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

- **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)

320GB/s intra-cube

4x120GB/s inter-cube
HMC: Hybrid Memory Cubes

Intra-cube bandwidth (GB/s)

320
HMC: Hybrid Memory Cubes

Intra-cube

Inter-cube

bandwidth (GB/s)
HMC: Hybrid Memory Cubes

Intra-cube

Inter-cube

bandwidth (GB/s)
HMC: Hybrid Memory Cubes

GraphP: A PIM-based Graph Processing Framework
HMC: Hybrid Memory Cubes

Bottleneck: Inter-cube communication

Intra-cube

Inter-group

bandwidth (GB/s)
Outline

• Motivation
  • Graph applications
  • Processing-In-Memory
  • The drawbacks of the current solution

• GraphP

• Evaluation
Current Solution: Tesseract

• First PIM-based graph processing architecture

Current Solution: Tesseract

• First PIM-based graph processing architecture

• Programming model
  • Vertex program

Current Solution: Tesseract

- First PIM-based graph processing architecture

- **Programming model**
  - Vertex program

- **Partition**
  - Based on vertex program

PageRank in Vertex Program

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

```java
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

```plaintext
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

GraphP: A PIM-based Graph Processing Framework

ALCHEM
alchem.usc.edu
Graph Partition

GraphP: A PIM-based Graph Processing Framework
Graph Partition

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

communication = # of cross-cube edges

GraphP: A PIM-based Graph Processing Framework
Drawback of Tesseract

• Excessive data communication

• Why?

Tesseract
Drawback of Tesseract

• Excessive data communication

• Why?

Programming Model \[\rightarrow\] Graph Partition \[\rightarrow\] Data Communication

Tesseract \[\uparrow\] \[?\]
Drawback of Tesseract

- Excessive data communication
- Why?

![Diagram showing the relationship between Programming Model, Graph Partition, and Data Communication with Tesseract indicating excessive data communication]

GraphP: A PIM-based Graph Processing Framework

alchem.usc.edu
Drawback of Tesseract

- Excessive data communication
- Why?

Diagram:

- Programming Model
- Graph Partition
- Data Communication

Tesseract

?
Outline

- Motivation
- GraphP
- Evaluation
• Consider graph partition first.

• Graph Partition
  • Source-Cut

• Programming model
  • Two-phase vertex program

• Reduces inter-cube communication
Source-Cut Partition

GraphP: A PIM-based Graph Processing Framework

hmc0

hmc1

1 vertex
Source-Cut Partition

GraphP: A PIM-based Graph Processing Framework

vertex intra edge
inter edge
Source-Cut Partition

GraphP: A PIM-based Graph Processing Framework
Source-Cut Partition

GraphP: A PIM-based Graph Processing Framework
Source-Cut Partition

GraphP: A PIM-based Graph Processing Framework

vertex
intra edge
replica
inter edge
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

GraphP: A PIM-based Graph Processing Framework
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

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

• **Strictly less** data communication
• Enables architecture optimizations
Less Communication

GraphP: A PIM-based Graph Processing Framework

Tesseract

GraphP

2

4

5
Less Communication

Tesseract

GraphP

GraphP: A PIM-based Graph Processing Framework

alchem.usc.edu
for (r: replicas) {
    put(r.id, function { r.next_rank = update});
}
Naïve Broadcast

• 15 point to point messages
Hierarchical communication

• 3 intergroup messages

GraphP: A PIM-based Graph Processing Framework
Other Optimizations

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

Please see the paper for more details
• 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)
GraphP: A PIM-based Graph Processing Framework

Performance

memory bandwidth

Tesseract
Performance

GraphP: A PIM-based Graph Processing Framework

GraphP-SC -BRD
GraphP: A PIM-based Graph Processing Framework

Performance

- DDR3
- Tesseract
- GraphP-SC
- GraphP-SC-BRD

- Speedup
- Memory bandwidth
- Data partition

<1.1x

1.7x
Communication Amount

Normalized to Tesseract

- Tesseract: 51.8%
- GraphP-SC: 48.2%
- GraphP-SC-BRD: 1.7%

Intra-group:
- Tesseract: 7.1%
- GraphP-SC: 7.0%
- GraphP-SC-BRD: 0.4%

Inter-group:
- Tesseract: 0%
- GraphP-SC: 25%
- GraphP-SC-BRD: 75%
Energy consumption

Normalized to Tesseract

100.0% 24.9% 15.9%

Tesseract GraphP-SC GraphP-SC-BRD

GraphP: A PIM-based Graph Processing Framework

alchem.usc.edu
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