GraphWave: A Highly-Parallel Compute-at-Memory Graph Processing Accelerator

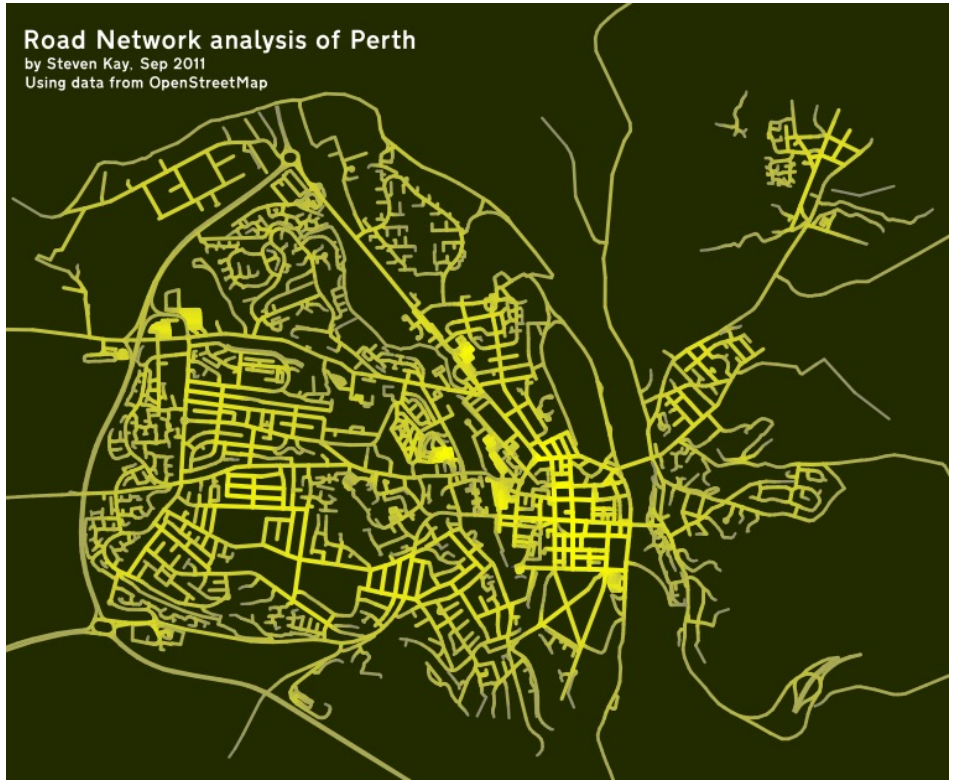
Jinho Lee, **Burin Amornpaisannon**, Tulika Mitra, Trevor E. Carlson

Design, Automation and Test in Europe Conference (DATE) 2022

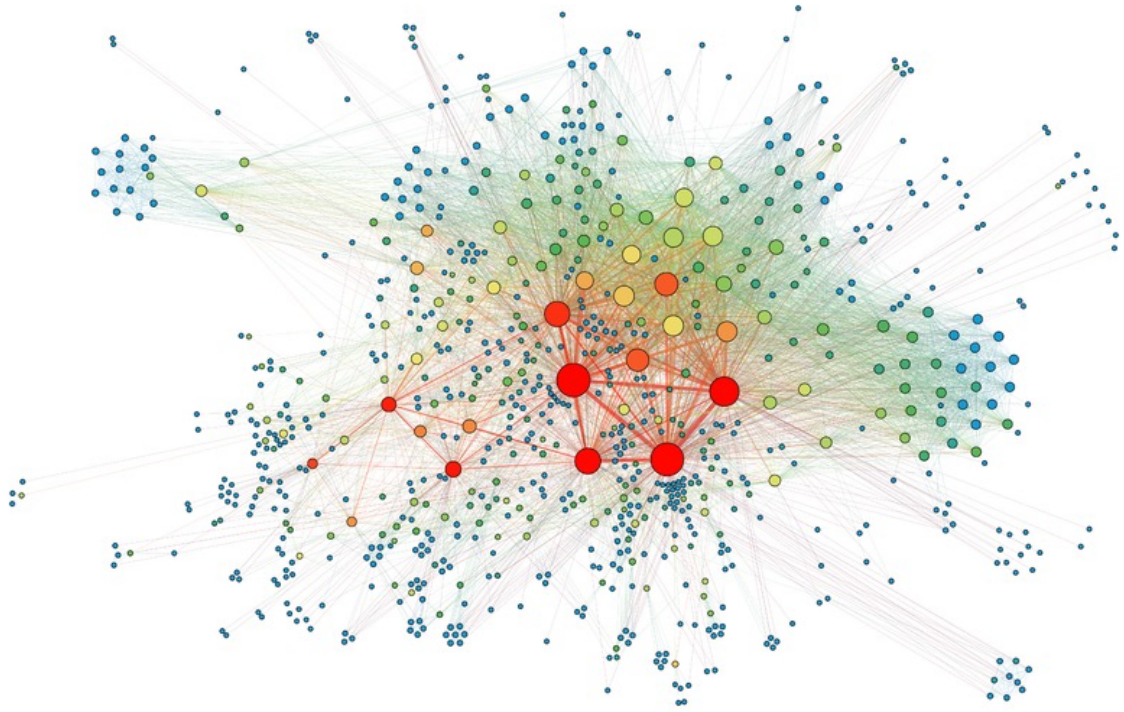
Overview

- Background: Graph processing
  - Vertex-centric approach
  - Challenges in vertex-centric approaches
  - Related works
- GraphWave
  - Methodology
  - Microarchitecture
  - Message Propagation & Congestion Avoidance
- Experimental Setup & Evaluation
- Conclusion

Background: graph proceccing



[1]

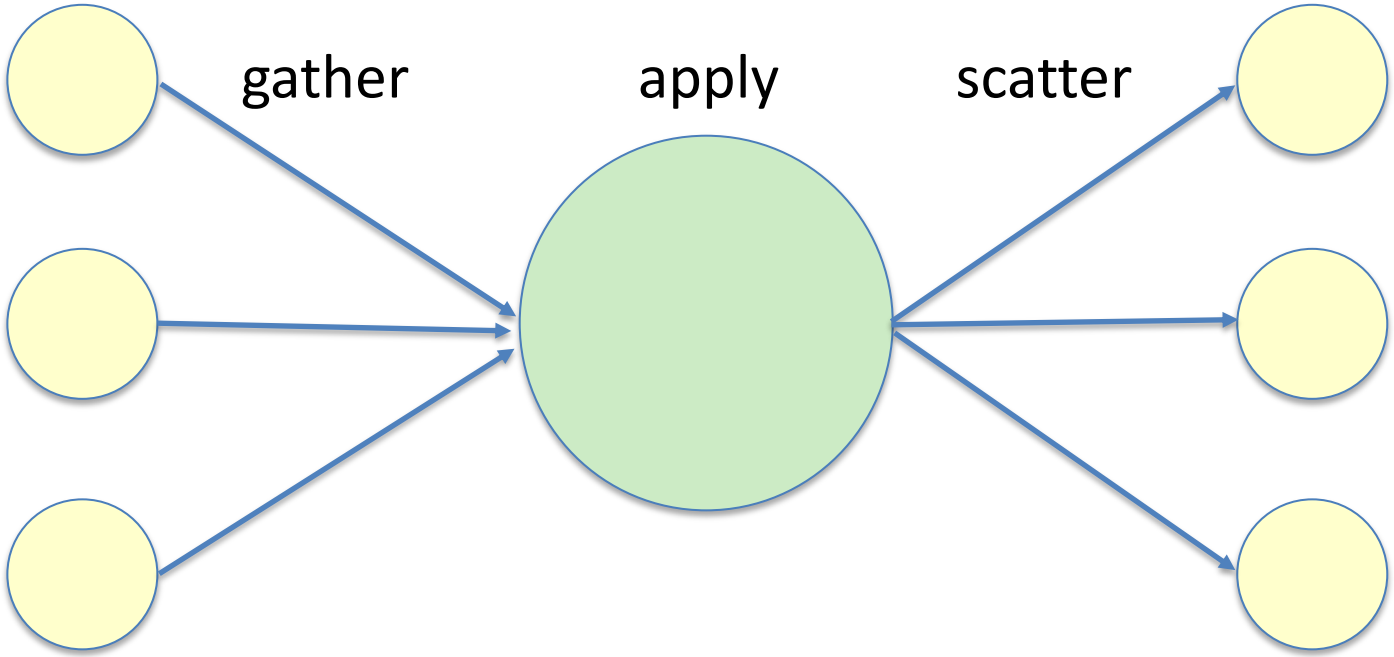


[2]

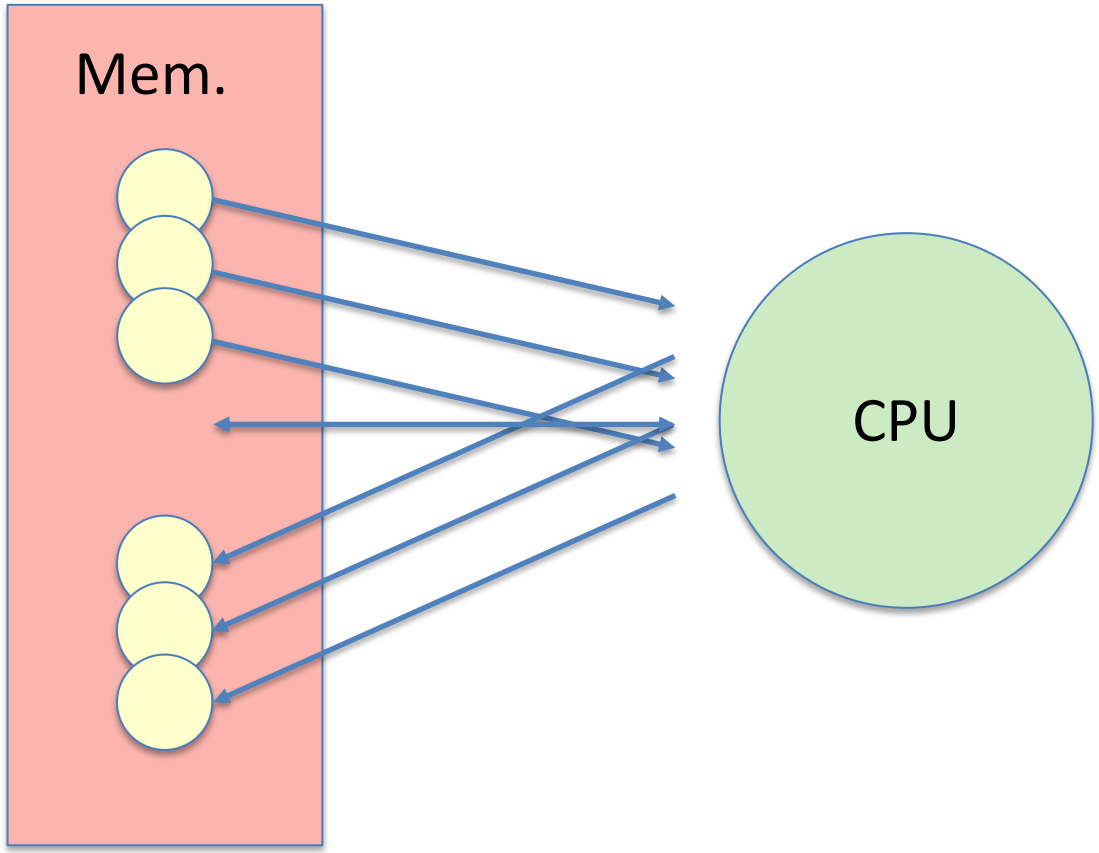
[1] Steven Kay, "perth road network analysis", 2011, <https://www.flickr.com/photos/stevefaeembra/6109227974>
[2] Grandjean Martin, "Introduction à la visualisation de données, l'analyse de réseau en histoire" *Geschichte und Informatik*, 18/19, 2015, 109-128.

Vertex-centric approach

- Also known as Think Like a Vertex (TLAV)
- Gather-Apply-Scatter (GAS) paradigm



Vertex-centric approach (continued)



- Gather: load requests to read inbound messages.
- Apply: updating status for each vertex.
- Scatter: write requests to the outbound links.

Challenges [3]

- Workload imbalance
  - A few vertices can have a very high degree while others have only a few neighbors.
- Data locality
  - Random memory access patterns can result when iterating neighboring vertices.
- Communication cost
  - Poor data locality tends to lead to frequent long latency off-chip memory accesses.

[3] A. Lumsdaine, D. Gregor, B. Hendrickson, and J. Berry, “Challenges in parallel graph processing.” Parallel Processing Letters, vol. 17, pp. 5–20, Mar. 2007.

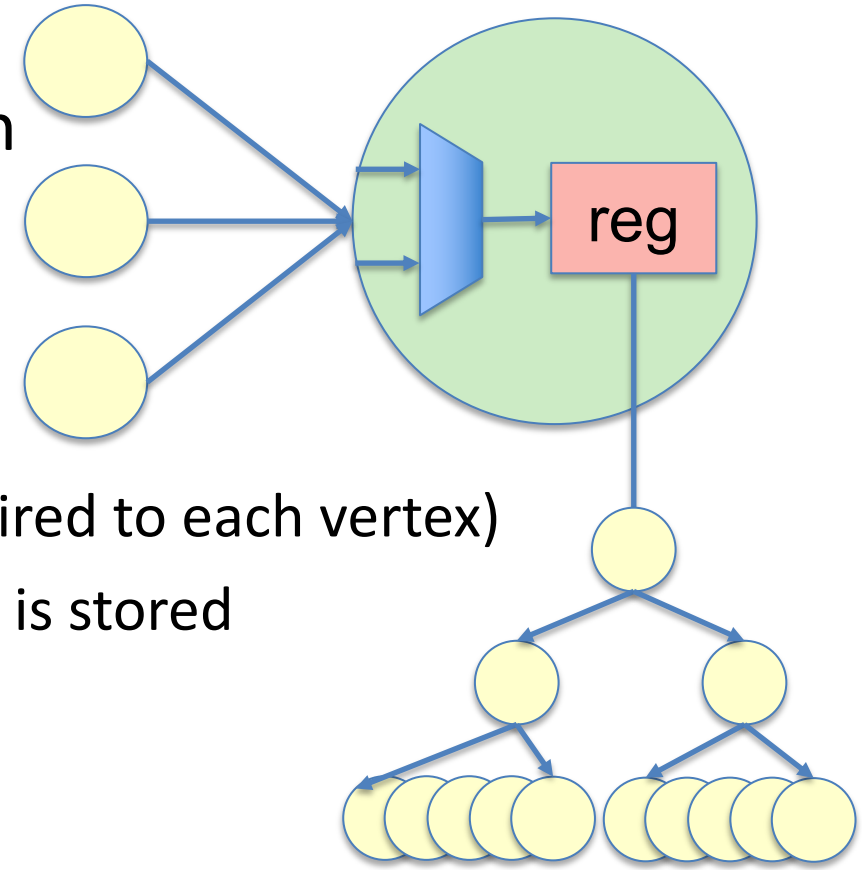
Related works

- Graphicionado [4]
  - Aims to reduce off-chip memory accesses with an on-chip eDRAM scratchpad.
- GraphH [5], GraphQ [6], and Tesseract [7]
  - Processing-in-Memory approaches are used to reduce access latency and improve bandwidth between storage and compute.
- PolyGraph [8]
  - Points out the importance of flexible graph processing e.g., flexibility in synchronicity as well as vertex scheduling.

[4] J. Ham, L. Wu, N. Sundaram, N. Satish, and M. Martonosi, “Graphicionado: A high-performance and energy-efficient accelerator for graph analytics,” in MICRO, 2016.
[5] Zhuo, C. Wang, M. Zhang, R. Wang, D. Niu, Y. Wang, and X. Qian,“GraphQ: Scalable PIM-based graph processing,” in MICRO, 2019.
[6] G. Dai, T. Huang, Y. Chi, J. Zhao, G. Sun, Y. Liu, Y. Wang, Y. Xie, and H. Yang, “GraphH: A processing-in-memory architecture for large-scale graph processing,”, TCAD, vol. 38, no. 4, pp. 640–653, 2019
[7] Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi, “A scalable processing-in-memory accelerator for parallel graph processing,” in ISCA, 2015.
[8] Dadu, S. Liu, and T. Nowatzki, “PolyGraph: Exposing the value of flexibility for graph processing accelerators,” in ISCA, 2021

Methodology: GraphWave

- Realization of the Think Like a Vertex abstraction
  - Message passing between vertices
  - Store intermediate status in each vertex
- Processing-at-memory
  - Scalable compute capabilities (one execution unit paired to each vertex)
  - The messages are processed where the vertex status is stored
- Efficient message propagation
  - Multi-level multicasting
- Results in propagation & reduction for all vertices in parallel



Our Approach: GraphWave (continued)

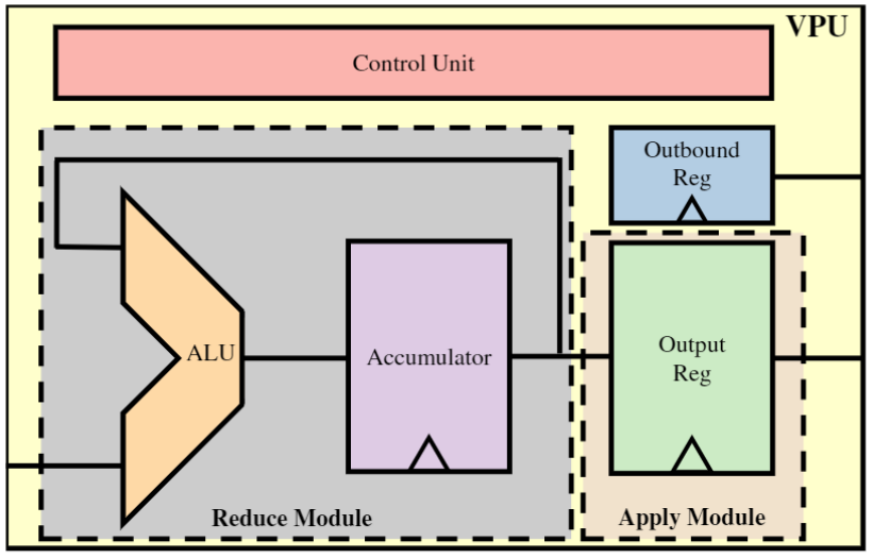
- Contributions
  - GraphWave microarchitecture
    - Highly-parallel vertex-level microarchitecture
  - Efficient multi-level data multicasting
    - Tackles both workload imbalance and high data communication cost

- Achieves one of the highest levels of efficiency
  - 63.94 GTEPS/W on a social network graph with 7k vertices & 9.4m edges

Microarchitecture

Vertex Processing Unit (VPU)

- Represents a single vertex in a graph
- Reduce Module
  - Processes received messages
  - Reduces the results
- Apply Module
  - Propagates the vertex data to other VPUs
  - Updates the vertex state at the end of a super-step

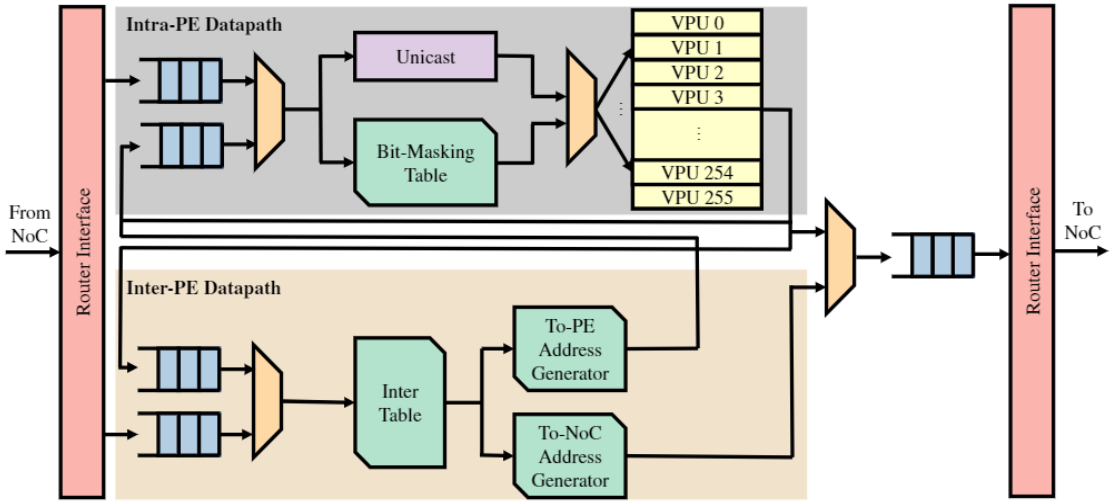


Vertex Processing Unit (VPU)

Microarchitecture

Processing Element (PE)

- 256 VPUs and routing tables
- Connected using Network-on-Chip (NoC)
- Intra-PE datapath
  - Computes messages inside the PE
- Inter-PE datapath
  - Generates packets for other VPUs

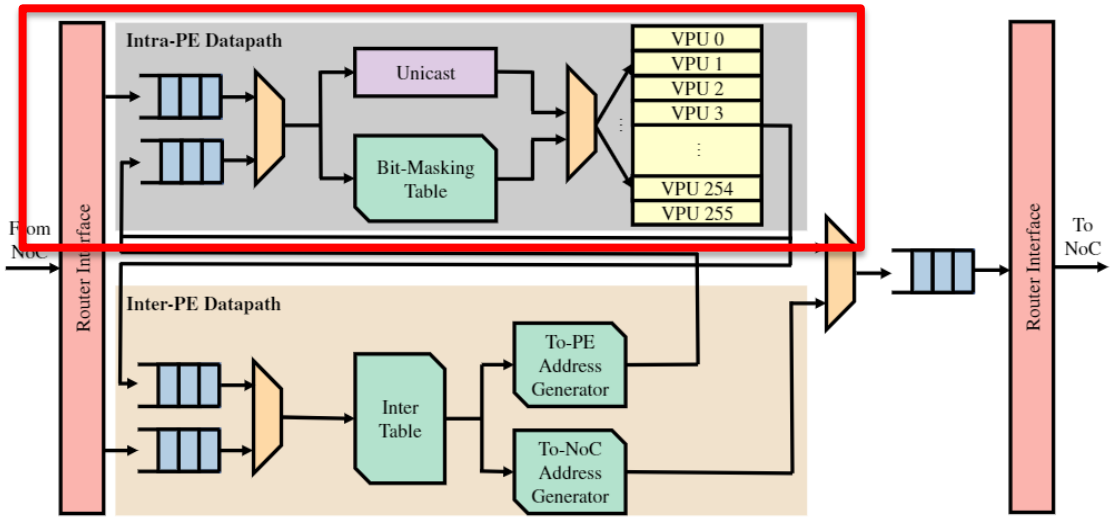


Processing Element (PE)

Microarchitecture

Intra-PE Datapath

- Receives messages from the NoC or inter-PE datapath
- Intra-PE multicasting
  - Relies on masking bits
  - Multicasts a message to multiple VPUs
- VPUs generate messages to inter-PE datapath or NoC

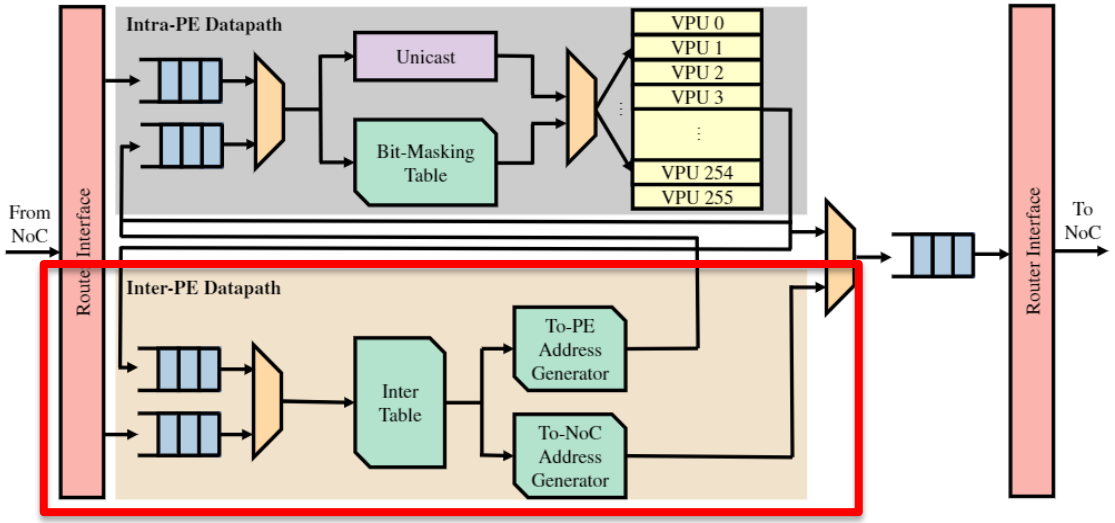


Processing Element (PE)

Microarchitecture

Inter-PE Datapath

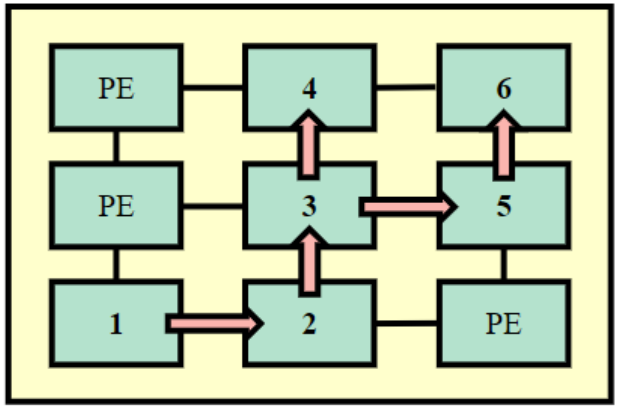
- Receives messages from the NoC or intra-PE datapath
- Inter table
  - Stores neighbors' destinations
- To-PE address generator
  - Generates a message for local computations
- To-NoC address generator
  - Generates packets for other PEs



Processing Element (PE)

Message Propagation

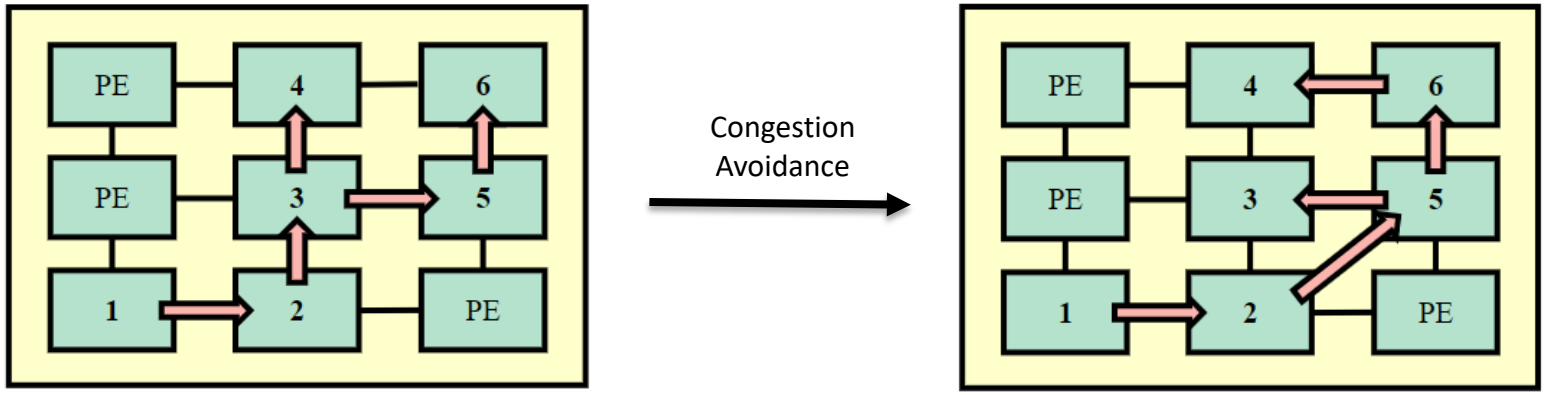
- Mapper statically predetermines the mapping and routing tables
  - Goal: minimize travel distance
  - Goal: increase throughput
- Inter-PE multicasting
  - Sends messages to its neighboring PEs
  - The neighboring PEs compute and relay messages to their neighbors



Message Propagation

Congestion Avoidance

- Center PEs can still suffer from network congestion
- Goal: Avoid overusing center PEs
- How: Considers travel distance and number of packets to be relayed
- In-flight reduce operation merges messages in a distributed way to reduce congestion



Message Propagation

Message Propagation & Congestion Avoidance

Experimental Setup

- GraphWave is implemented in SystemVerilog with power gating
  - Evaluated with Page Rank, Breadth First Search and Connected Components
- OpenSMART NoC [9] is used as the network connecting the PEs
- The system has 42 PEs with 22nm technology at 200 MHz
- Total area is less than 92mm^2

Graph	|V|	|E|	D_{avg}	domain
gnutella05 (GT)	9k	31k	3.4	web
wiki-vote (WV)	7k	103k	14.7	web
grid-yeast (GY)	6k	314k	52.3	biology
spam-detection (SD)	9k	506k	56.2	web
reality (RE)	7k	9,429k	1,344.4	social

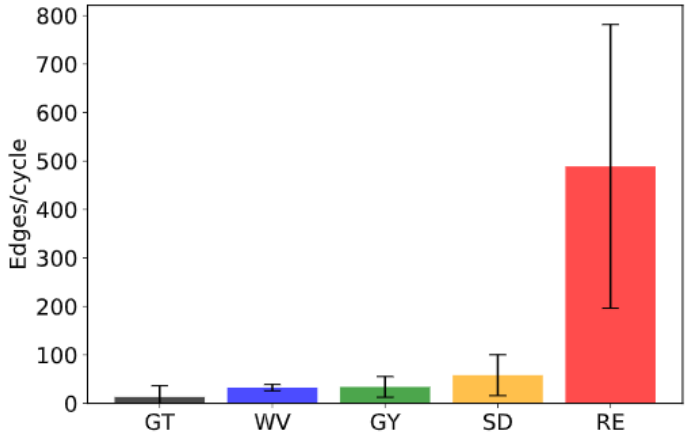
Input graph dataset

D_{avg} is average vertex degree

[9] H. Kwon and T. Krishna, "OpenSMART: Single-cycle multi-hop NoC generator in BSV and Chisel," 2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2017, pp. 195-204

Performance Evaluation

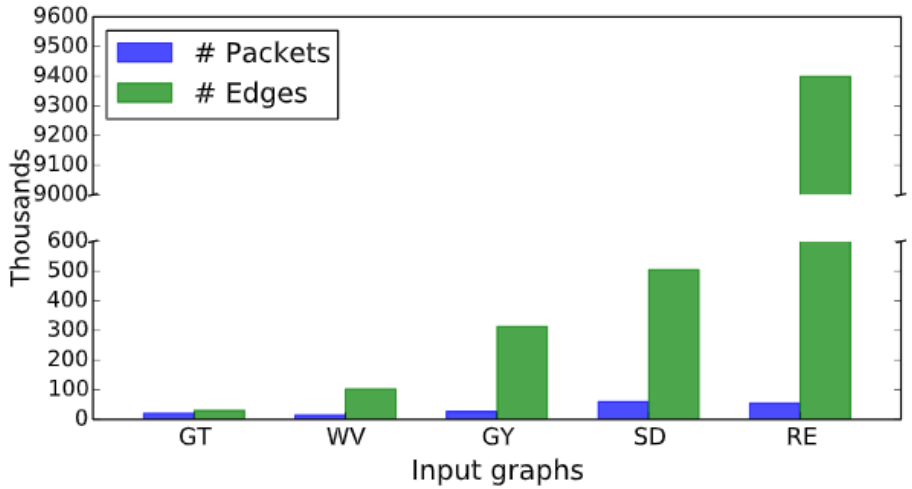
- Achieves 2.9-97.8 GTEPS depending on the edge count
- GraphWave can take advantage of higher edge counts and scale with the benchmark
- Compared to the version without intra-PE and inter-PE multicasting
  - Intra-PE muticasting has 36x higher throughput
  - Intra-PE and inter-PE multicasting achieves 39x higher throughput



Edges/cycle for each input graph with standard deviation

Packet vs Edge Count

- On the reality graph (RE), 200k packets are transferred for 9.4M edges
  - Eliminates 97.8% of messages from being transmitted through the NoC compared to the version without inter-PE and intra-PE multicasting
- Network congestion is significantly reduced due to inter-PE multicasting that multicasts a message to multiple VPUs



#Edges vs #packets

Storage Usage & Scalability

- A graph with a high degree tends to have higher compression rate
  - Lower streaming time and fewer hardware resources used
- Maximum number of packets = VPU-count · PE-count
  - Significantly lower than edge count in dense graphs

Graph	Savings
gnutella05 (GT)	1.23%
wiki-vote (WV)	31.45%
grid-yeast (GY)	23.84%
spam-detection (SD)	30.19%
reality (RE)	83.93%

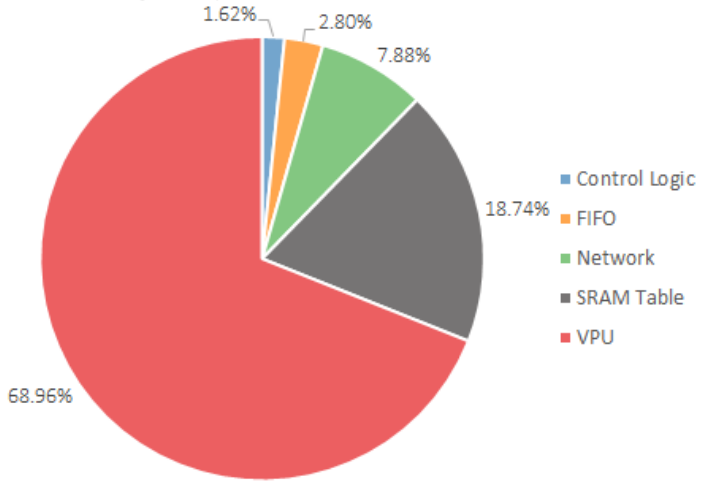
Storage savings for the input graphs

|V|	|E|	D_{avg}	GTEPS	#Packets	Savings
4k	0.04M	10	1.74	27,957	-25.69%
4k	0.40M	100	8.33	59,548	37.95%
4k	4.00M	1,000	83.80	60,000	93.80%
4k	8.00M	2,000	166.61	60,000	96.82%
4k	16.00M	3,999	377.01	60,000	98.41%

Scalability with large degree synthetic graphs
D_{avg} is average vertex degree

Power Breakdown

- Power consumption ranges from 1.36W to 2.05W
- For the reality graph (RE), GraphWave consumes 1.54W
  - Leads to 63.94 GTEPS/W, one of the highest levels of efficiency compared to the current state-of-the-art graph accelerators [6, 8]



Power breakdown of GraphWave based on reality graph

[6] G. Dai, T. Huang, Y. Chi, J. Zhao, G. Sun, Y. Liu, Y. Wang, Y. Xie, and H. Yang, "GraphH: A processing-in-memory architecture for large-scale graph processing,", TCAD, vol. 38, no. 4, pp. 640–653, 2019
[8] Dadu, S. Liu, and T. Nowatzki, "PolyGraph: Exposing the value of flexibility for graph processing accelerators," in ISCA, 2021

Conclusion

- Efficiently handles workload imbalance
  - A VPU needs to send only one message regardless of the number of outbound edges due to multicasting
- Eliminates frequent irregular memory accesses and minimizes communication latency
  - Achieved by the inter-PE and intra-PE multicasting techniques
- Attains high performance and efficiency
  - Up to 97.8 GTEPS at 63.94 GTEPS/W

Thank you