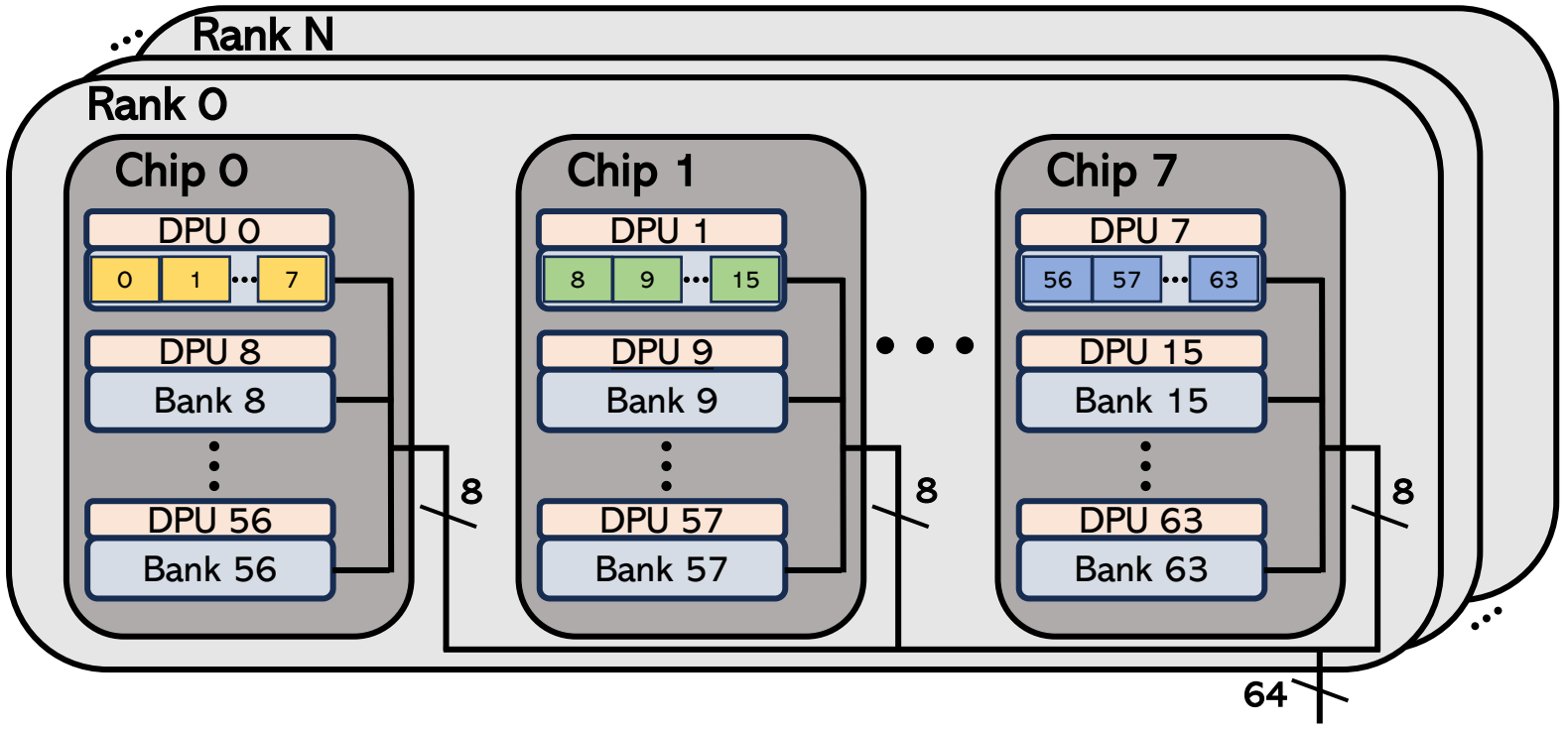
- 8-bit bus per chip to form 64-bit channel bus
- The same bank of each chip in a rank are accessed at once
- We name the group of banks accessed together as “Entangled Group”

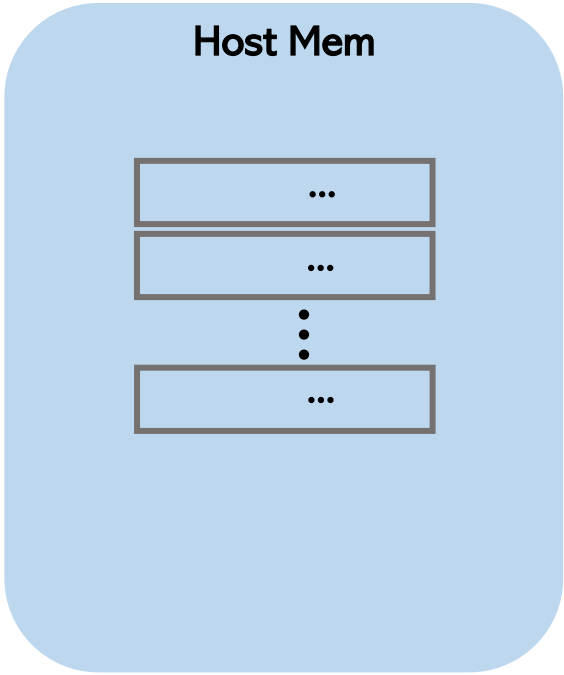




PIM-enabled DIMMs

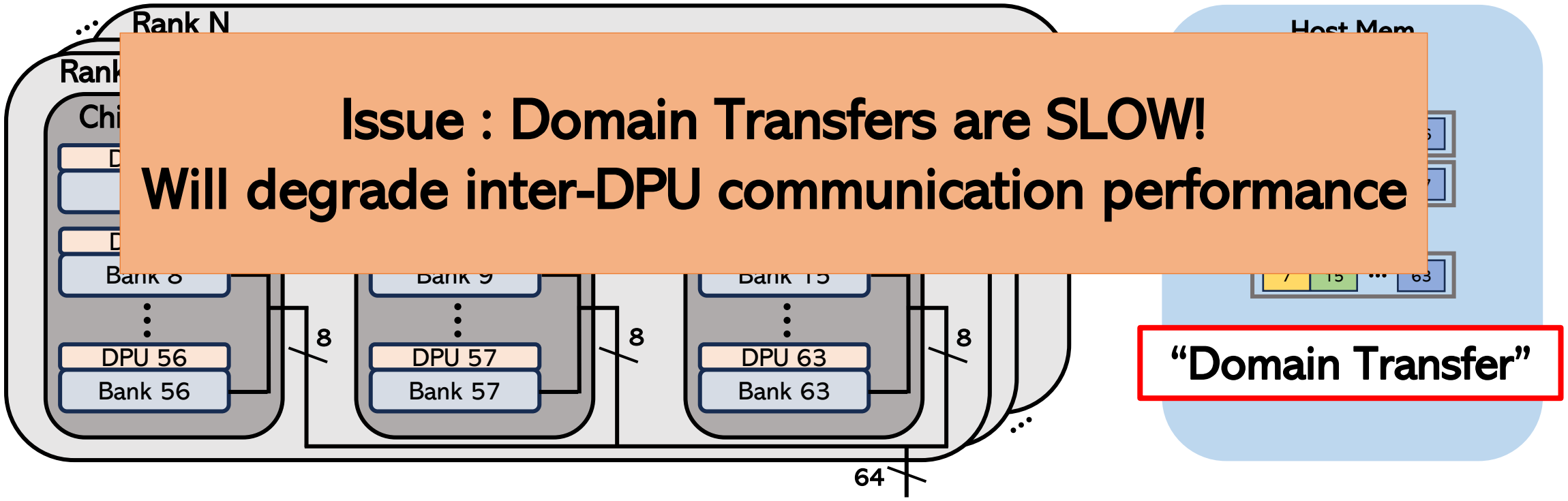
- 8-bit bus per chip to form 64-bit channel bus
- The same bank of each chip in a rank are accessed at once
- We name the group of banks accessed together as “Entangled Group”





PIM-enabled DIMMs

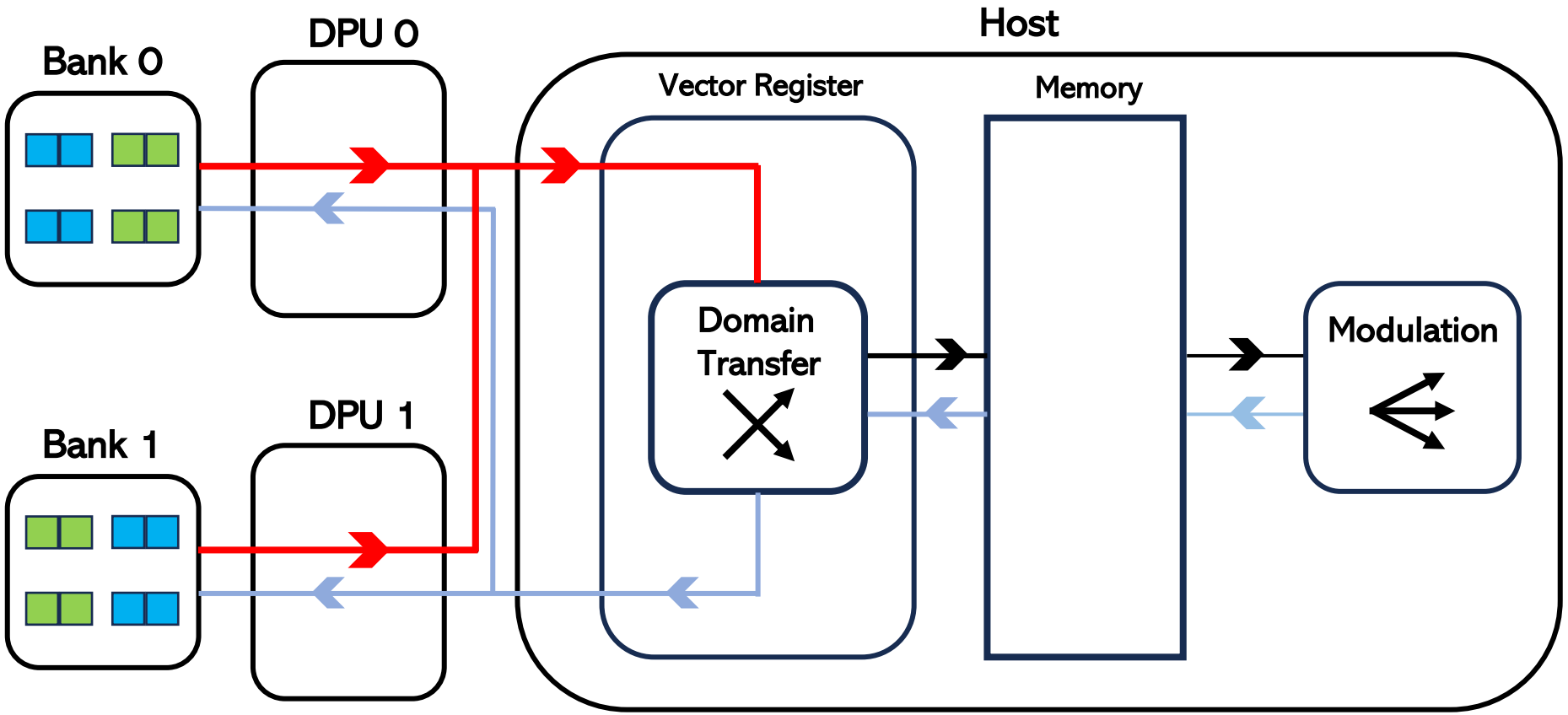
- 8-bit bus per chip to form 64-bit channel bus
- The same bank of each chip in a rank are accessed at once
- We name the group of banks accessed together as “Entangled Group”



Issue : Domain Transfers are SLOW!
Will degrade inter-DPU communication performance

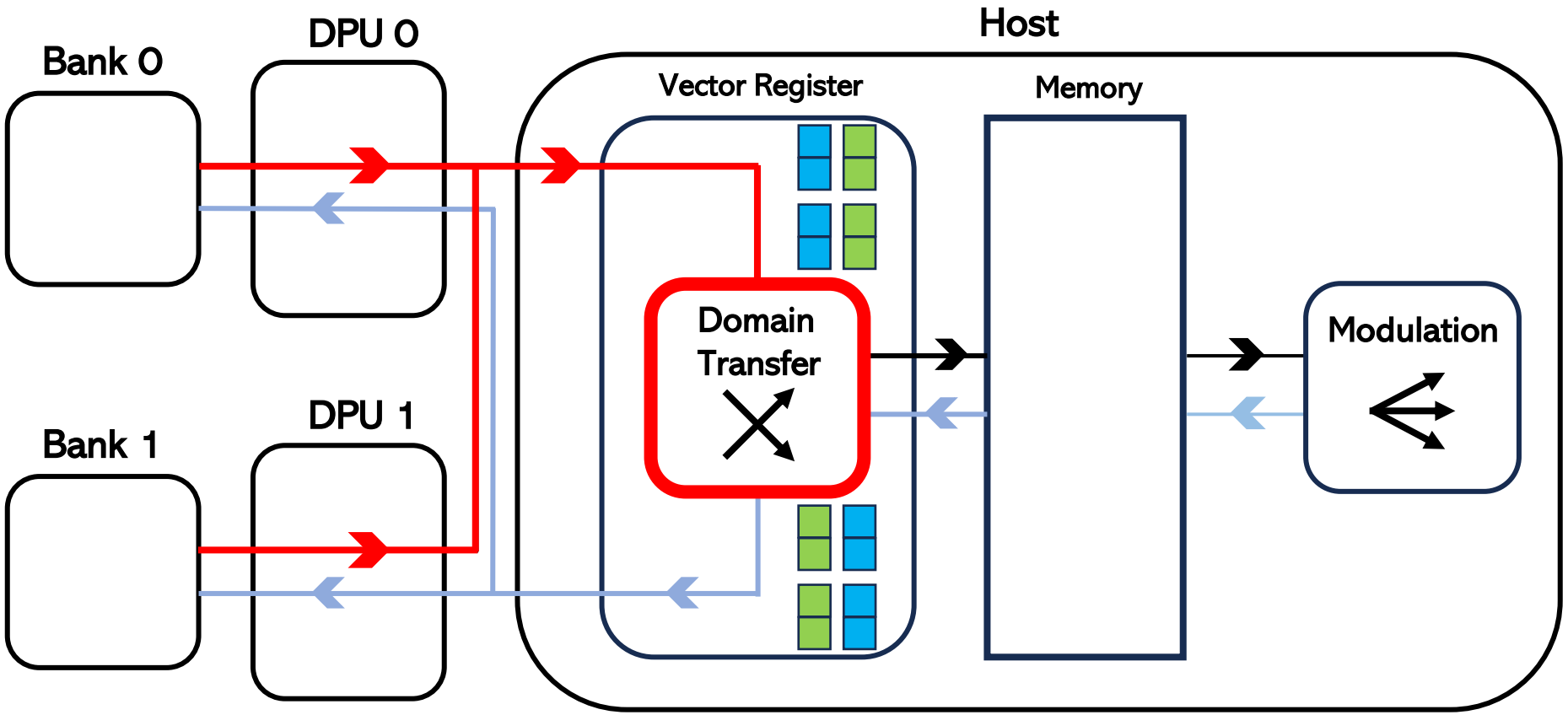
Conventional Inter-DPU Communication

- Data domain-transferred in the vector register and saved in host memory
- Data sent back to bank after modulation in the host processor



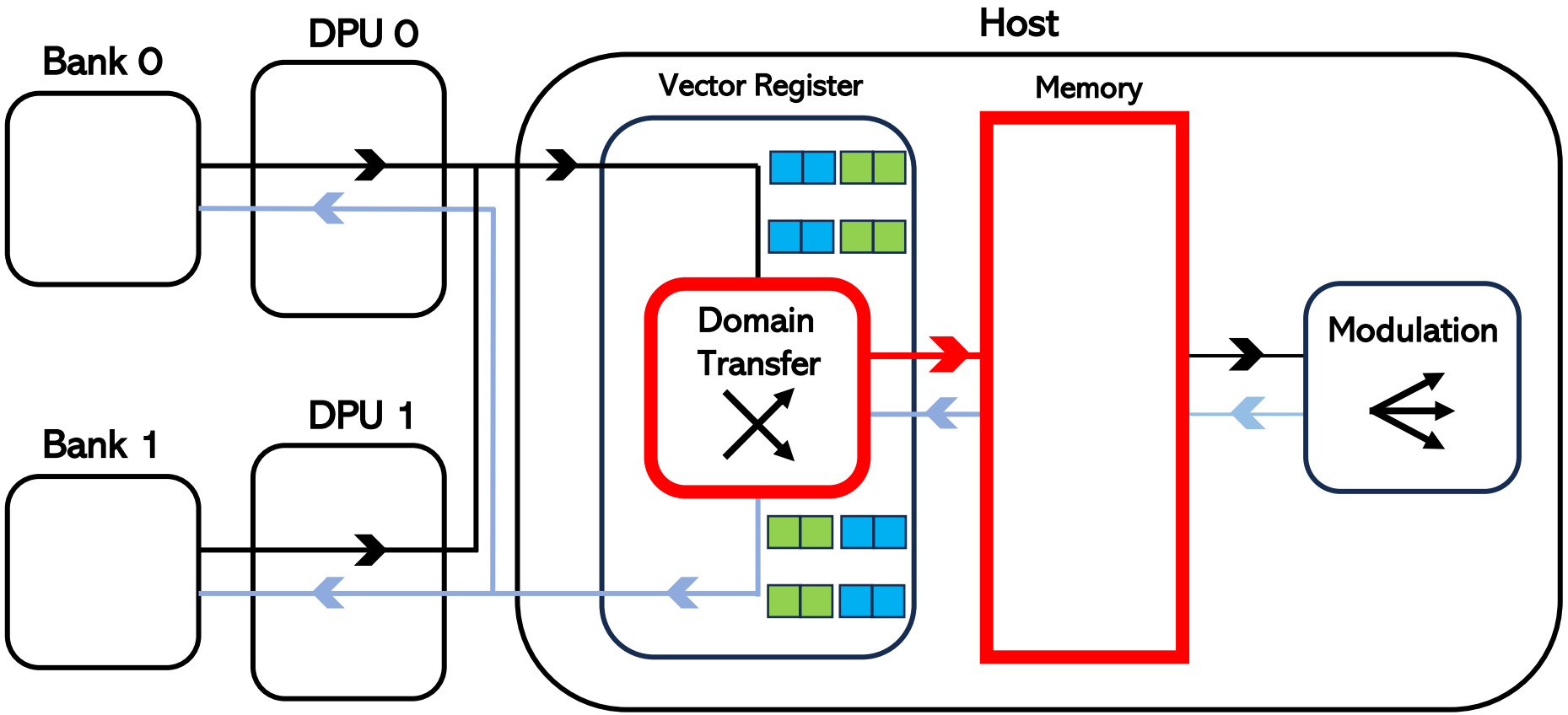
Conventional Inter-DPU Communication

- Data domain-transferred in the vector register and saved in host memory
- Data sent back to bank after modulation in the host processor



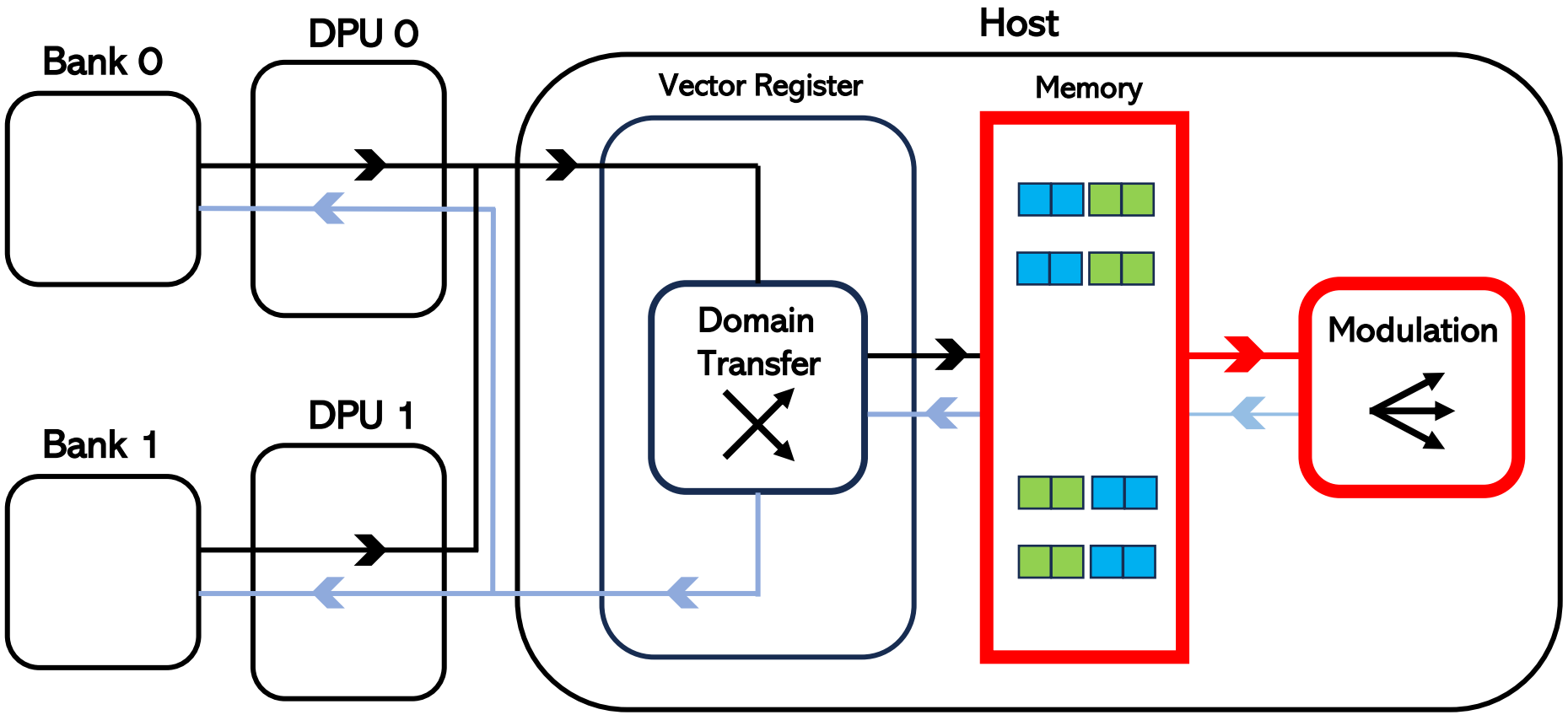
Conventional Inter-DPU Communication

- Data domain-transferred in the vector register and saved in host memory
- Data sent back to bank after modulation in the host processor



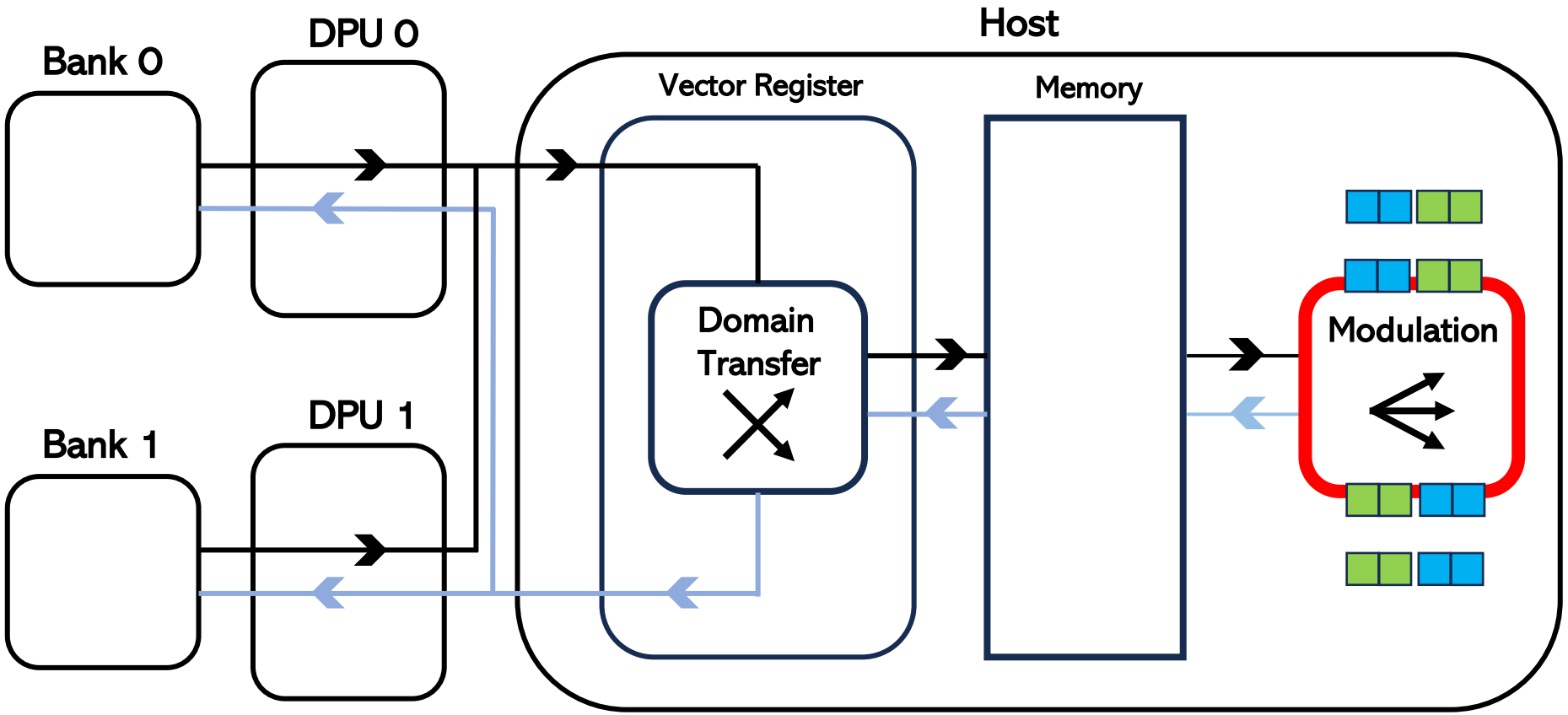
Conventional Inter-DPU Communication

- Data domain-transferred in the vector register and saved in host memory
- Data sent back to bank after modulation in the host processor



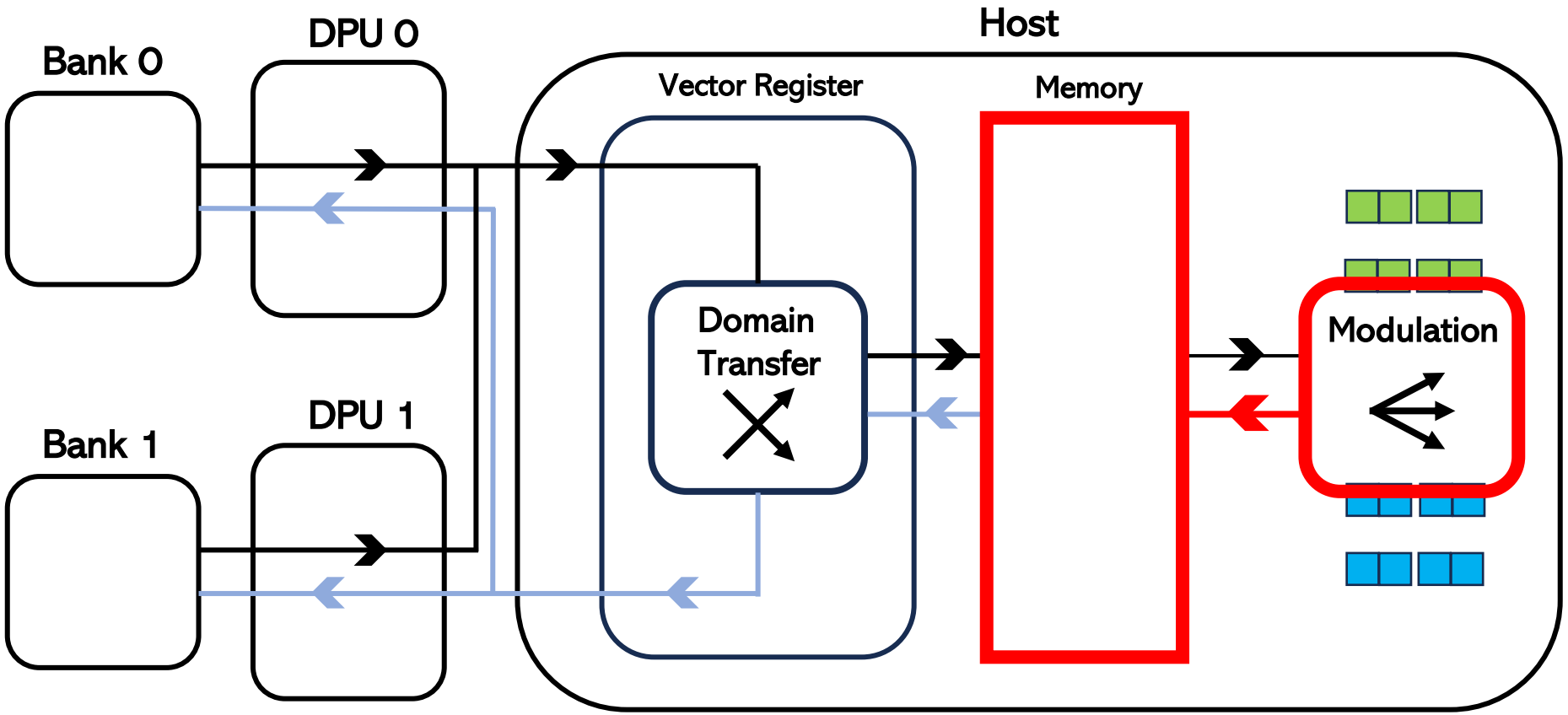
Conventional Inter-DPU Communication

- Data domain-transferred in the vector register and saved in host memory
- Data sent back to bank after modulation in the host processor



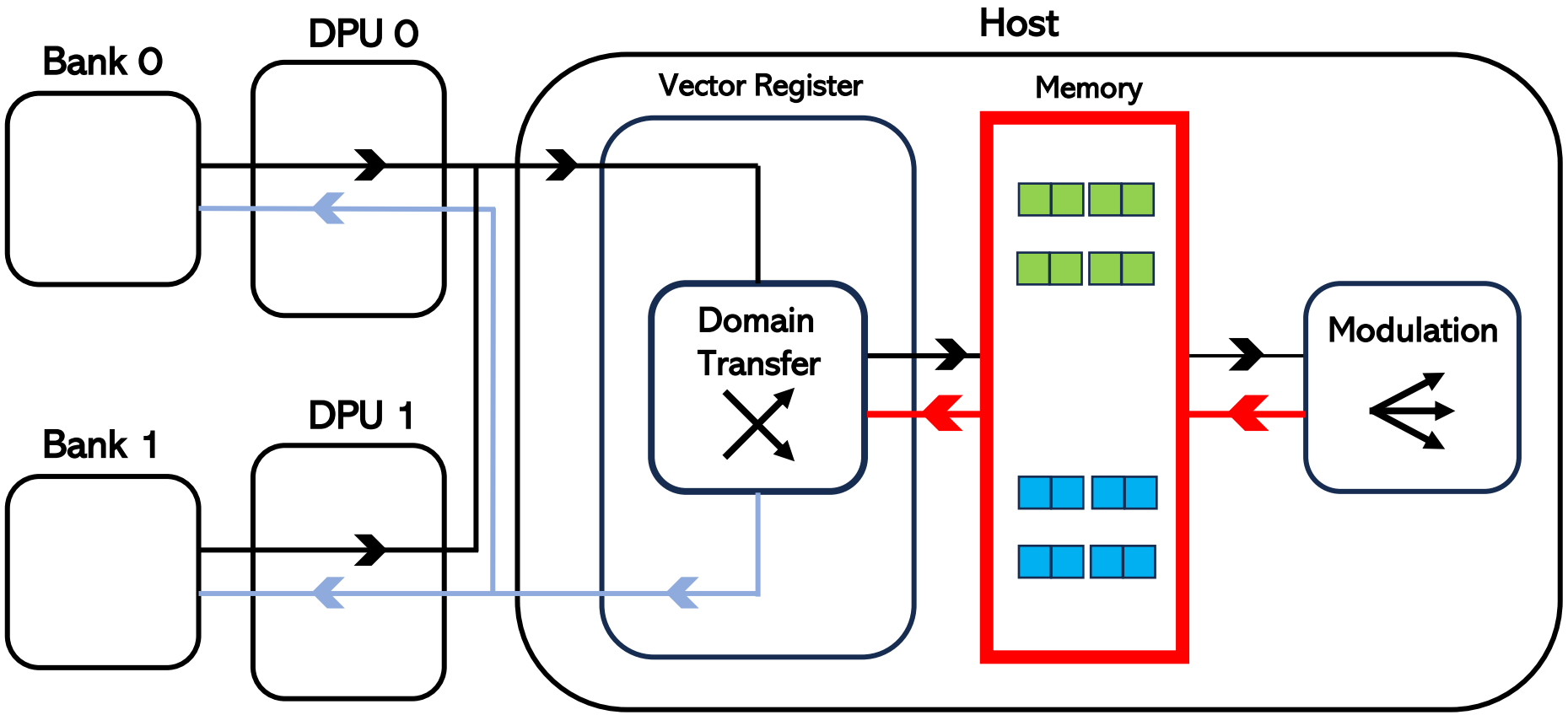
Conventional Inter-DPU Communication

- Data domain-transferred in the vector register and saved in host memory
- Data sent back to bank after modulation in the host processor



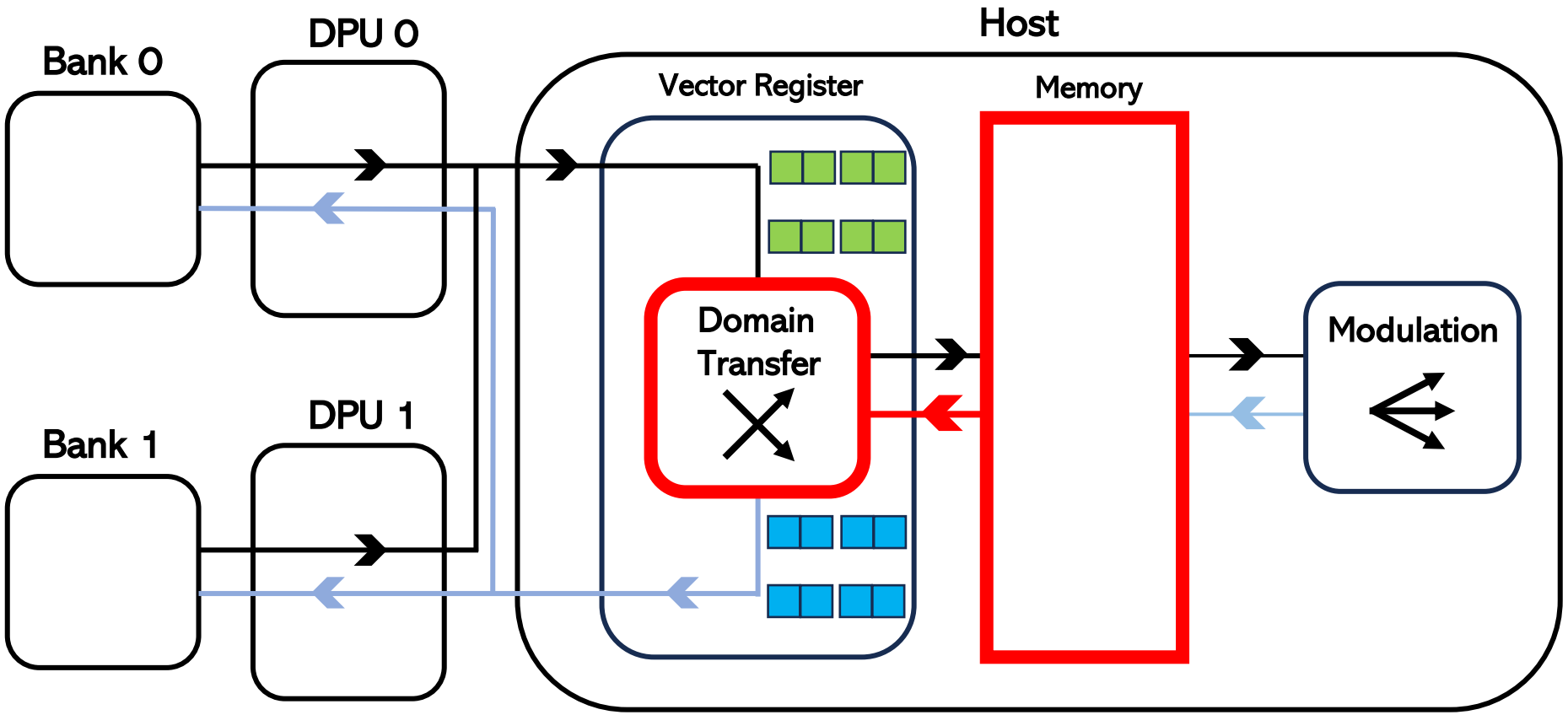
Conventional Inter-DPU Communication

- Data domain-transferred in the vector register and saved in host memory
- Data sent back to bank after modulation in the host processor



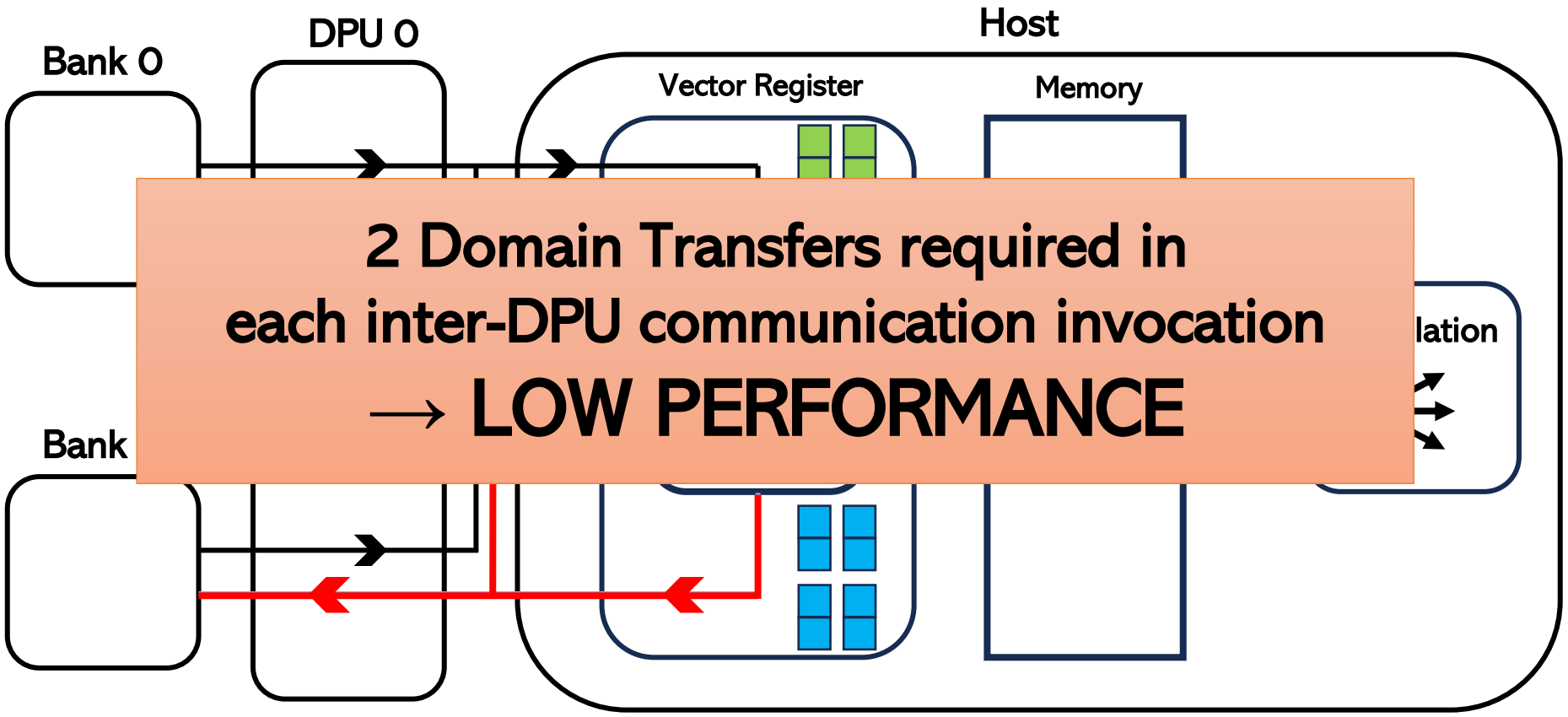
Conventional Inter-DPU Communication

- Data domain-transferred in the vector register and saved in host memory
- Data sent back to bank after modulation in the host processor



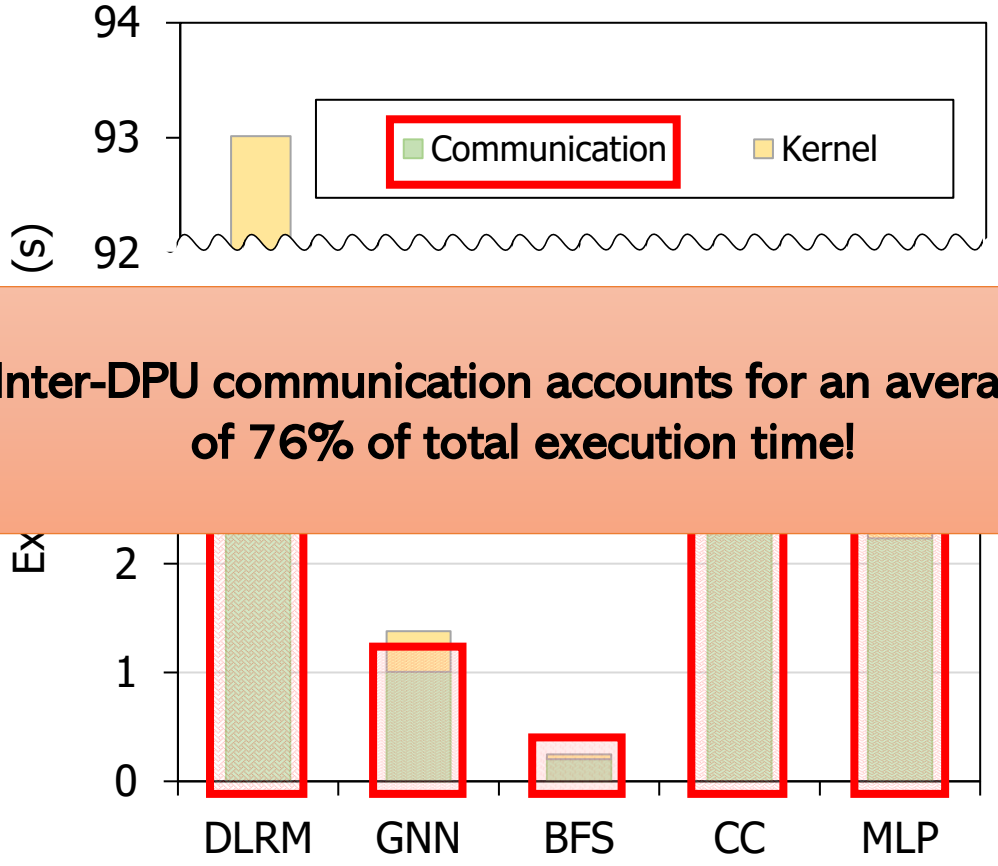
Conventional Inter-DPU Communication

- Data domain-transferred in the vector register and saved in host memory
- Data sent back to bank after modulation in the host processor



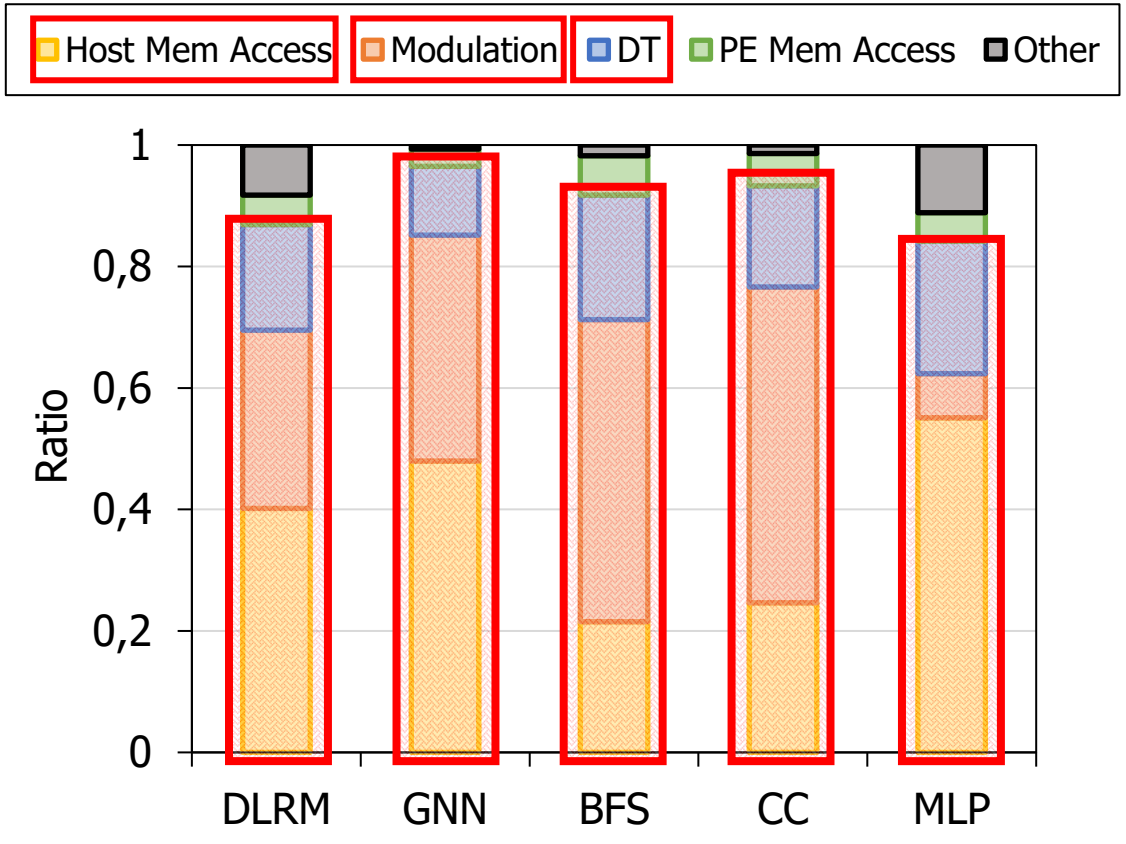
Performance of Naïve inter-DPU communications

- Breakdown of 5 benchmark applications (DLRM, GNN, BFS, CC, MLP)
- Communication takes up 76% of total execution time



Inter-DPU communication accounts for an average of 76% of total execution time!

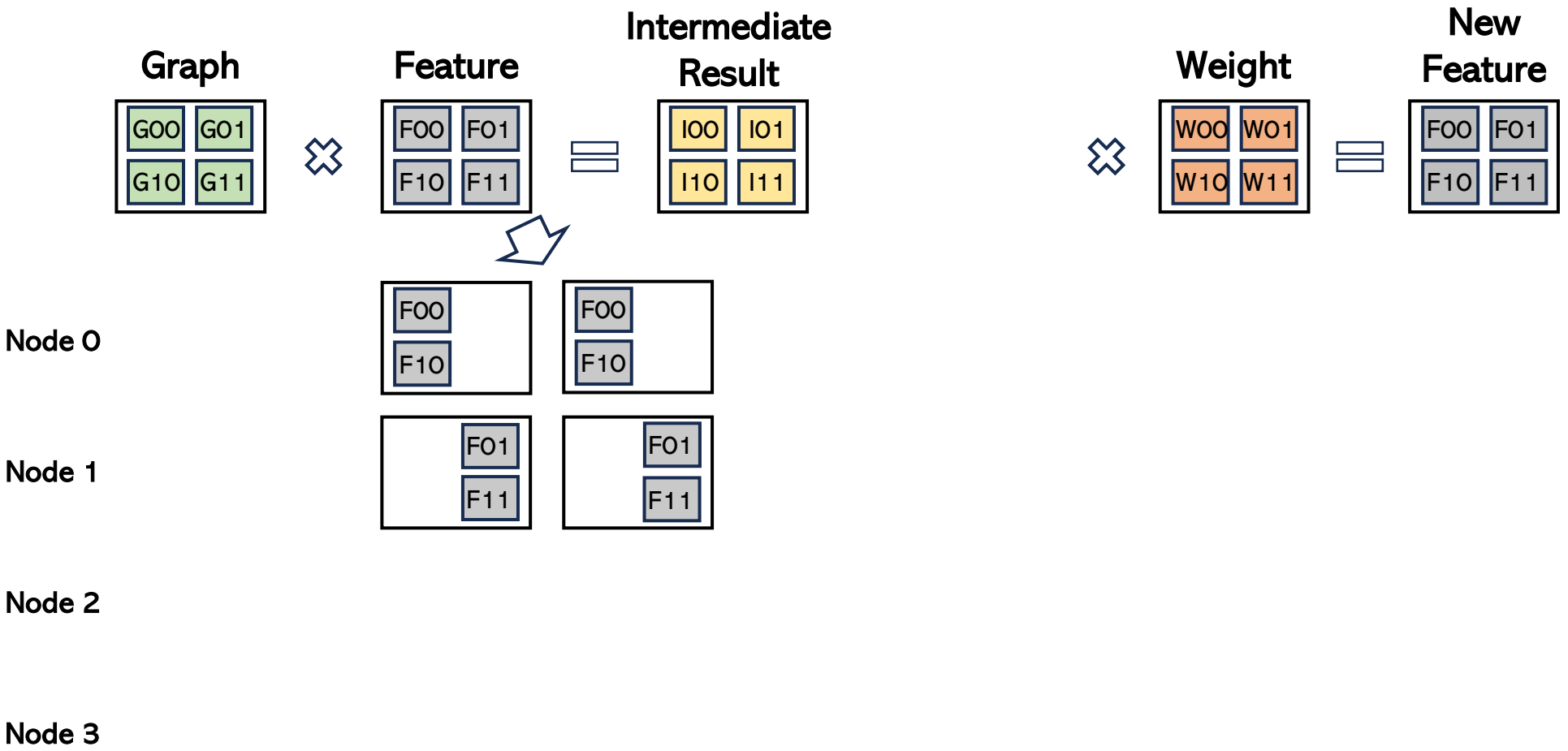
Execution time breakdown of benchmark applications



Execution time breakdown of communication overhead in benchmark applications

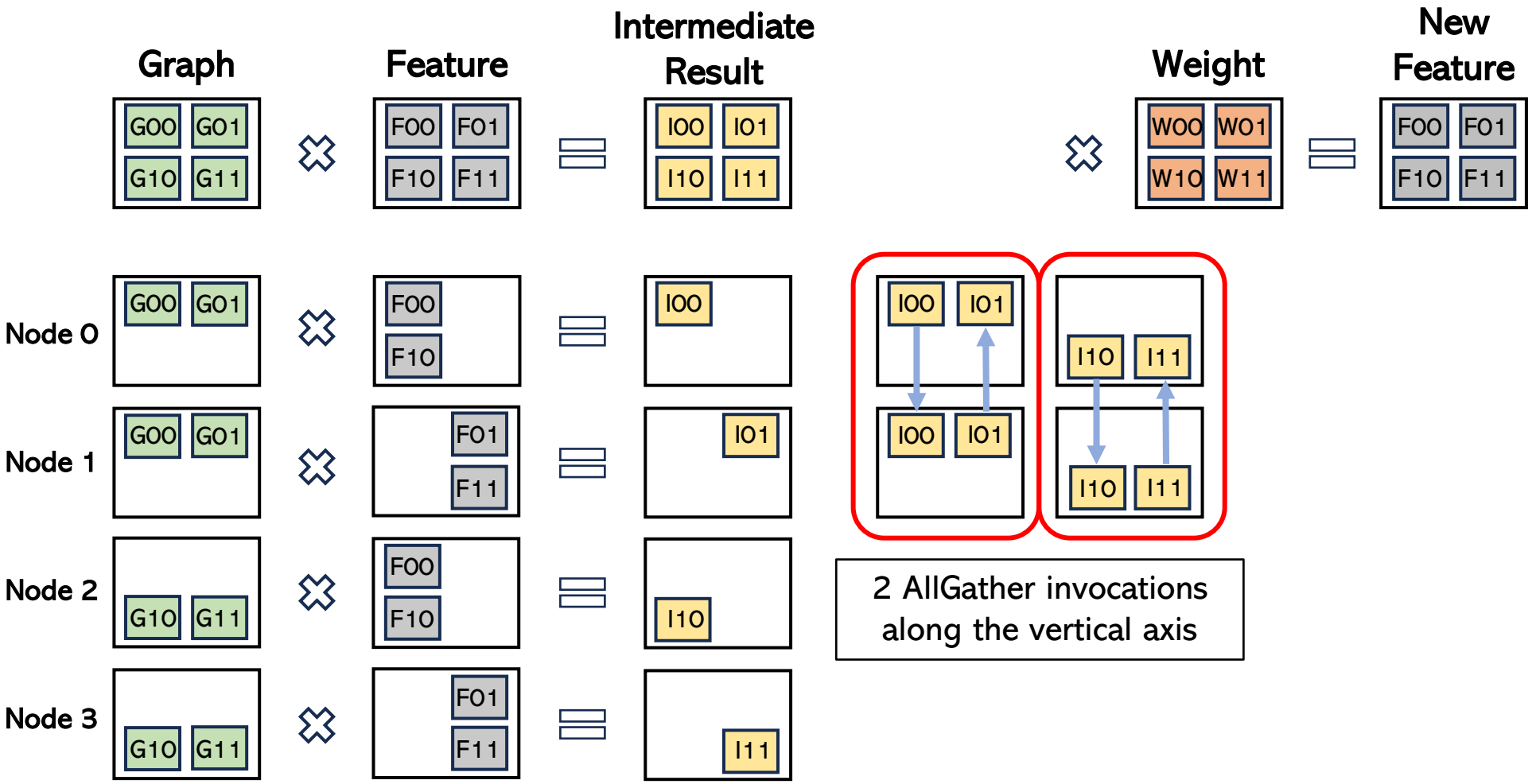
Lack of a Flexible Communication Model

- Assume a simple GNN inference application on 4 nodes



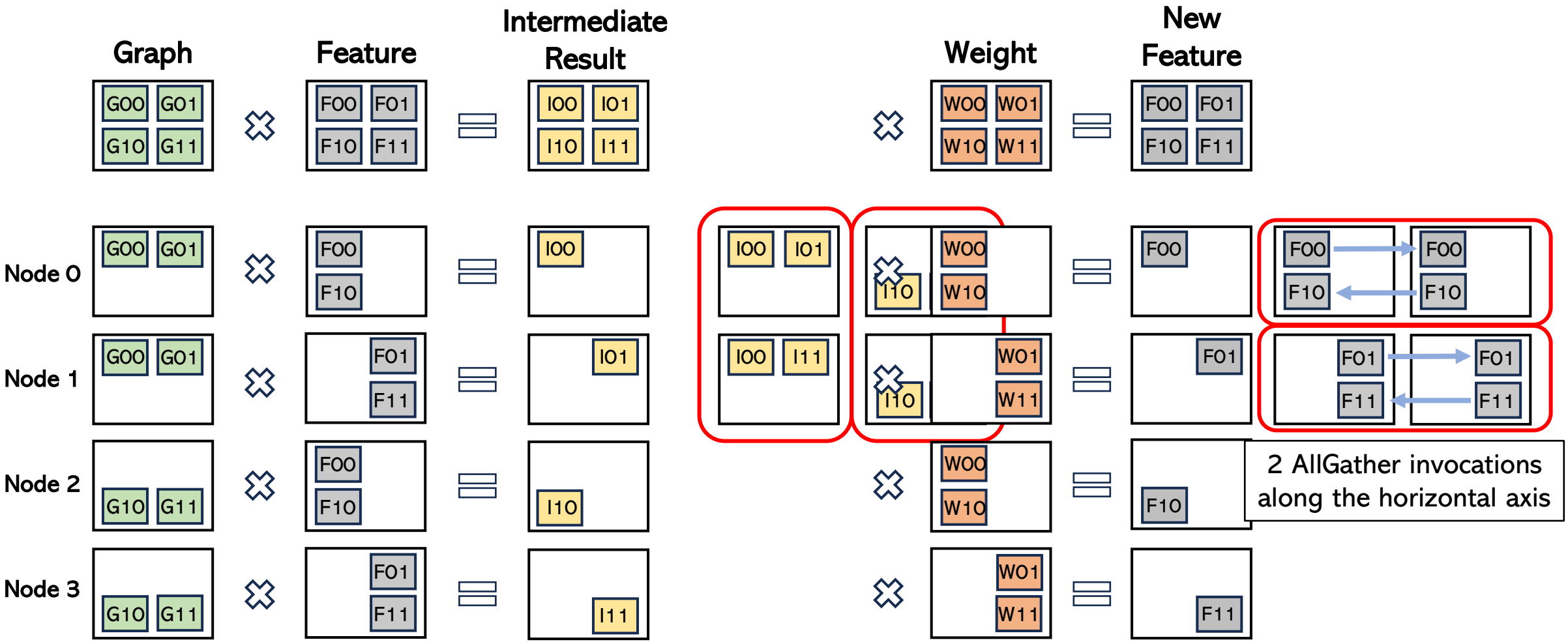
Lack of a Flexible Communication Model

- Assume a simple GNN inference application on 4 nodes



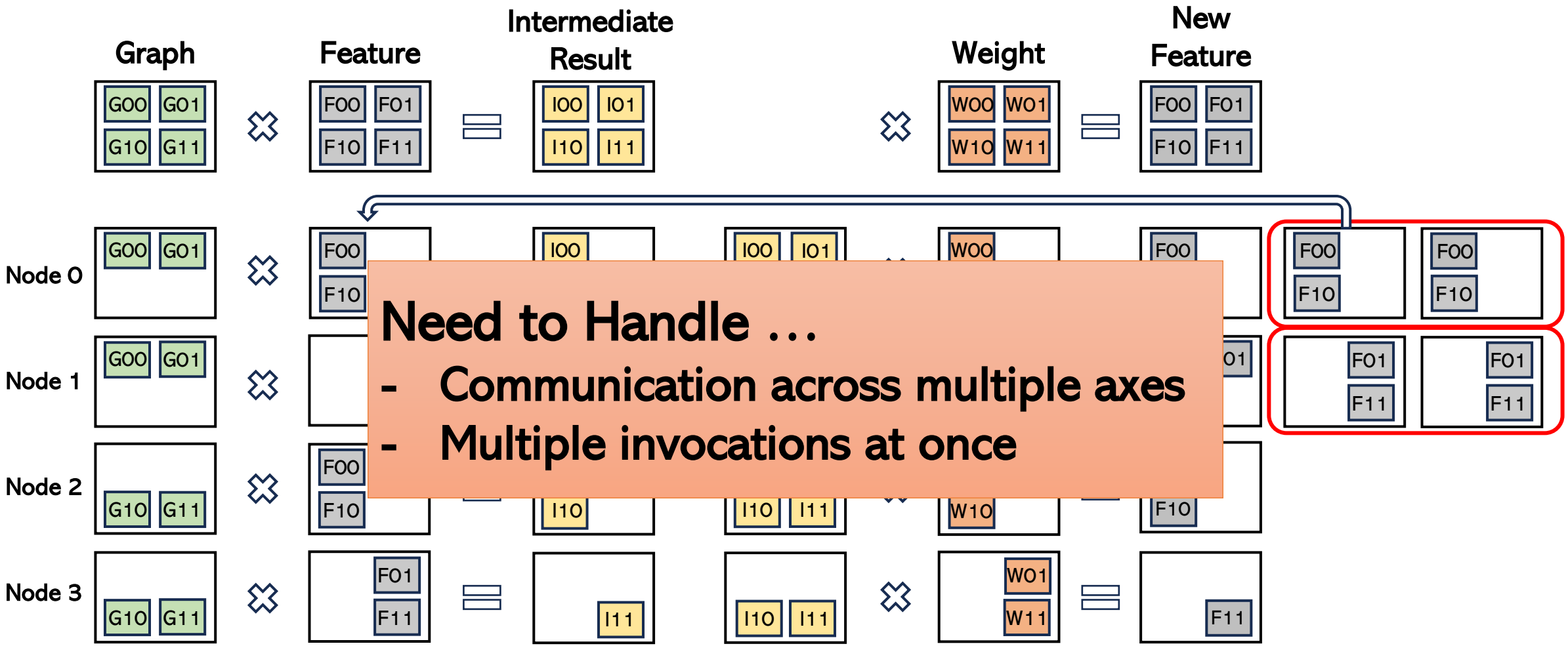
Lack of a Flexible Communication Model

- Assume a simple GNN inference application on 4 nodes



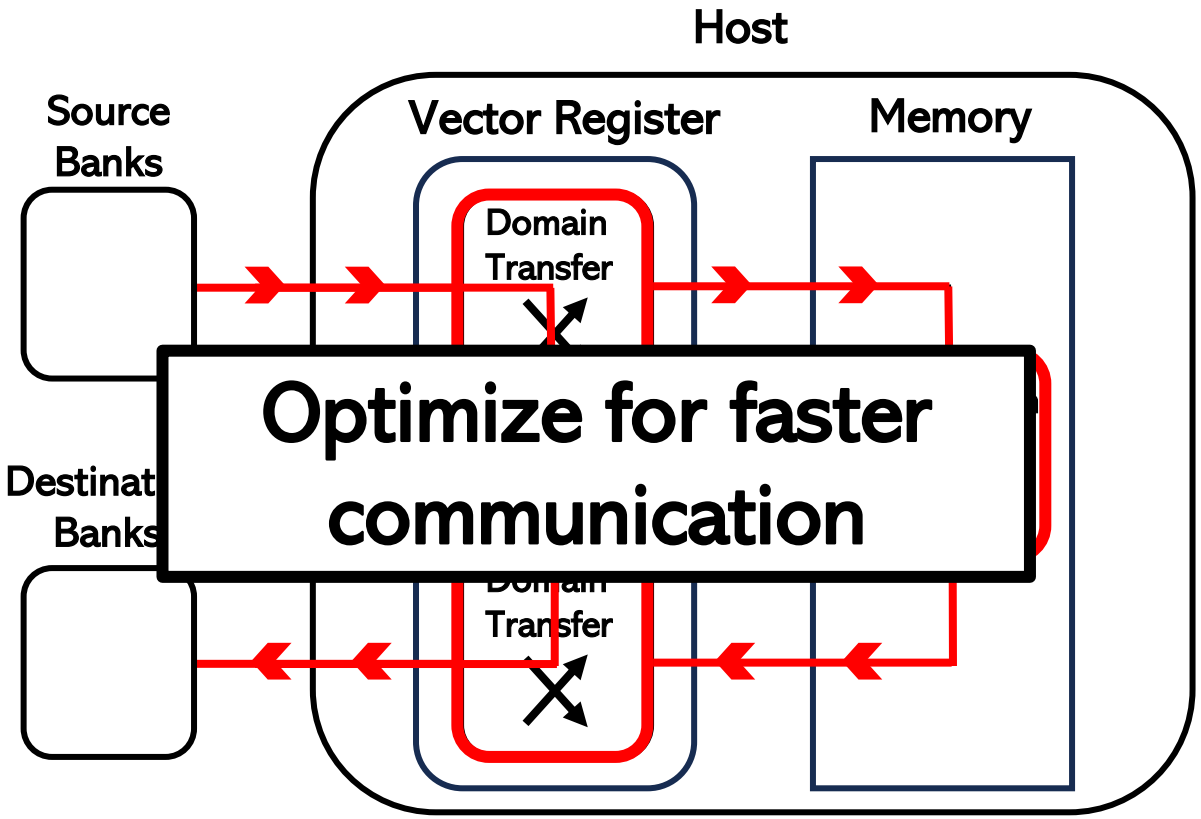
Lack of a Flexible Communication Model

- Assume a simple GNN inference application on 4 nodes

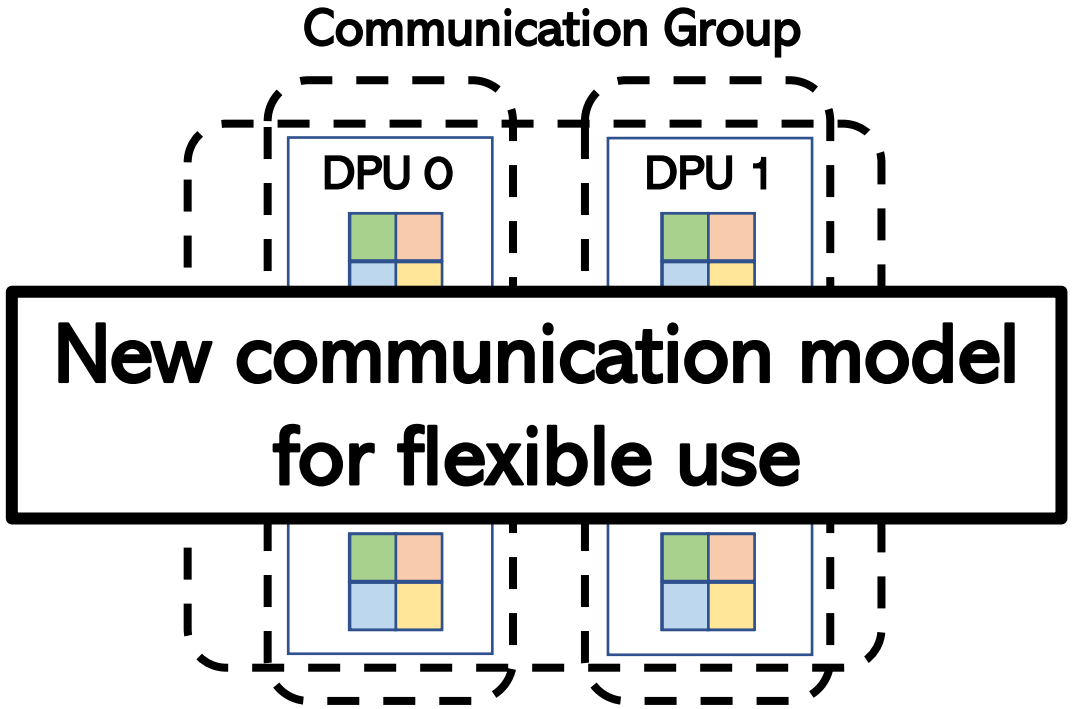


Overview

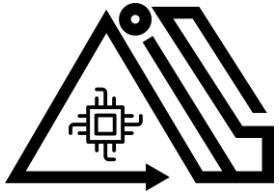
- Conventional inter-DPU communication is slow, but there is room for enhancement
- We aim to make them fast and enable flexible use



Optimization of inter-DPU communication



Flexible communication between DPUs



4. PID-Comm

0) Conventional inter-PIM Communication

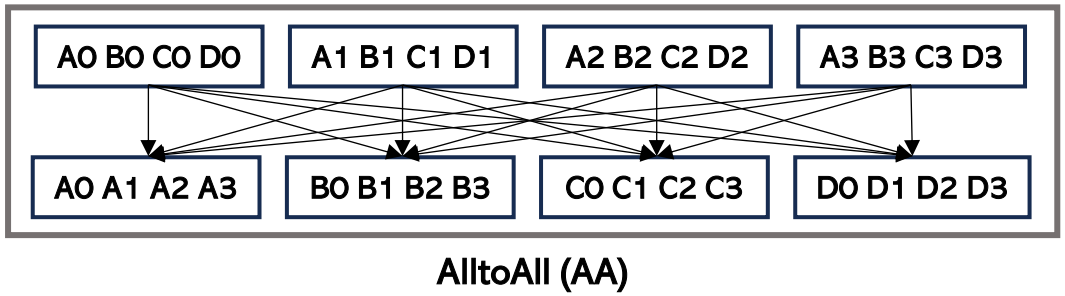
1) PIM-assisted Reordering

2) In-register Modulation

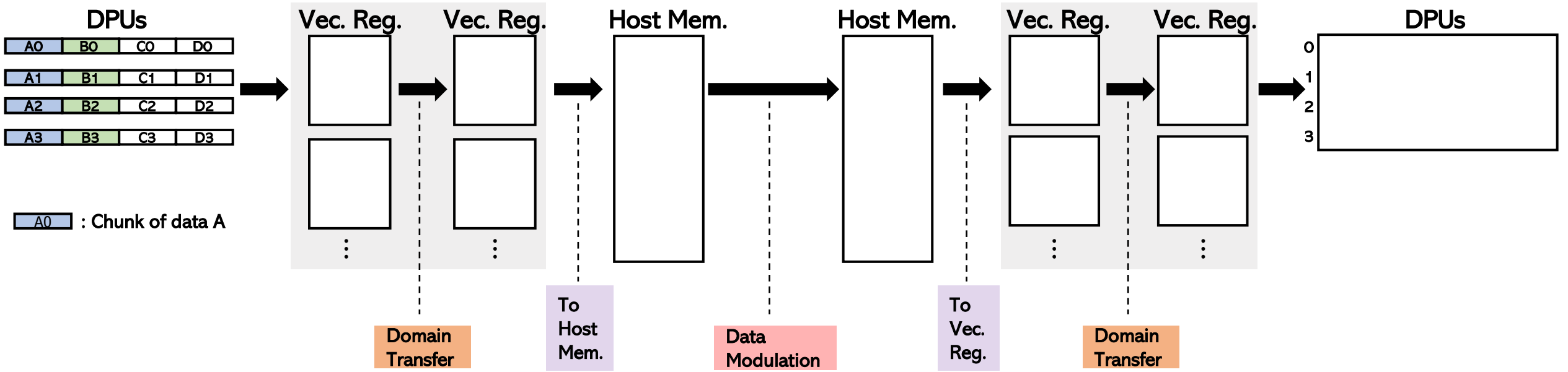
3) Cross-domain Modulation

4) The Hypercube Model

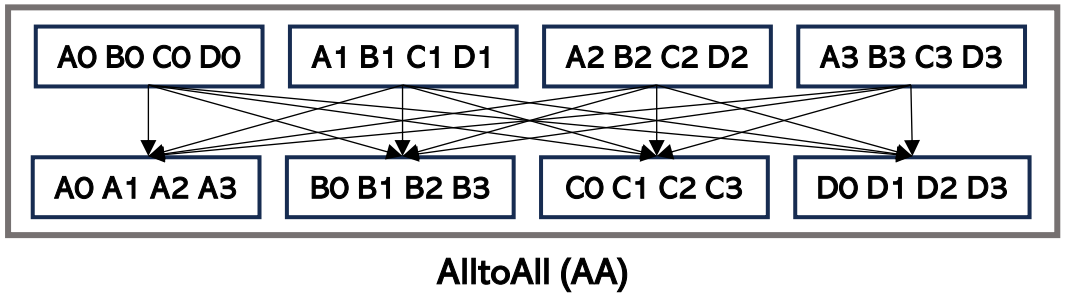
Conventional Inter-DPU communication



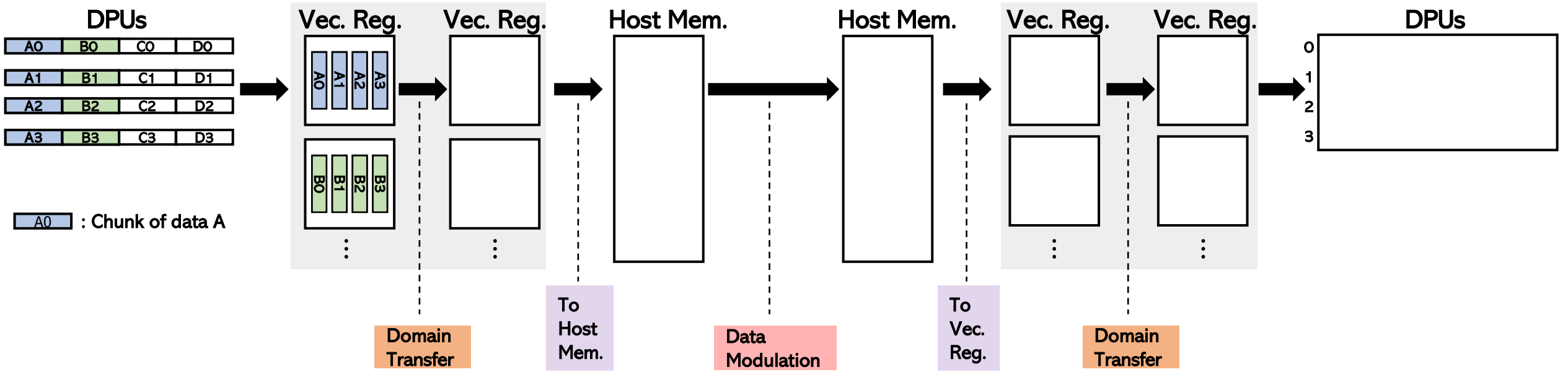
AlltoAll (AA)



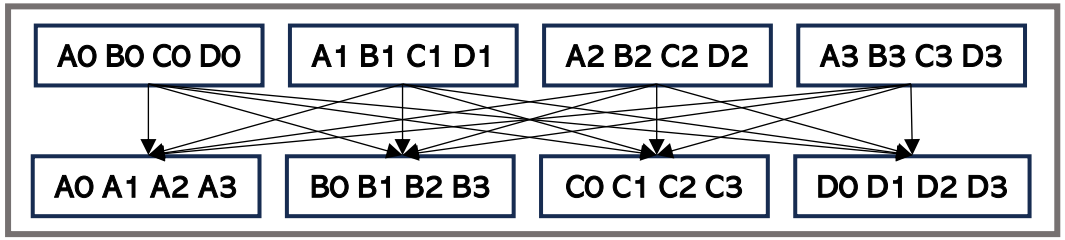
Conventional Inter-DPU communication



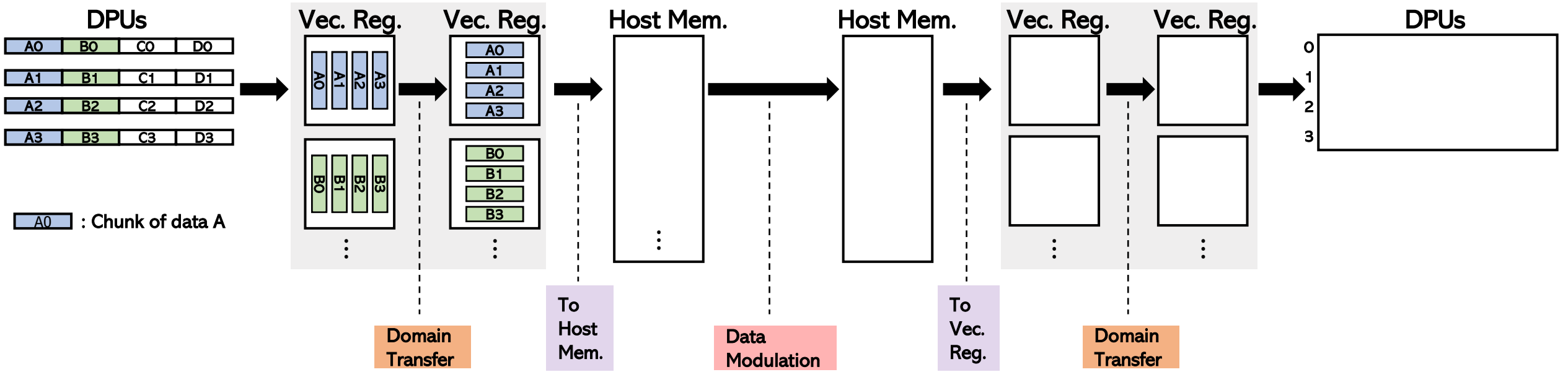
AlltoAll (AA)



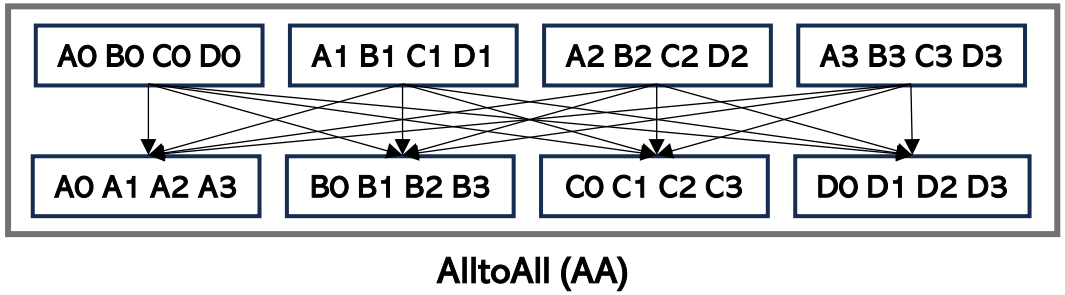
Conventional Inter-DPU communication



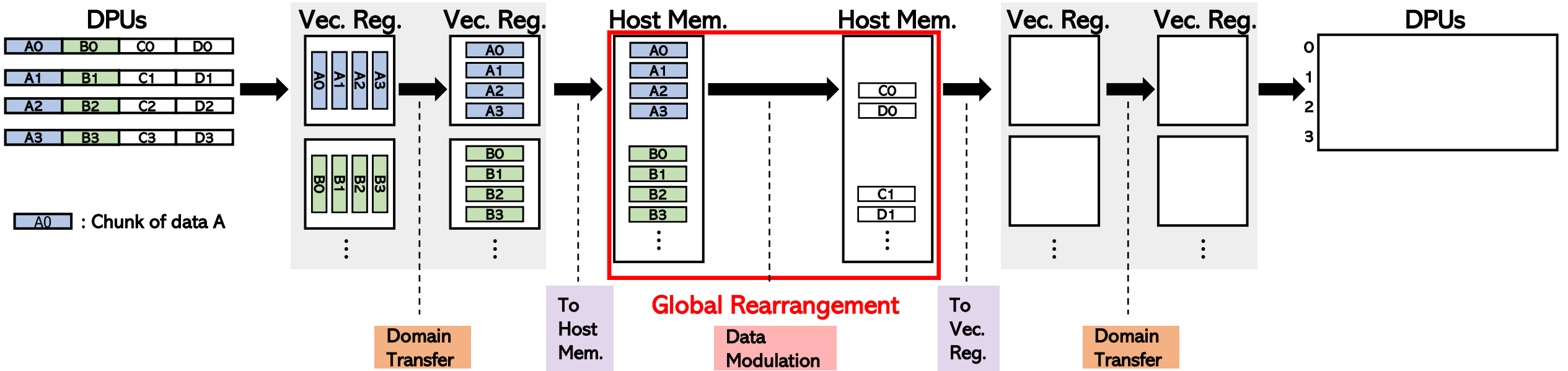
AlltoAll (AA)



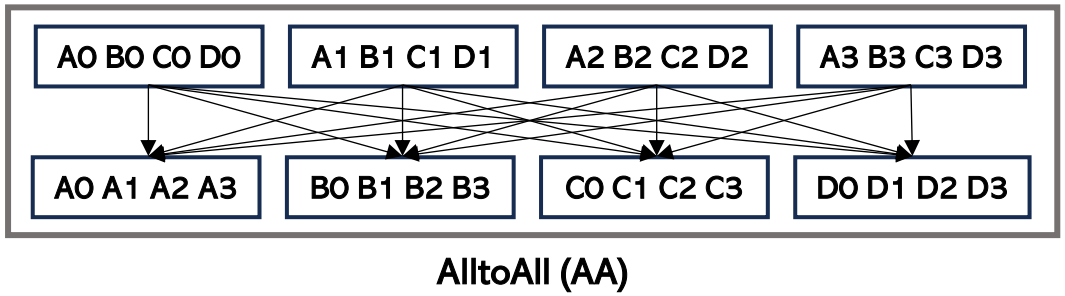
Conventional Inter-DPU communication



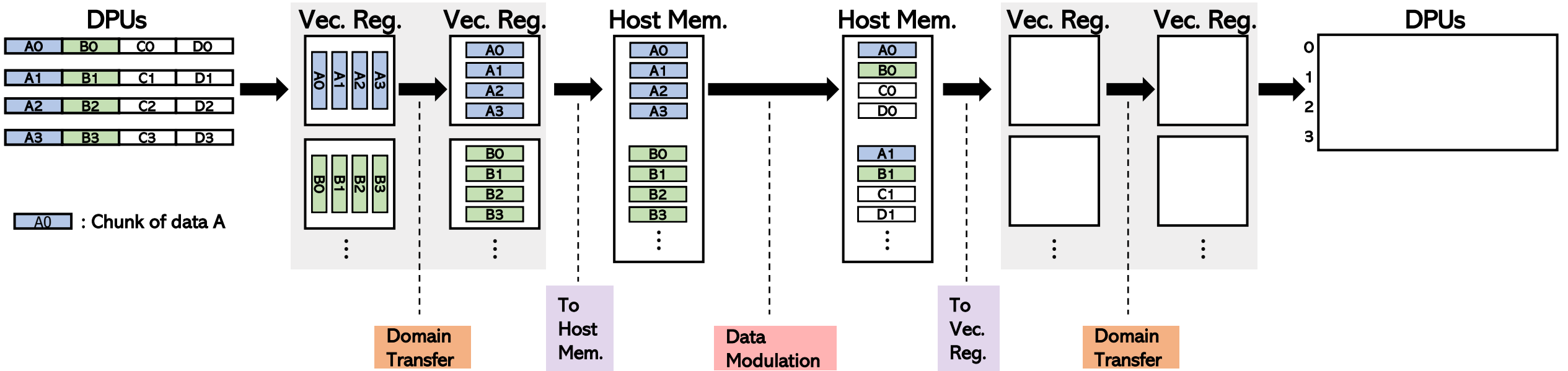
AlltoAll (AA)



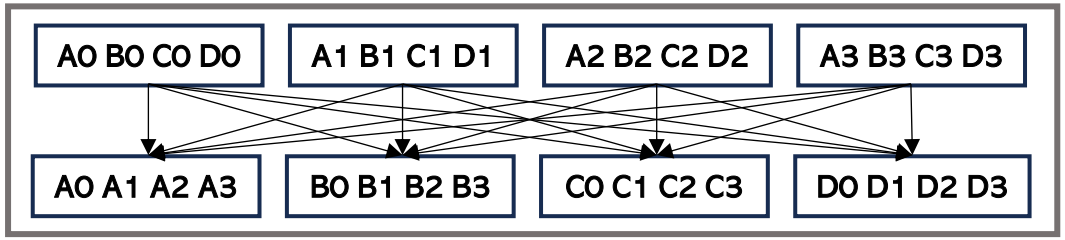
Conventional Inter-DPU communication



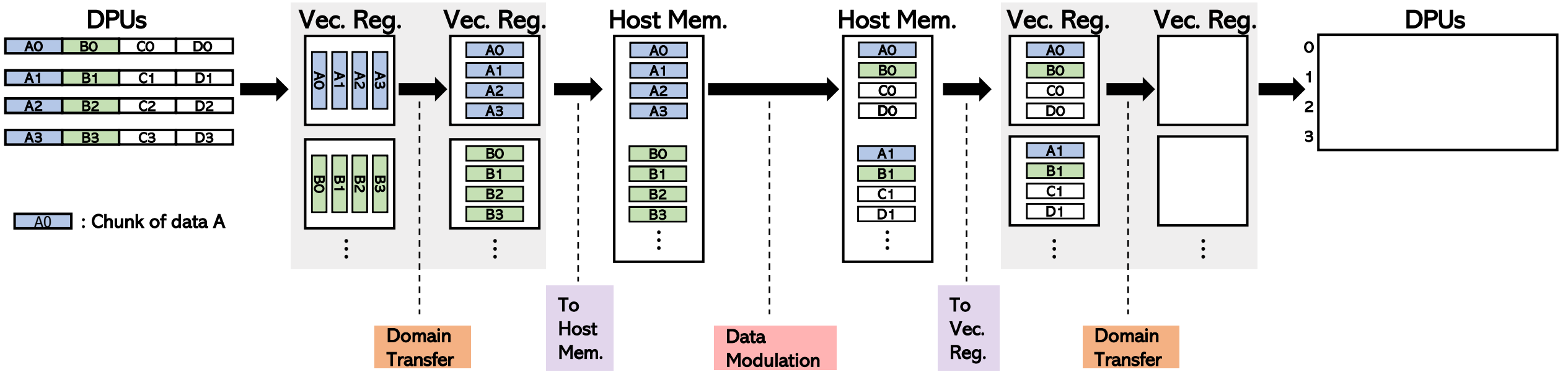
AlltoAll (AA)



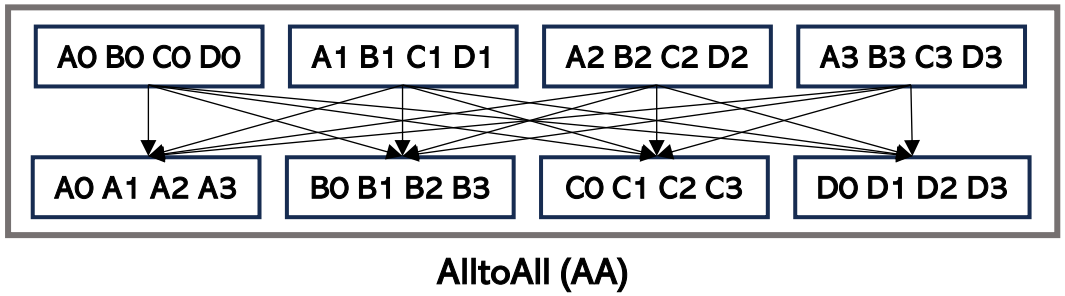
Conventional Inter-DPU communication



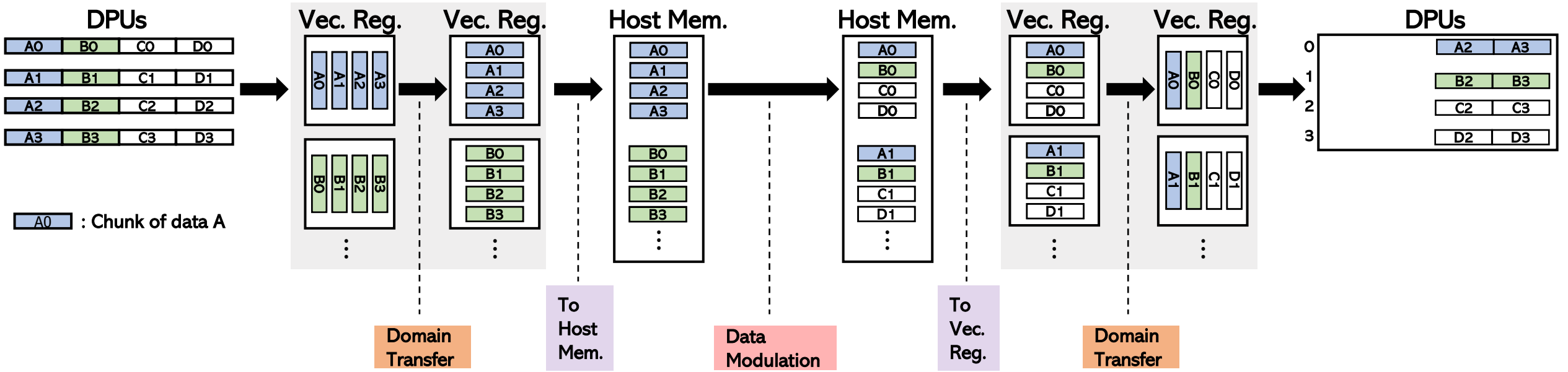
AlltoAll (AA)



Conventional Inter-DPU communication

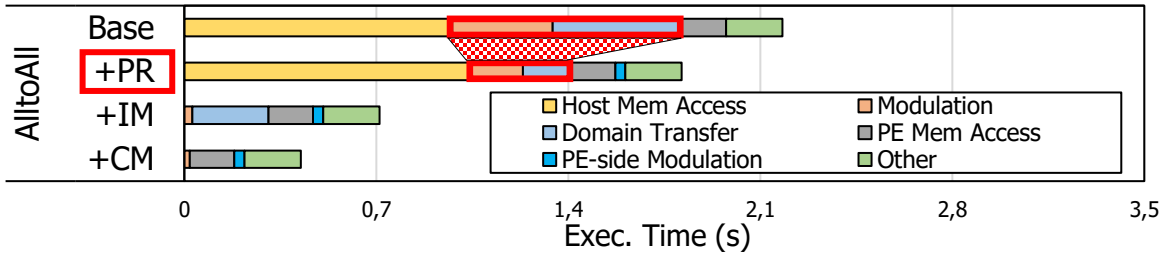


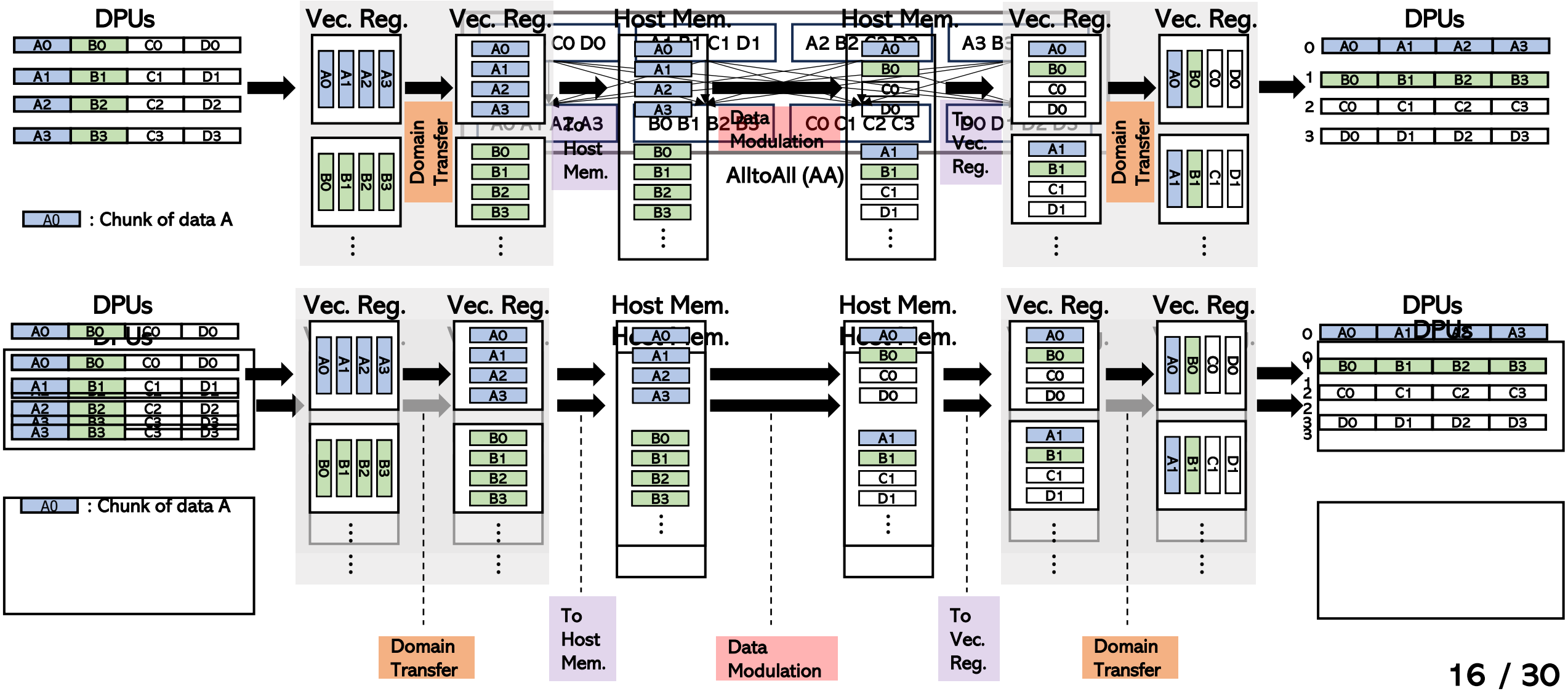
AlltoAll (AA)



1. PIM-assisted Reordering

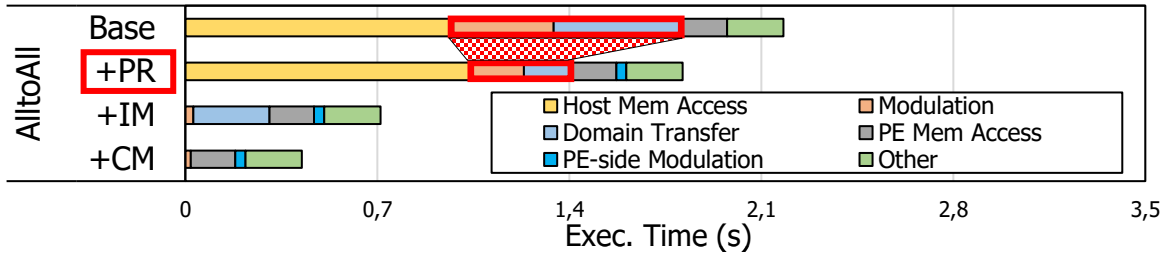
- Reorder data inside DPUs to enhance data locality

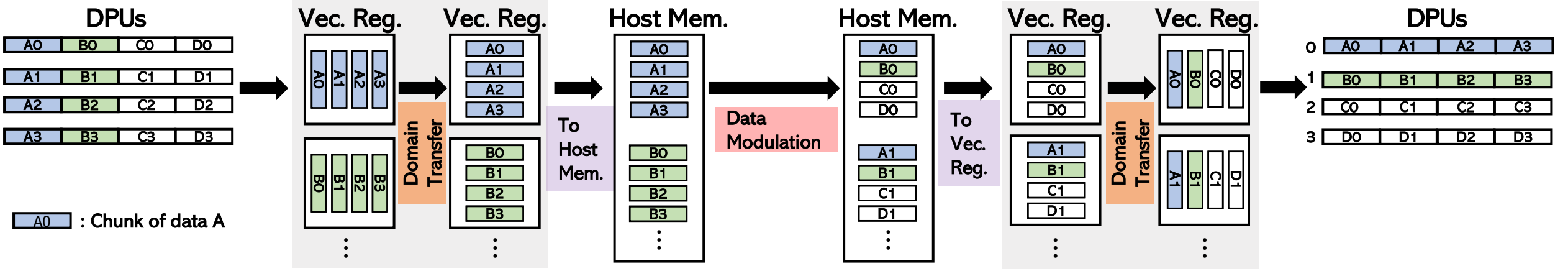


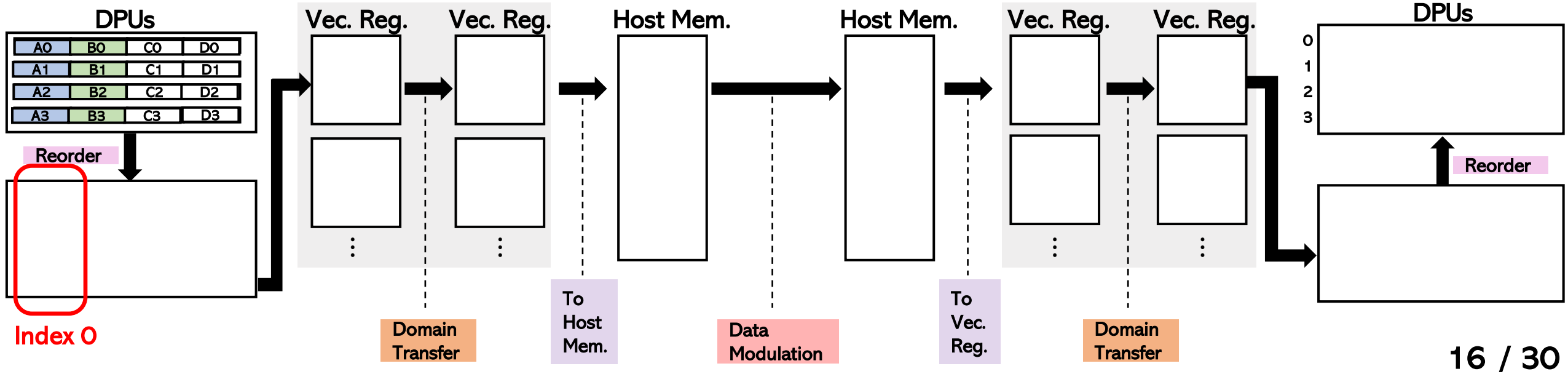


1. PIM-assisted Reordering

- Reorder data inside DPUs to enhance data locality

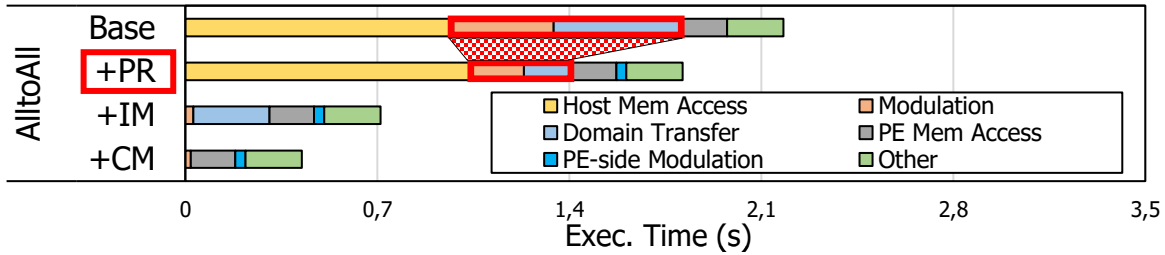


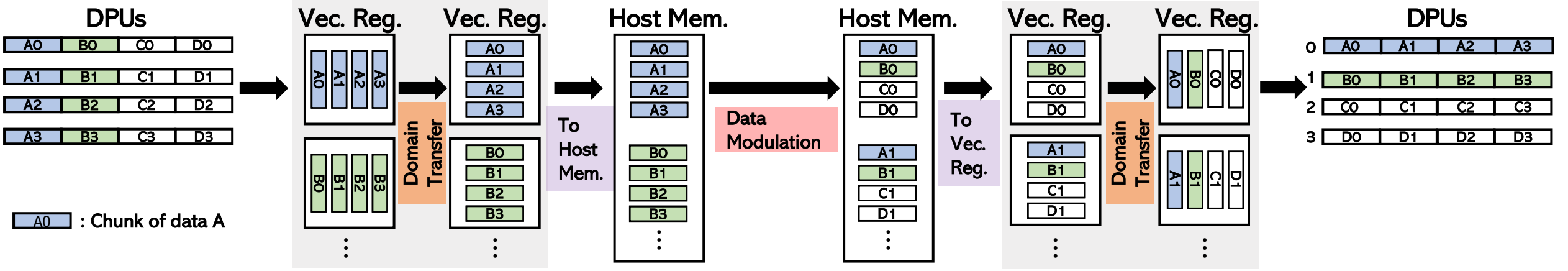


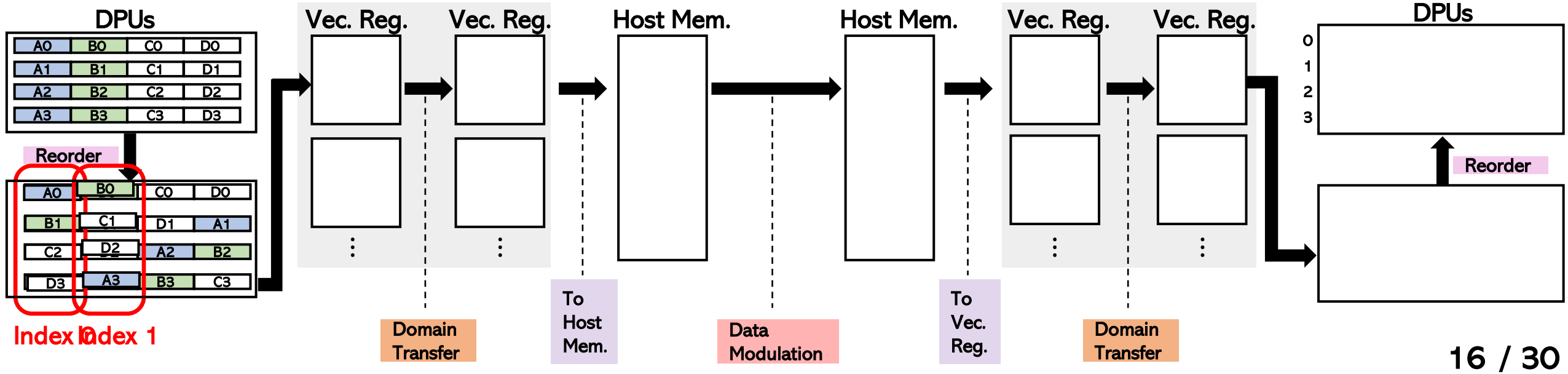


1. PIM-assisted Reordering

- Reorder data inside DPUs to enhance data locality

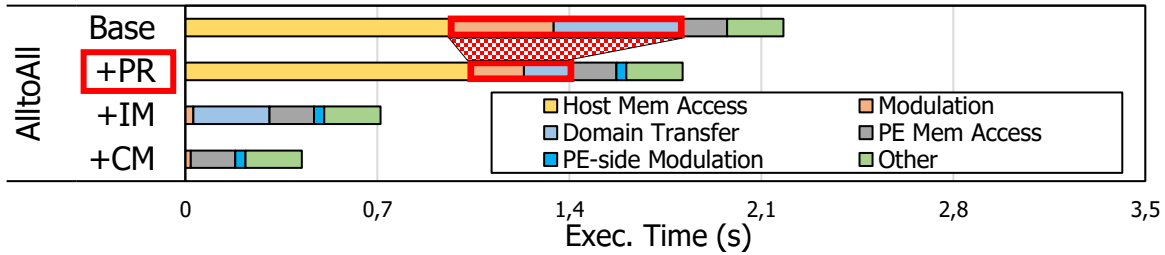


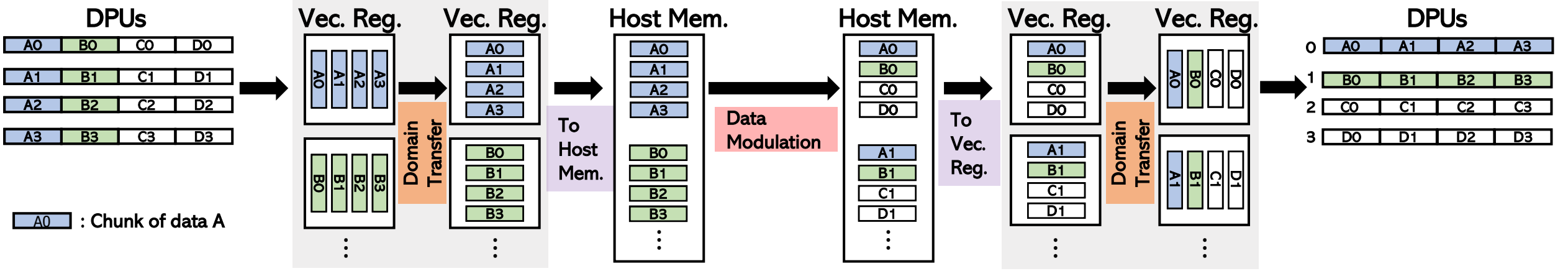


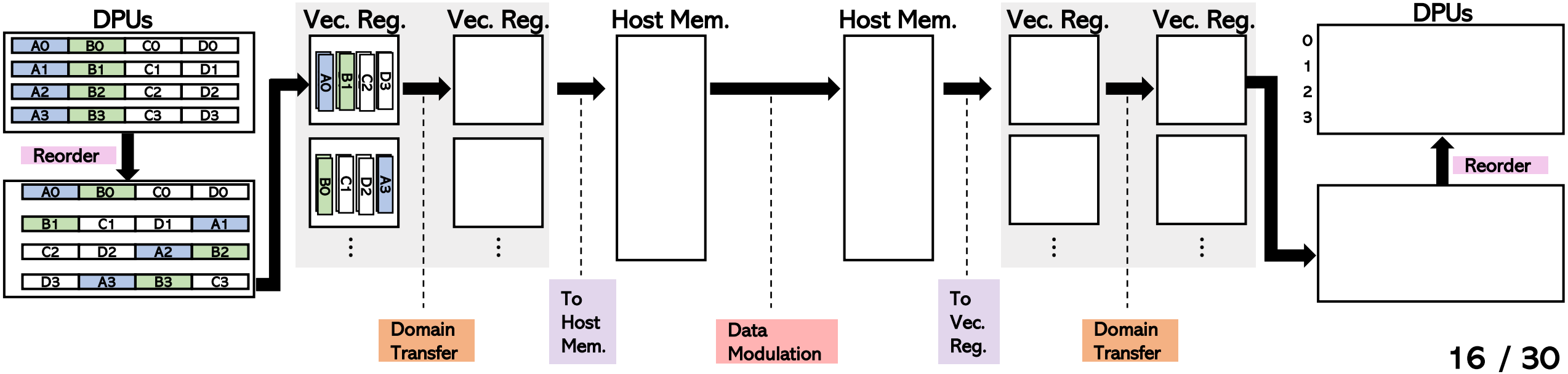


1. PIM-assisted Reordering

- Reorder data inside DPUs to enhance data locality

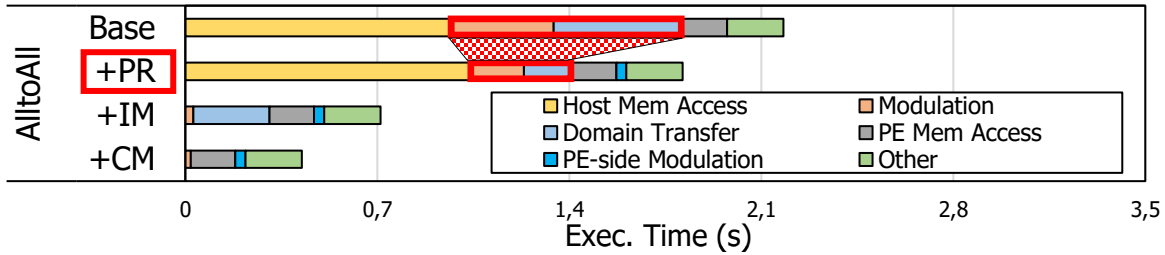


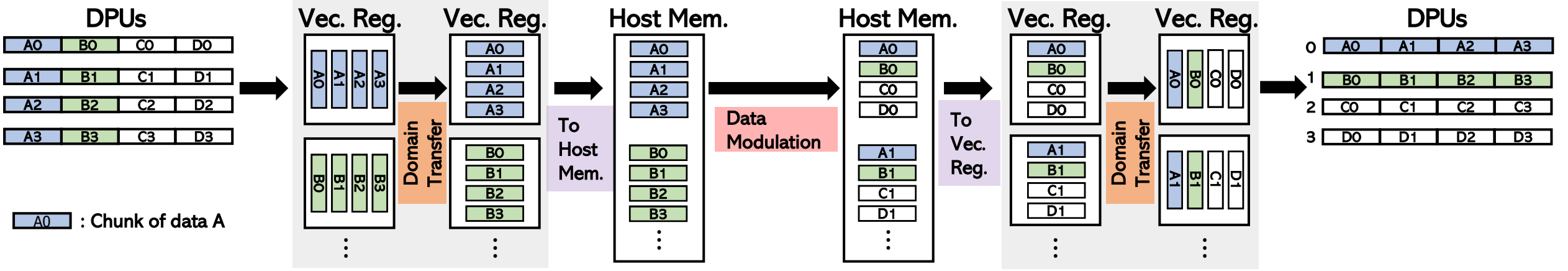


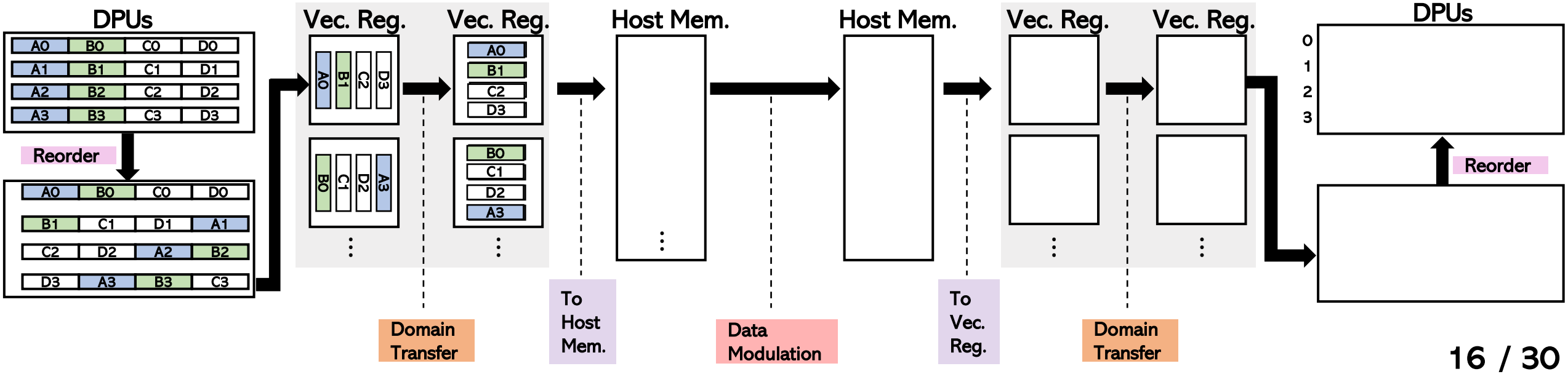


1. PIM-assisted Reordering

- Reorder data inside DPUs to enhance data locality

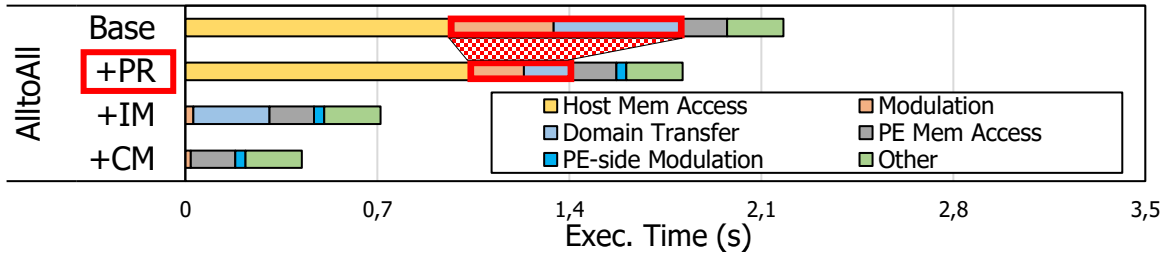


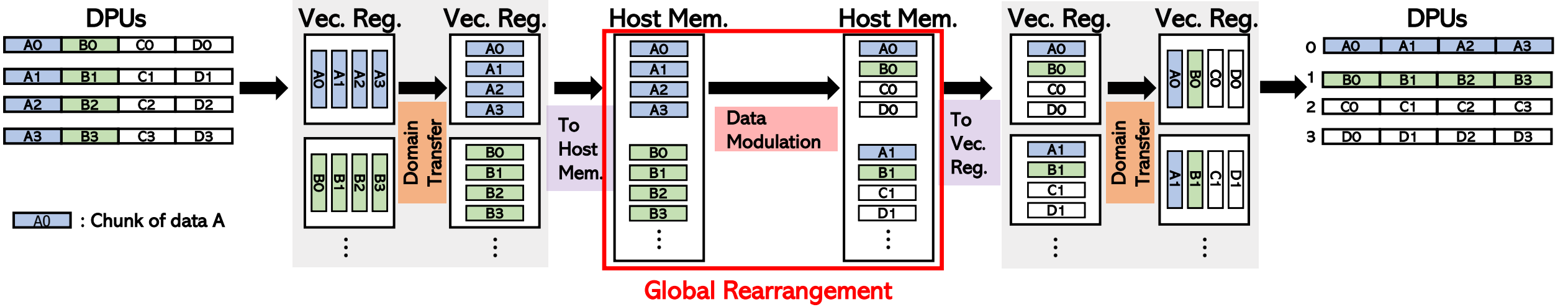


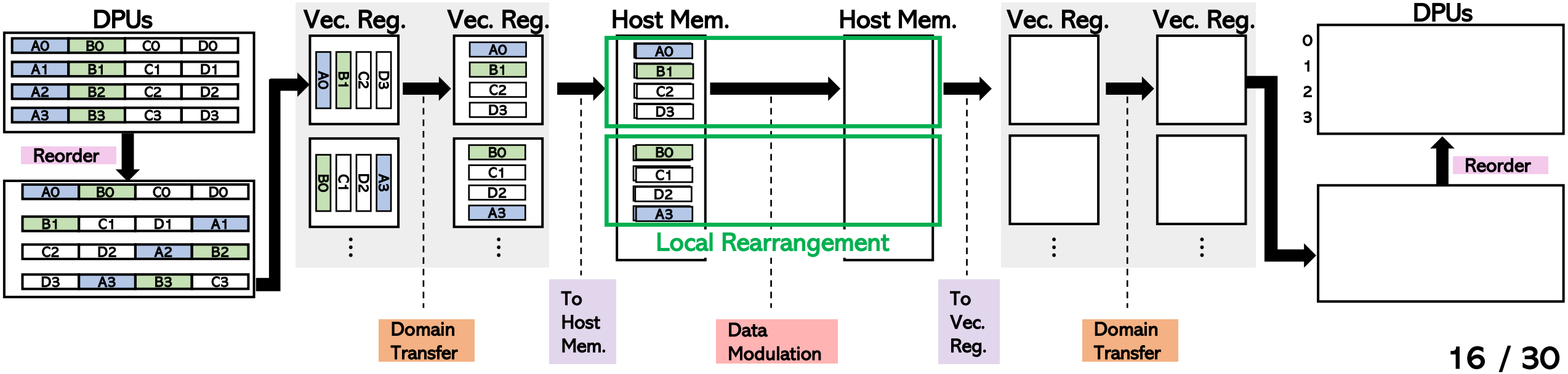


1. PIM-assisted Reordering

- Reorder data inside DPUs to enhance data locality

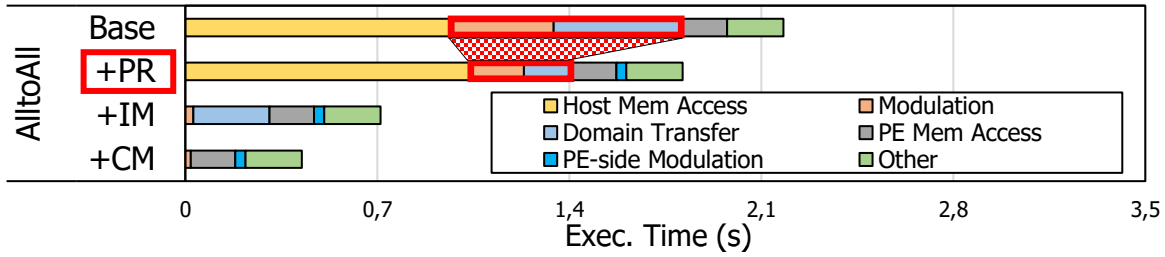


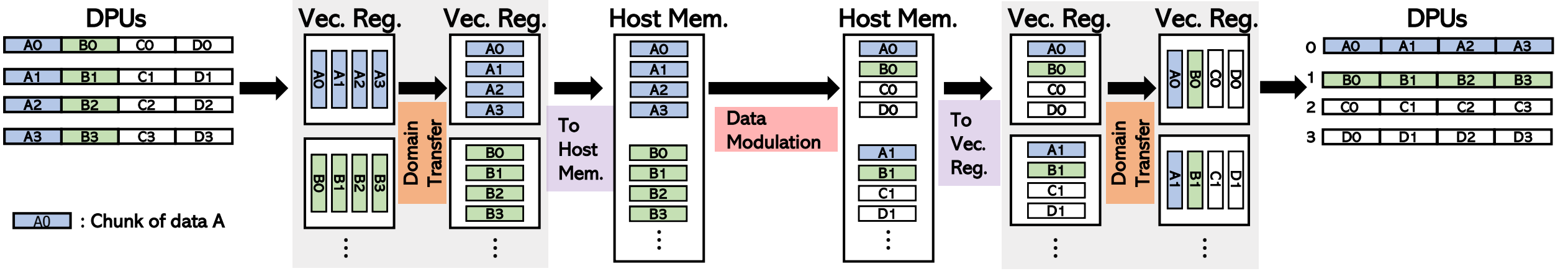


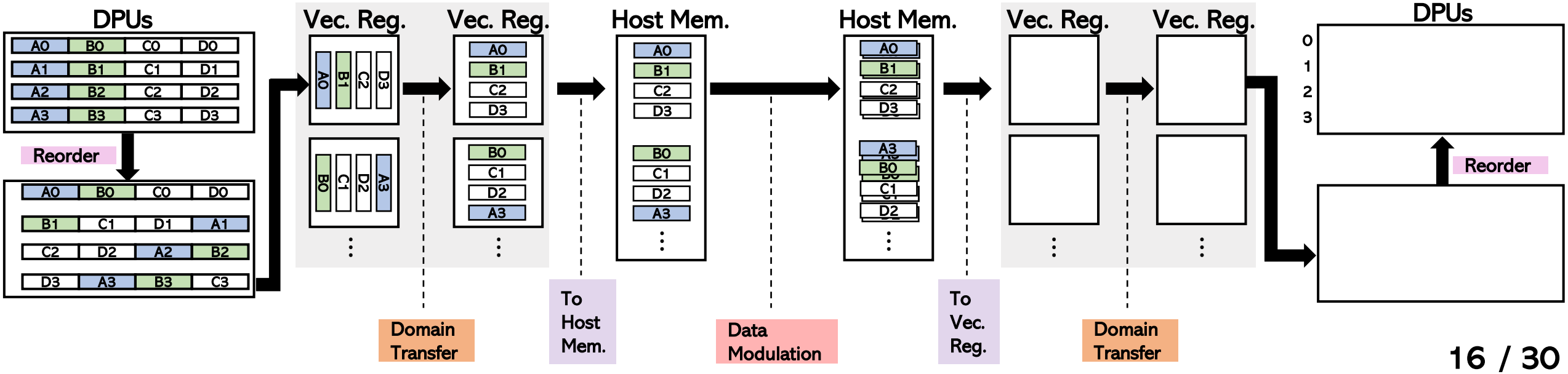


1. PIM-assisted Reordering

- Reorder data inside DPUs to enhance data locality

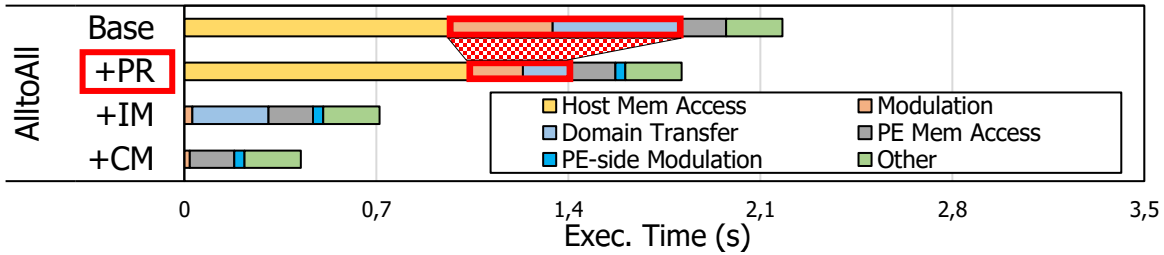


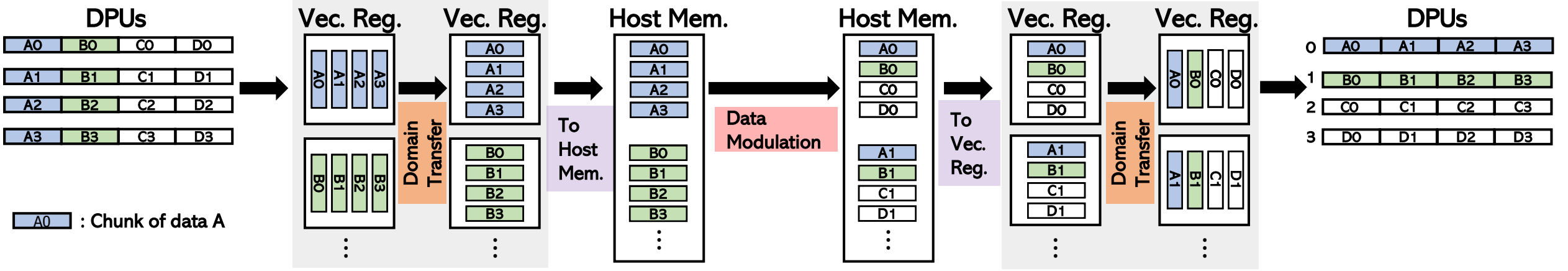


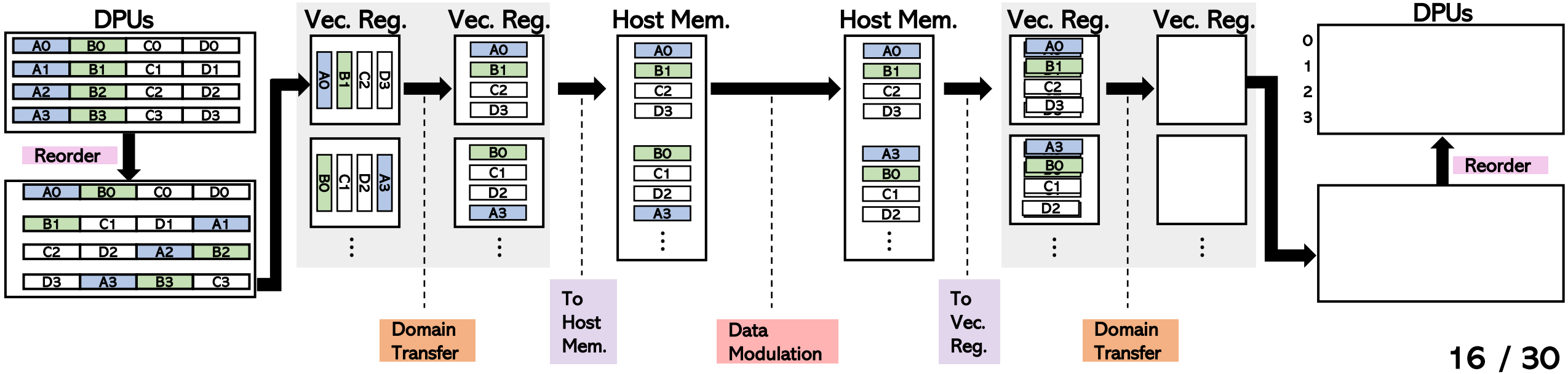


1. PIM-assisted Reordering

- Reorder data inside DPUs to enhance data locality

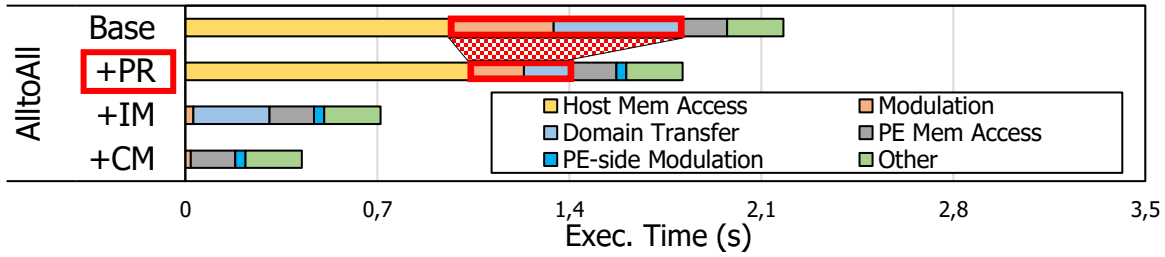


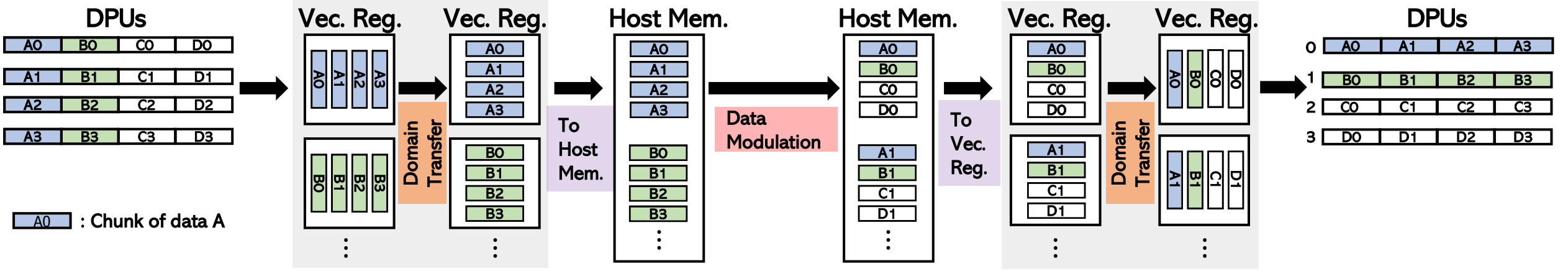


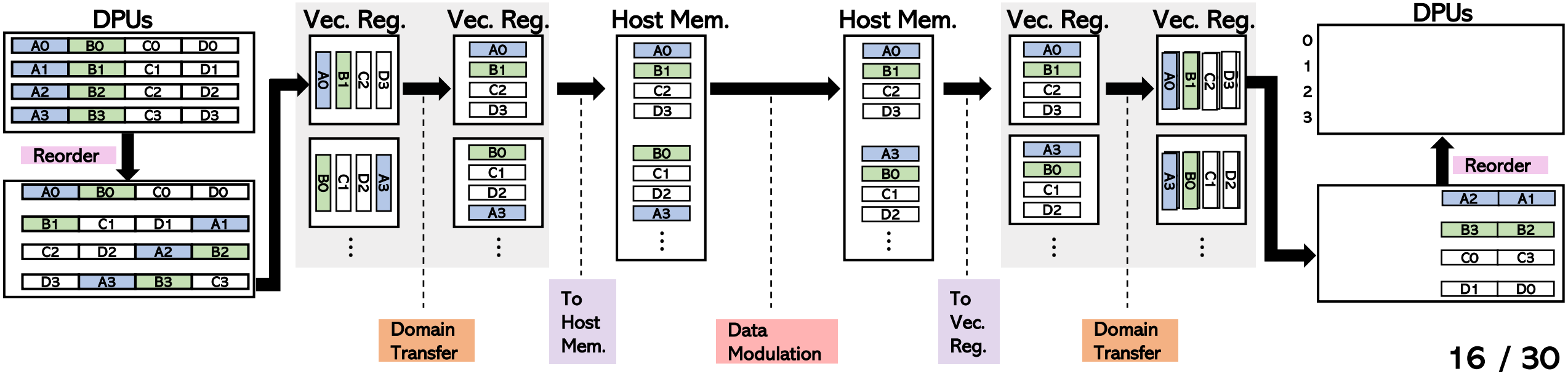


1. PIM-assisted Reordering

- Reorder data inside DPUs to enhance data locality

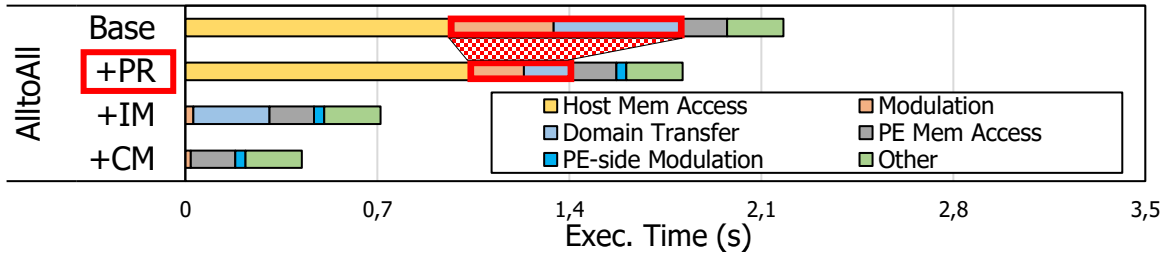


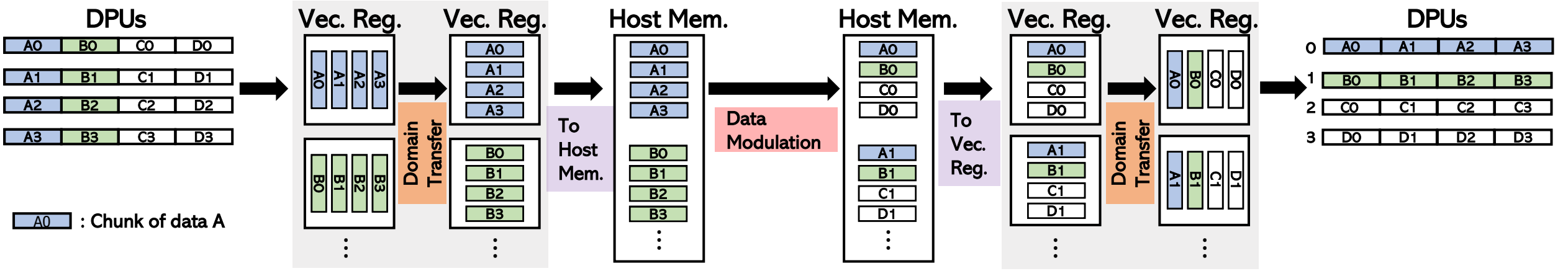


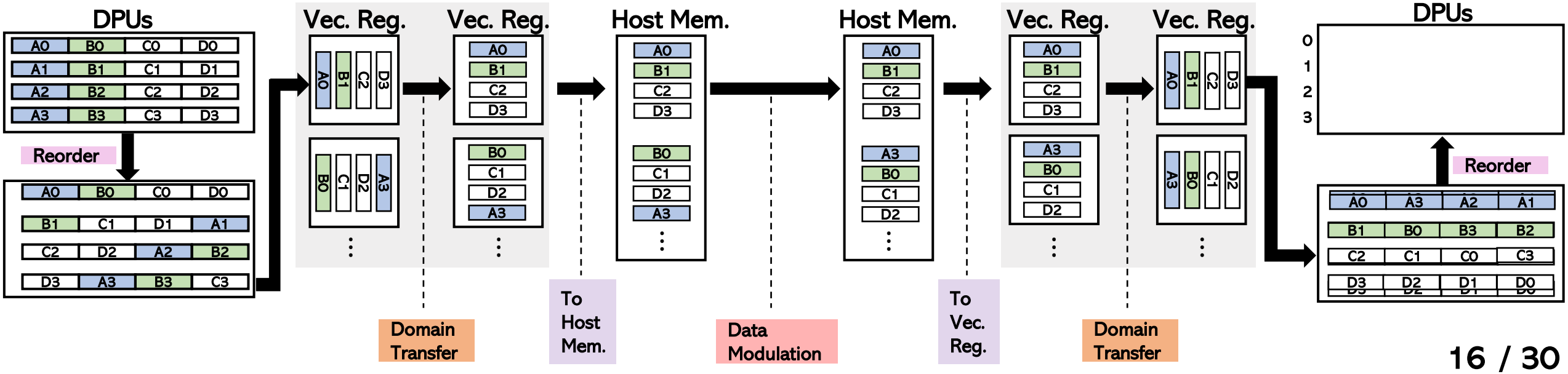


1. PIM-assisted Reordering

- Reorder data inside DPUs to enhance data locality

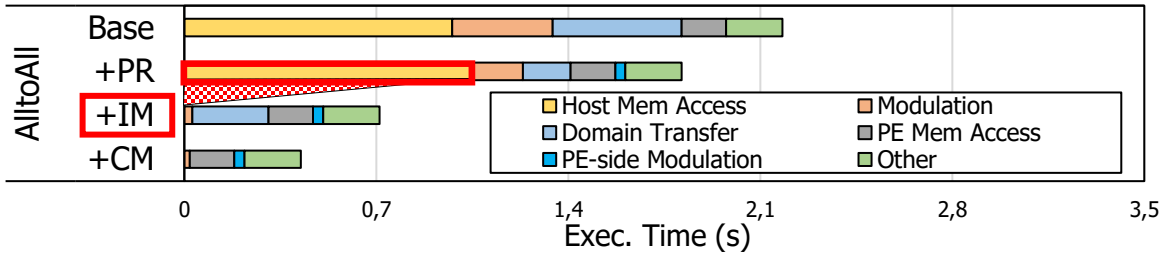


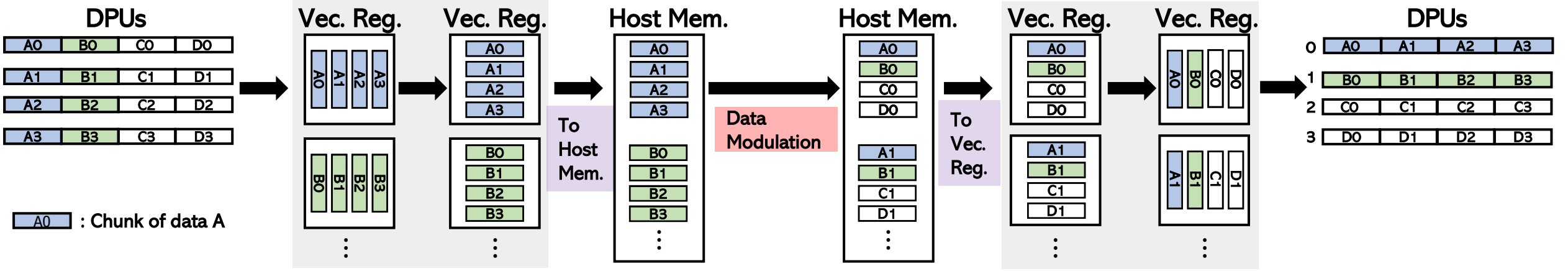




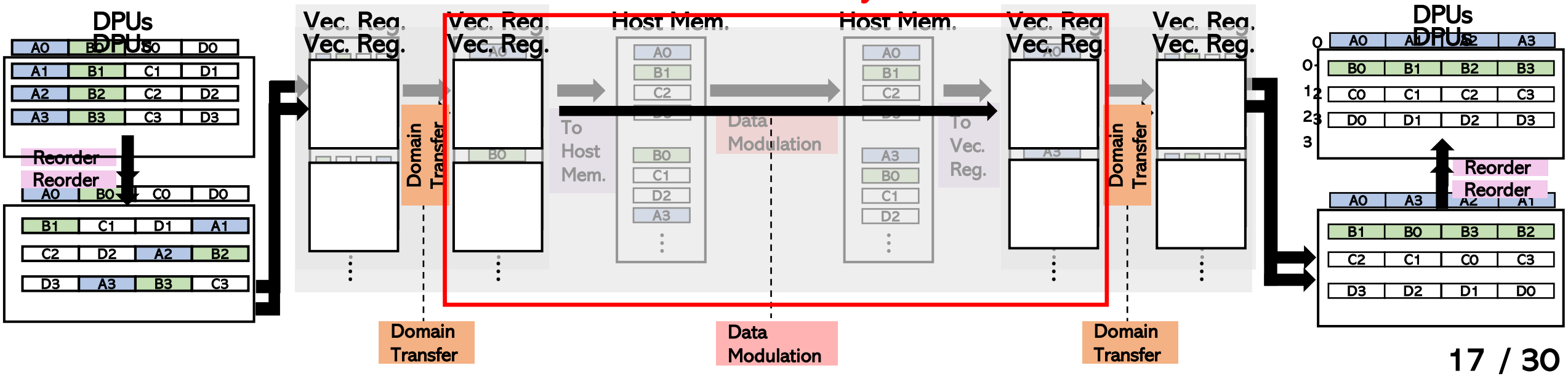
2. In-register Modulation

- Modulate data within the vector register



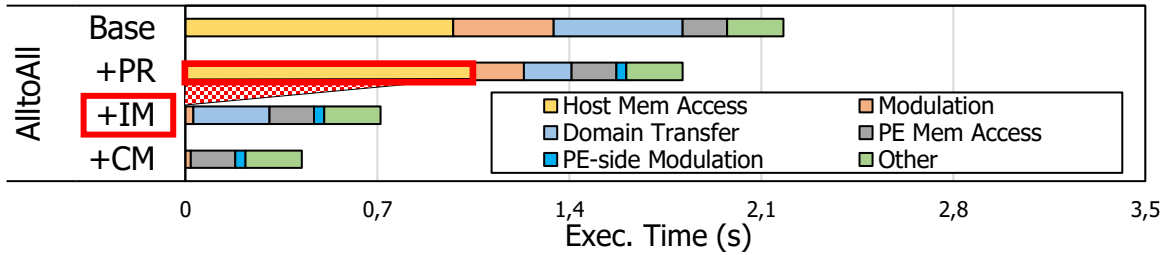


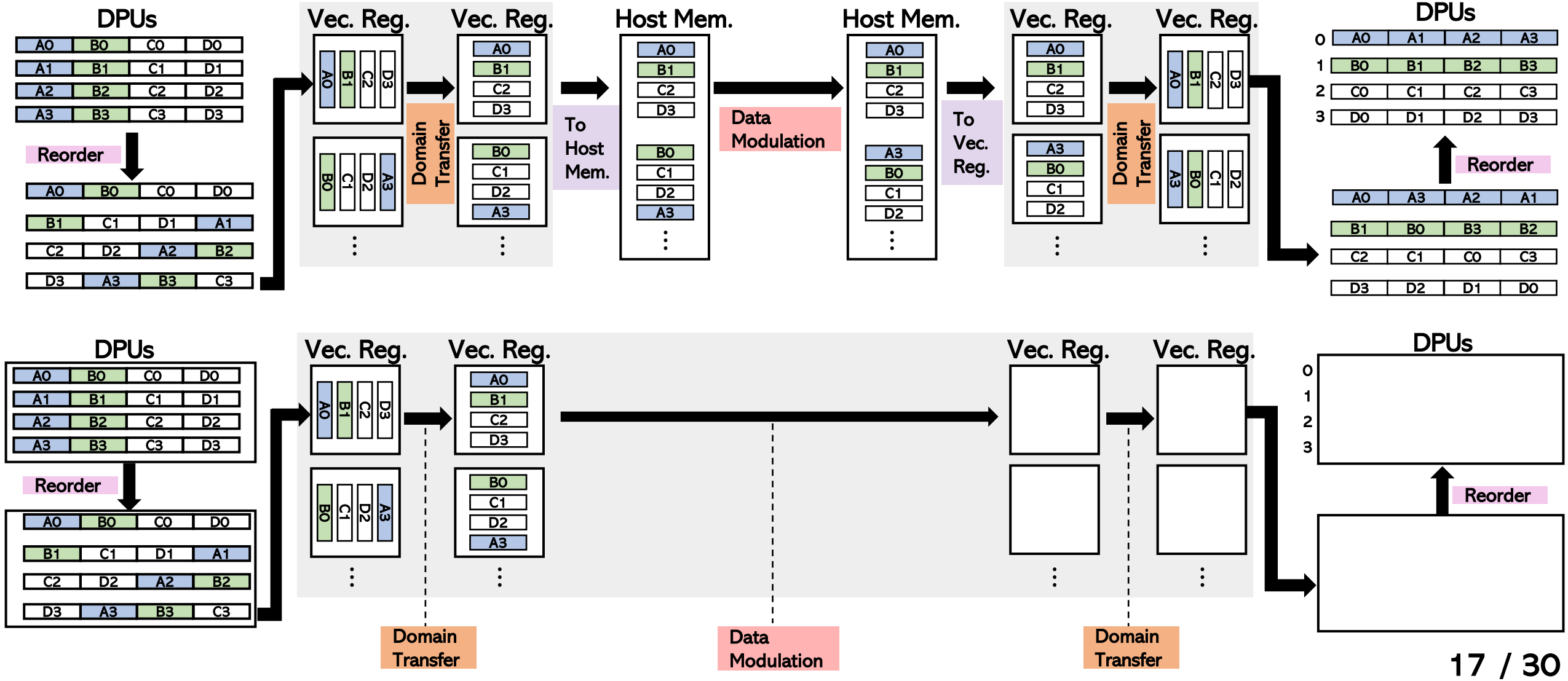
Remove host memory access



2. In-register Modulation

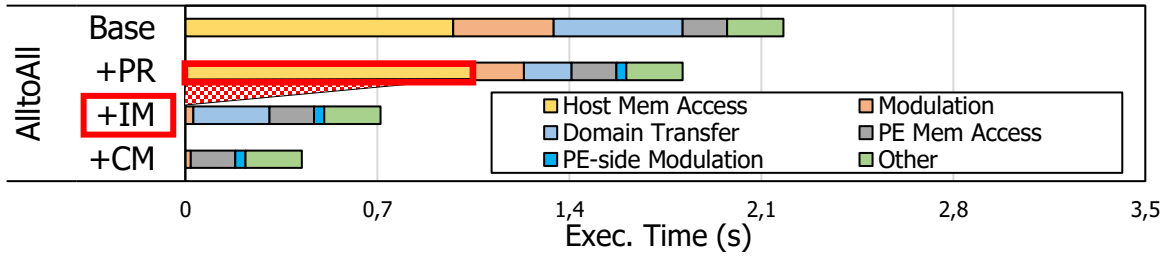
- Modulate data within the vector register

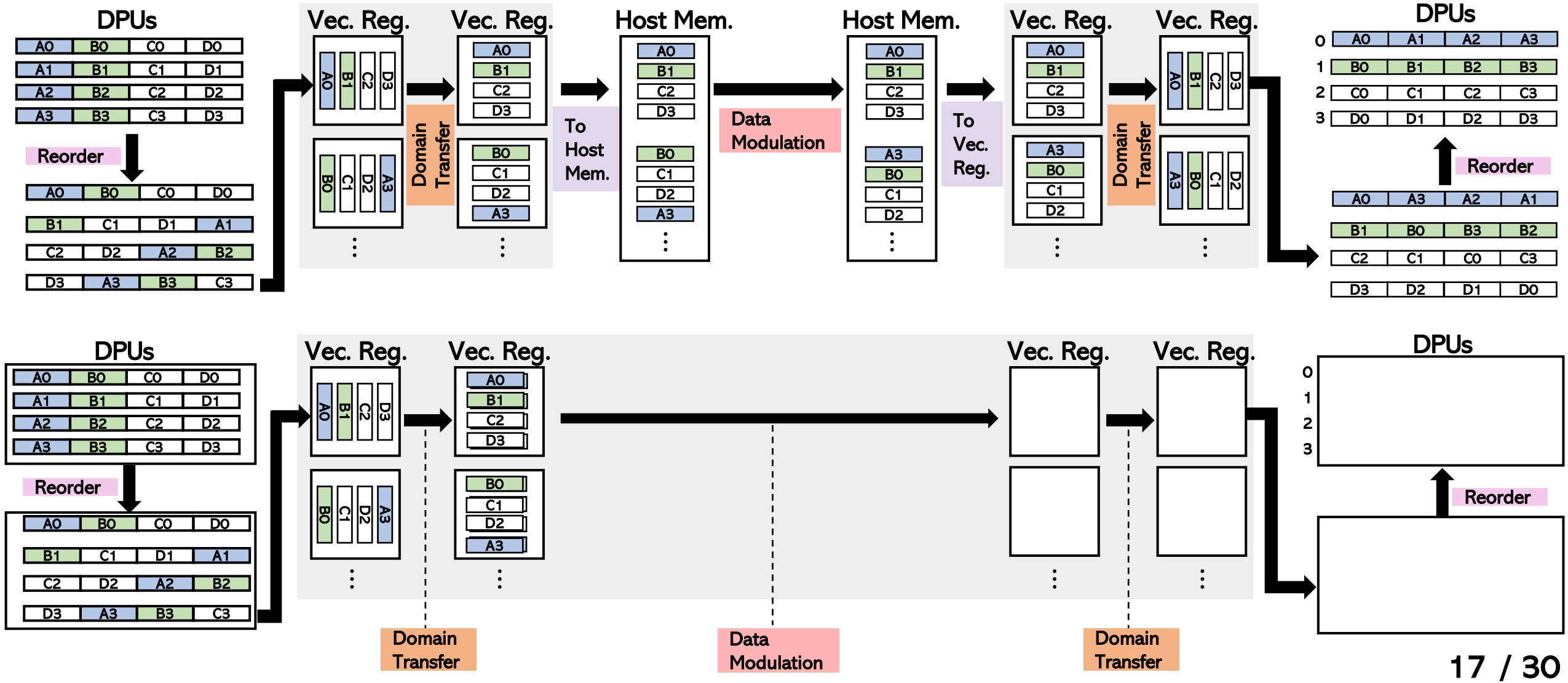




2. In-register Modulation

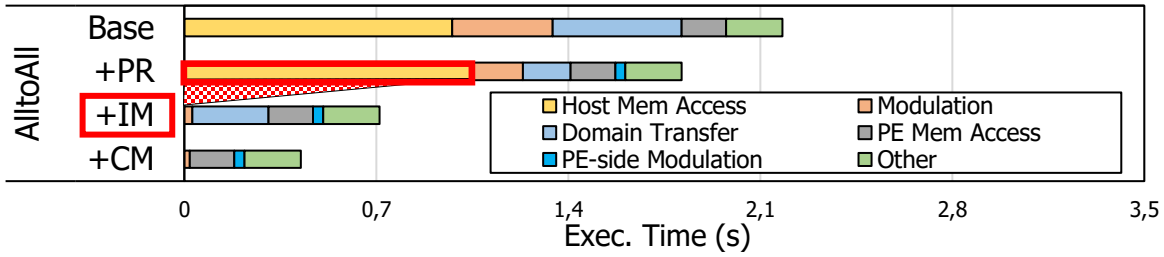
- Modulate data within the vector register

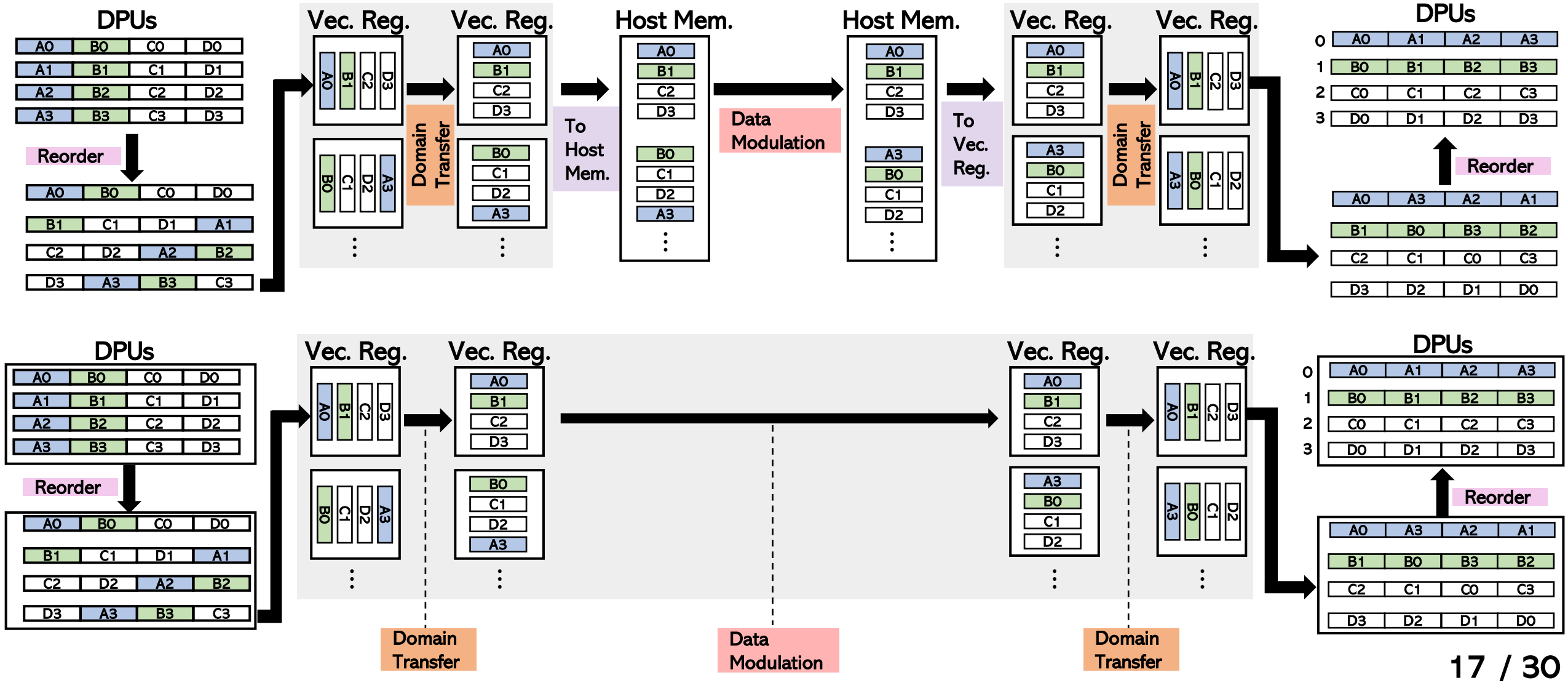




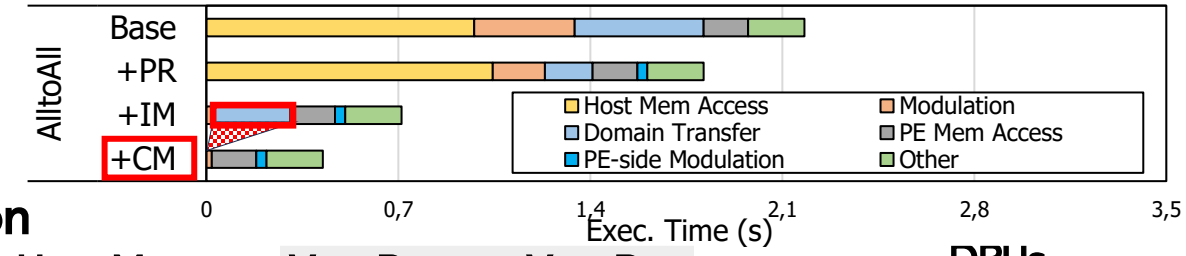
2. In-register Modulation

- Modulate data within the vector register

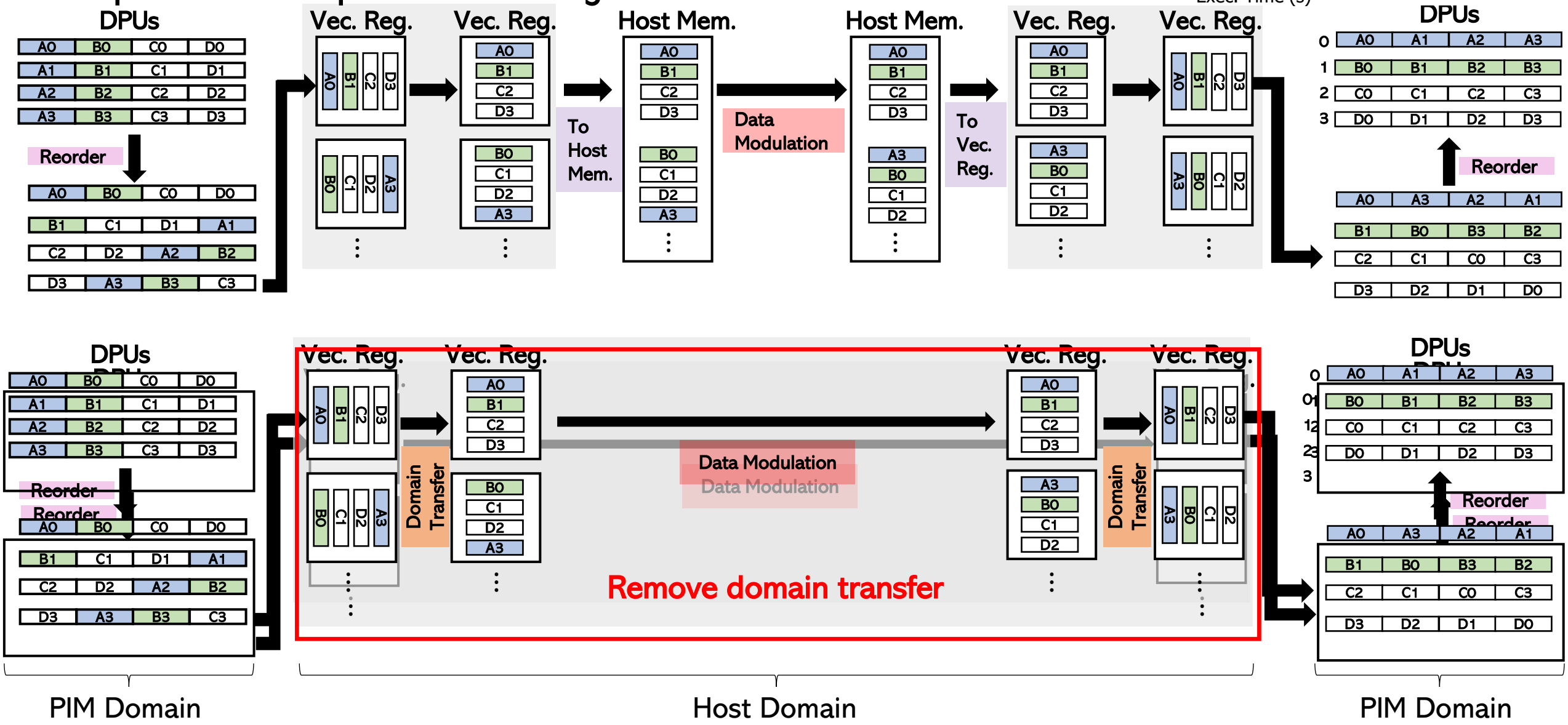




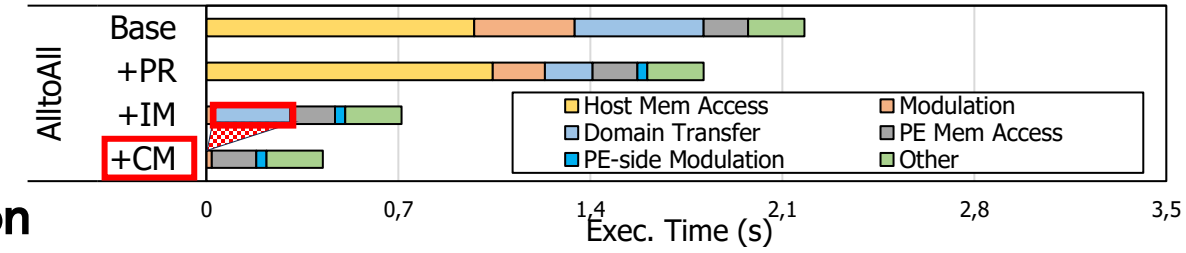
3. Cross-domain Modulation



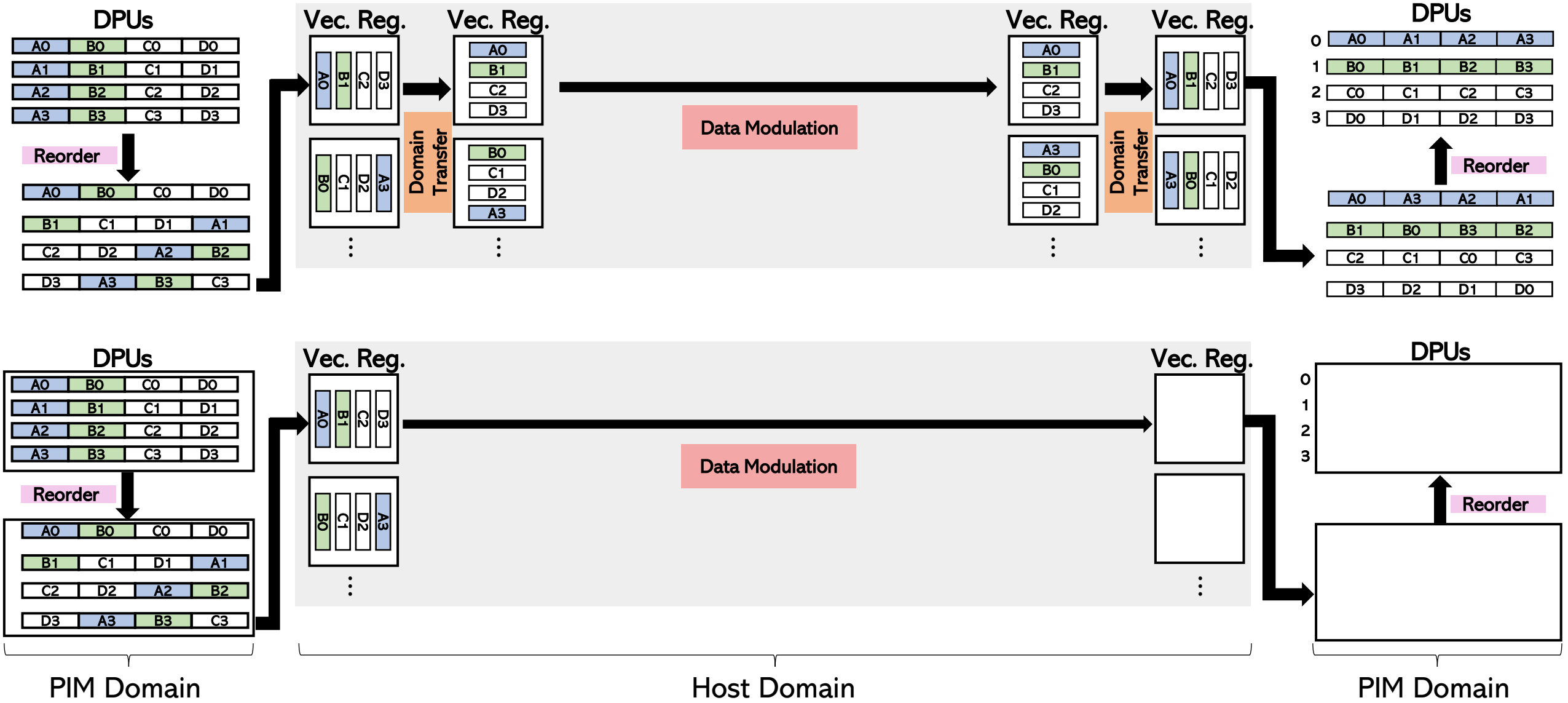
- Replace inefficient procedures with light bitwise rotation



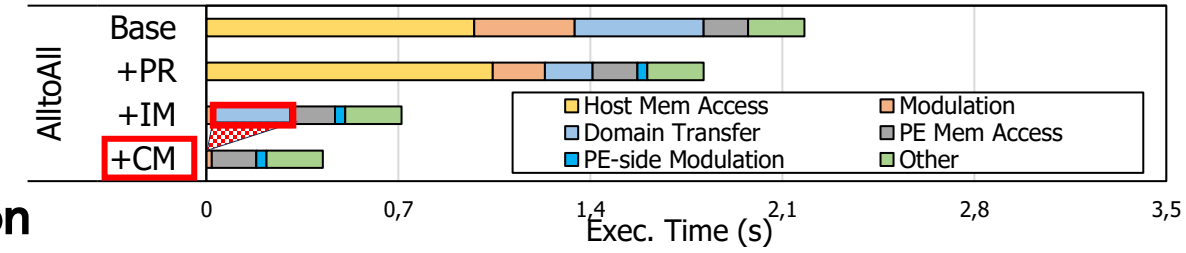
3. Cross-domain Modulation



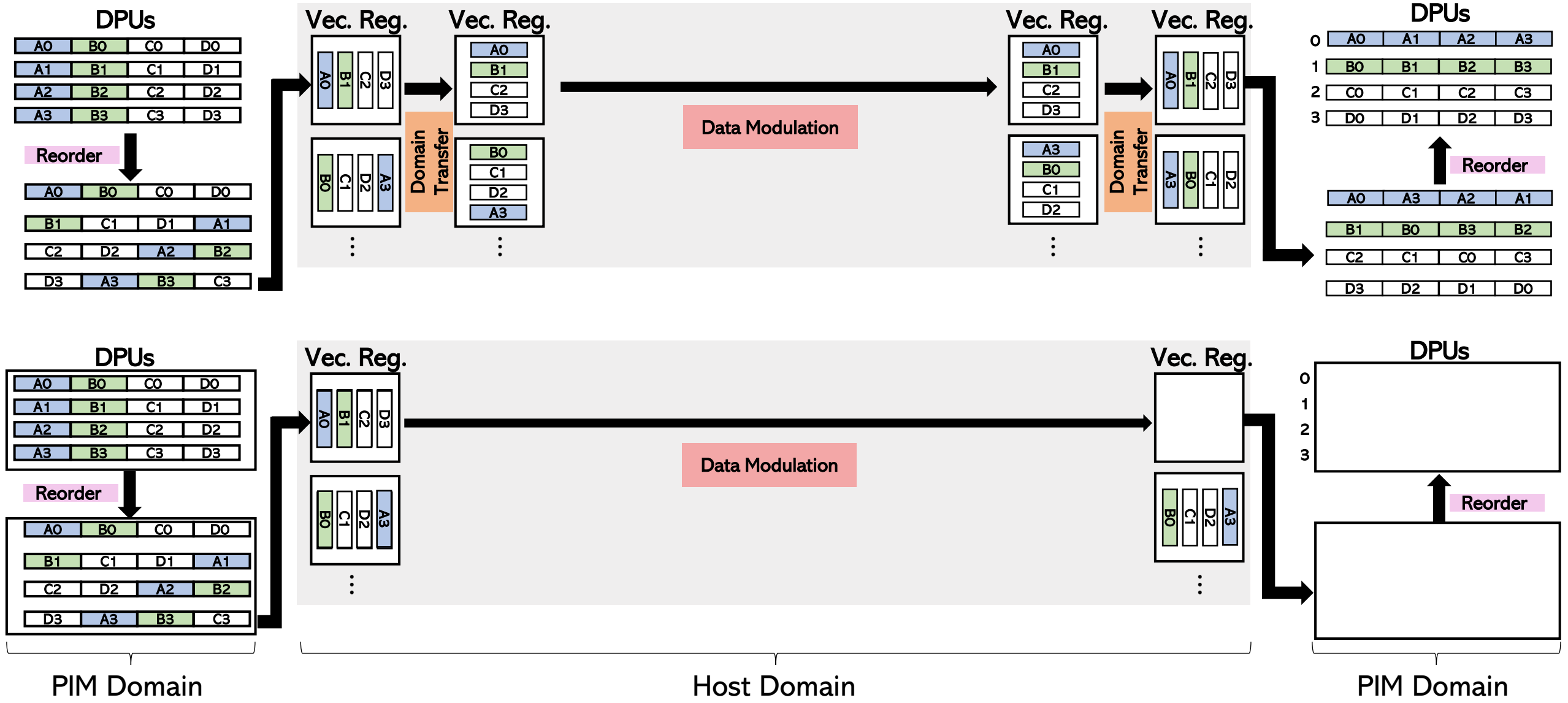
- Replace inefficient procedures with light bitwise rotation



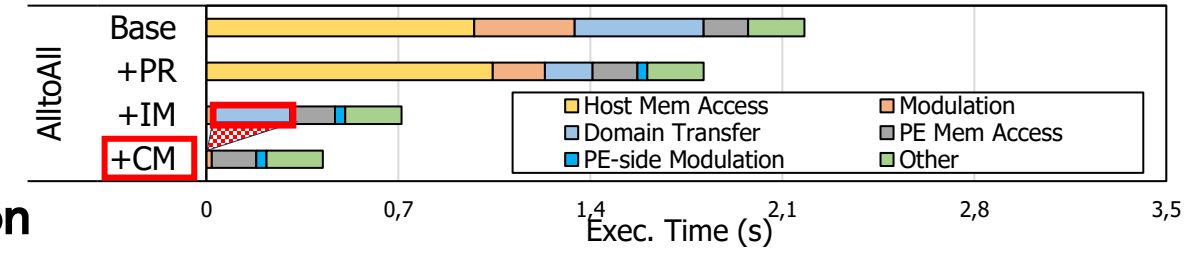
3. Cross-domain Modulation



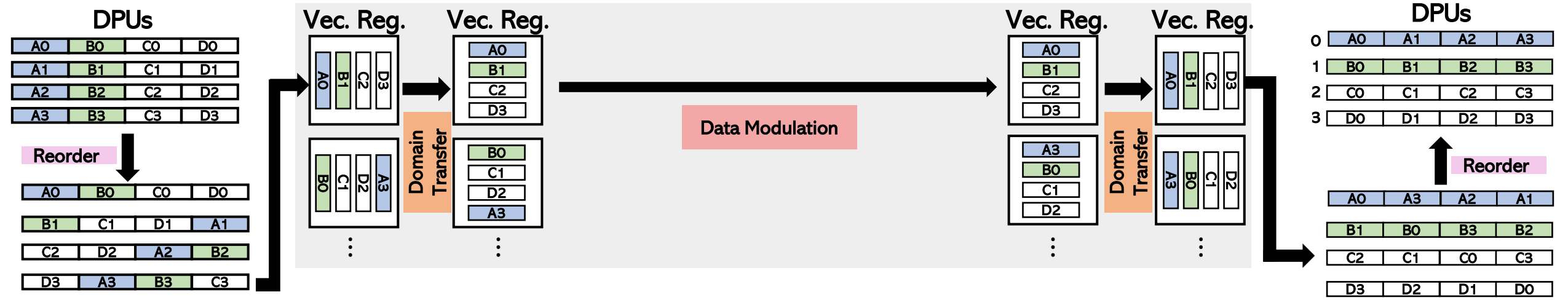
- Replace inefficient procedures with light bitwise rotation

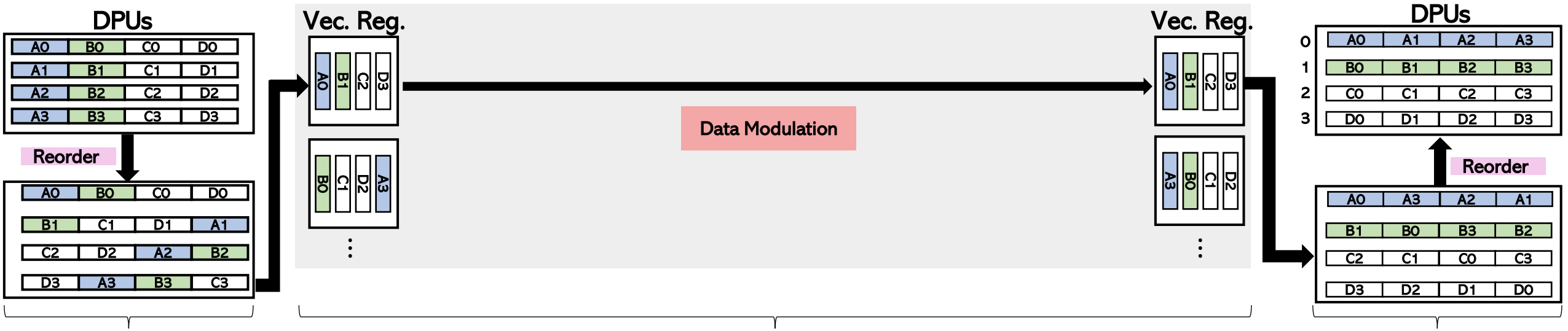


3. Cross-domain Modulation



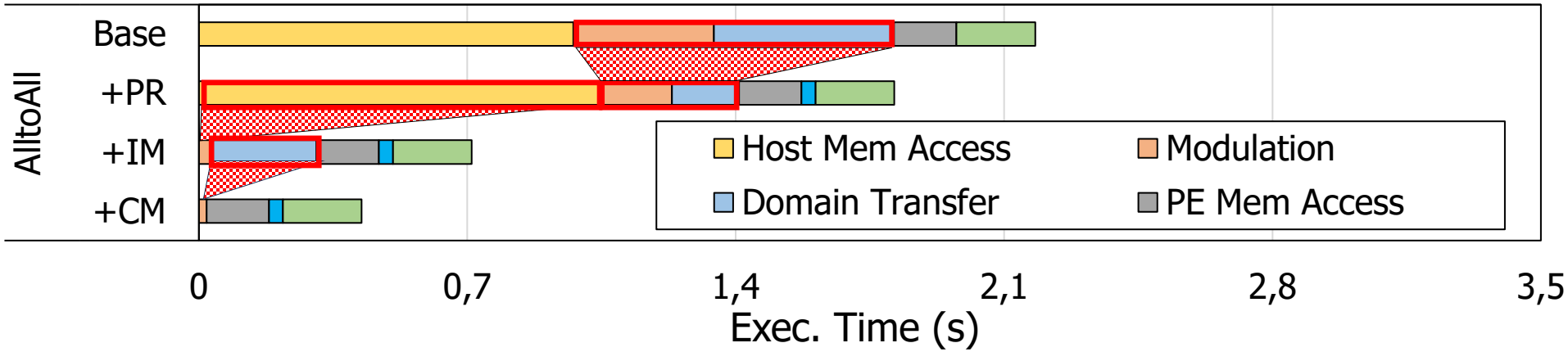
- Replace inefficient procedures with light bitwise rotation





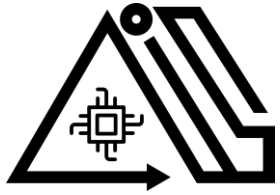
PID-Comm's Optimization

- **Three optimization techniques reduce their target bottleneck**
  - PIM-assisted Reordering (PR) targets data modulation
  - In-register Modulation (IM) targets host memory access
  - Cross-domain Modulation (CM) targets domain transfer

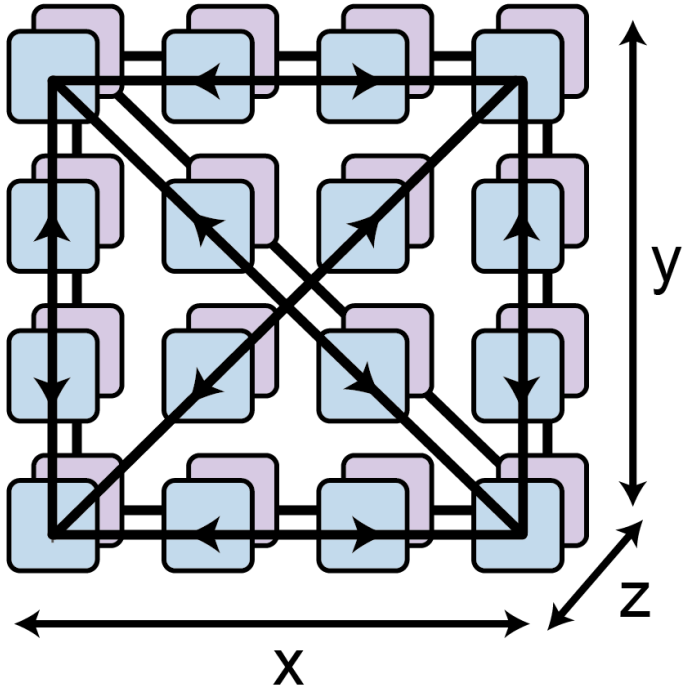


- **PID-Comm is fast but for practical use we need a model that supports**
  - Diverse Communication Groups
  - Multi-instance invocation

4. The Hypercube Model



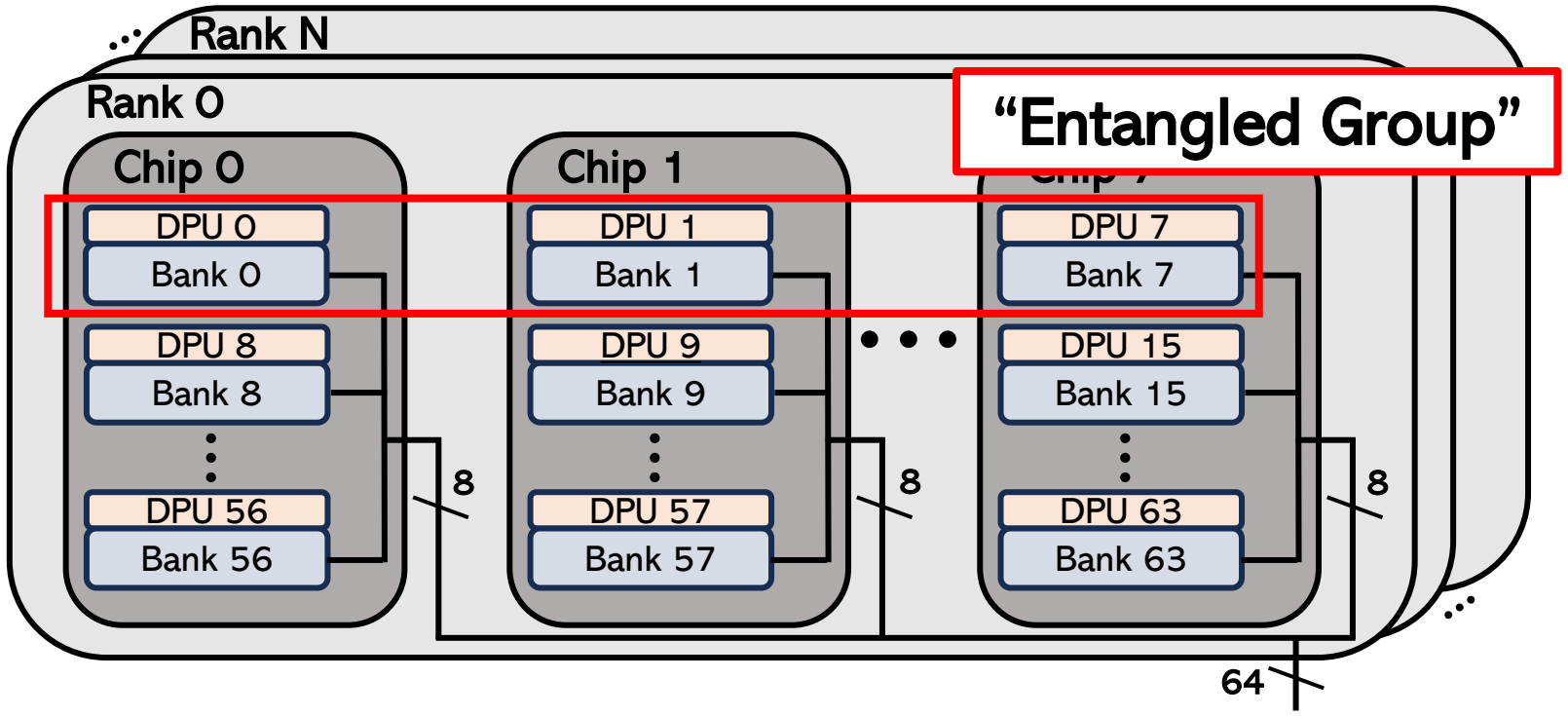
- Organize the DPUs into a virtual hypercube model
- DPUs form communication groups, communicate within them



Hypercube (4, 4, 2)

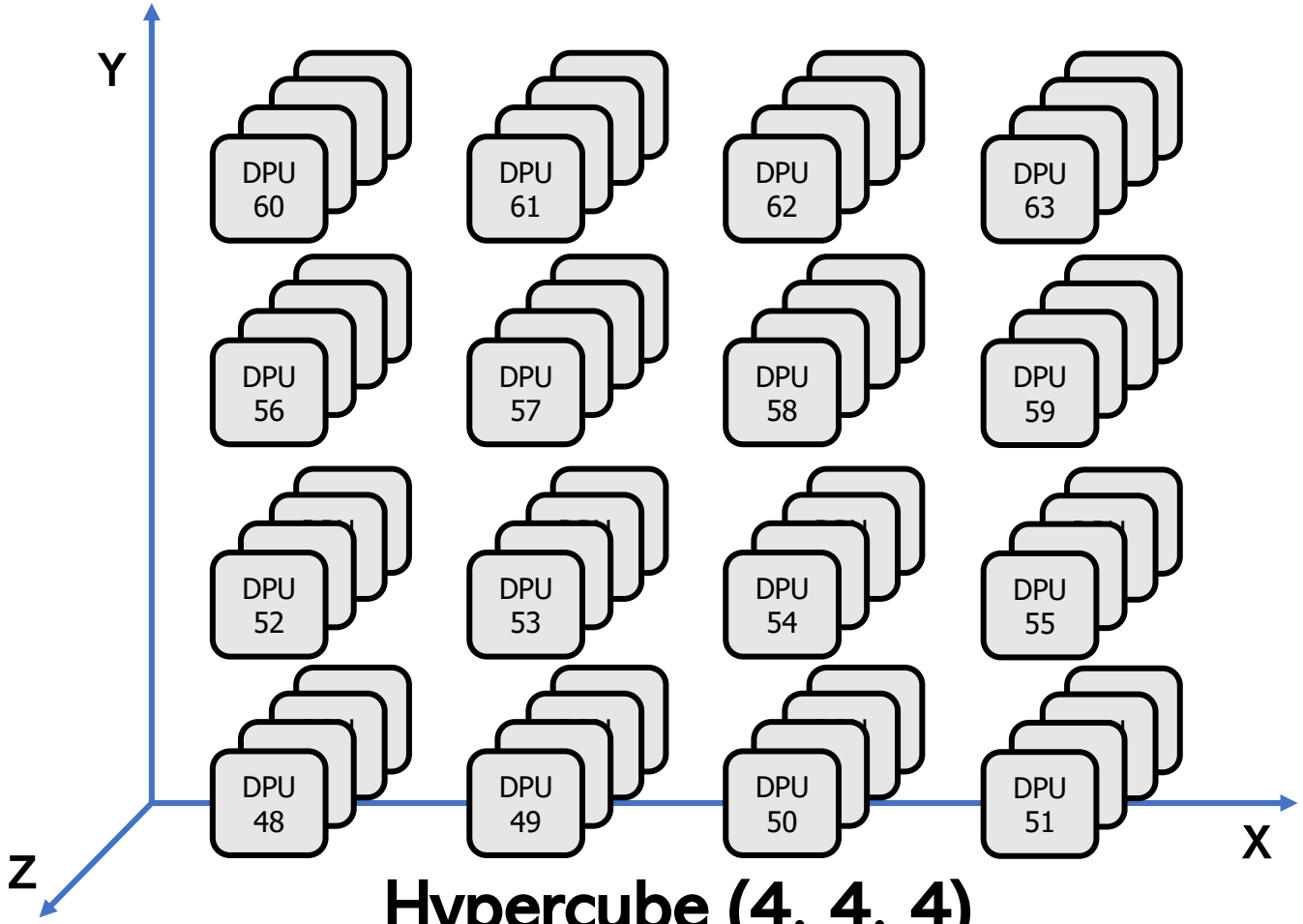
4. The Hypercube Model

- **Banks in an entangled group are always accessed together at once**
  - Entangled groups as building blocks for the hypercube



4. The Hypercube Model

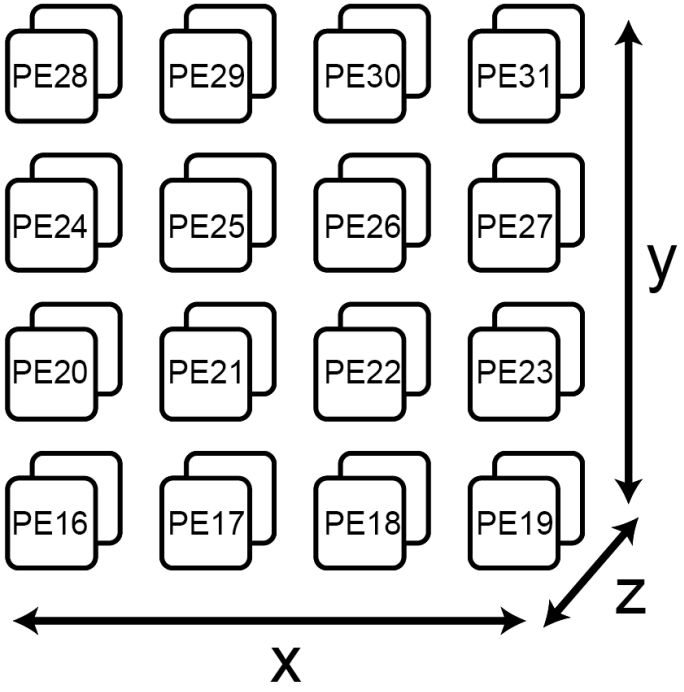
- **Banks in an entangled group are always accessed together at once**
  - Entangled groups as building blocks for the hypercube



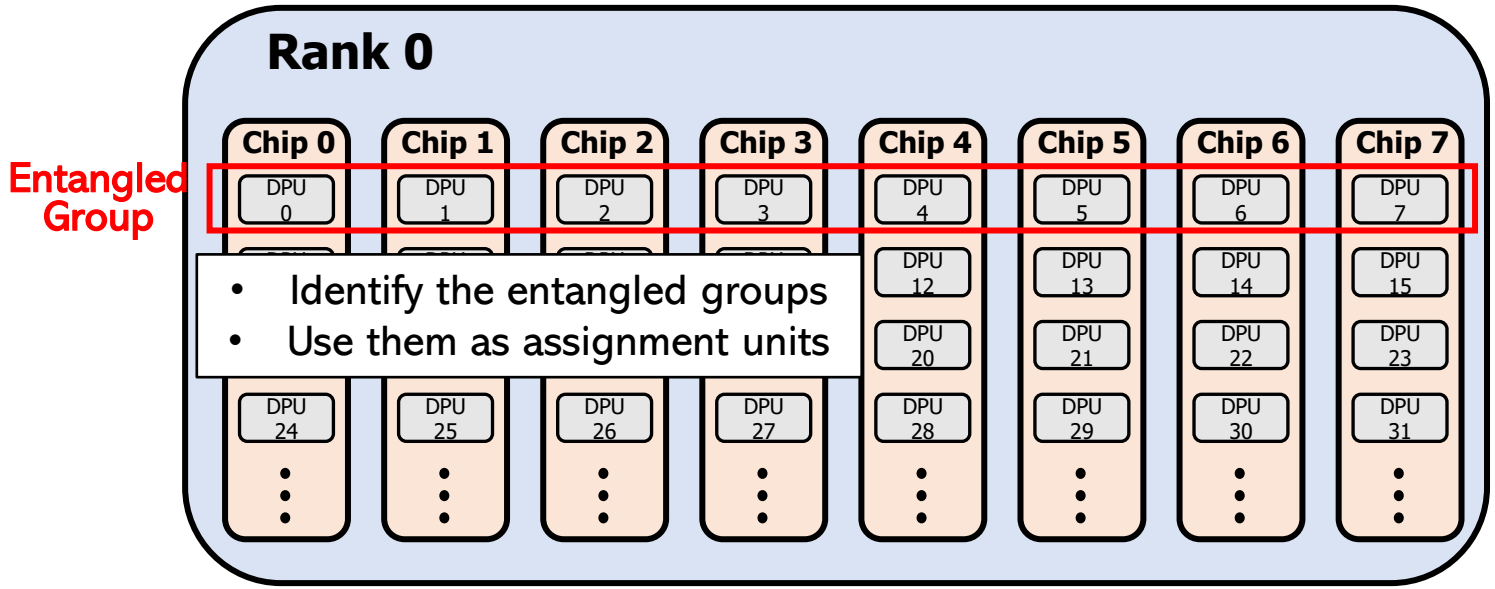
Hypercube (4, 4, 4)

4. The Hypercube Model

- Maintaining both optimal performance and high flexibility
- Mapping virtual hypercube to physical banks

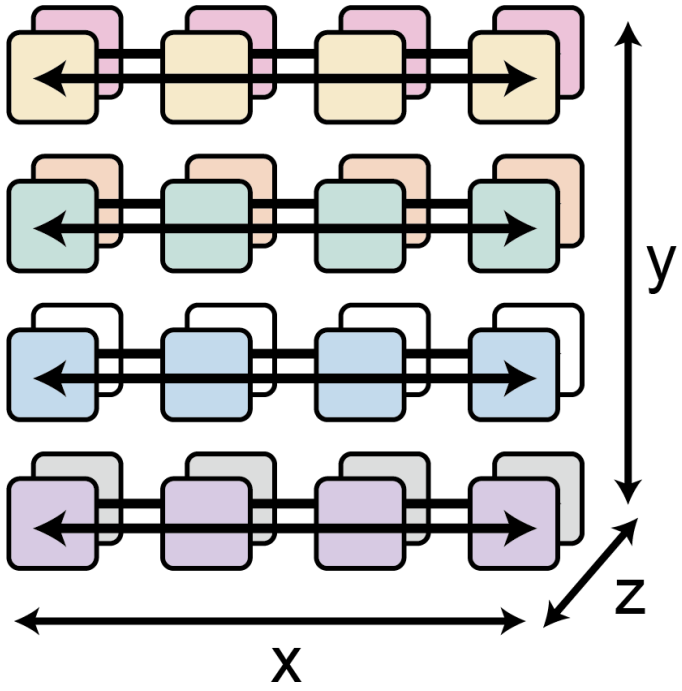


Hypercube (4, 4, 2)

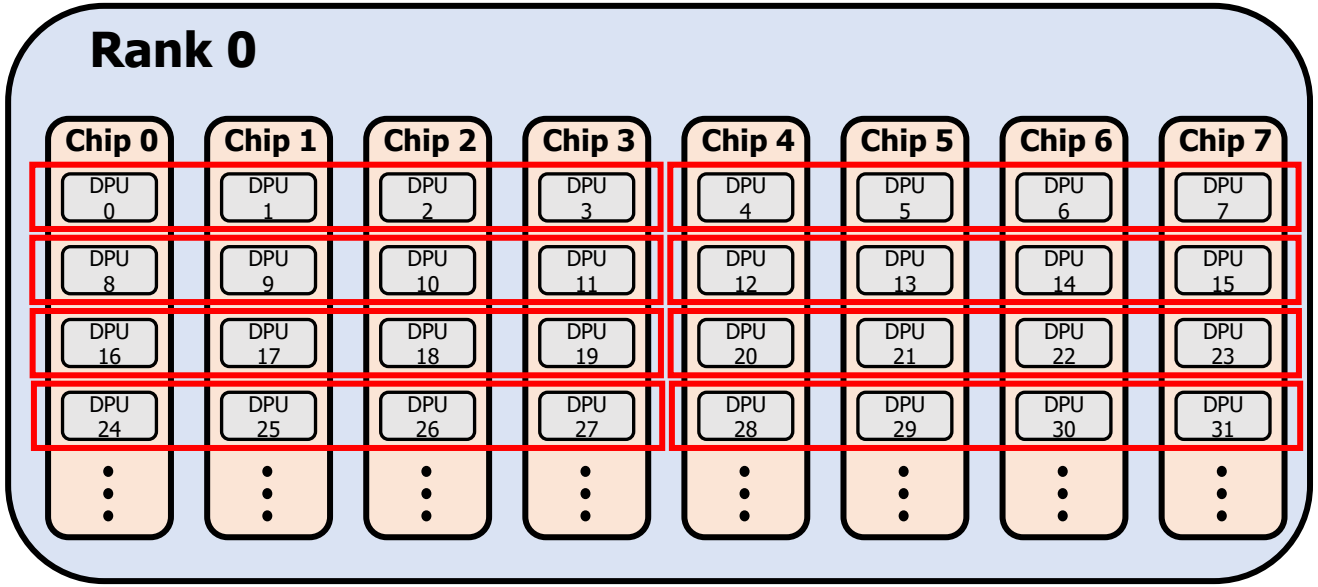


4. The Hypercube Model

- Allows multiple communication invocations
- Support diverse communication groups



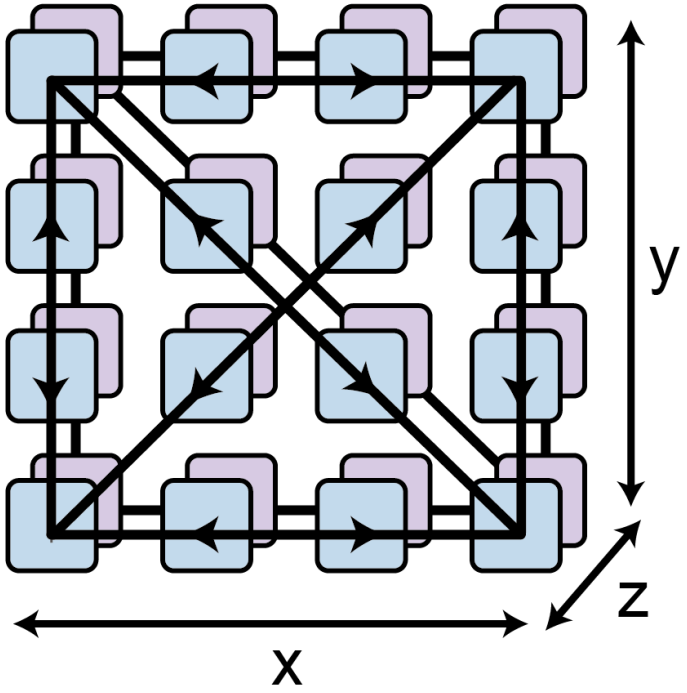
Hypercube (4, 4, 2)



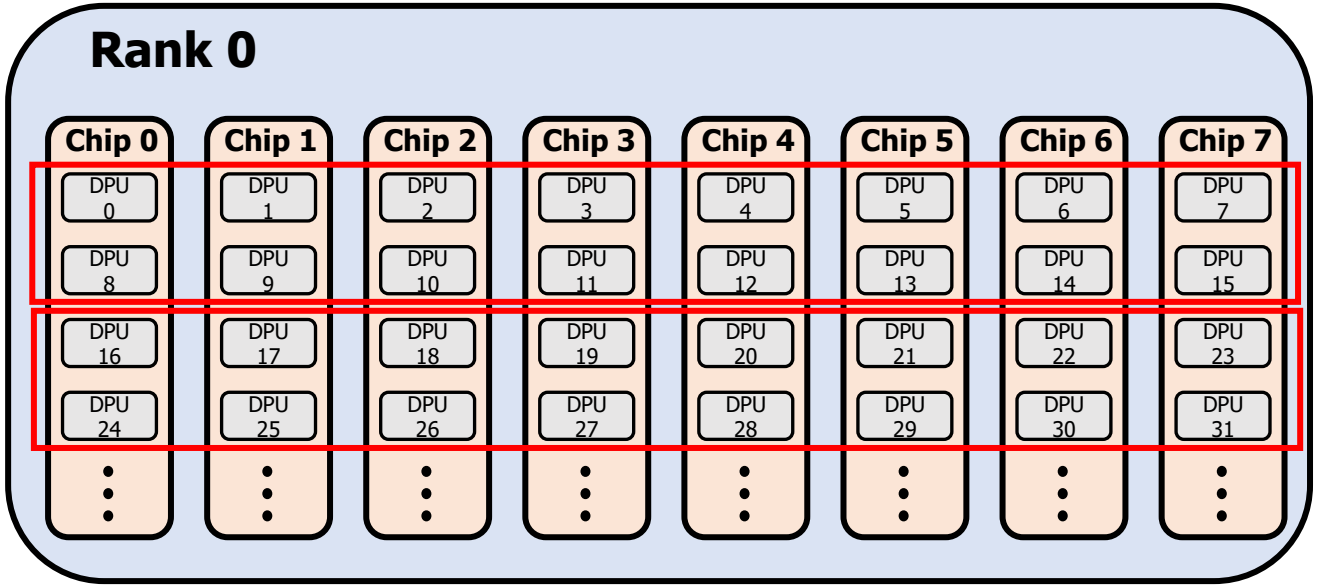
Communication group configuration.

4. The Hypercube Model

- Allows multiple communication invocations
- Support diverse communication groups



Hypercube (4, 4, 2)



Communication group configuration.

PID-Comm Library

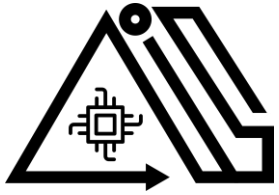
- Using PID-Comm, we can compress complex implementations into just a single line of code.

```c
//Receive the data for each DPU
DPU_FOREACH_ENTANGLED_GROUP(dpu_set, dpu, each_dpu, nr_dpus){
    DPU_ASSERT(dpu_prepare_xfer(dpu, original_data+each_dpu*data_num_per_dpu));
}
DPU_ASSERT(dpu_push_xfer(dpu_set, DPU_XFER_FROM_DPU, DPU_MRAM_HEAP_POINTER_NAME, 0,\
                                    data_size_per_dpu, DPU_XFER_DEFAULT));

//Perform AllReduce on the host CPU
uint32_t *conv_data = (uint32_t*)calloc(data_num_per_dpu*nr_dpus, sizeof(uint32_t));
#pragma omp parallel for collapse(2)
for(uint32_t jk=0; jk<axis_len[1]*axis_len[2]; jk++){
    for(uint32_t dst_dpu=0; dst_dpu<axis_len[0]; dst_dpu++){
        for(uint32_t src_dpu=0; src_dpu<axis_len[0]; src_dpu++){
            uint32_t dst_dpu_id = jk * axis_len[0] + dst_dpu;
            uint32_t src_dpu_id = jk * axis_len[0] + src_dpu;

            for(uint32_t data_index=0; data_index<data_num_per_dpu; data_index++){
                conv_data[(dst_dpu_id * data_num_per_dpu + data_index)] += \
                original_data[(src_dpu_id * data_num_per_dpu + data_index)];
            }
        }
    }
}

//Send data to each DPU
DPU_FOREACH_ENTANGLED_GROUP(dpu_set, dpu, each_dpu, nr_dpus){
    DPU_ASSERT(dpu_prepare_xfer(dpu, conv_data+each_dpu*data_num_per_dpu));
}
DPU_ASSERT(dpu_push_xfer(dpu_set, DPU_XFER_TO_DPU, DPU_MRAM_HEAP_POINTER_NAME, 0,\
                                    data_size_per_dpu, DPU_XFER_DEFAULT));
```

Conventional AllReduce Implementation.

```c
//Perform AllReduce by utilizing PID-Comm
pidcomm_all_reduce(hypercube_manager, "100", data_size_per_dpu, \
        start_offset, target_offset, buffer_offset, sizeof(T), 0);
```

PID-Comm AllReduce Implementation.



5. Evaluation

- Environment
- Results

Environment

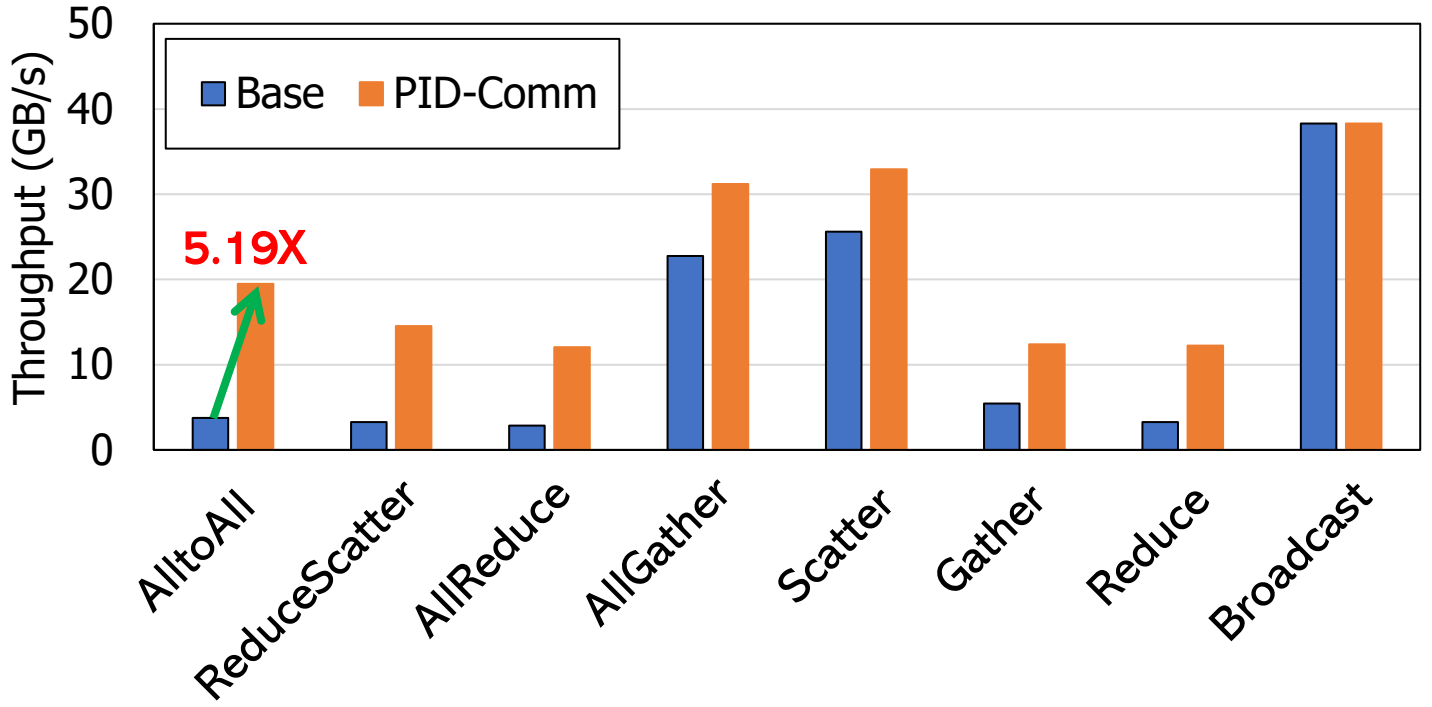
- **Experimental Setup**
  - Intel Xeon Gold 5125 CPU (Double socket, 10 cores each)
  - 4 Channels of UPMEM DIMMs (1024 DPUs) on a **REAL SYSTEM**
- **Benchmark Applications**

- **Baseline**
  - SimplePIM (for AllGather, AllReduce, Scatter, Gather, Broadcast)
  - UPMEM SDK based implementation (for AlltoAll, ReduceScatter, Reduce)

App.	Hyper. Dim.	Communication Primitives							
		AlltoAll	Reduce Scatter	All Reduce	All Gather	Scatter	Gather	Reduce	Broad cast
DLRM	3	✓	✓			✓	✓		✓
GNN	2			✓	✓	✓	✓		
BFS	1			✓		✓		✓	
CC	1			✓		✓		✓	
MLP	1		✓			✓		✓	

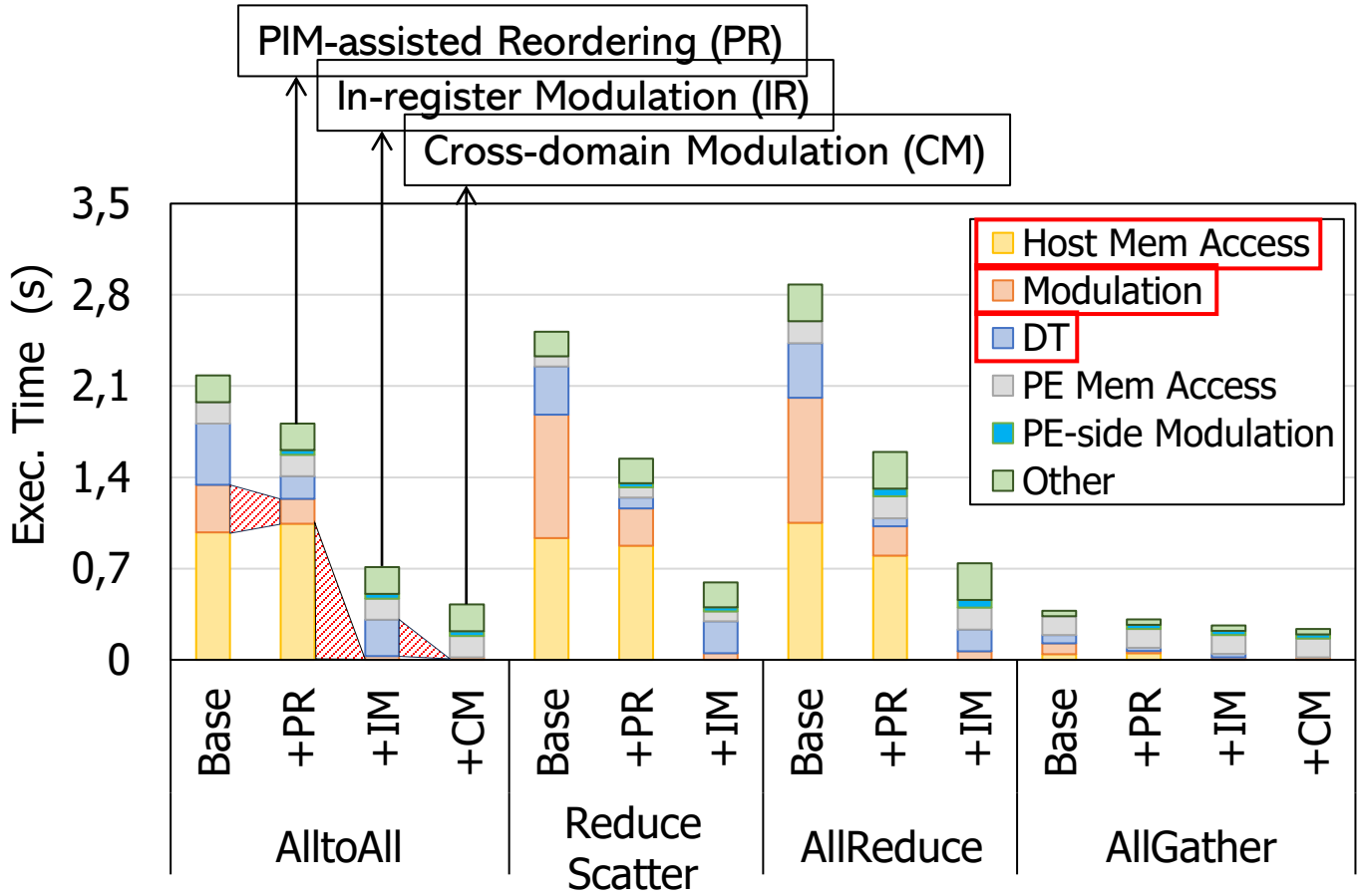
Performance of Primitives

- Up to 5.19x higher throughput compared to PIM baseline
- Geomean Speedup of 2.83X



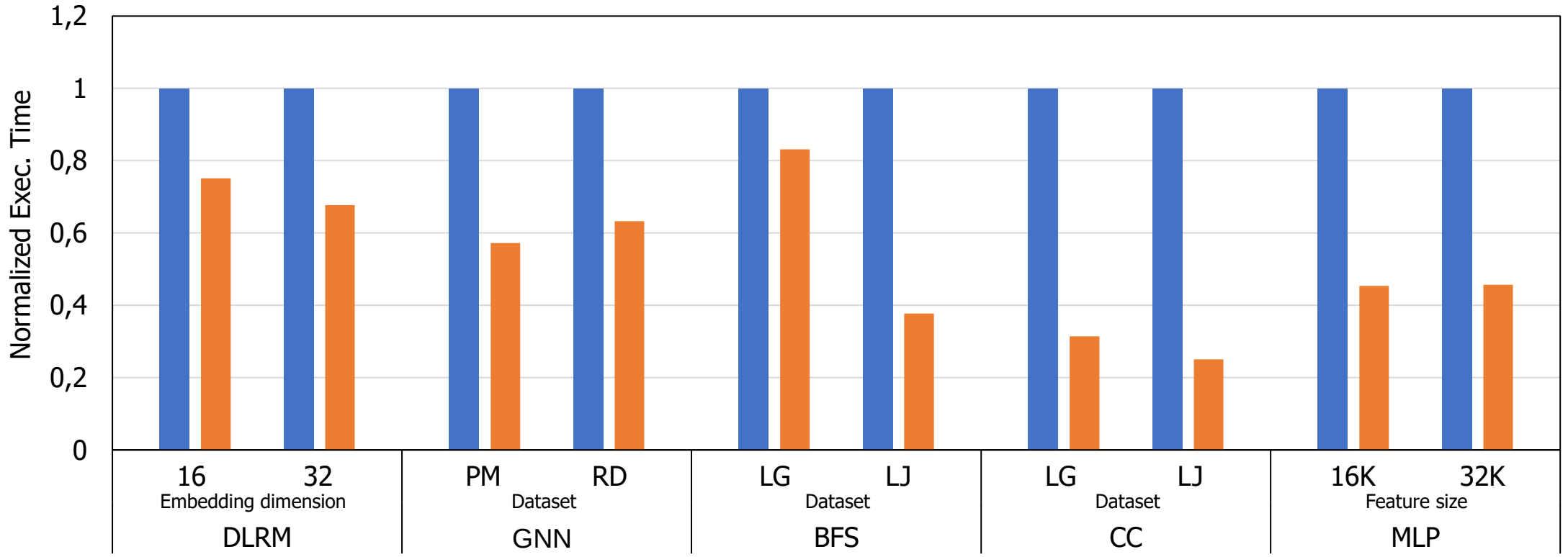
Ablation Study & Breakdown

- Average speedup of 1.48x, 2.03x, and 1.42x for PIM-assisted Reordering (+PR), In-register Modulation(+IM), and Cross-domain Modulation (+CM)



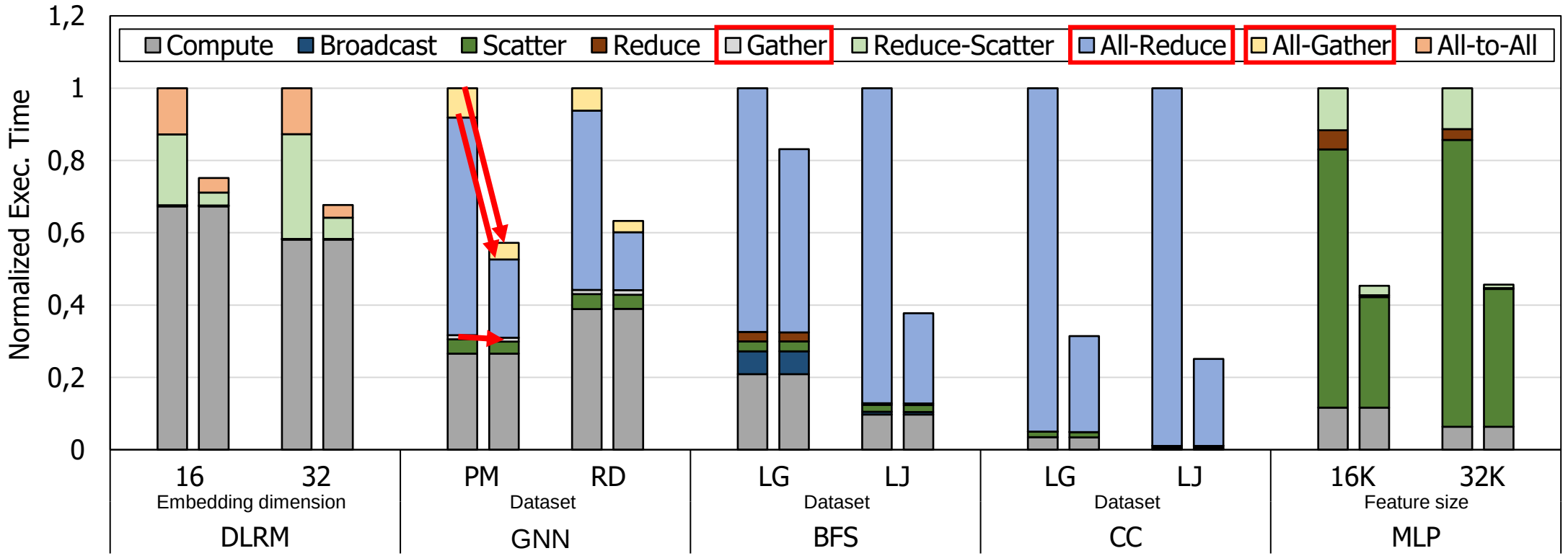
Benchmark Applications

- Evaluated on different datasets / embedding dimensions / feature sizes
- Up to 3.99x speedup compared to conventional communication schemes
- Geo-mean speedup of 1.99x

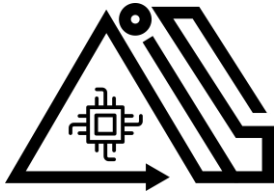


Benchmark Applications

- Evaluated on different datasets / embedding dimensions / feature sizes
- Up to 3.99x speedup compared to conventional communication schemes
- Geo-mean speedup of 1.99x



6. Conclusion



PID-Comm is..

1. **The 1st full-fledged collective communication library for PIM-enabled DIMMs**
   - Supports 8 types of communication primitives (sharing the scope of NCCL)

2. **Provides**
   - Micro-level optimizations to accelerate inter-DPU communications
   - A hypercube communication model for flexible communications

3. **Primitives outperform PIM baseline by 2.83x in geomean**

4. **Open source: https://github.com/AIS-SNU/PID-Comm**

Thank you.

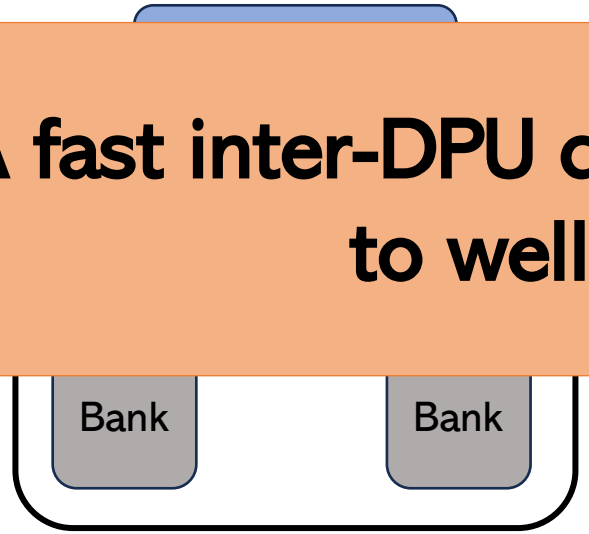
Si Ung Noh @ Seoul National University

Email: siung98@snu.ac.kr

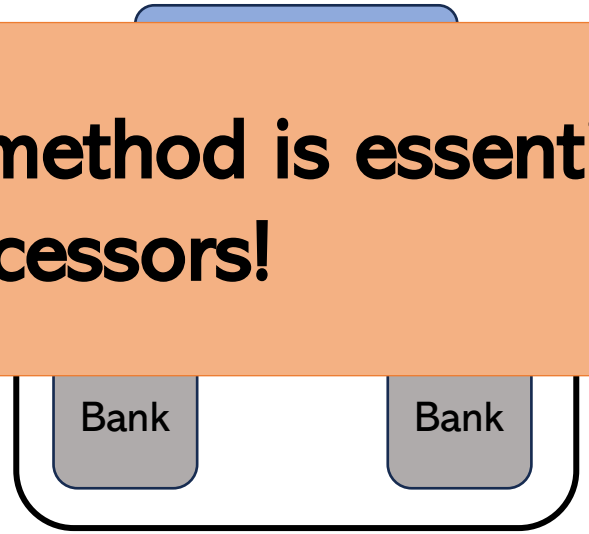
Inter-DPU Communications

- No direct path between each DRAM processing elements (DPUs)
- Host processor becomes the medium for inter-DPU communication
- Inter-DPU communications are becoming the bottleneck of major applications

A fast inter-DPU communication method is essential to well-utilize PIM processors!



Lack of a direct path between DPUs



Existing path passes the host processor