GraphP: Reducing Communication for PIM-based Graph Processing with Efficient Data Partition

Mingxing Zhang, **Youwei Zhuo (equal contribution)**,
Chao Wang, Mingyu Gao, Yongwei Wu, Kang Chen,
Christos Kozyrakis, Xuehai Qian

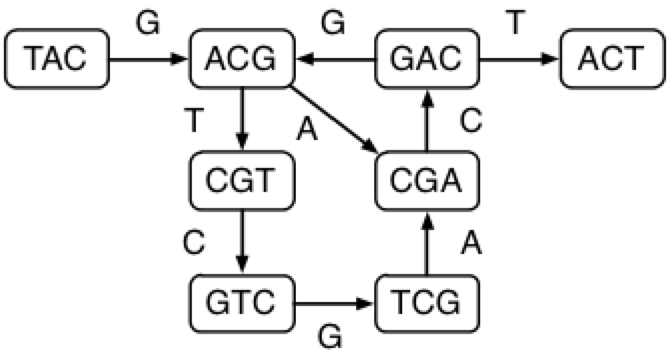
Tsinghua University
University of Southern California
Stanford University

Outline

- **Motivation**
  - Graph applications
  - Processing-In-Memory
  - The drawbacks of the current solution
- GraphP
- Evaluation

Graph Applications

- Social network analytics
- Recommendation system
- Bioinformatics
- …

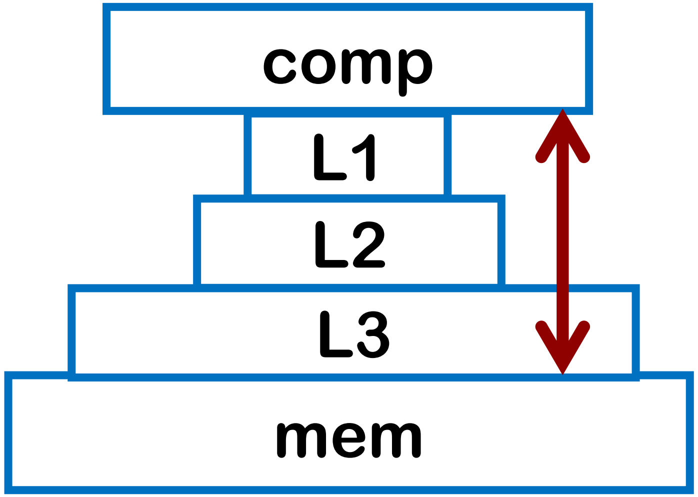


Challenges

- High bandwidth requirement
  - Small amount of computation per vertex
  - Data movement overhead

Challenges

- High bandwidth requirement
  - Small amount of computation per vertex
  - Data movement overhead



PIM: Processing-In-Memory

PIM: Processing-In-Memory

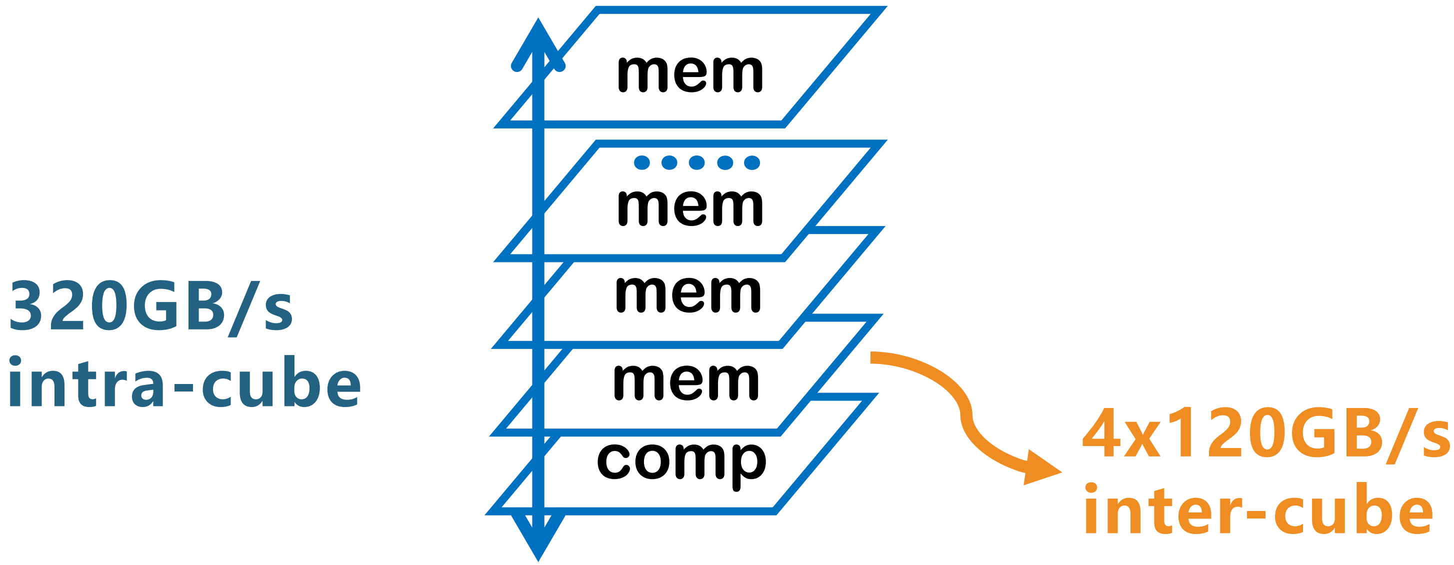
- **Idea**: Computation logic inside memory
- **Advantage**: High memory bandwidth

PIM: Processing-In-Memory

- **Idea**: Computation logic inside memory
- **Advantage**: High memory bandwidth
- **Example**: Hybrid Memory Cubes (HMC)

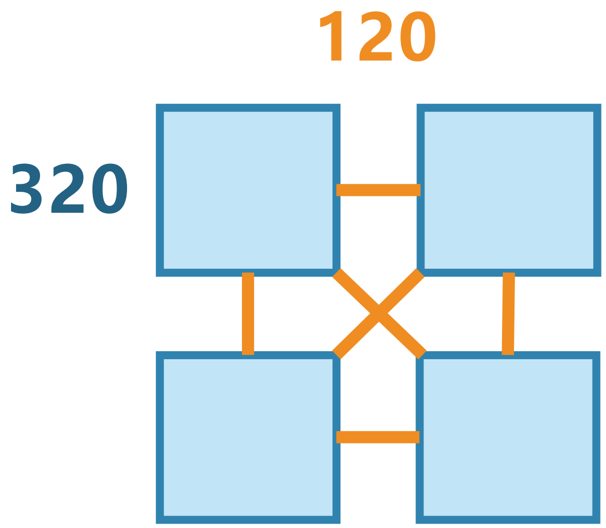
PIM: Processing-In-Memory

- **Idea**: Computation logic inside memory
- **Advantage**: High memory bandwidth
- **Example**: Hybrid Memory Cubes (HMC)



HMC: Hybrid Memory Cubes

HMC: Hybrid Memory Cubes

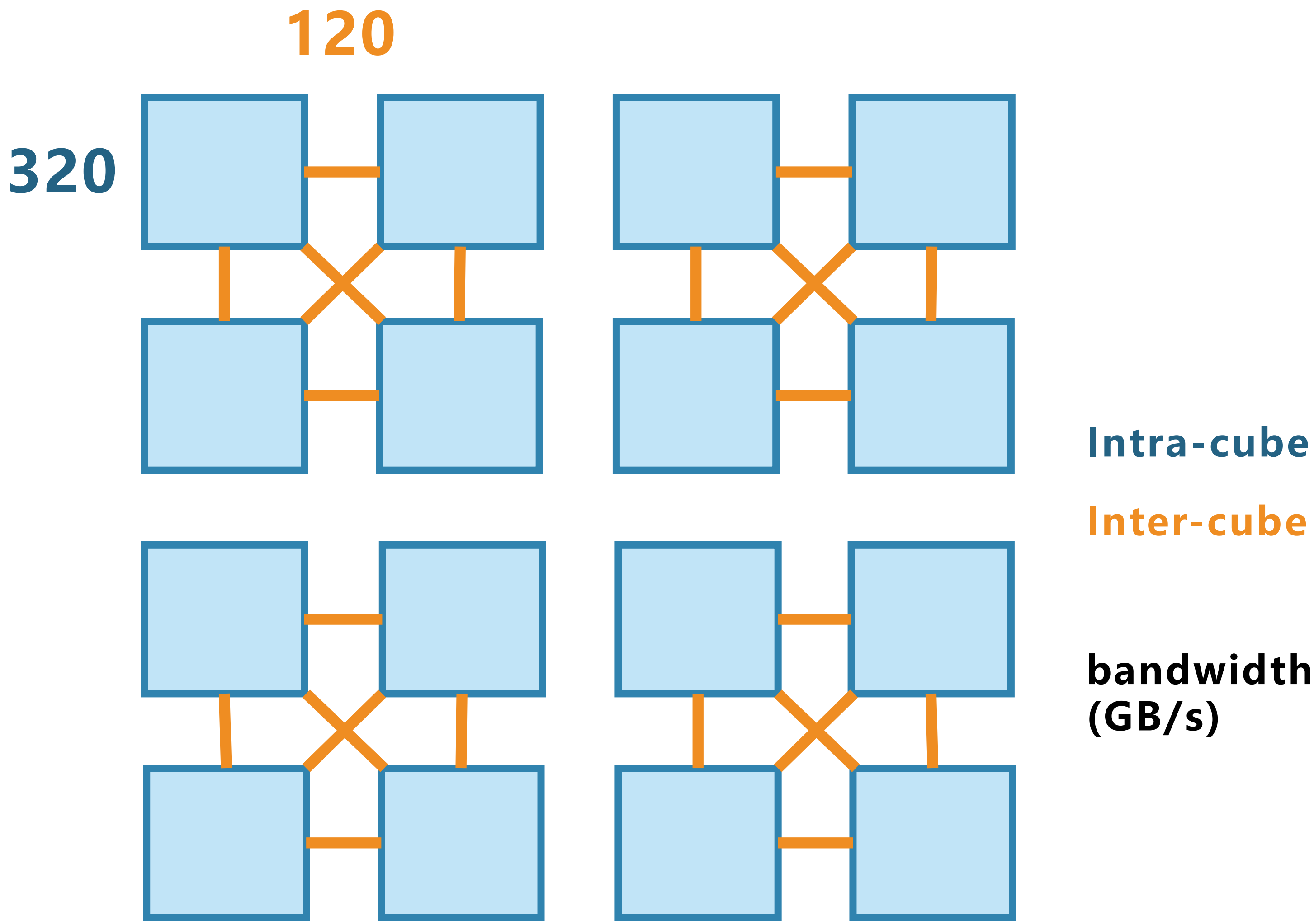


Intra-cube

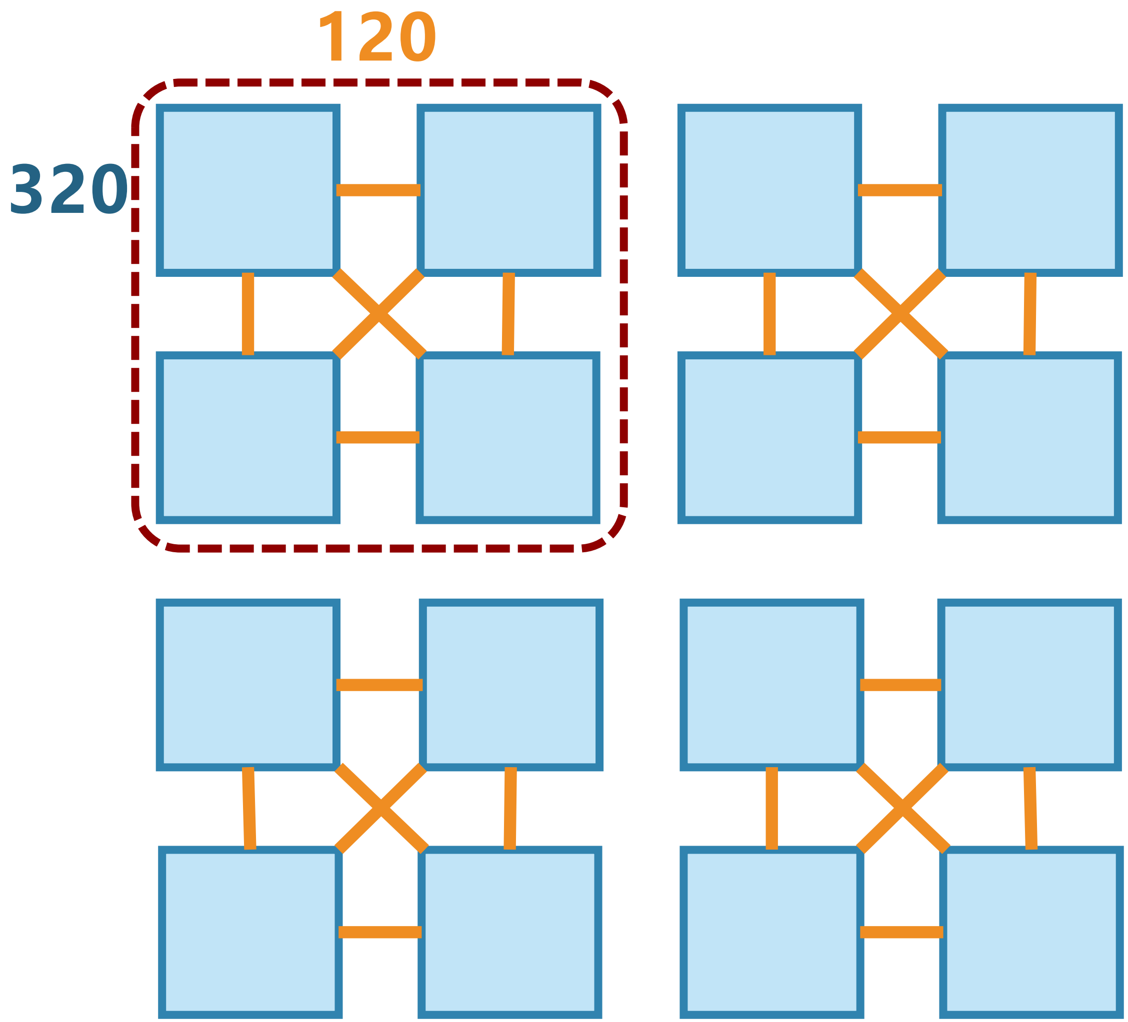
Inter-cube

bandwidth (GB/s)

HMC: Hybrid Memory Cubes



HMC: Hybrid Memory Cubes

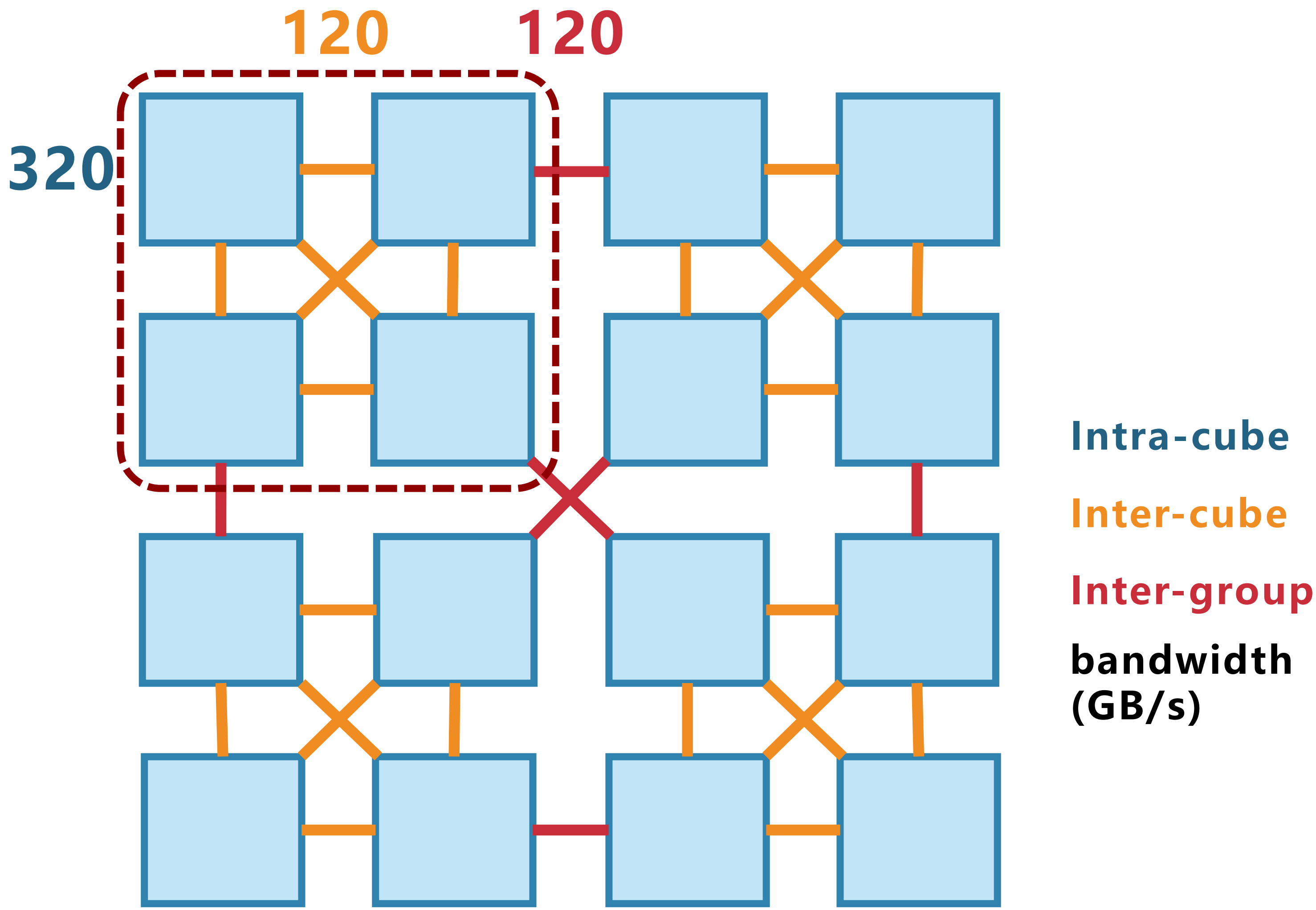


Intra-cube

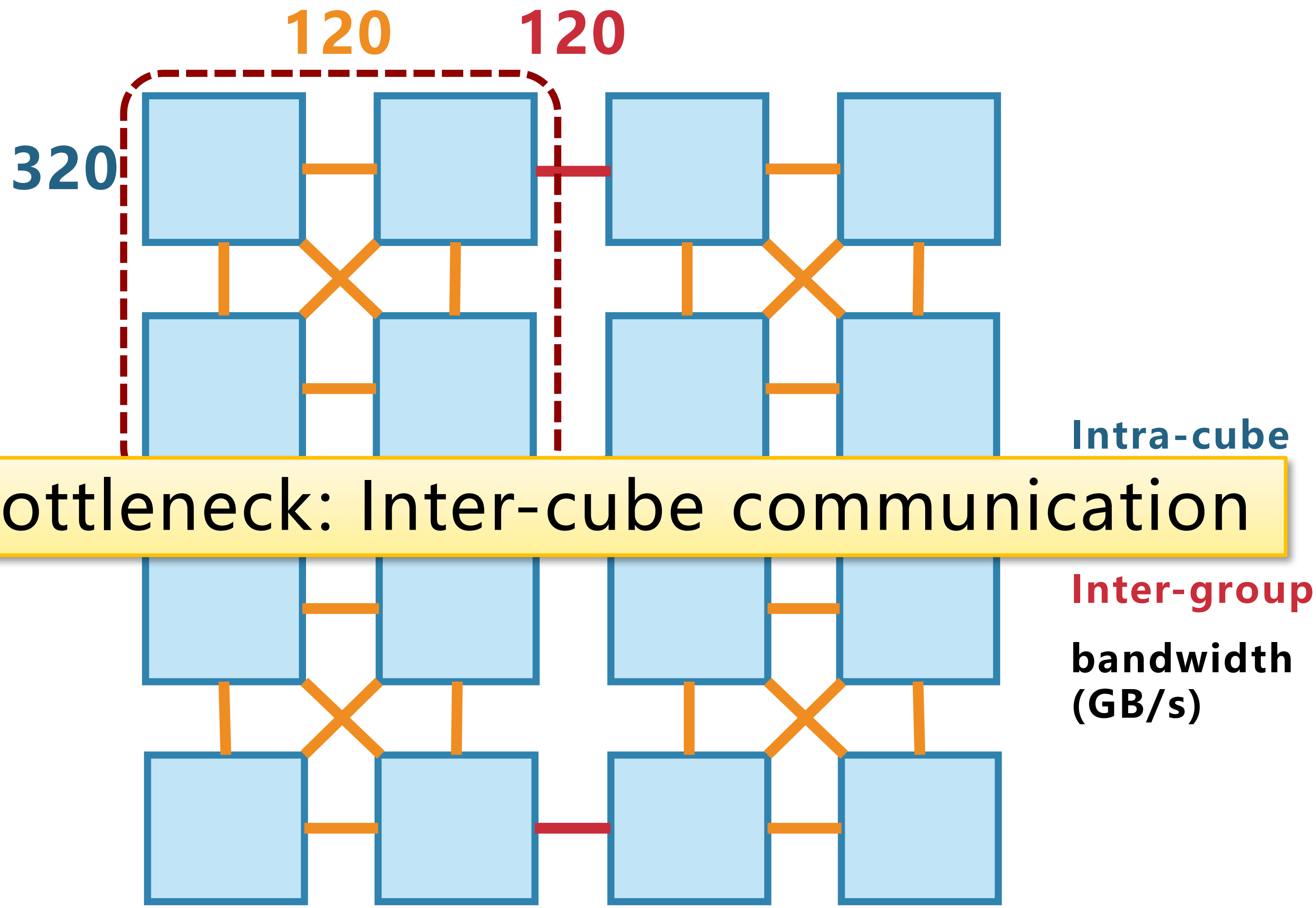
Inter-cube

bandwidth (GB/s)

HMC: Hybrid Memory Cubes



HMC: Hybrid Memory Cubes



Bottleneck: Inter-cube communication

Outline

- **Motivation**
  - Graph applications
  - Processing-In-Memory
  - **The drawbacks of the current solution**
- GraphP
- Evaluation

Current Solution: Tesseract

- First PIM-based graph processing architecture

Ahn, J., Hong, S., Yoo, S., Mutlu, O., & Choi, K. A scalable processing-in-memory accelerator for parallel graph processing. In *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*.

Current Solution: Tesseract

- First PIM-based graph processing architecture
- **Programming model**
  - Vertex program

Ahn, J., Hong, S., Yoo, S., Mutlu, O., & Choi, K. A scalable processing-in-memory accelerator for parallel graph processing. In *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*.

Current Solution: Tesseract

- First PIM-based graph processing architecture
- **Programming model**
  - Vertex program
- **Partition**
  - Based on vertex program

Ahn, J., Hong, S., Yoo, S., Mutlu, O., & Choi, K. A scalable processing-in-memory accelerator for parallel graph processing. In *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*.

PageRank in Vertex Program

```
for (v: vertices) {
    update = 0.85 * v.rank / v.out_degree;
}
```

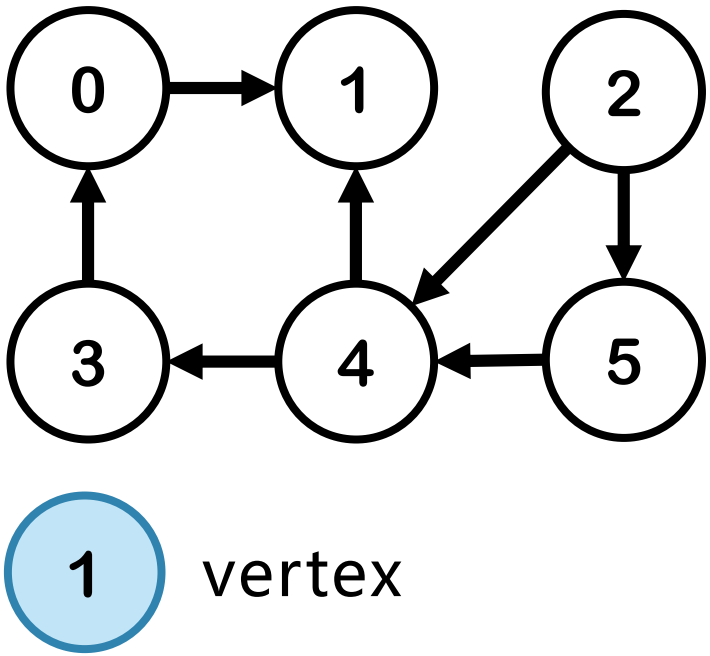
PageRank in Vertex Program

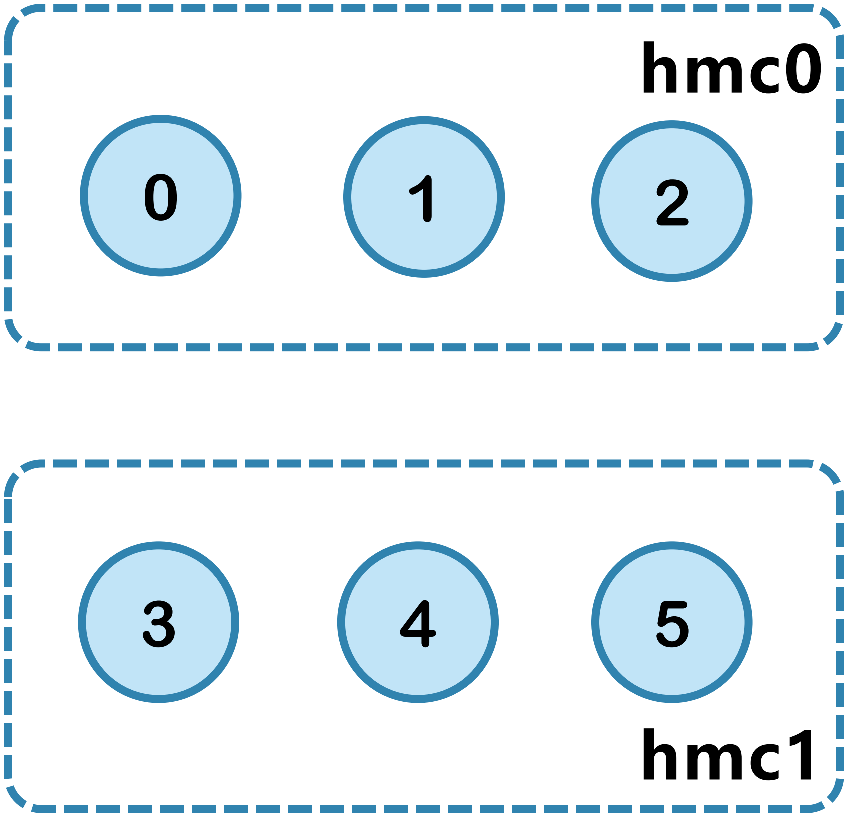
```
for (v: vertices) {
    update = 0.85 * v.rank / v.out_degree;
    for (w: edges.destination) {
        put(w.id, function{ w.next_rank += update; });
    }
}
```

PageRank in Vertex Program

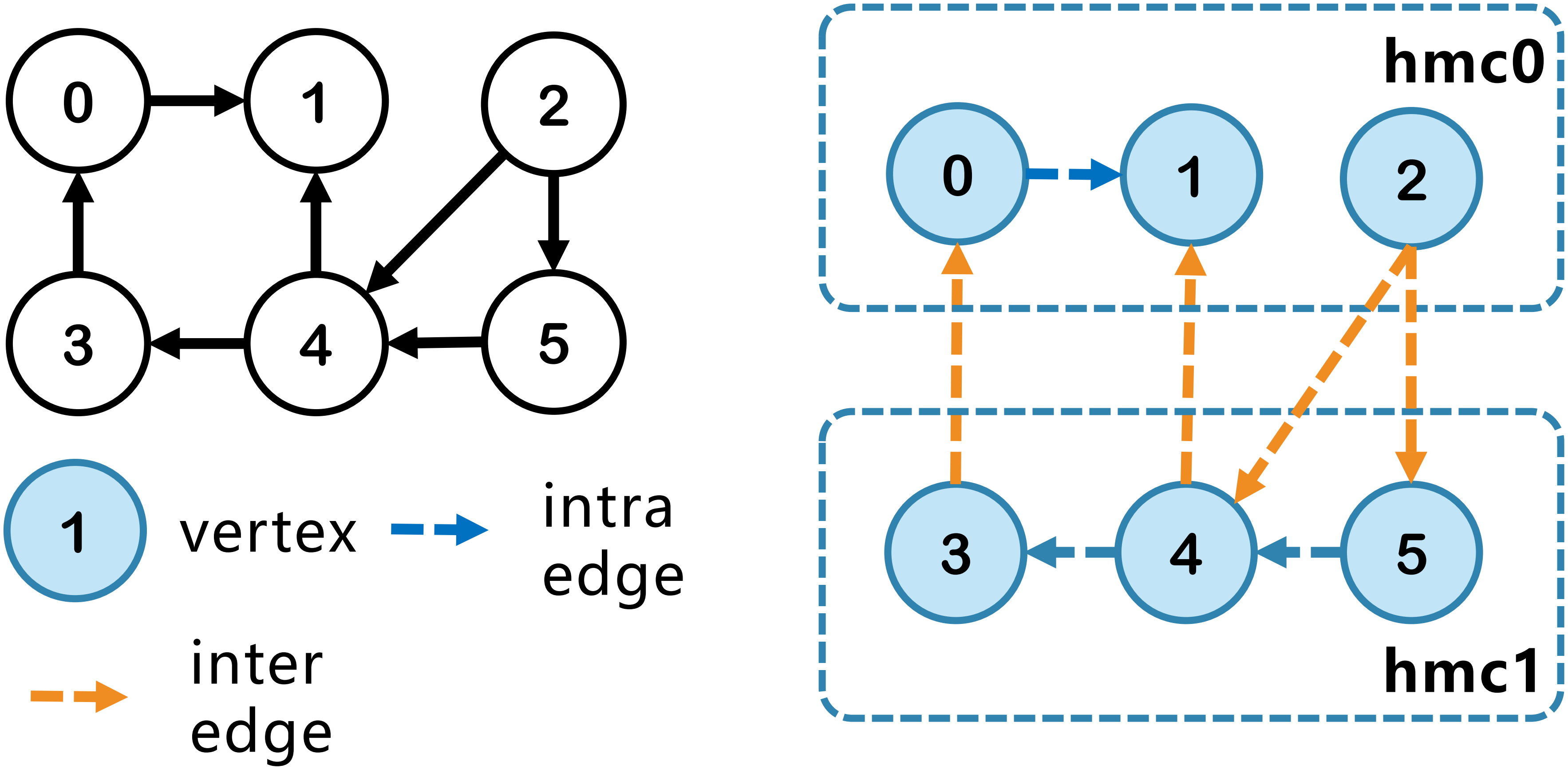
```
for (v: vertices) {
    update = 0.85 * v.rank / v.out_degree;
    for (w: edges.destination) {
        put(w.id, function{ w.next_rank += update; });
    }
}
barrier();
```

Graph Partition

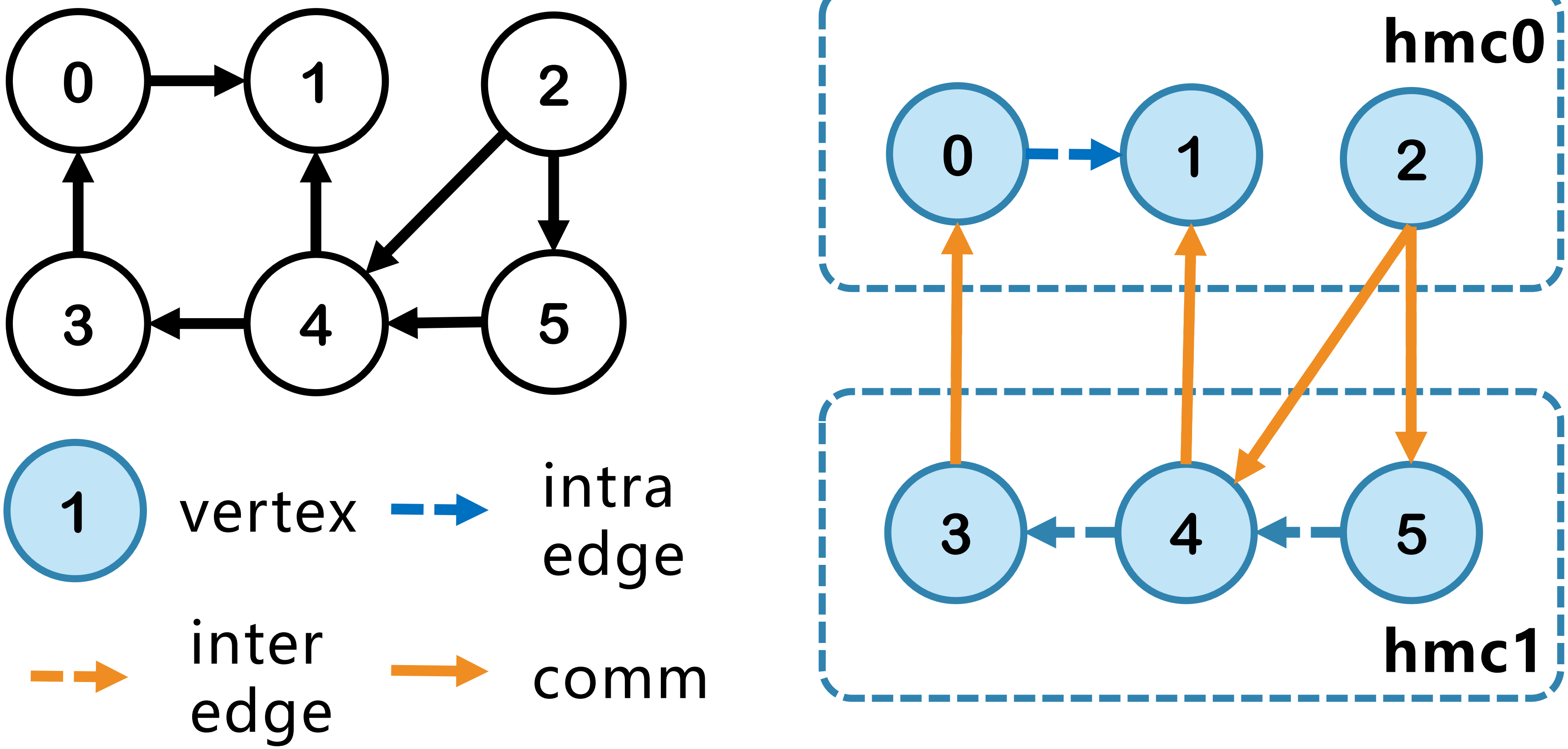




Graph Partition

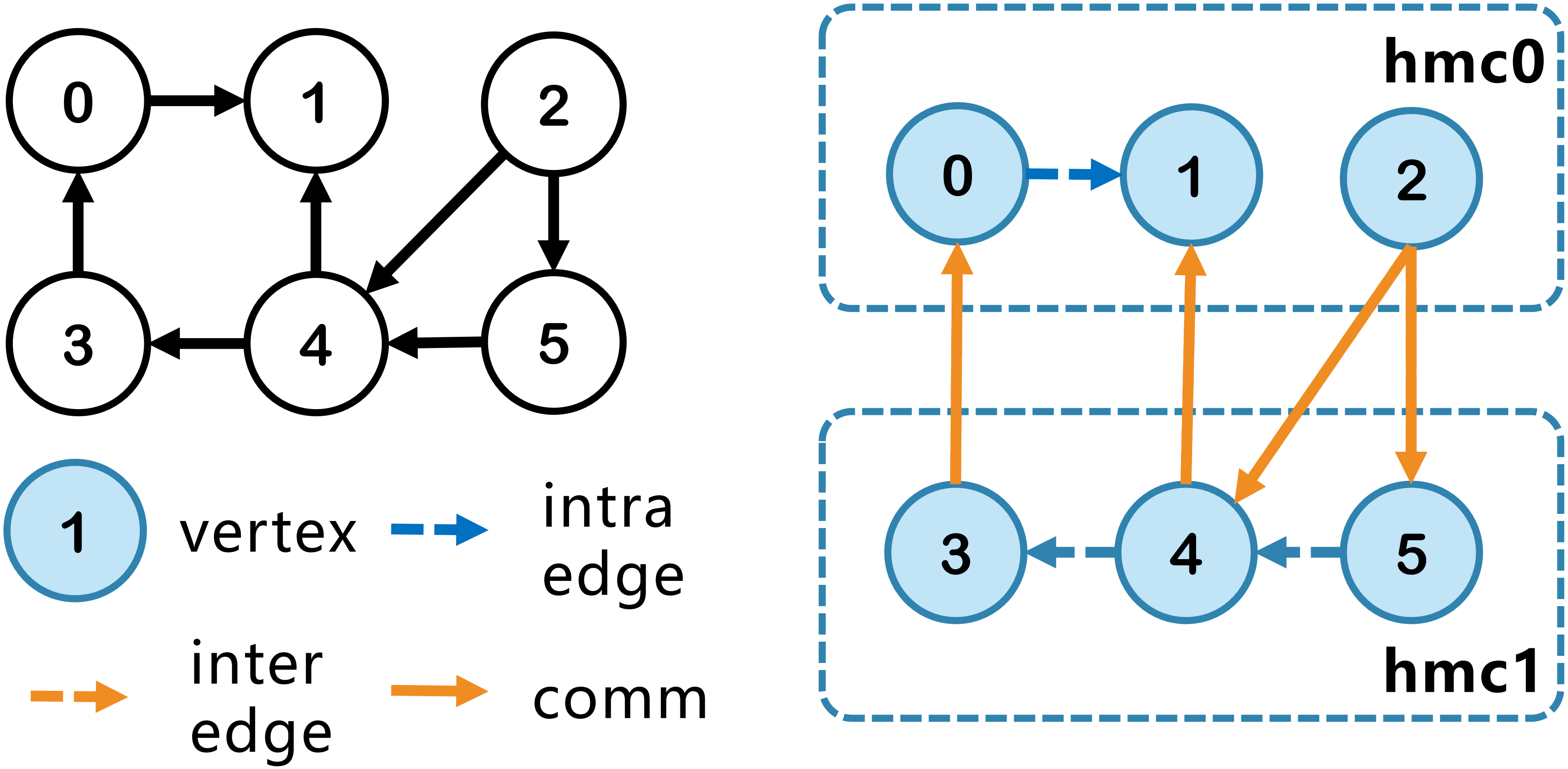


Graph Partition




```
put(w.id, function{ w.next_rank += update; });
```

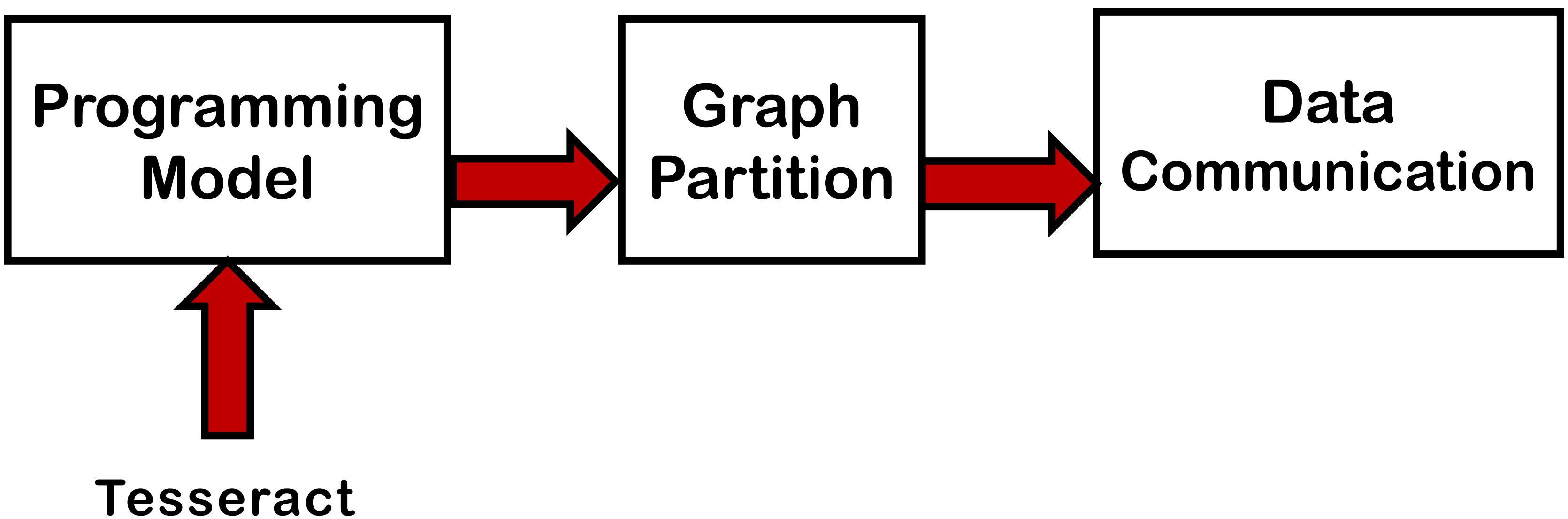
Graph Partition



communication = # of cross-cube edges

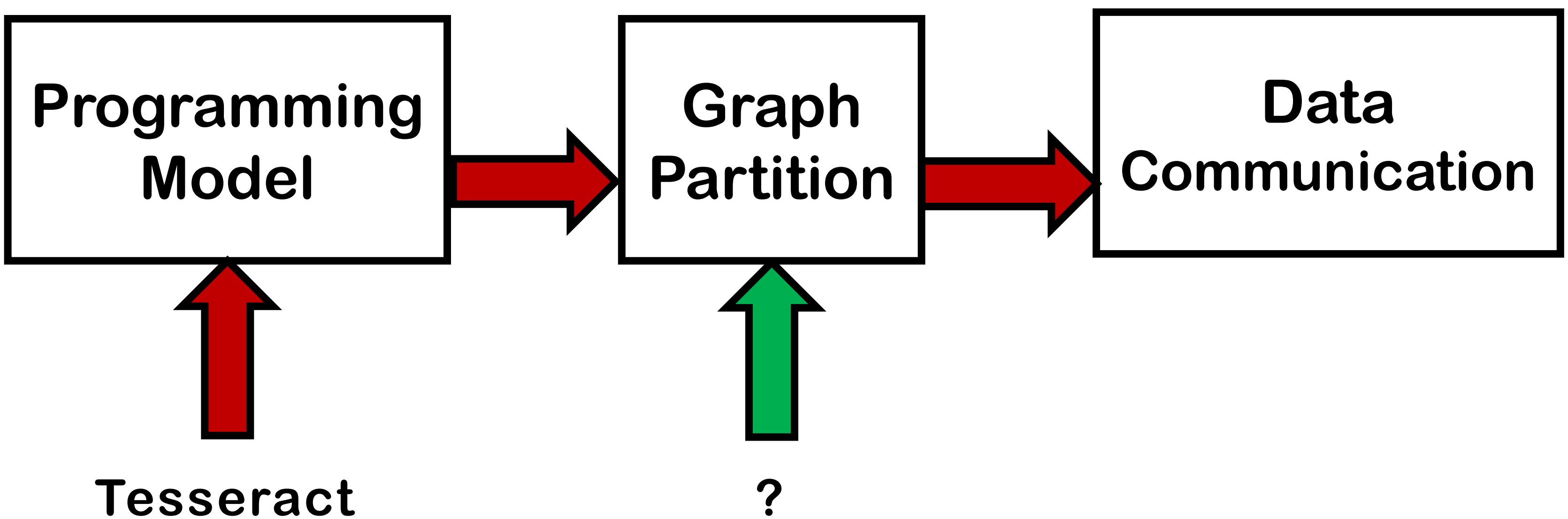
Drawback of Tesseract

- Excessive data communication
- Why?



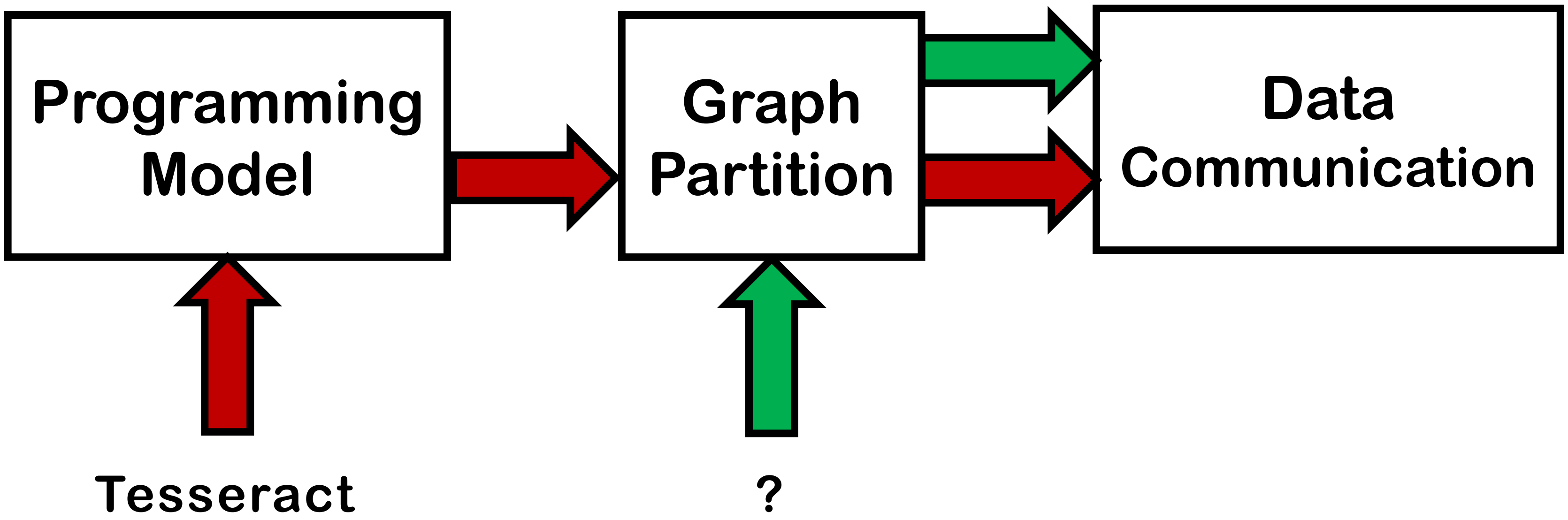
Drawback of Tesseract

- Excessive data communication
- Why?



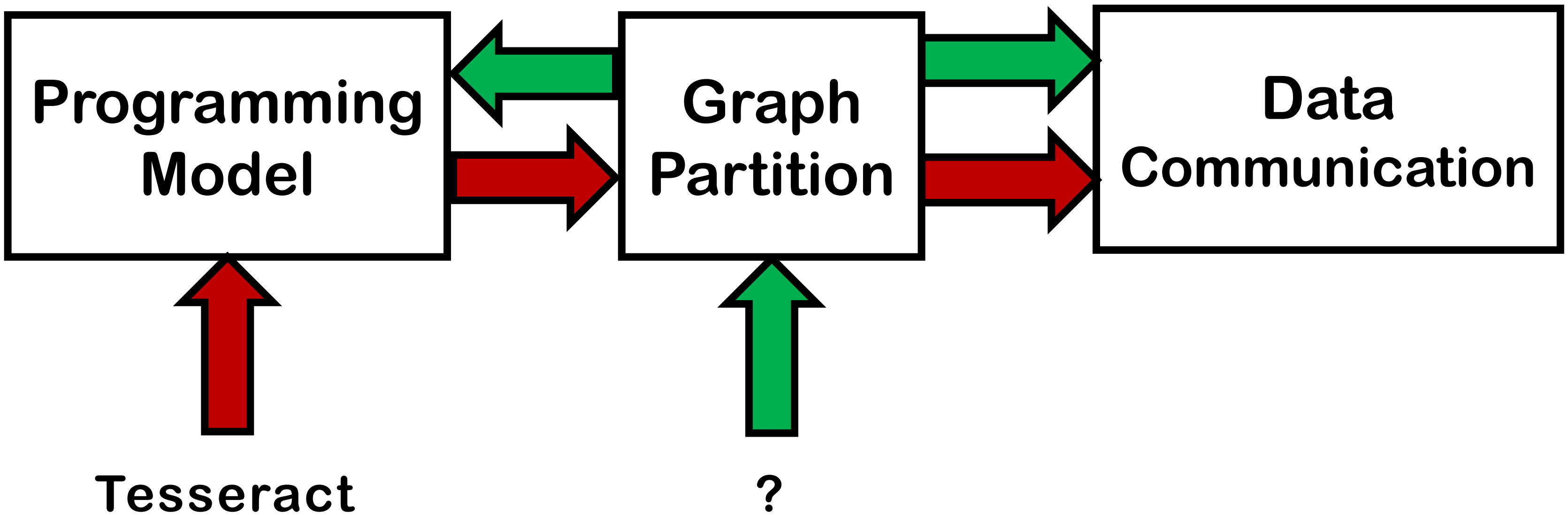
Drawback of Tesseract

- Excessive data communication
- Why?



Drawback of Tesseract

- Excessive data communication
- Why?



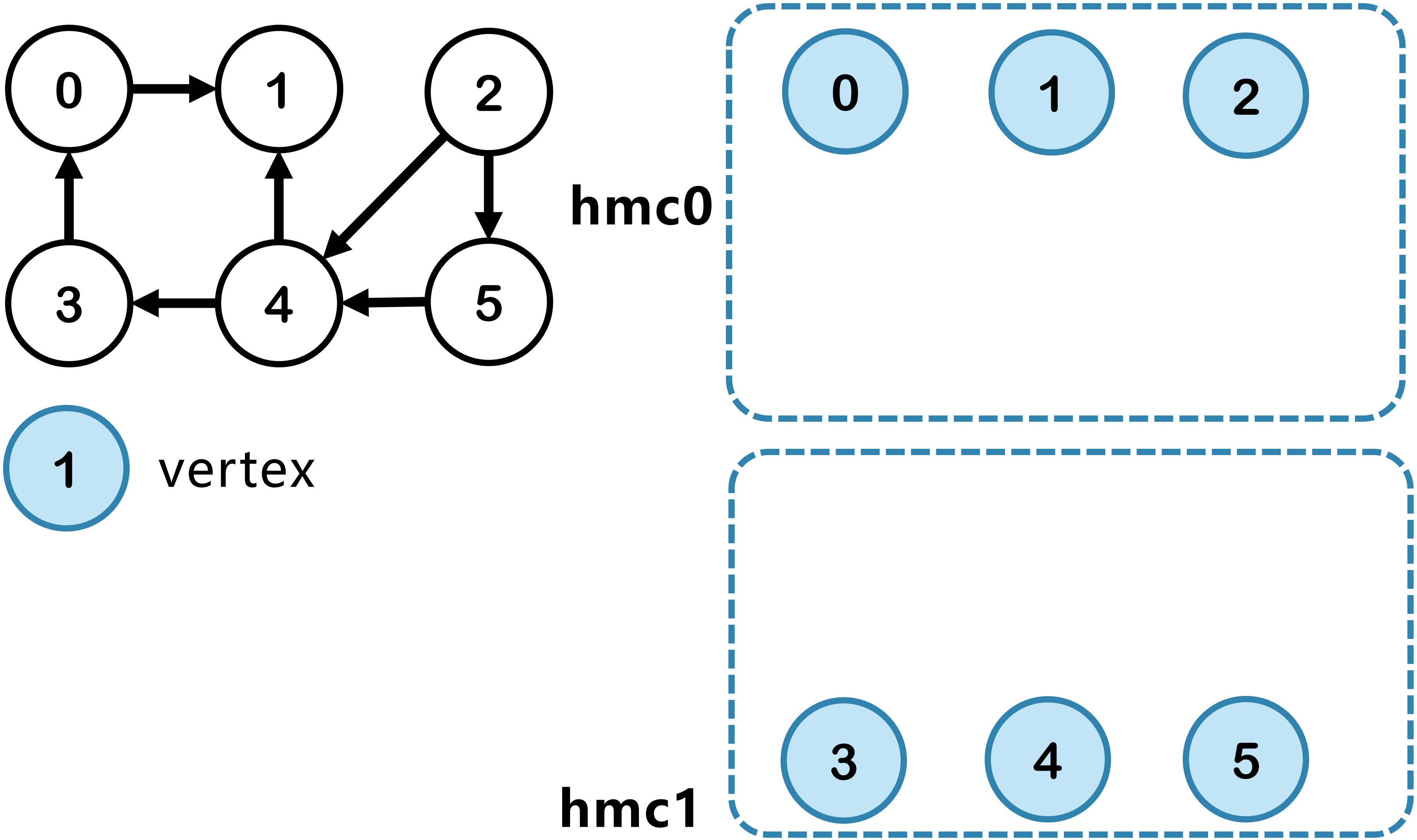
Outline

- Motivation
- **GraphP**
- Evaluation

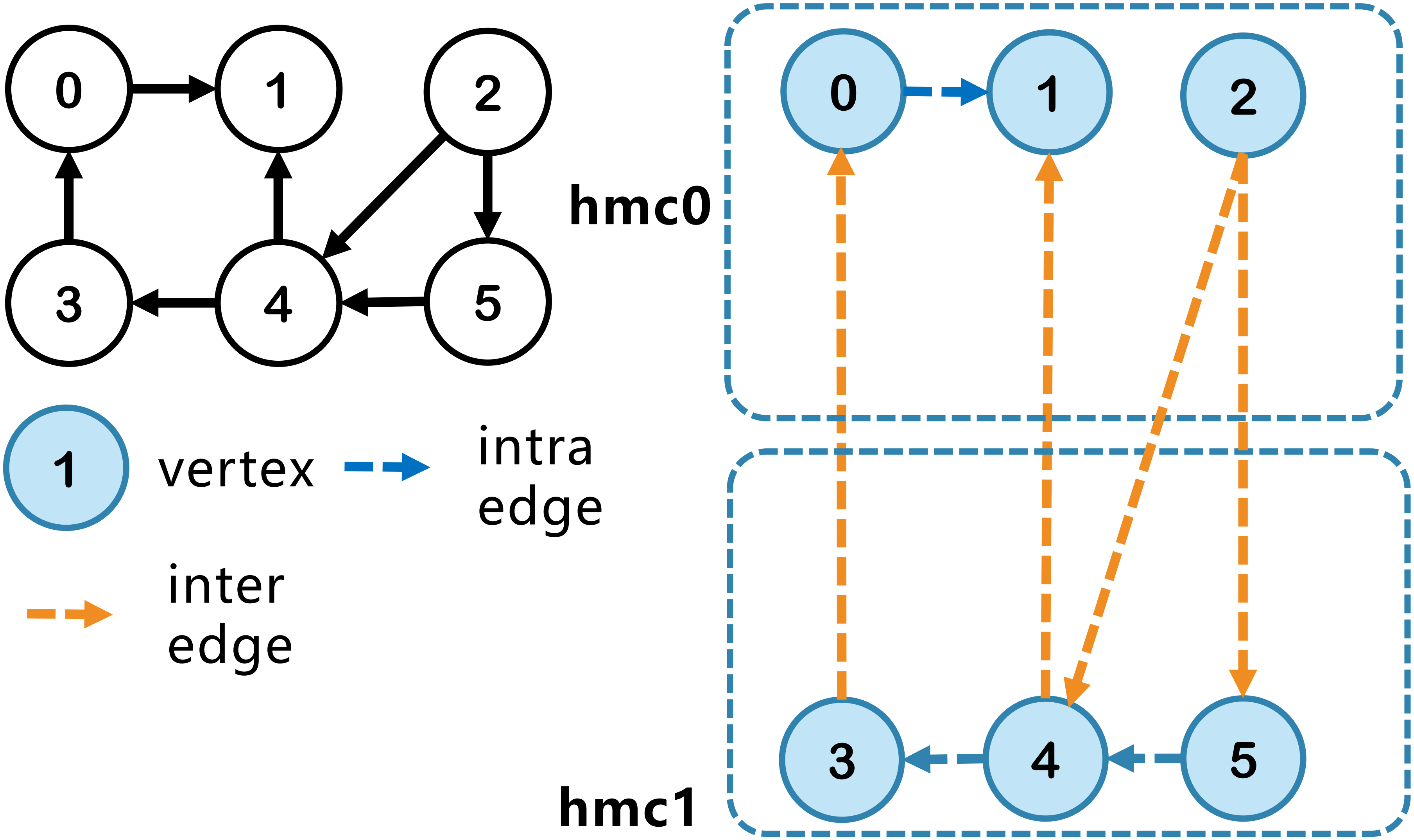
GraphP

- **Consider graph partition first.**
- **Graph Partition**
  - Source-Cut
- **Programming model**
  - Two-phase vertex program
- Reduces inter-cube communication

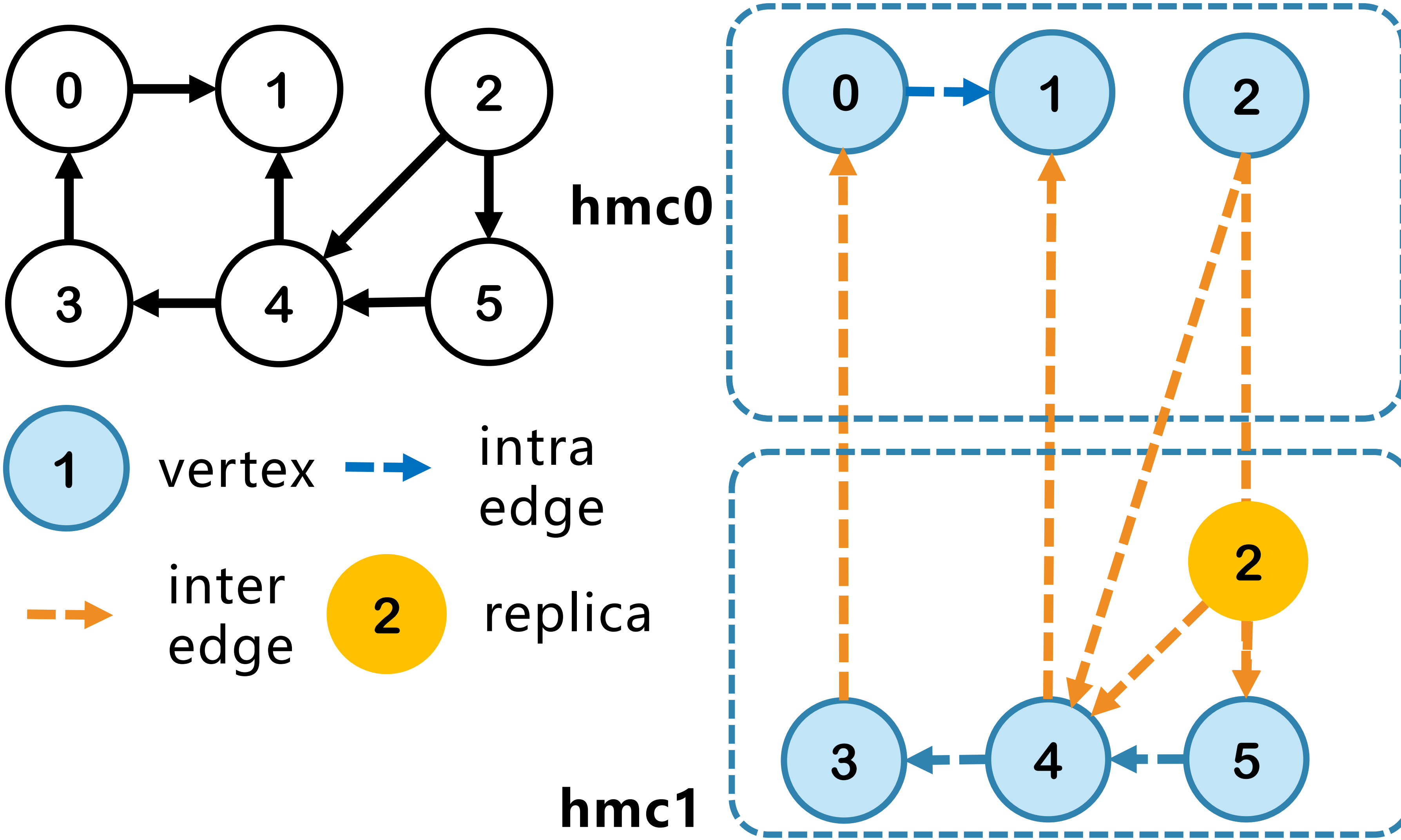
Source-Cut Partition



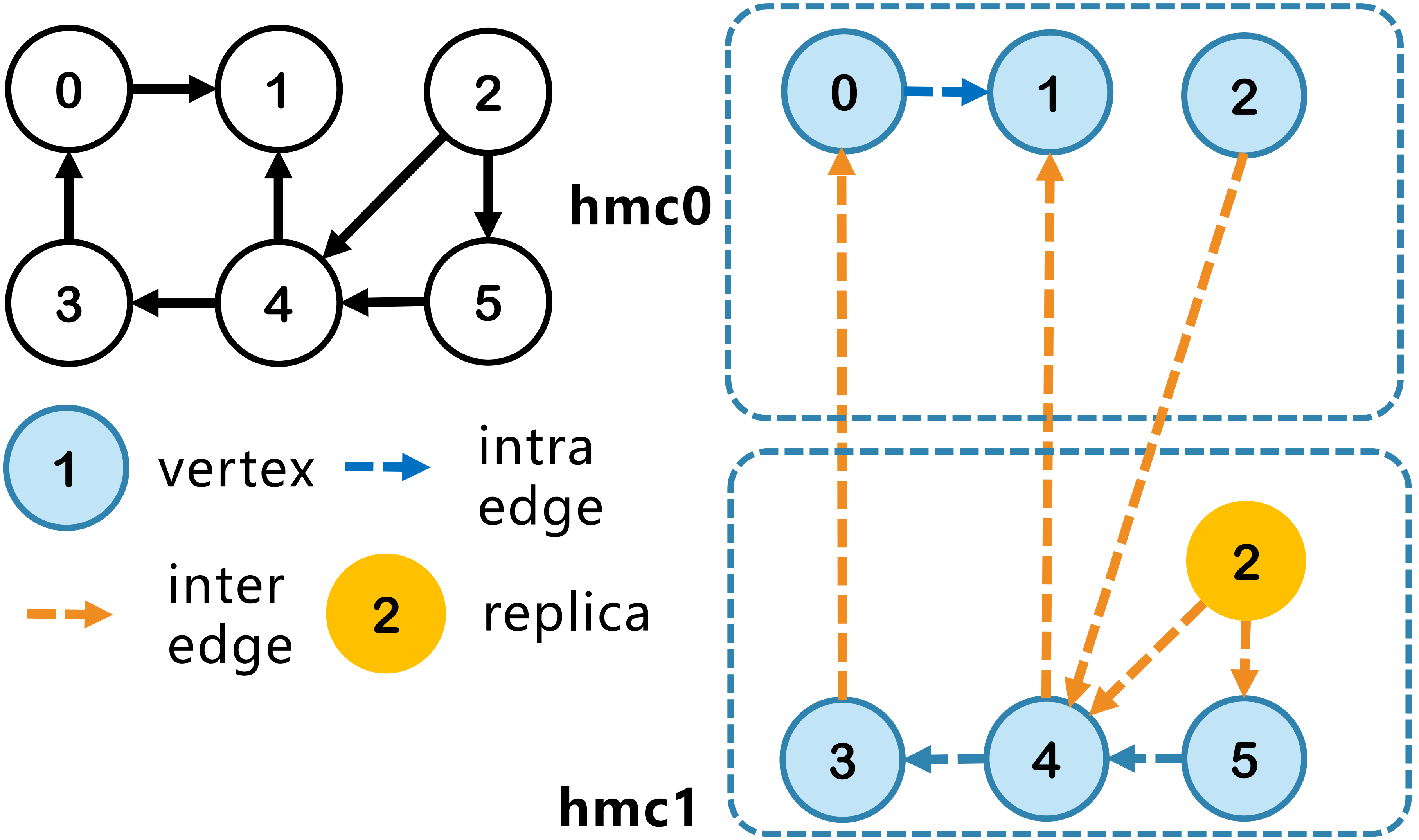
Source-Cut Partition



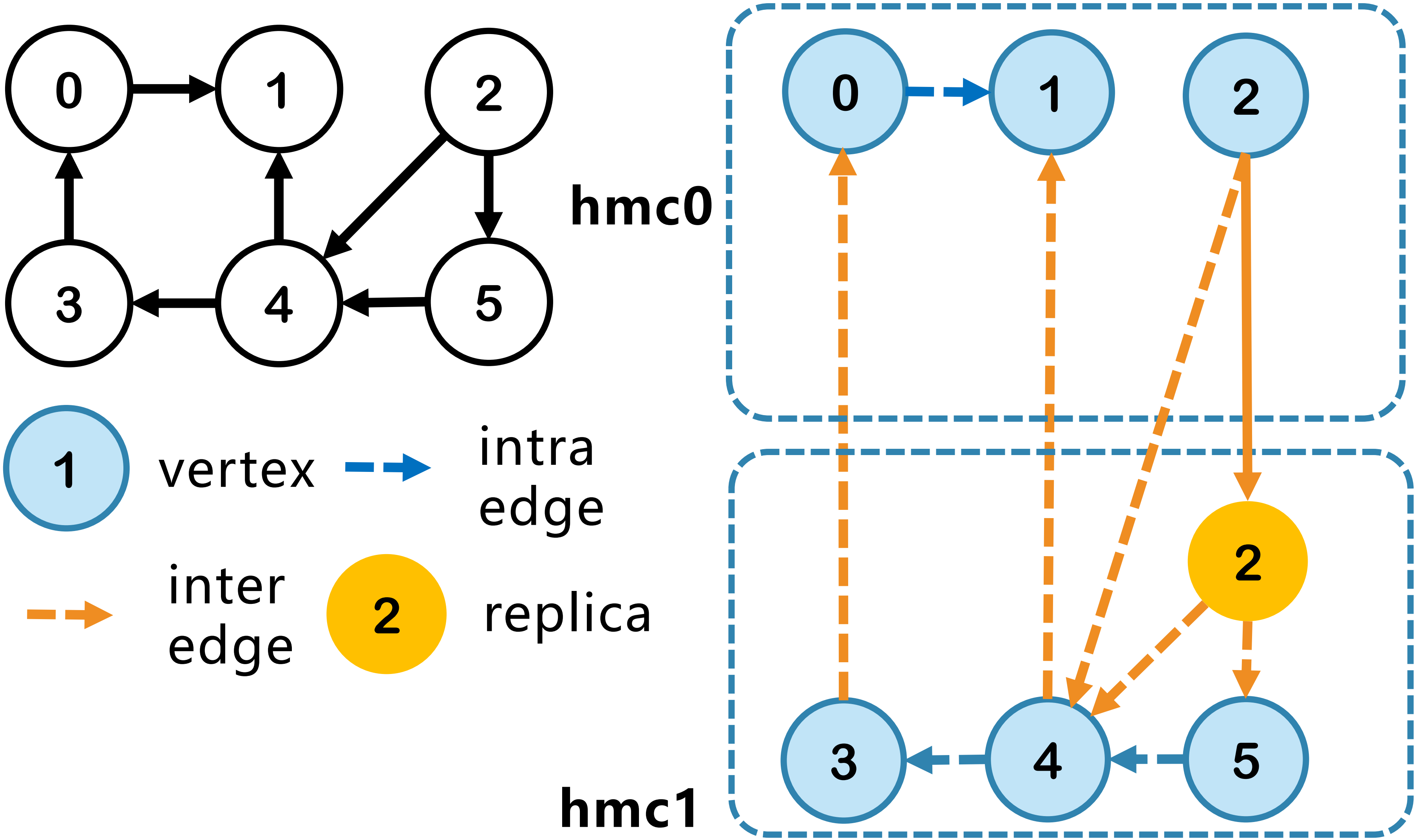
Source-Cut Partition



Source-Cut Partition



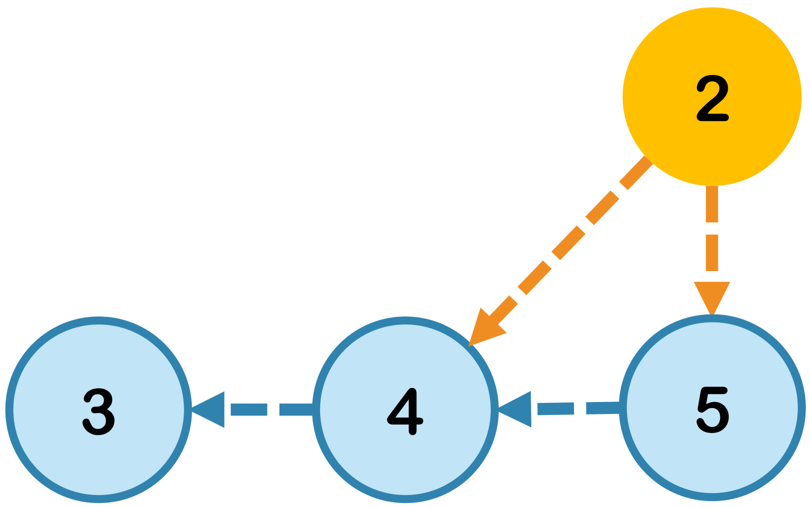
Source-Cut Partition



Two-Phase Vertex Program

```
for (r: replicas) {
    r.next_rank = 0.85 * r.next_rank / r.out_degree;
}
```

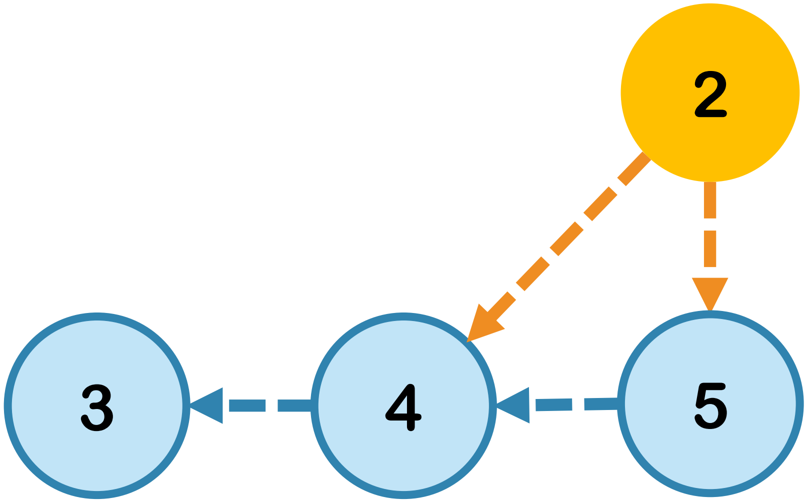
```
//apply updates from previous iterations
```



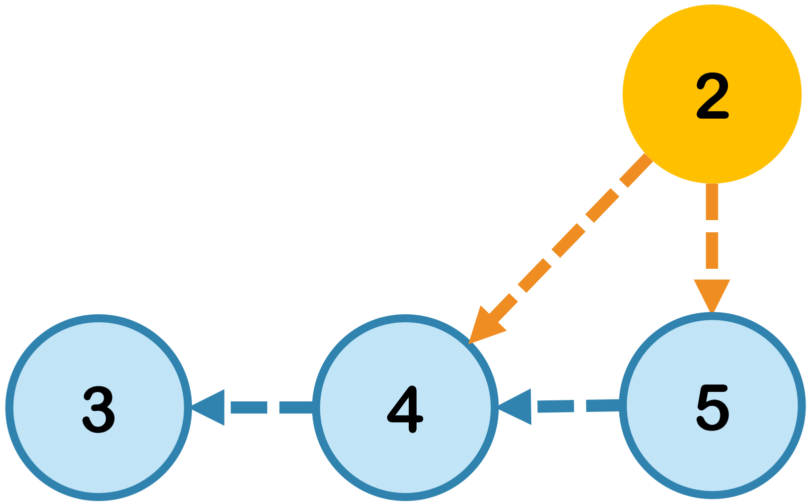
Two-Phase Vertex Program

```
for (r: replicas) {
    r.next_rank = 0.85 * r.next_rank / r.out_degree;
}
```

```
//apply updates from previous iterations
```



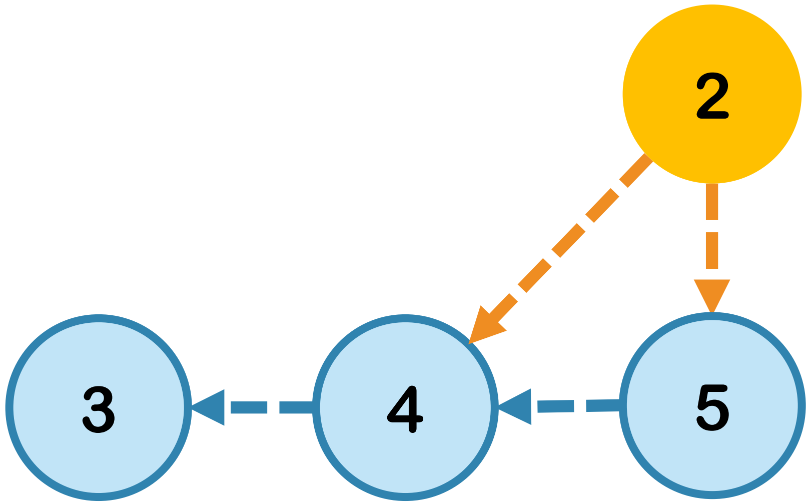
Two-Phase Vertex Program



Two-Phase Vertex Program

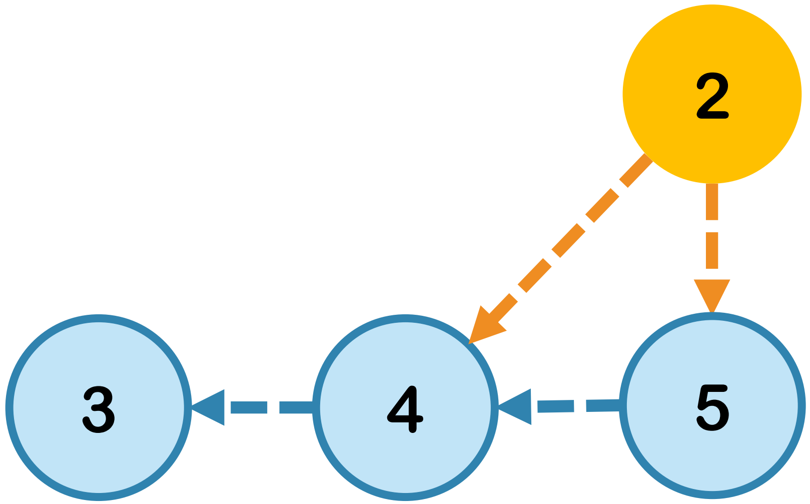
```
for (v: vertices) {
   for (u: edges.sources) {


   }
```



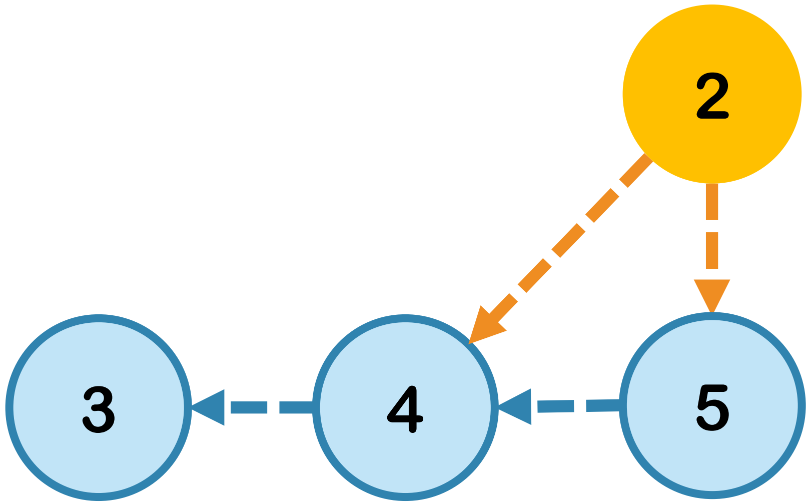
Two-Phase Vertex Program

```
for (v: vertices) {
  for (u: edges.sources) {
    update += u.rank;
  }
```



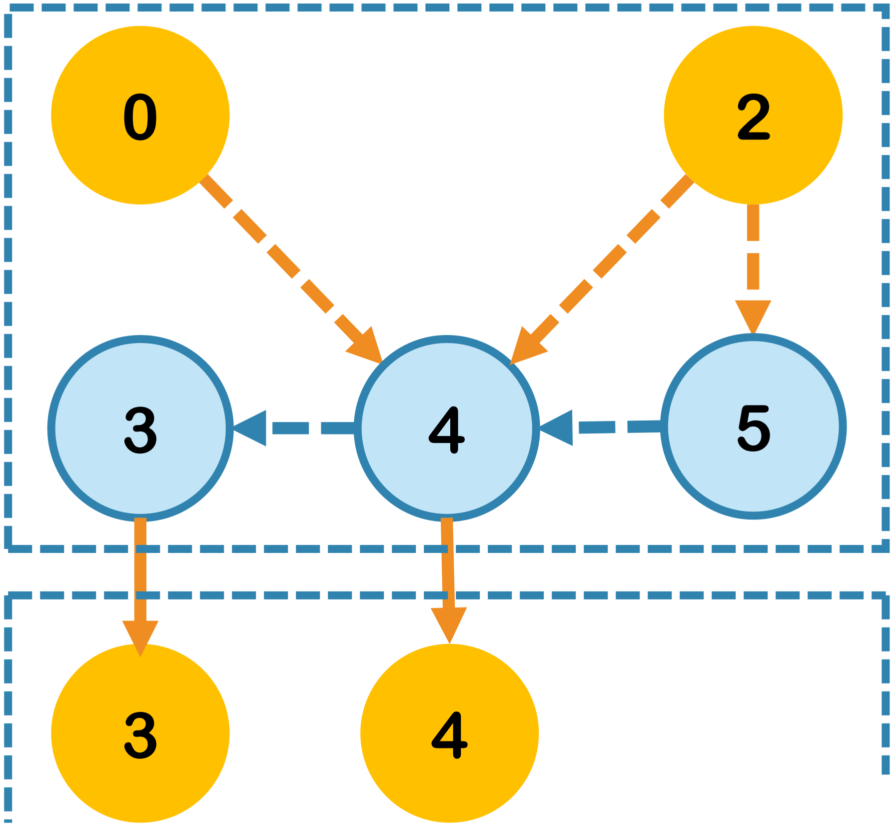
Two-Phase Vertex Program

```
for (v: vertices) {
  for (u: edges.sources) {
    update += u.rank;
  }
```



Two-Phase Vertex Program

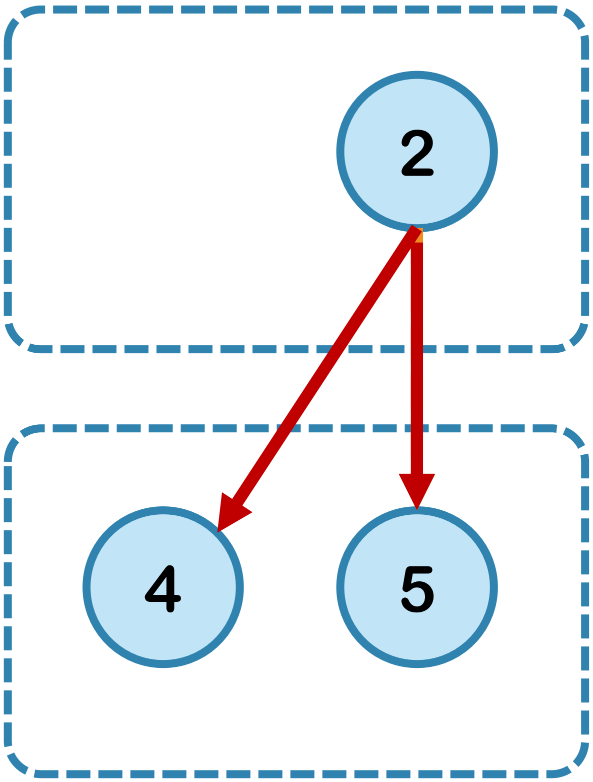
```
for (r: replicas) {
    put(r.id, function { r.next_rank = update});
  }
}
barrier();
```



Benefits

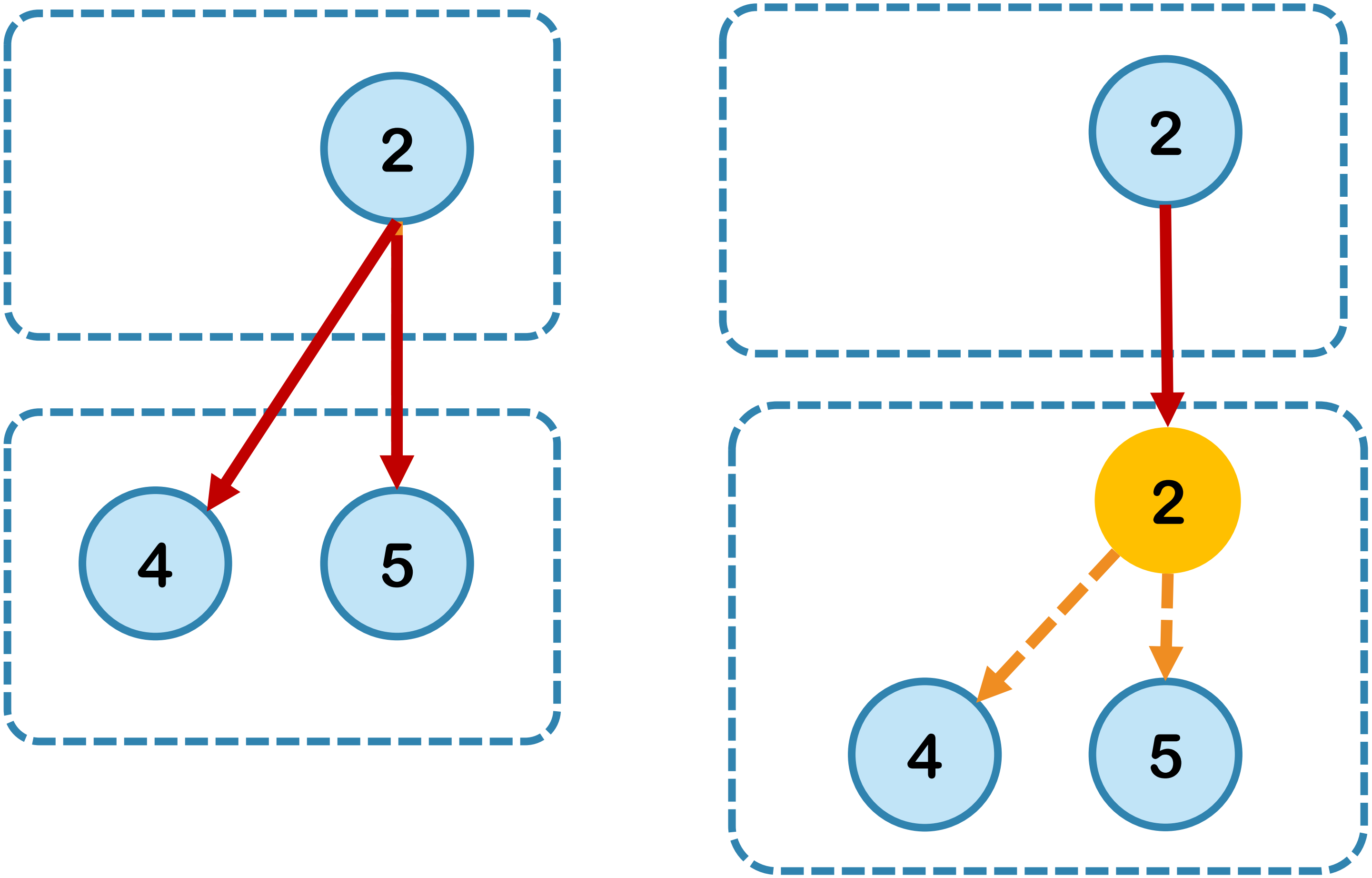
- **Strictly less** data communication
- Enables architecture optimizations

Less Communication



Tesseract **GraphP**

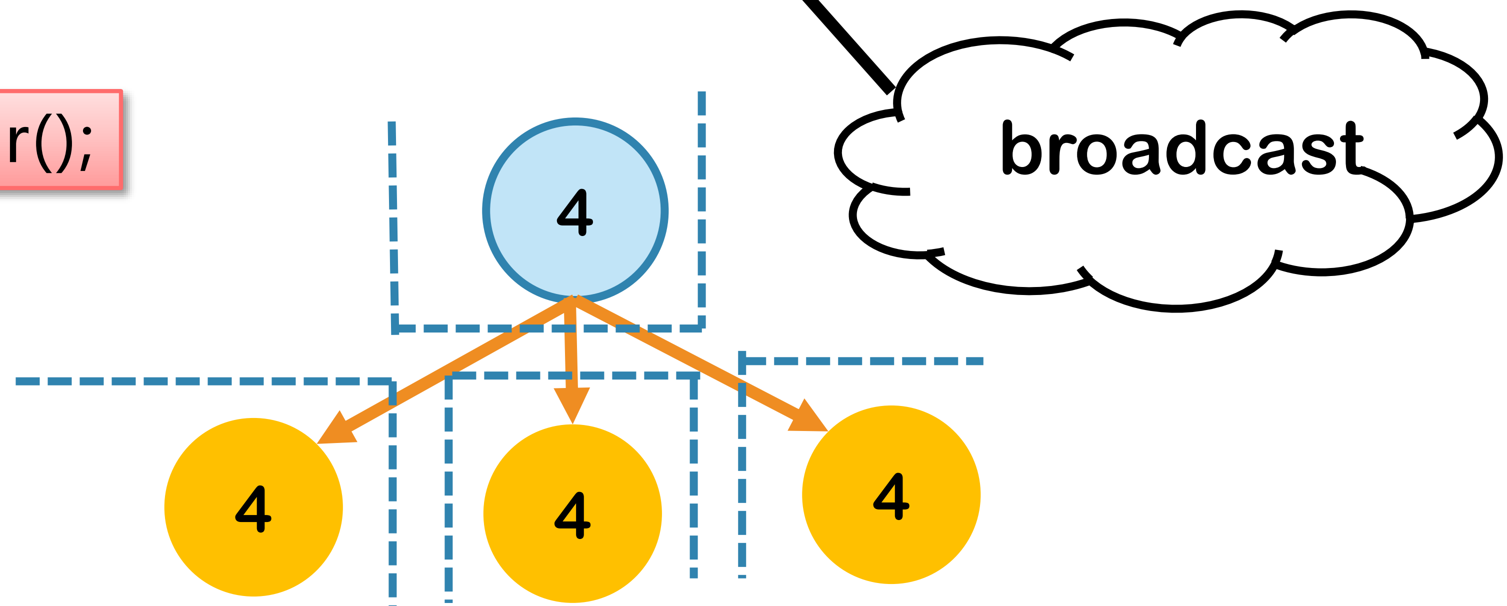
Less Communication



Tesseract **GraphP**

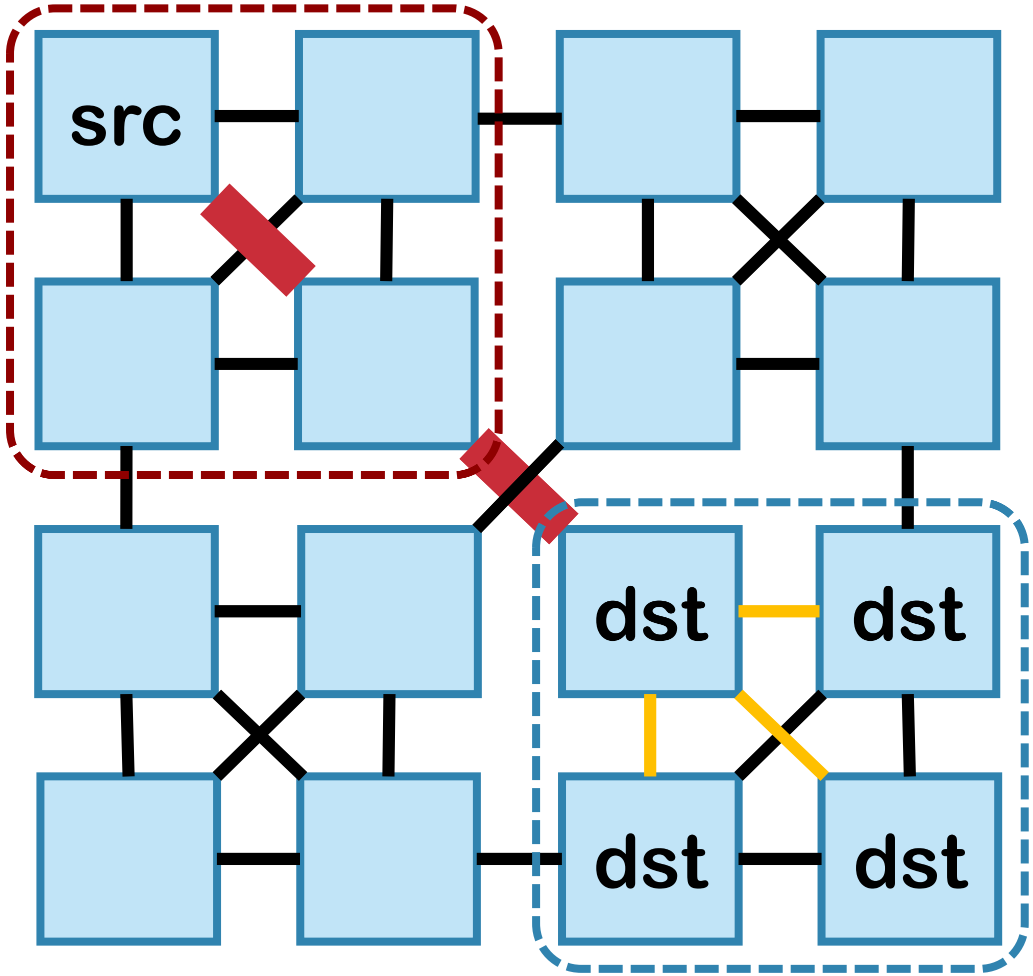
Broadcast Optimization

```
for (r: replicas) {
    put(r.id, function { r.next_rank = update});
}
}
barrier();
```



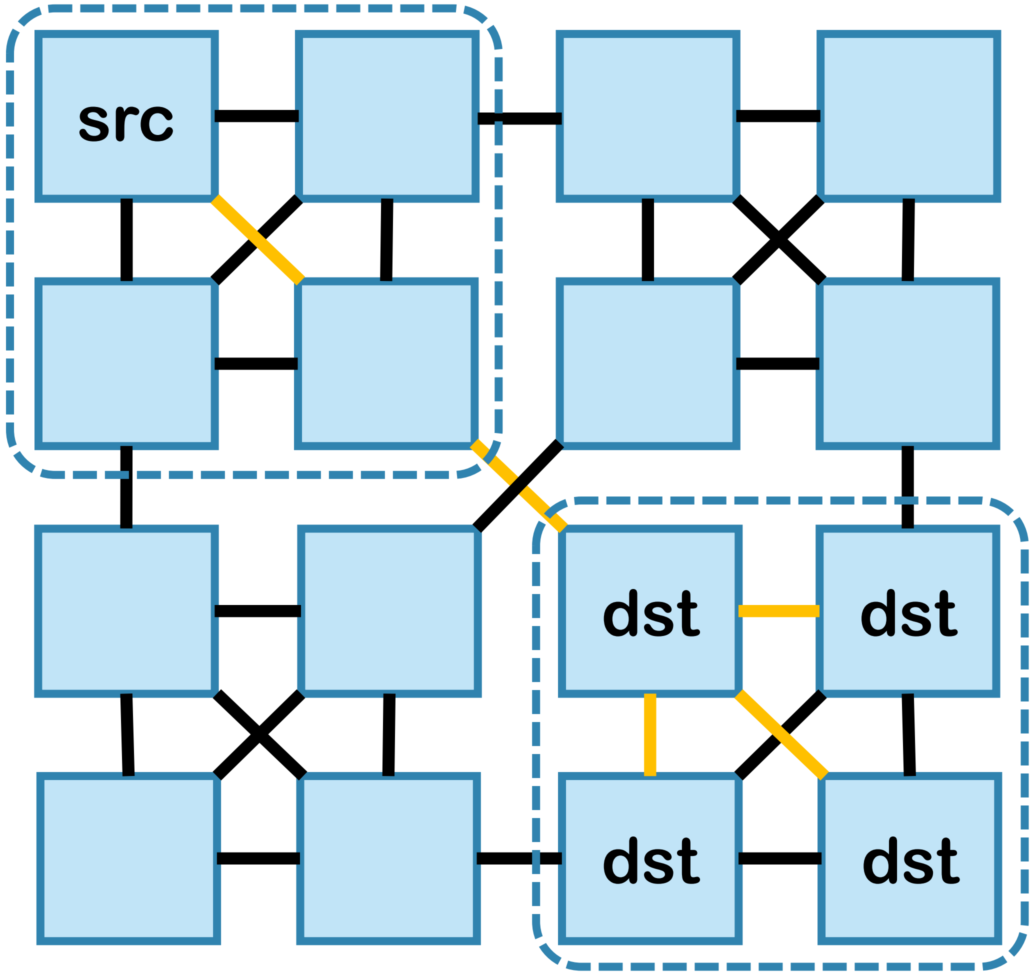
Naïve Broadcast

- 15 point to point messages



Hierarchical communication

- 3 intergroup messages



Other Optimizations

- Computation/communication overlap
- Leveraging low-power state of SerDes

Please see the paper for more details

Outline

- Motivation
- GraphP
- **Evaluation**

Evaluation Methodology

- Simulation Infrastructure
  - zSim with HMC support
  - ORION for NOC Energy modeling
- Configurations
  - Same as Tesseract
  - 16 HMCs
  - Interconnection: Dragonfly and Mesh2D
  - 512 CPUs
    - Single-issue in-order cores
    - Frequency: 1GHz

Workloads

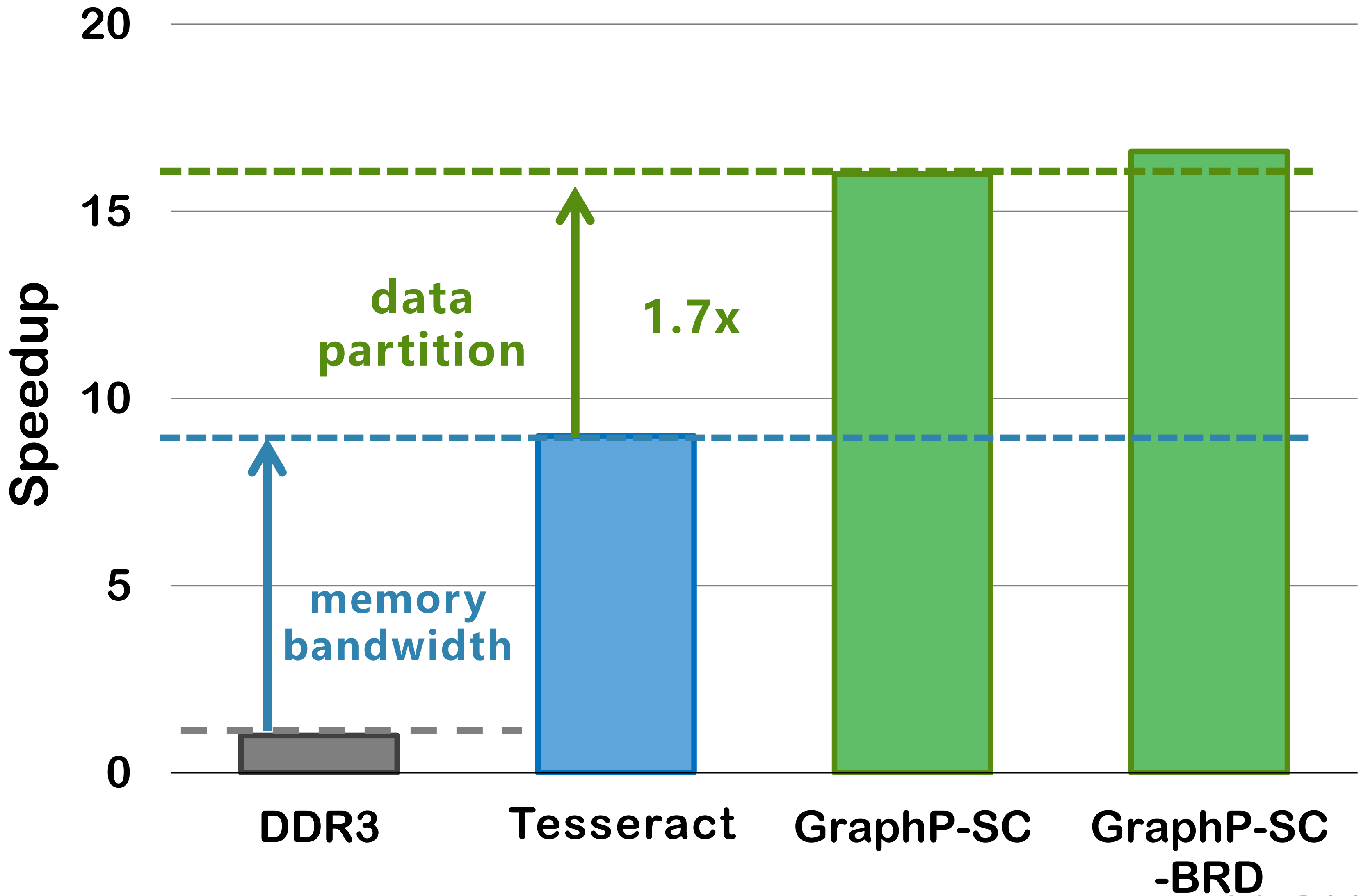
- 4 graph algorithms
- 5 real-world graphs

Workloads

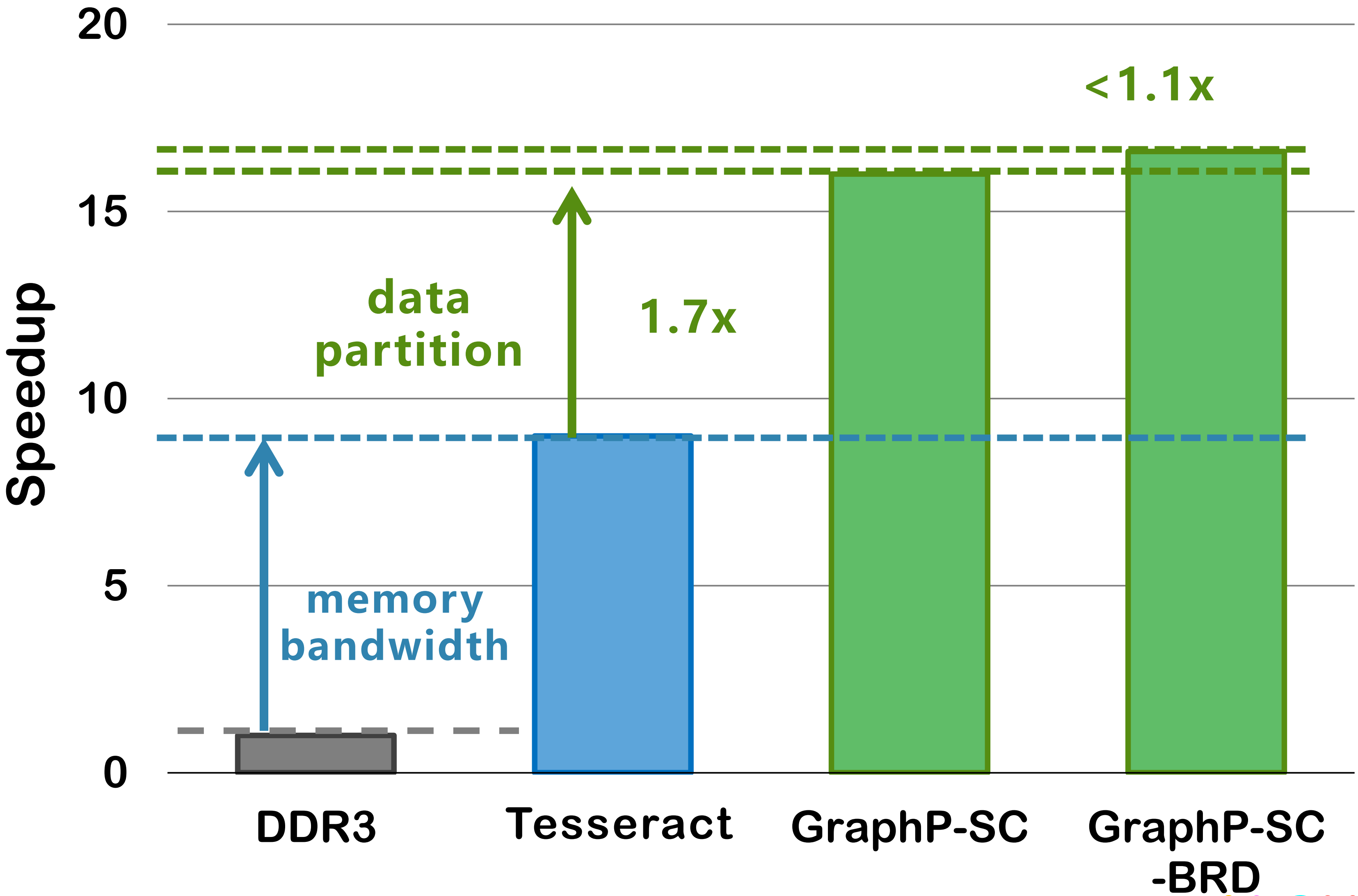
- 4 graph algorithms
  - Breadth First Search
  - Single Source Shortest Path
  - Weakly Connected Component
  - PageRank
- 5 real-world graphs
  - Wiki-Vote (WV)
  - ego-Twitter (TT)
  - Soc-Slashdot0902 (SD)
  - Amazon0302 (AZ)
  - ljournal-2008 (LJ)

Performance

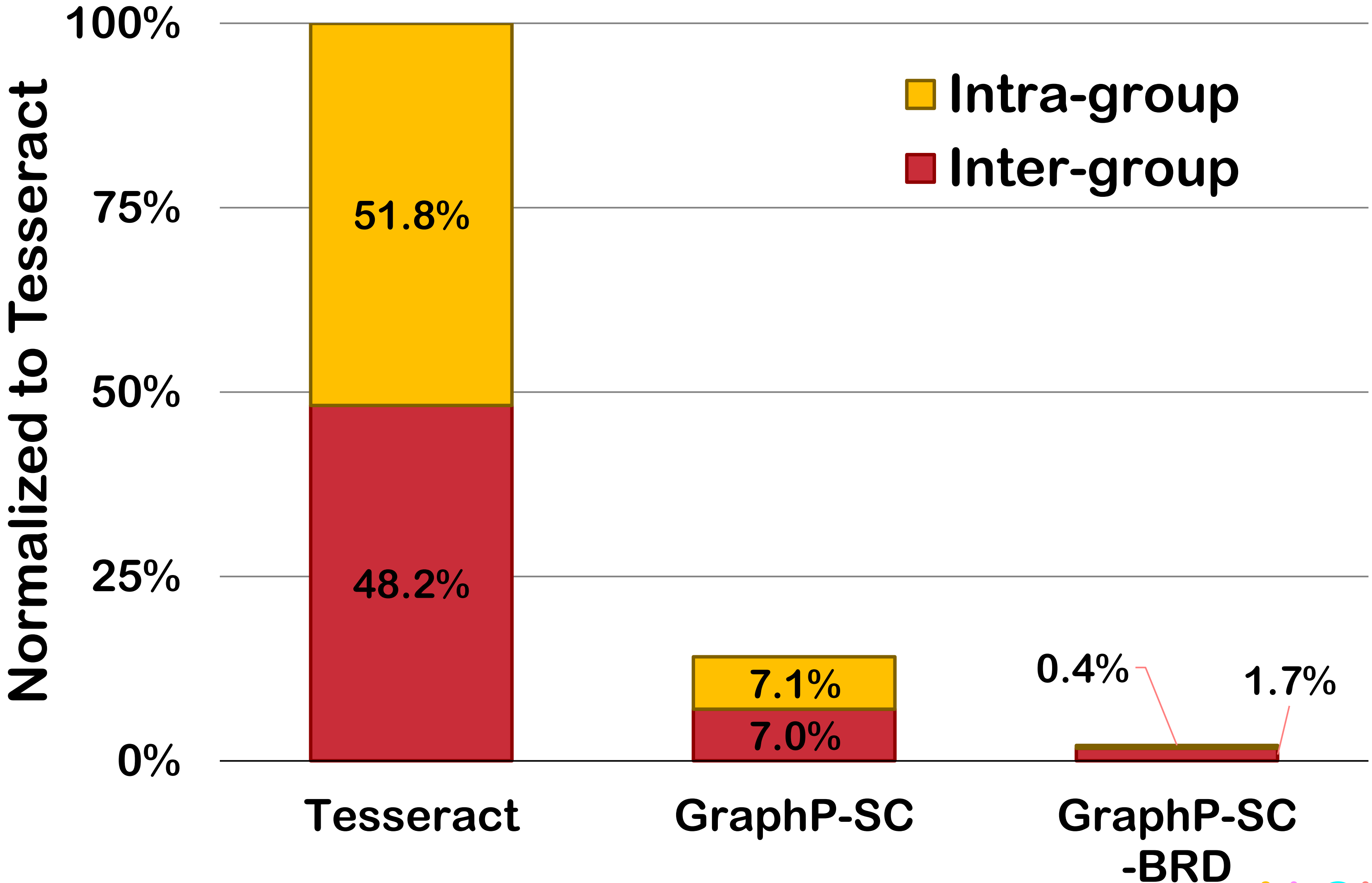
Performance



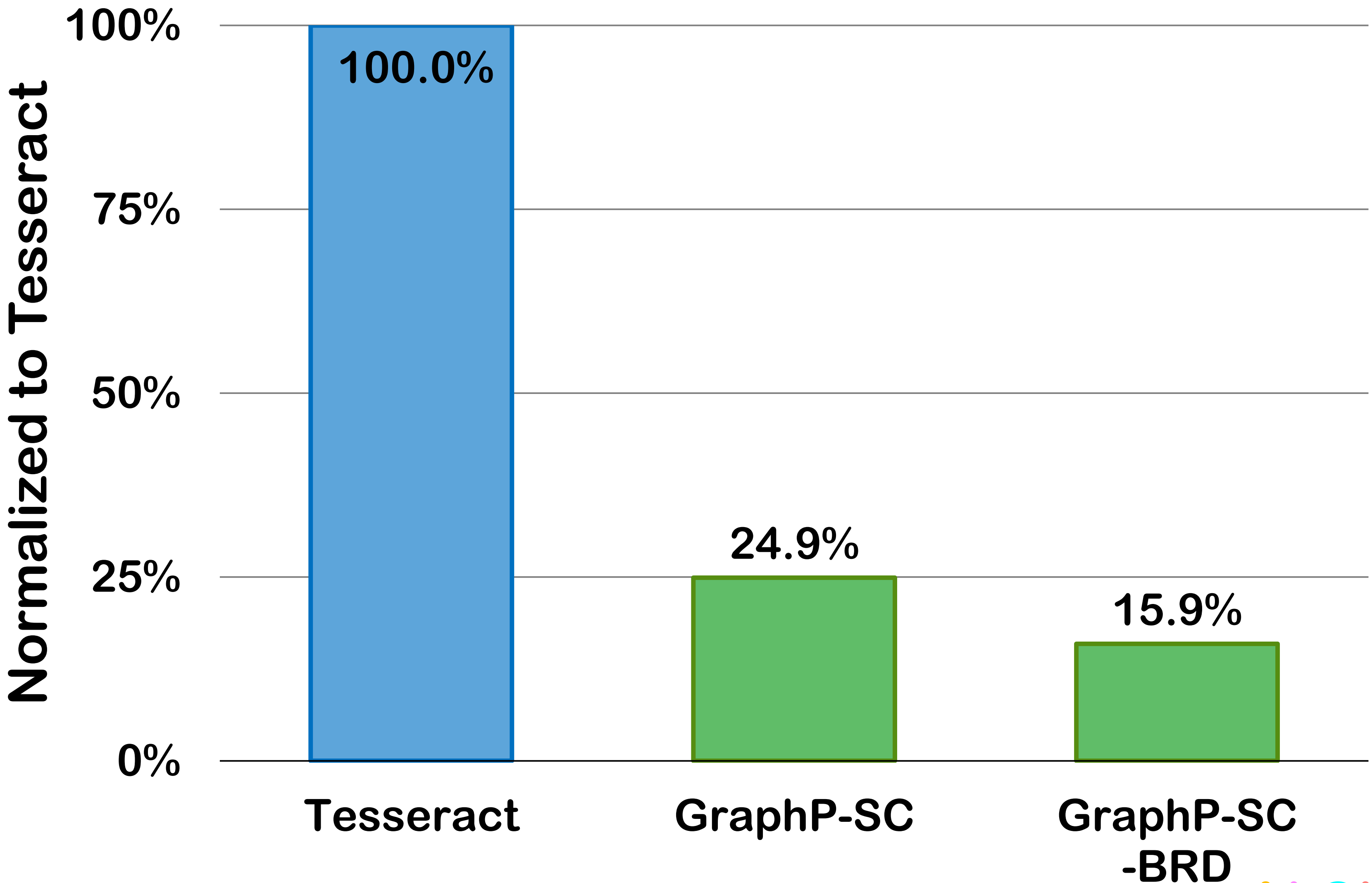
Performance



Communication Amount



Energy consumption



Other results

- Bandwidth utilization
- Scalability
- Replication overhead

Please see the paper for more details

Conclusions

- We propose **GraphP**
  - A new PIM-based graph processing framework
- Key contributions

Conclusions

- We propose **GraphP**
  - A new PIM-based graph processing framework
- Key contributions
  - Data partition as first-order design consideration

Conclusions

- We propose **GraphP**
  - A new PIM-based graph processing framework
- Key contributions
  - Data partition as first-order design consideration
  - Source-cut partition

Conclusions

- We propose **GraphP**
  - A new PIM-based graph processing framework
- Key contributions
  - Data partition as first-order design consideration
  - Source-cut partition
  - Two-phase vertex program

Conclusions

- We propose **GraphP**
  - A new PIM-based graph processing framework
- Key contributions
  - Data partition as first-order design consideration
  - Source-cut partition
  - Two-phase vertex program
  - Enable additional architecture optimizations

Conclusions

- We propose **GraphP**
  - A new PIM-based graph processing framework
- Key contributions
  - Data partition as first-order design consideration
  - Source-cut partition
  - Two-phase vertex program
  - Enable additional architecture optimizations
- GraphP drastically reduces inter-cube communication and improves energy efficiency.

GraphP: Reducing Communication for PIM-based Graph Processing with Efficient Data Partition

Mingxing Zhang, **Youwei Zhuo (equal contribution)**,
Chao Wang, Mingyu Gao, Yongwei Wu, Kang Chen,
Christos Kozyrakis, Xuehai Qian

Tsinghua University
University of Southern California
Stanford University

Workload Size & Capacity

- 128 GB (16 * 8GB)
- ~16 billion edges
- ~400 million edges (SNAP)
- ~7 billion edges (WebGraph)

https://snap.stanford.edu/data/
http://law.di.unimi.it/datasets.php

Two-phase vertex program

- Equivalent Expressiveness as vertex programs