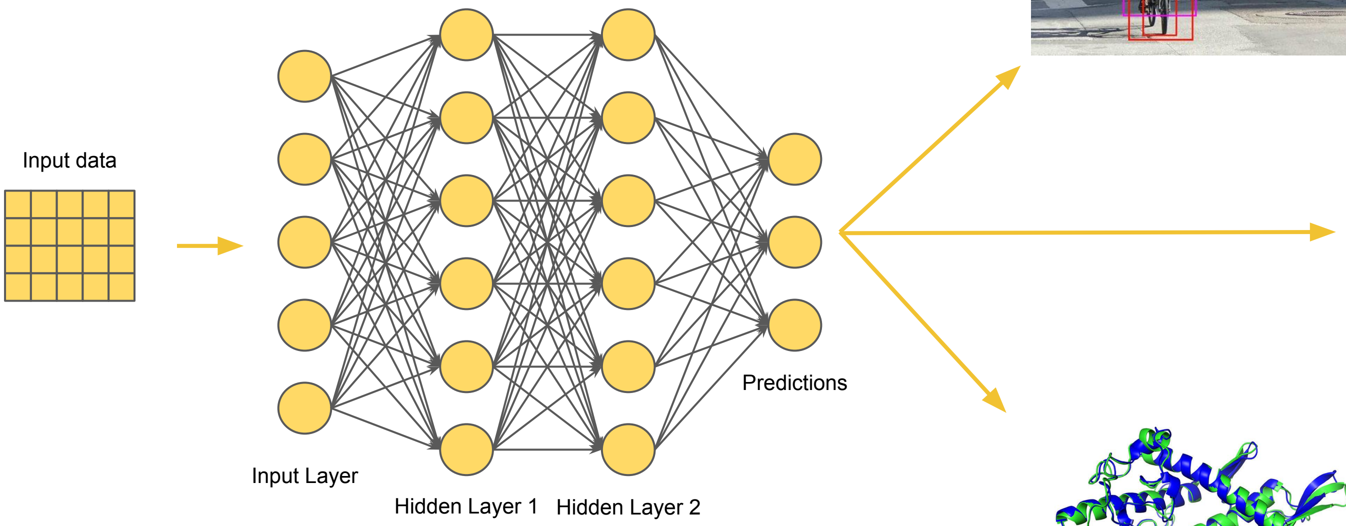
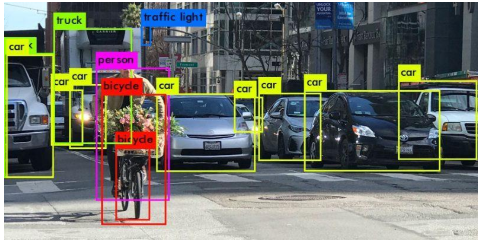
Dorylus: Affordable, Scalable, and Accurate GNN Training with Distributed CPU Servers and Serverless Threads

John Thorpe*, Yifan Qiao*, Jonathan Eyolfson, Shen Teng, Guanzhou Hu, Zhihao Jia, Jinliang Wei, Keval Vora, Ravi Netravali, Miryung Kim, Harry Xu

UCLA University of Wisconsin CMU Google Brain Simon Fraser Princeton University

Machine Learning



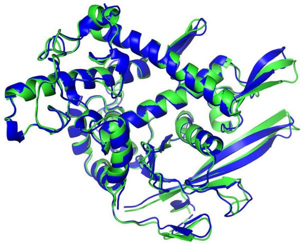


In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called "showers".

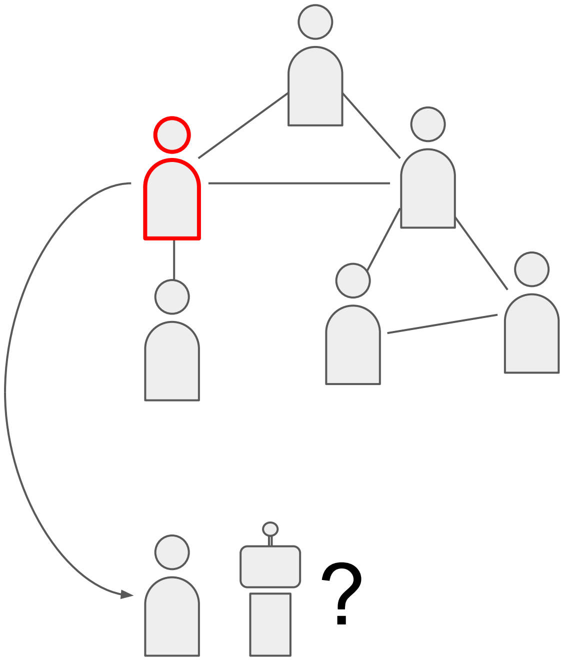
What causes precipitation to fall?
gravity

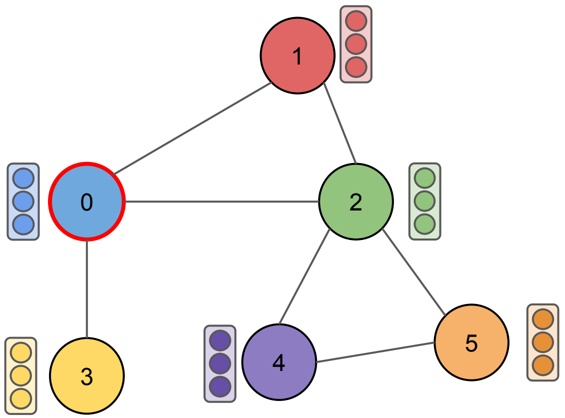
What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud



Graph Neural Networks

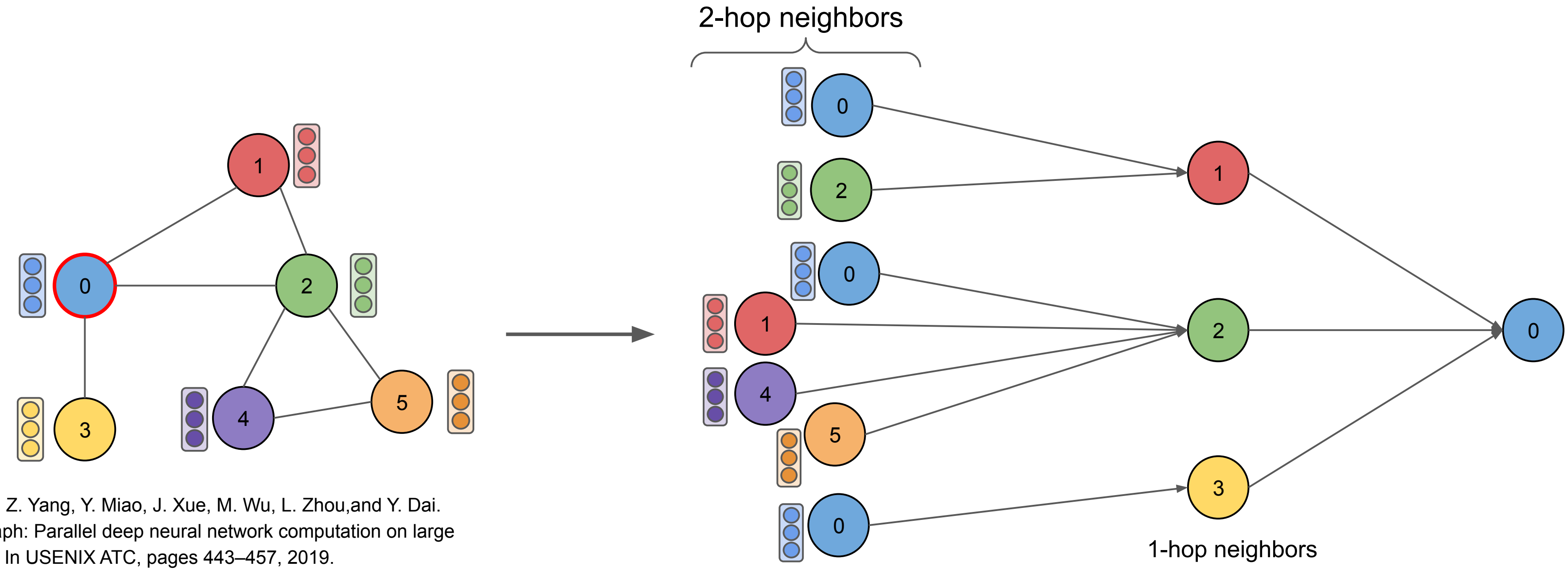




Goals:

- Affordability
- Scalability
- Performance

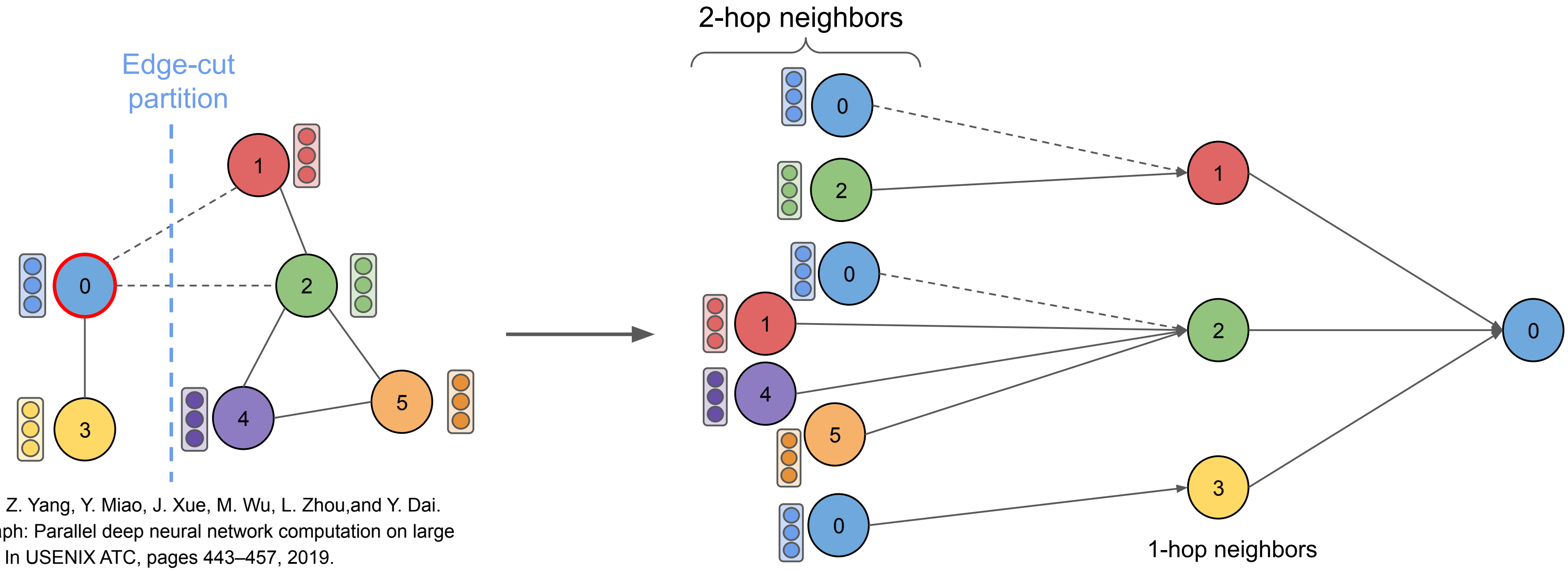
Stages of a Graph Neural Network



[1] L. Ma, Z. Yang, Y. Miao, J. Xue, M. Wu, L. Zhou,and Y. Dai. NeuGraph: Parallel deep neural network computation on large graphs. In USENIX ATC, pages 443–457, 2019.

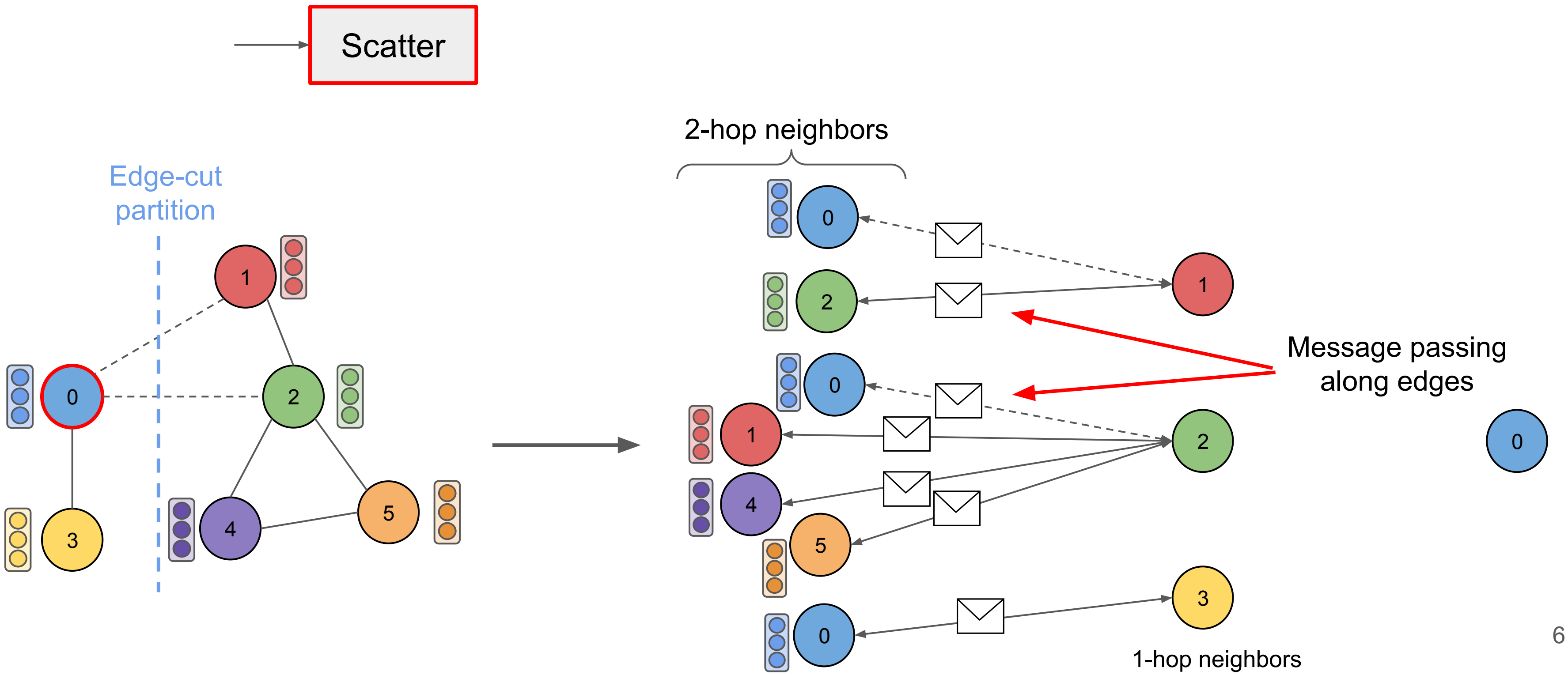
UCLA

Stages of a Graph Neural Network

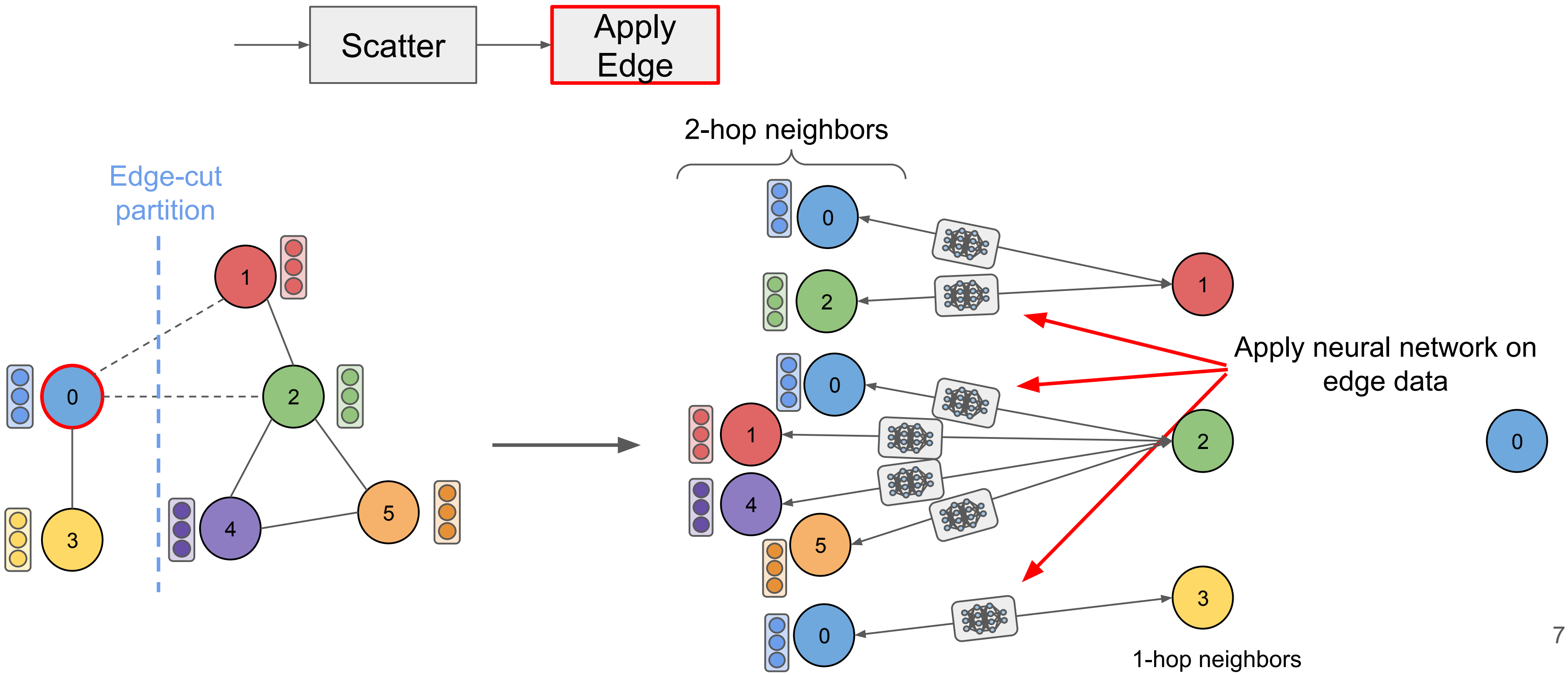


[1] L. Ma, Z. Yang, Y. Miao, J. Xue, M. Wu, L. Zhou,and Y. Dai. NeuGraph: Parallel deep neural network computation on large graphs. In USENIX ATC, pages 443–457, 2019.

Stages of a Graph Neural Network

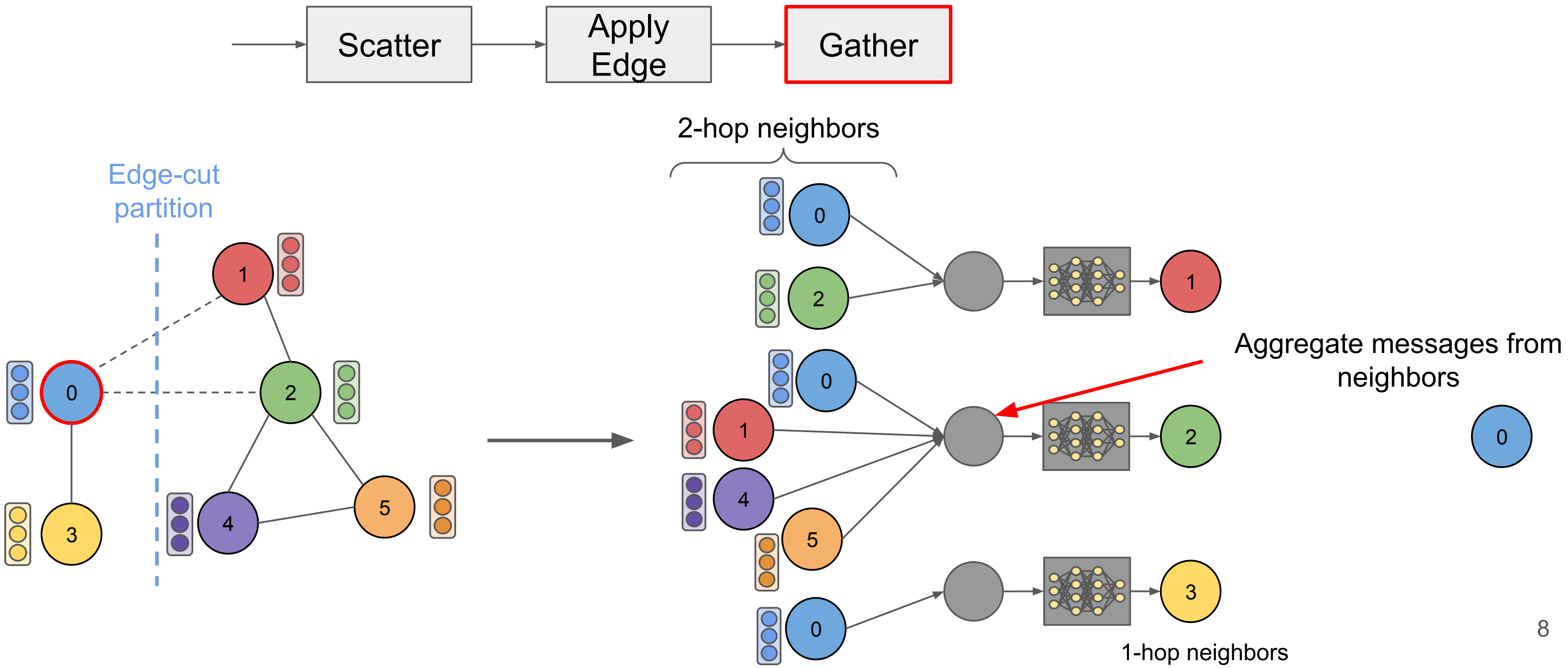


Stages of a Graph Neural Network

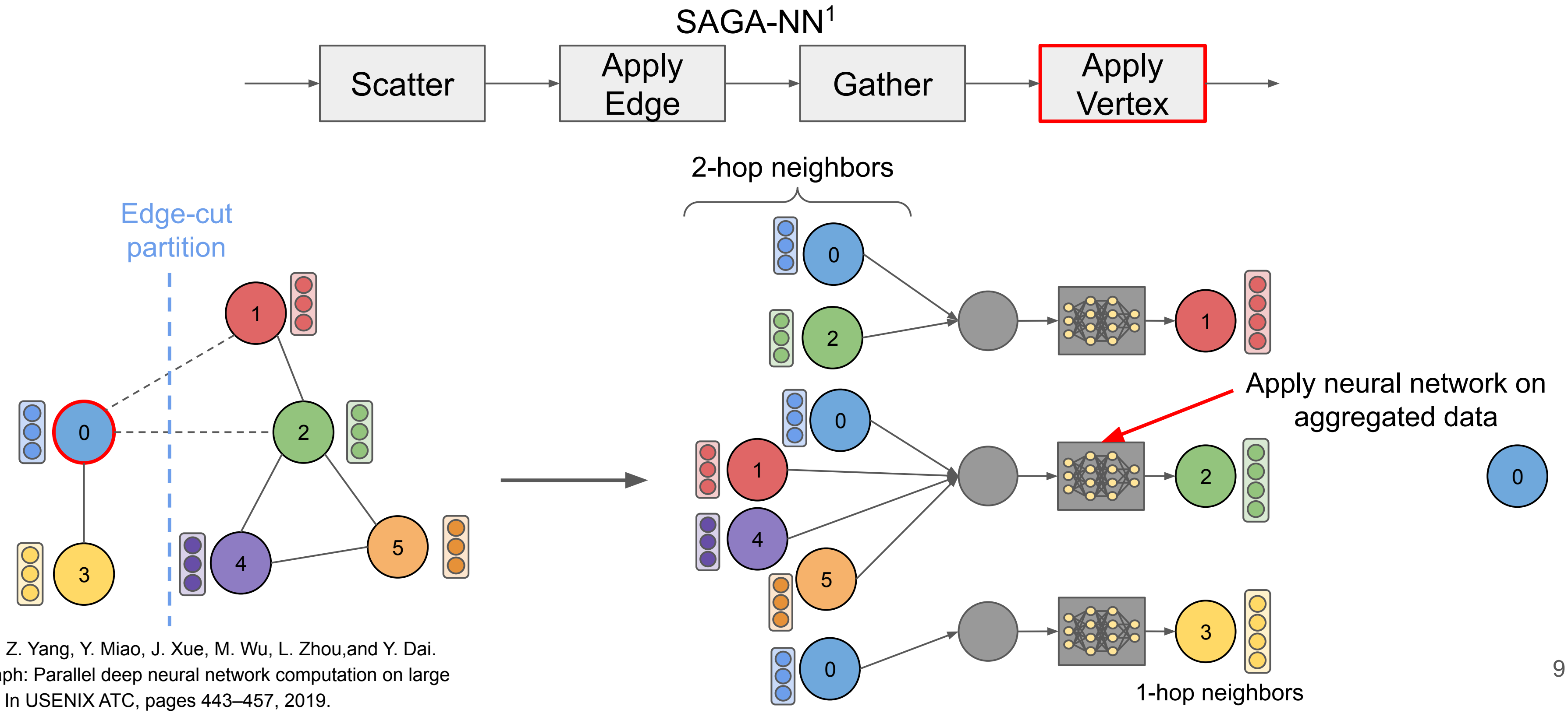


UCLA

Stages of a Graph Neural Network

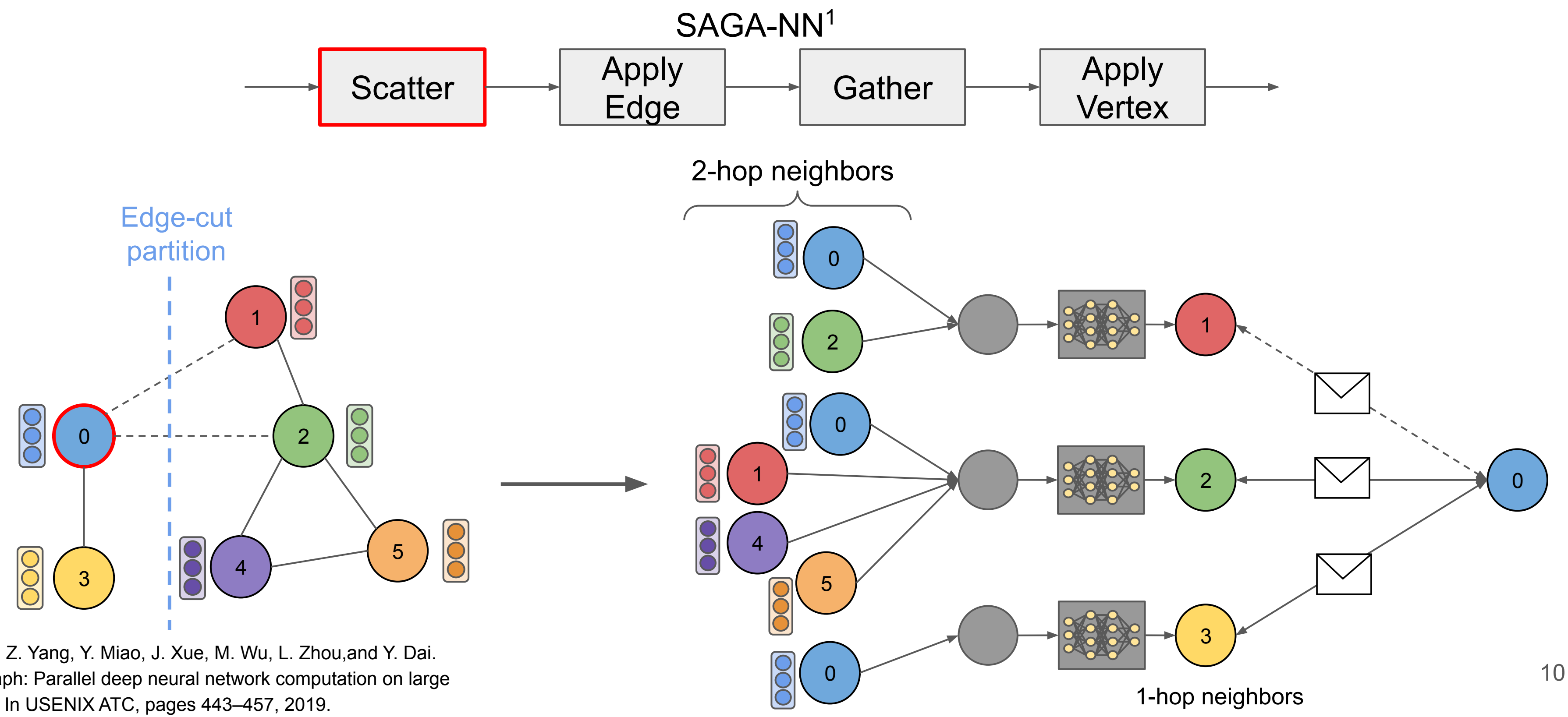


Stages of a Graph Neural Network



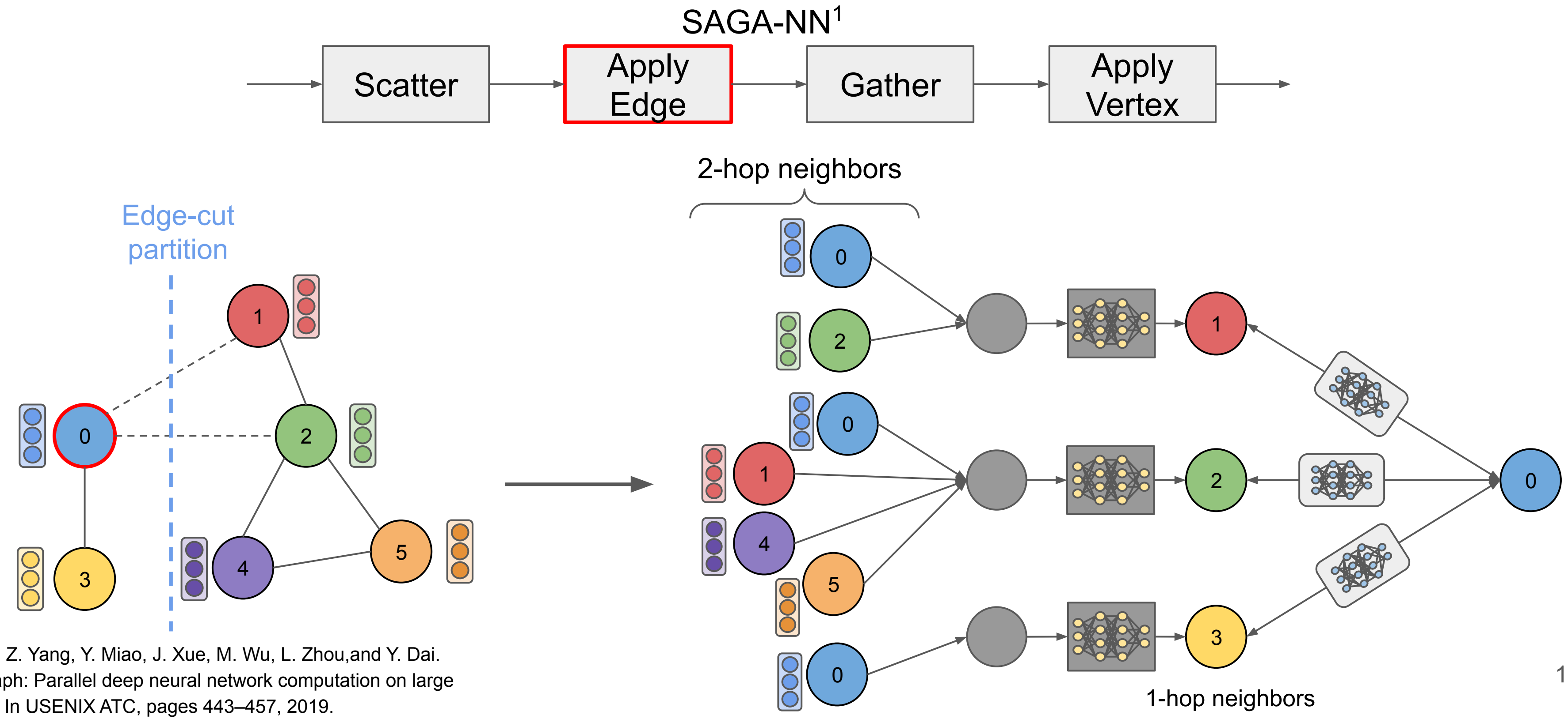
[1] L. Ma, Z. Yang, Y. Miao, J. Xue, M. Wu, L. Zhou,and Y. Dai. NeuGraph: Parallel deep neural network computation on large graphs. In USENIX ATC, pages 443–457, 2019.

Stages of a Graph Neural Network



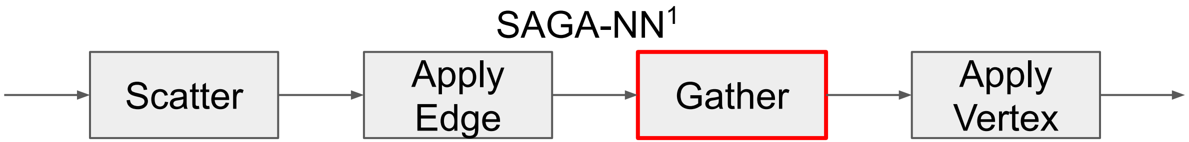
[1] L. Ma, Z. Yang, Y. Miao, J. Xue, M. Wu, L. Zhou,and Y. Dai. NeuGraph: Parallel deep neural network computation on large graphs. In USENIX ATC, pages 443–457, 2019.

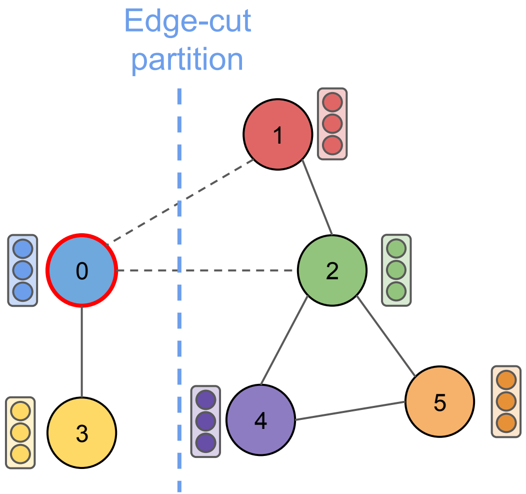
Stages of a Graph Neural Network

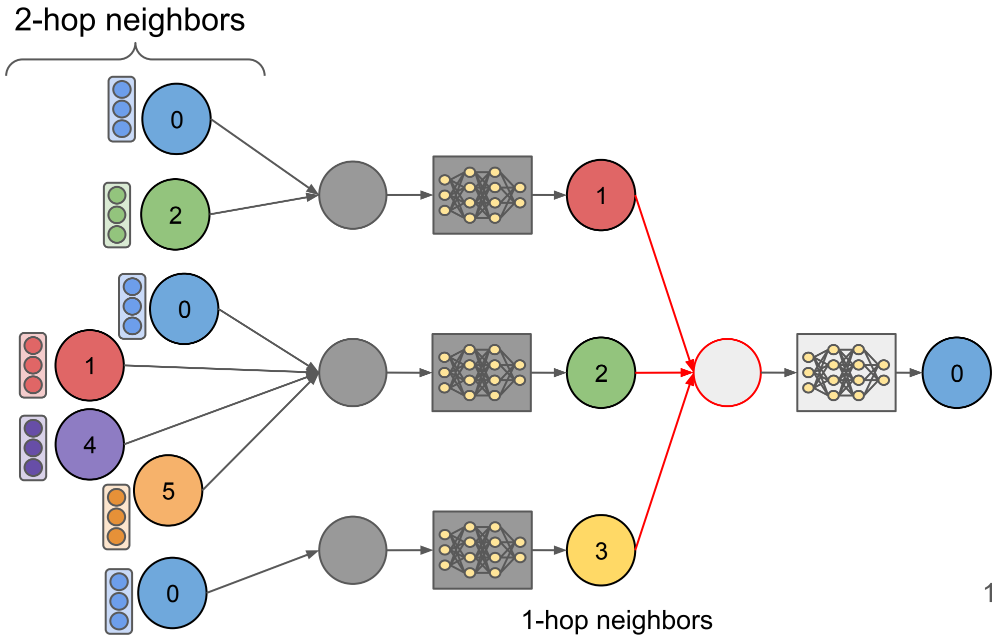


[1] L. Ma, Z. Yang, Y. Miao, J. Xue, M. Wu, L. Zhou,and Y. Dai. NeuGraph: Parallel deep neural network computation on large graphs. In USENIX ATC, pages 443–457, 2019.

Stages of a Graph Neural Network

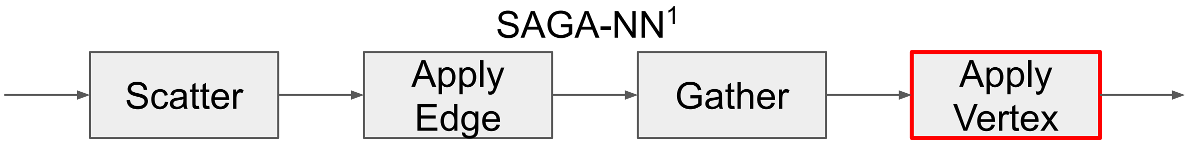






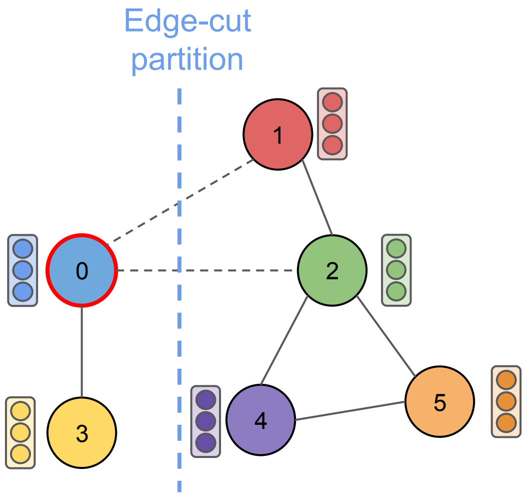
[1] L. Ma, Z. Yang, Y. Miao, J. Xue, M. Wu, L. Zhou,and Y. Dai. NeuGraph: Parallel deep neural network computation on large graphs. In USENIX ATC, pages 443–457, 2019.

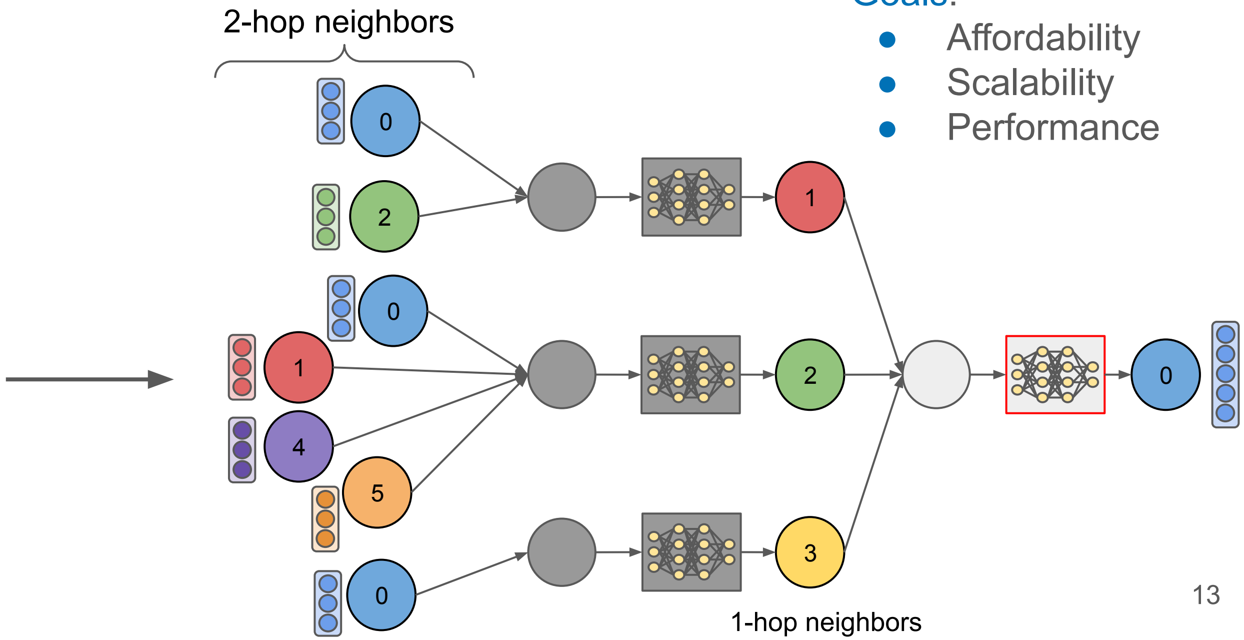
Stages of a Graph Neural Network



Goals:

- Affordability
- Scalability
- Performance

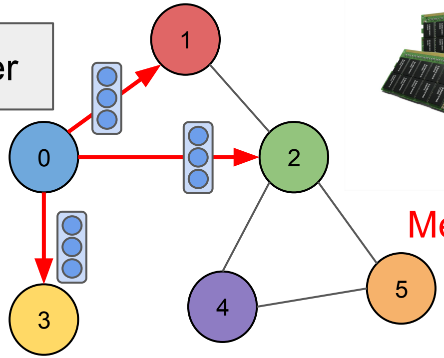


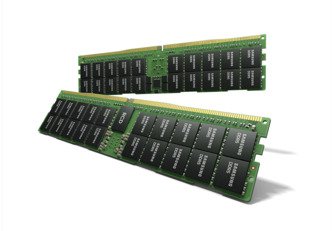


[1] L. Ma, Z. Yang, Y. Miao, J. Xue, M. Wu, L. Zhou,and Y. Dai. NeuGraph: Parallel deep neural network computation on large graphs. In USENIX ATC, pages 443–457, 2019.

GNNs Comprise Very Different Workloads

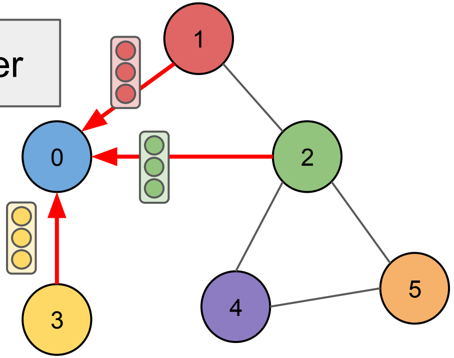
Scatter





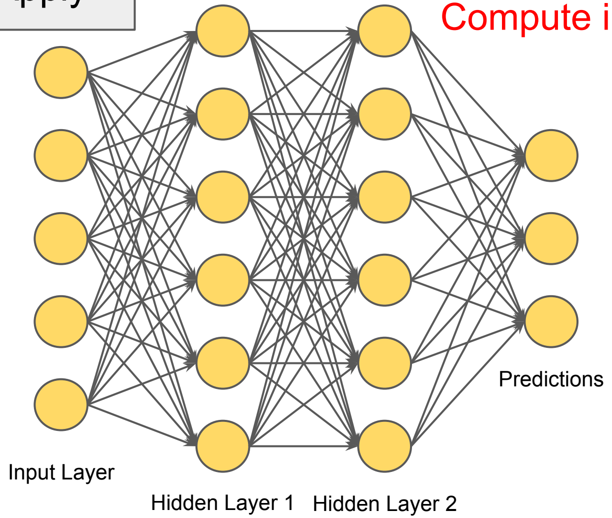
Memory intensive!

Gather



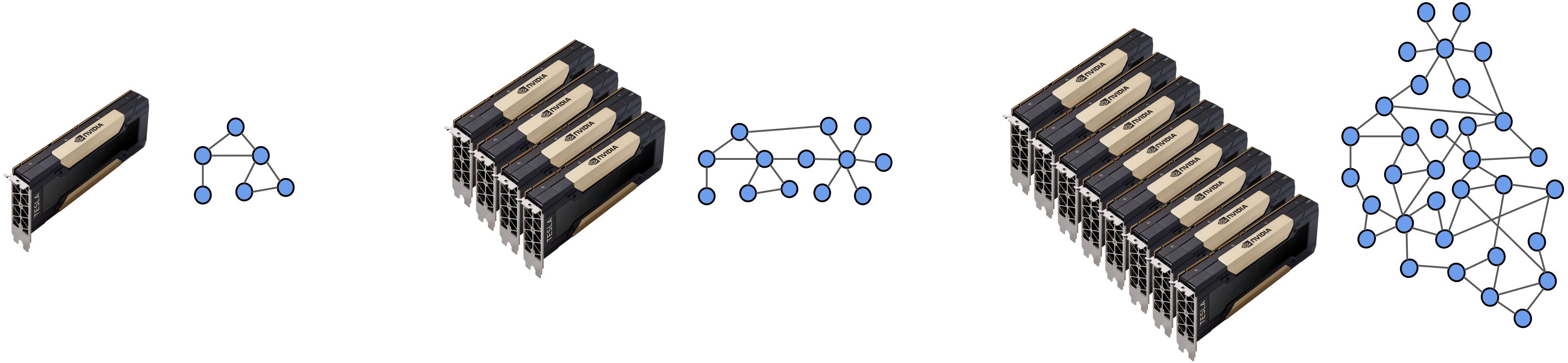
Apply

Compute intensive!

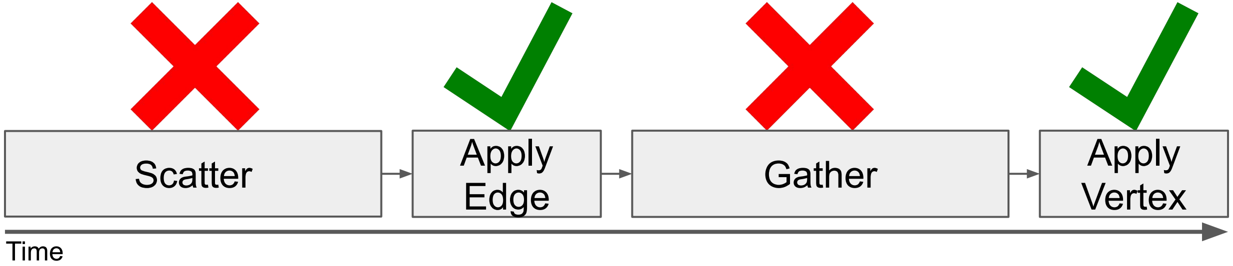


GPUs Are Not a Good Fit for Graph Operations

Limited device memory + large adjacency matrix = poor scalability!



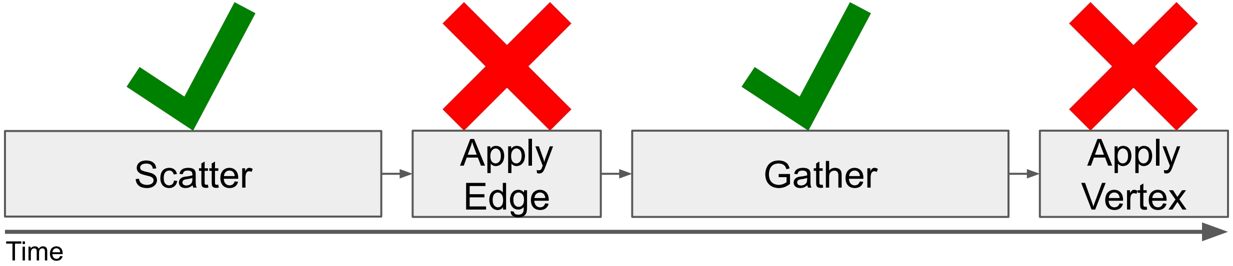
GPUs Are Not a Good Fit for Graph Operations



GPUs work very well for tensor computation

- Less efficient for Gather
- Idle for Scatter across partitions

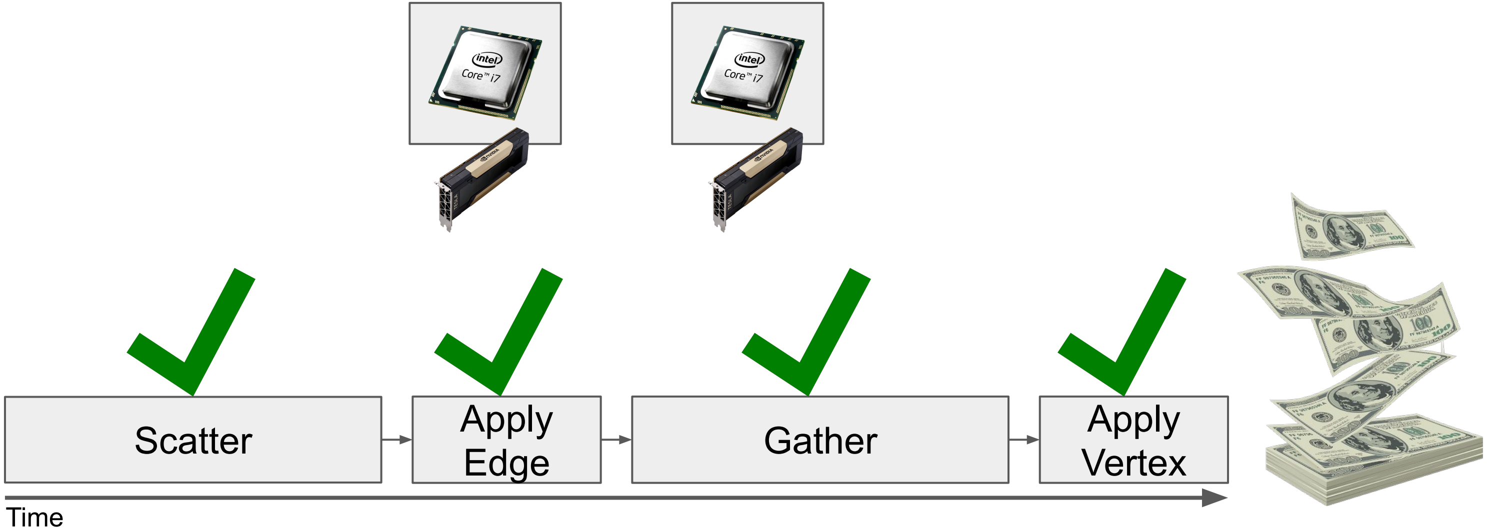
CPUs Are Not Efficient for Tensor Workloads



CPUs provide scalability for graph operations

- Not optimized for highly parallel computation

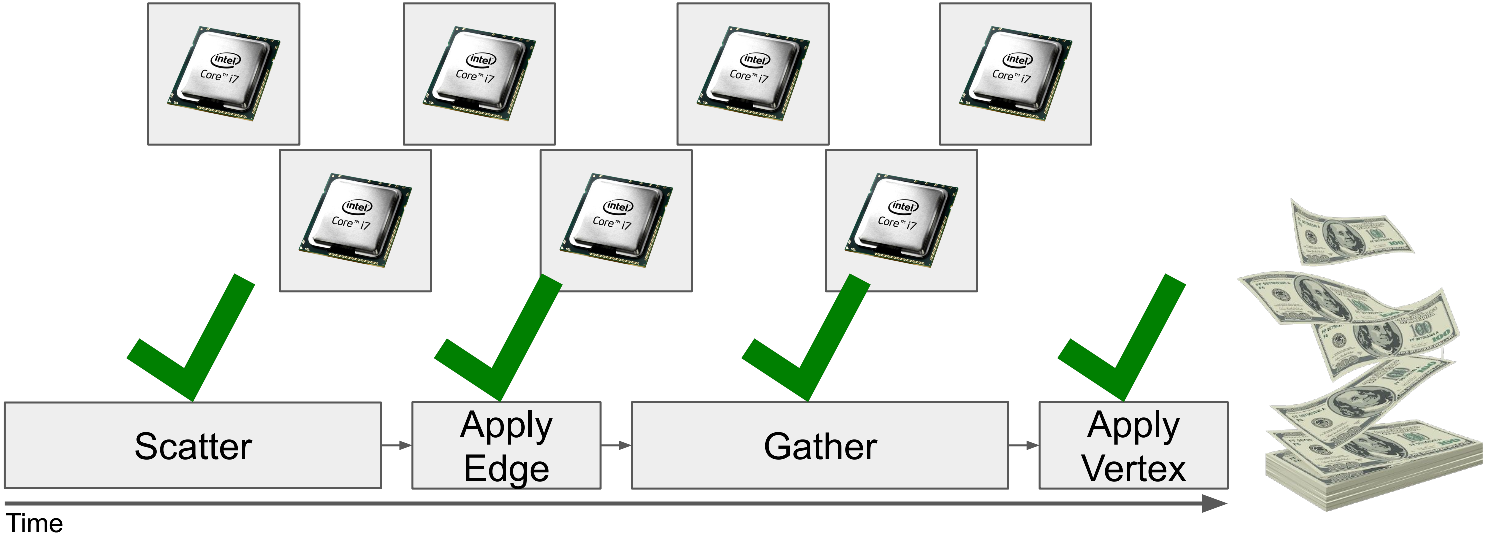
Combining CPUs and GPUs is Cost-Ineffective



Get the scalability of CPUs with performance of GPUs

- GPUs under-utilized during graph operations

Using Many CPU Servers Can Still Be Expensive



Allocating many CPU servers increases parallelism at the expense of cost

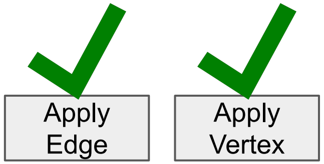
- Many unnecessary resources allocated along with CPU machines

Key Insight: Serverless Fits Our Goals

Serverless: cloud execution model that provisions resources on demand

Highly scalable interface fits needs of tensor computation

- Invoke thousands of threads in parallel

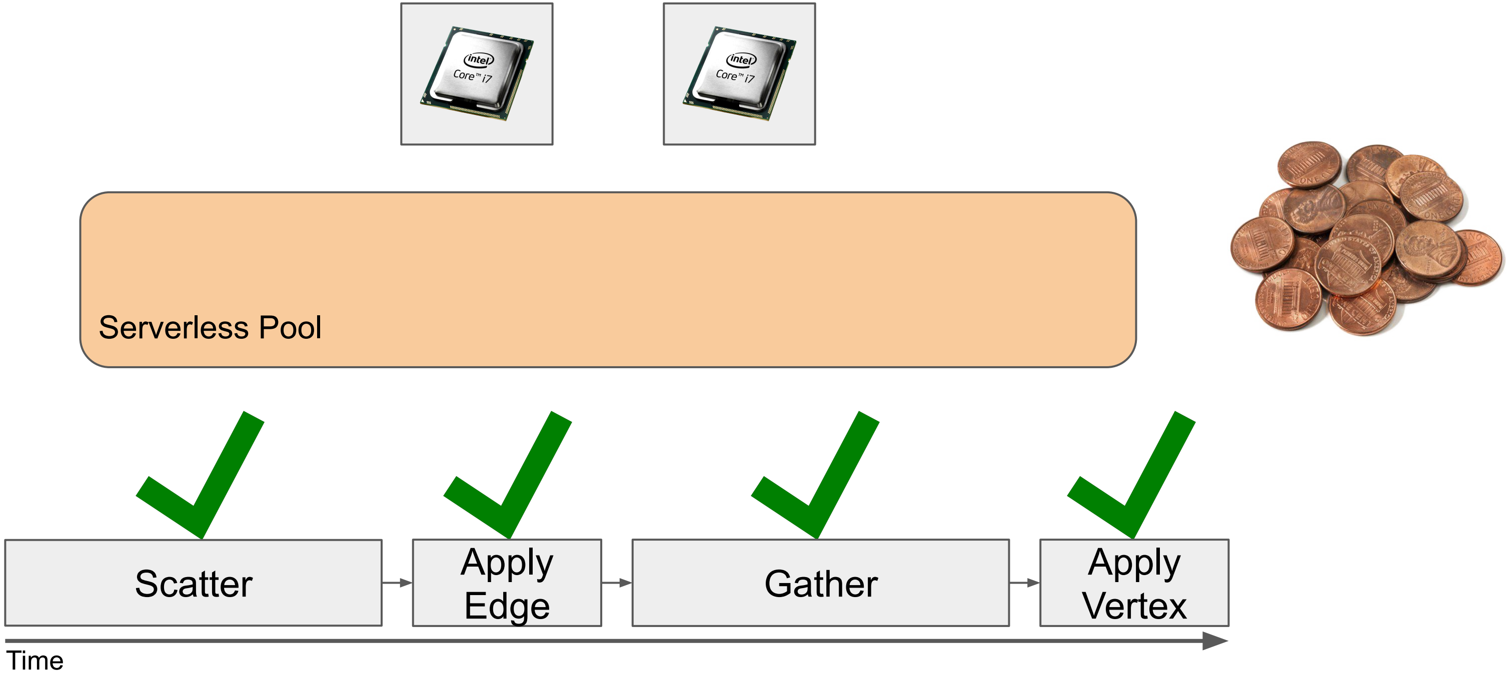


Low-cost, flexible pricing model

Fine grained: Only pay for compute resources on millisecond basis

Provide high **performance-per-dollar (value)**

Serverless Achieves Low-Cost, Scalable Efficiency



Challenges with Using Serverless

- Each thread has limited resources
    - Weak CPU, limited memory
- Limited network
    - Design to handle light asynchronous tasks

UCLA

Challenge 1: Limited Resources

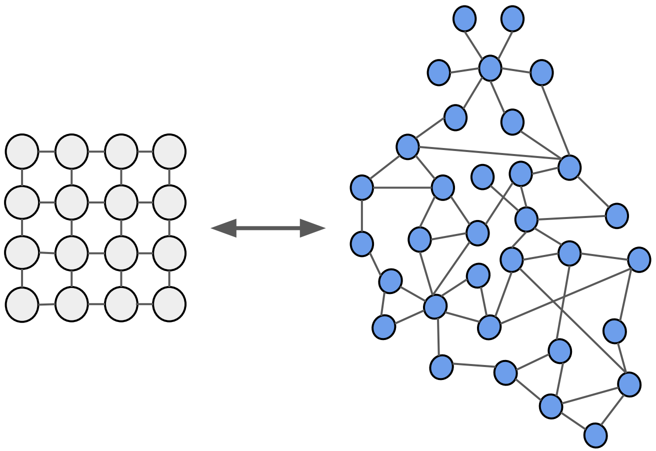
Each serverless thread has limited memory and compute

- Better for highly parallel computation without dependencies

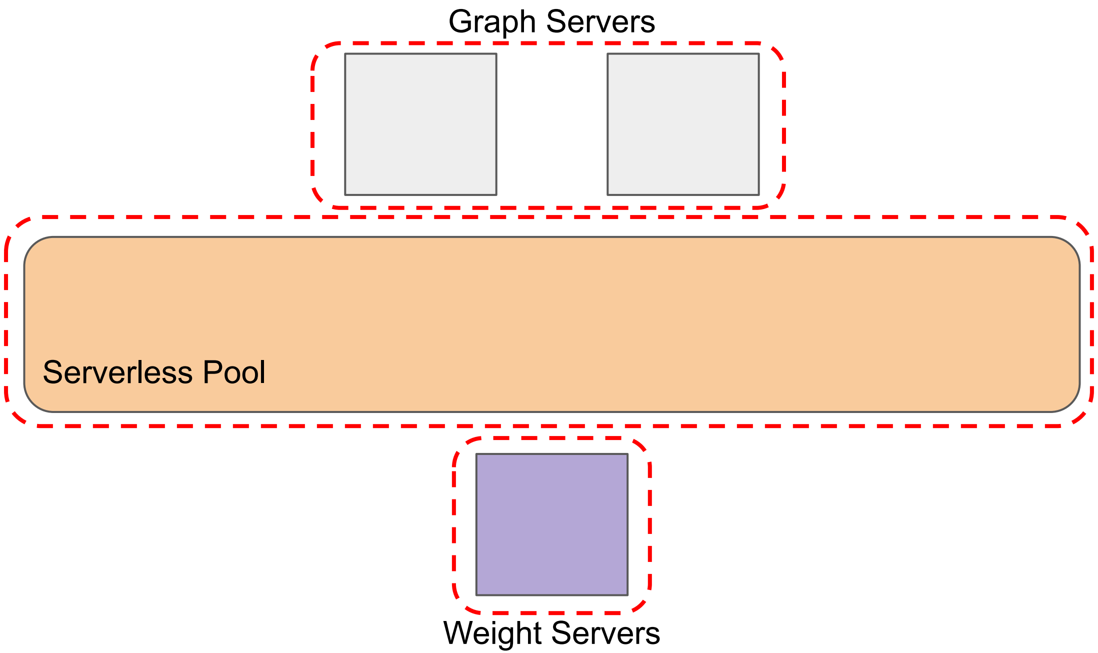
Solution: Computation Separation

Separation of graph and tensor computation

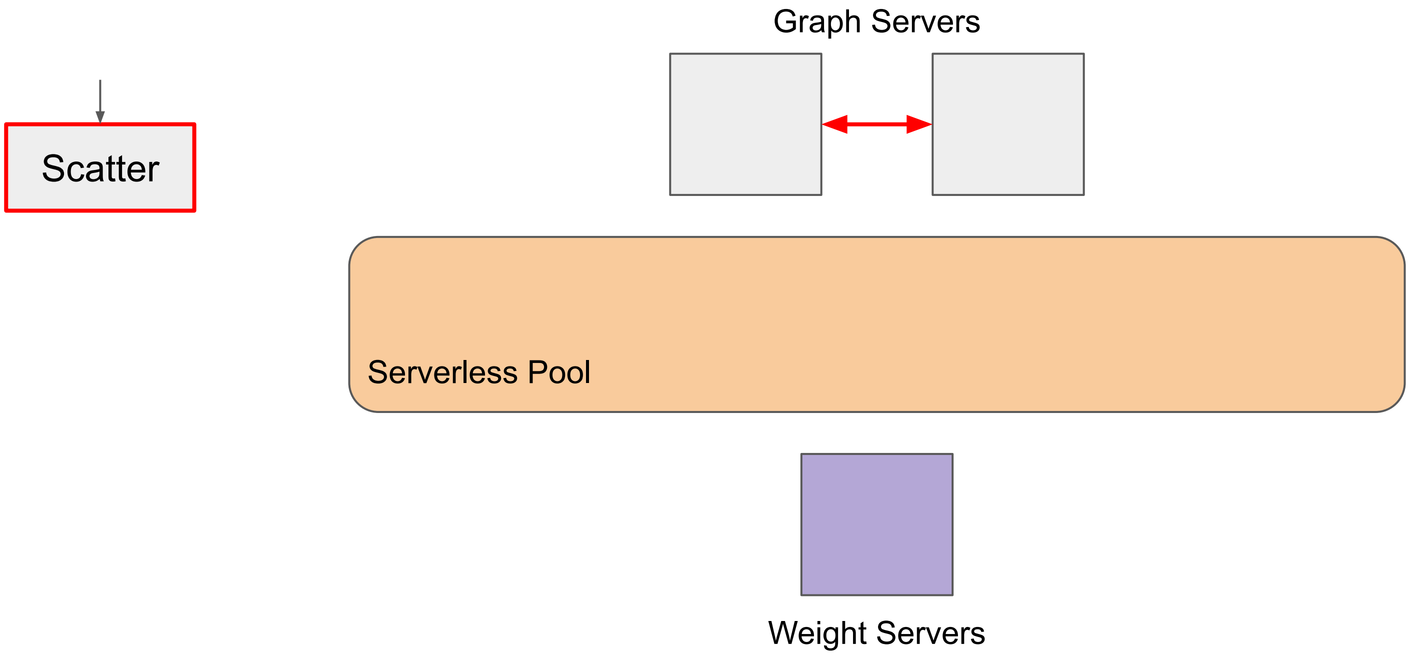
- Scale graph operations on CPU servers



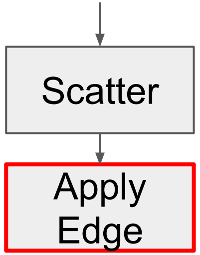
Dorylus Architecture

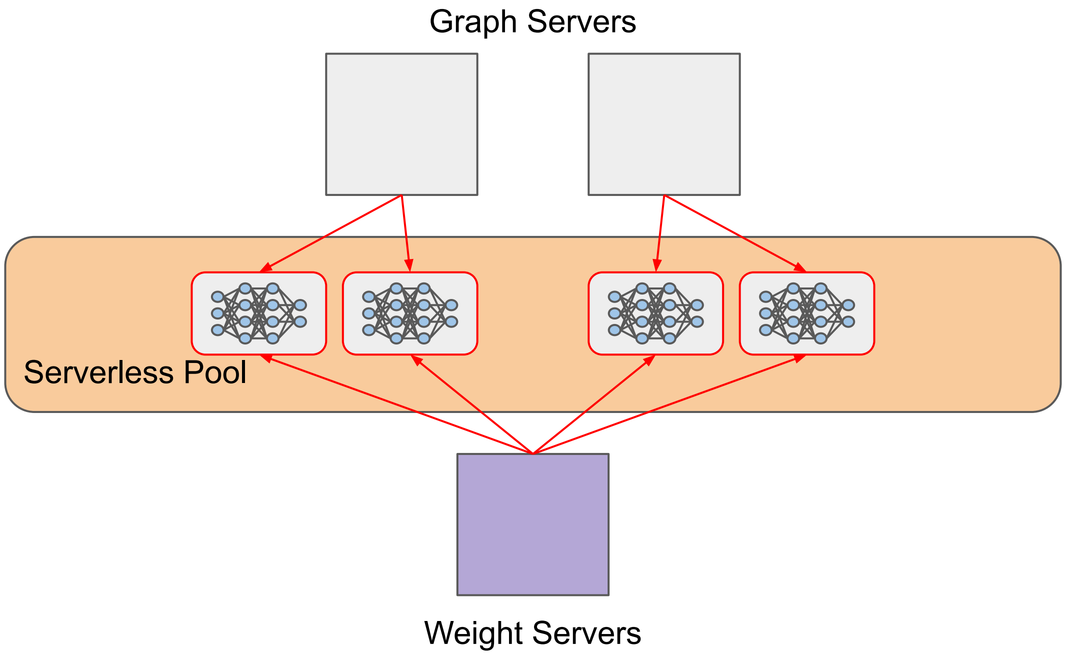


Dorylus Architecture

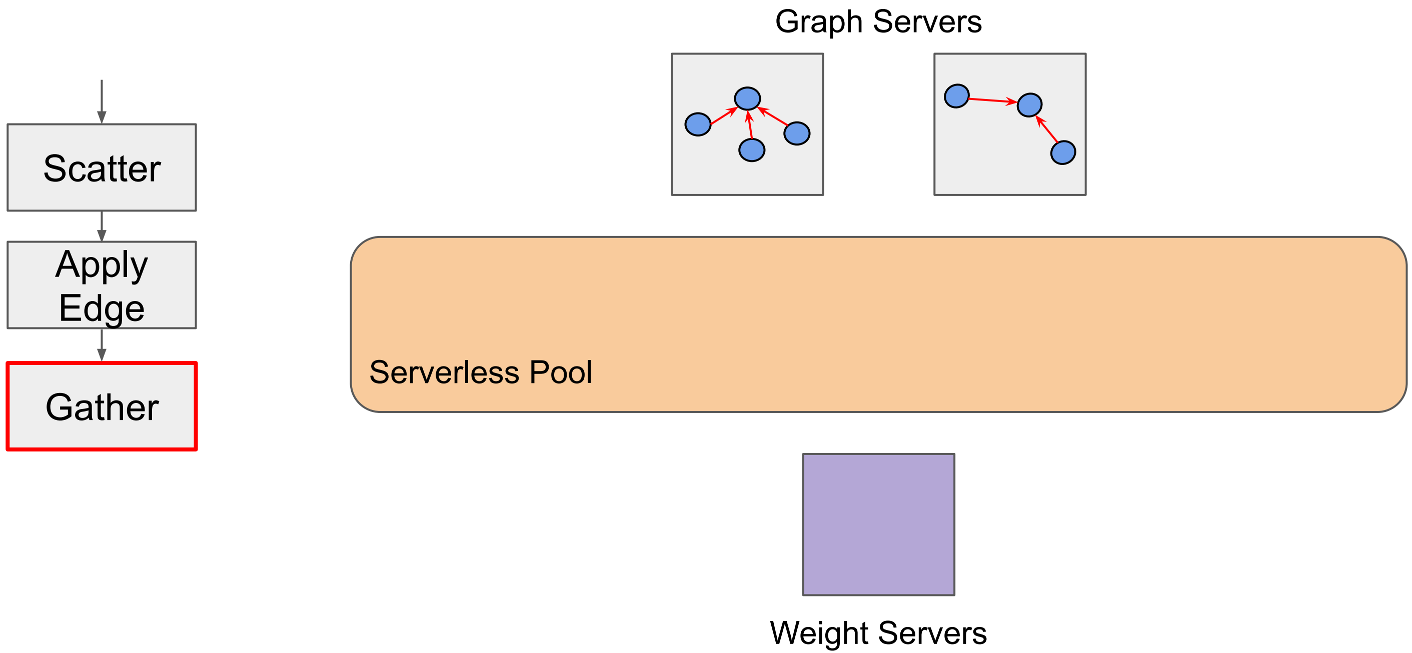


Dorylus Architecture

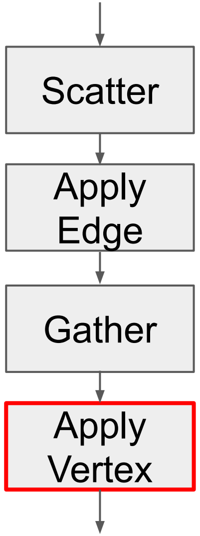


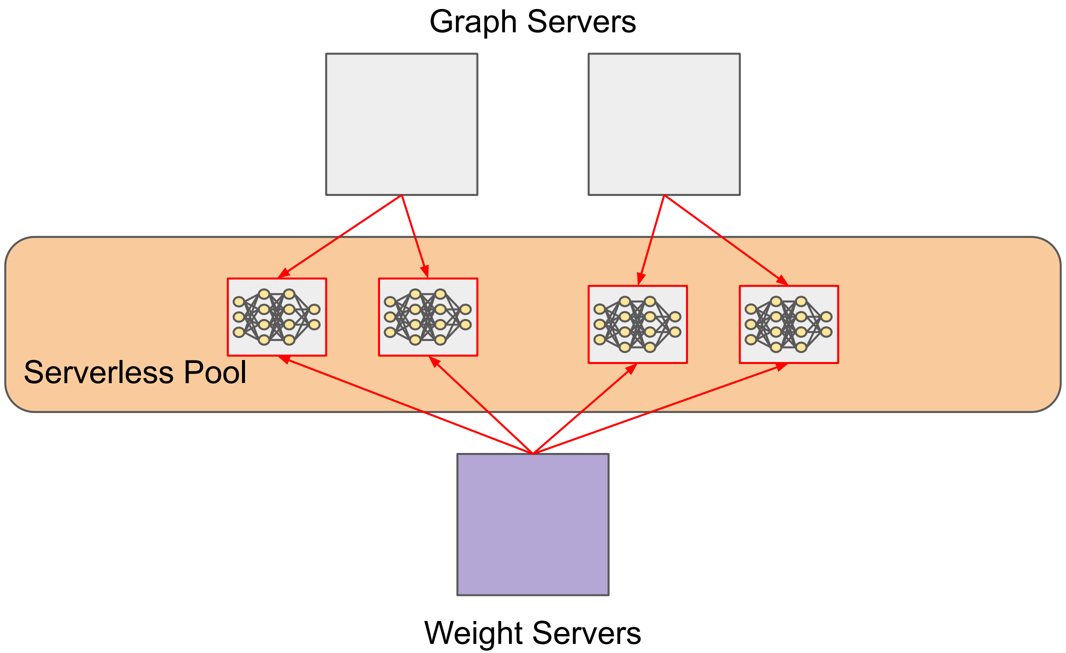


Dorylus Architecture

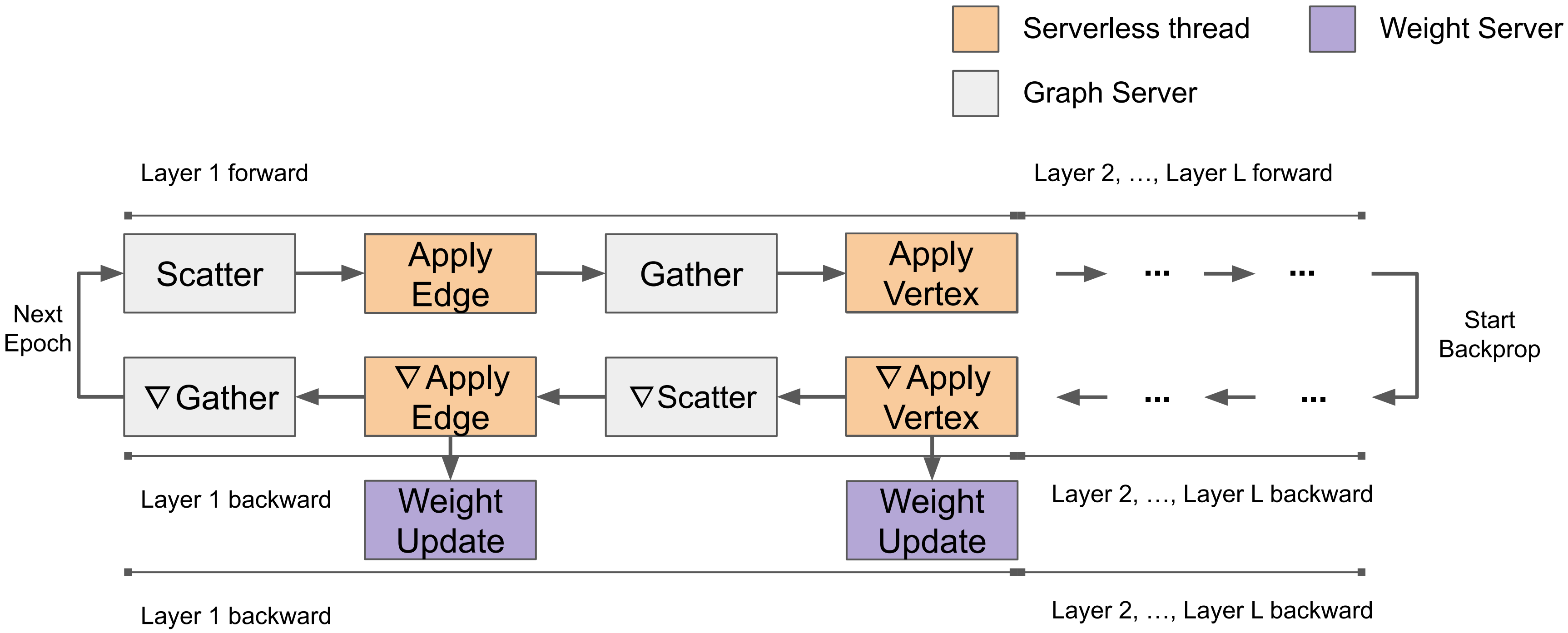


Dorylus Architecture





Flow of Decomposed Tasks



Challenge 2: Limited Network

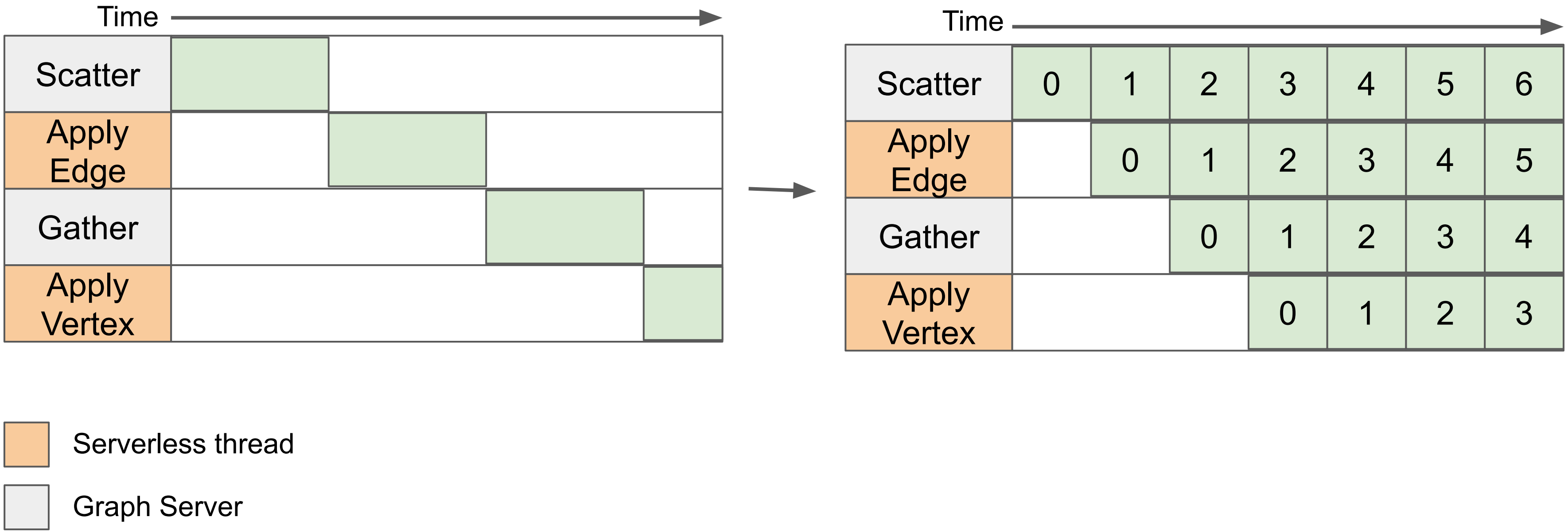
Network latency has high overhead

- Significantly hinders performance

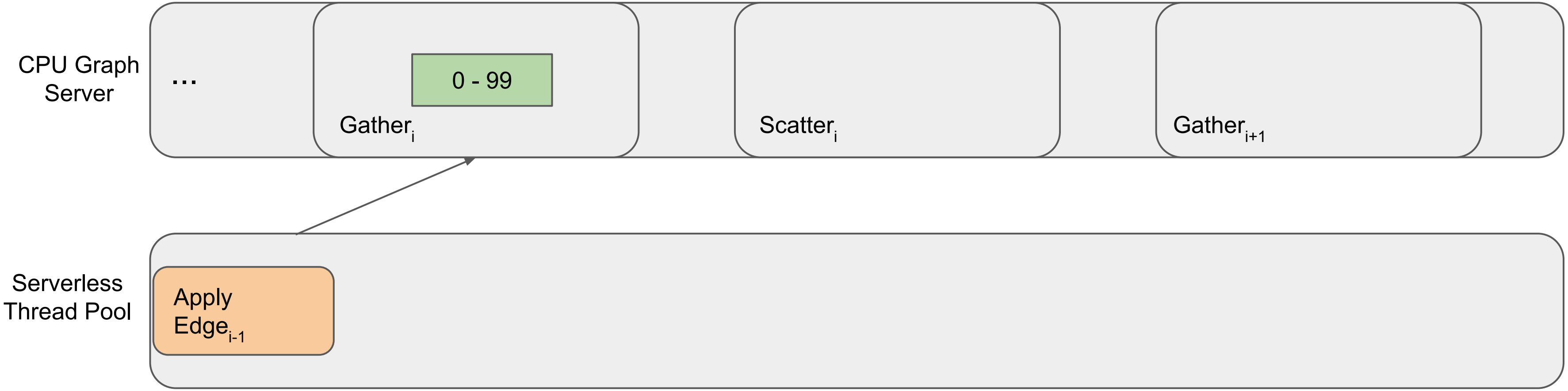
Running sequentially leads to stalls

UCLA

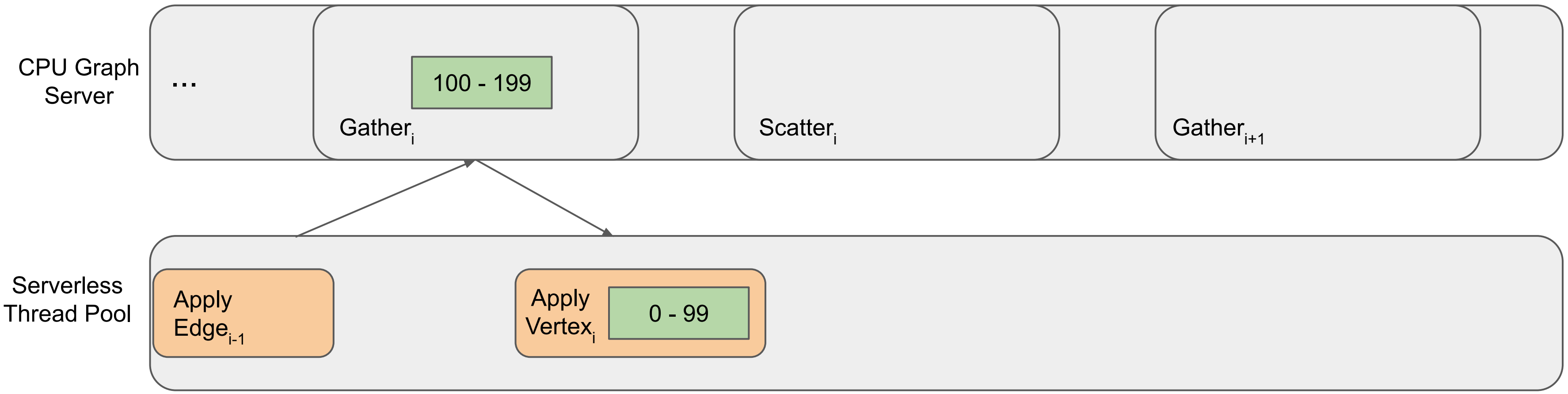
Solution: Create Pipeline of Decomposed Tasks



Data Chunks Moving Through Layer of Pipeline

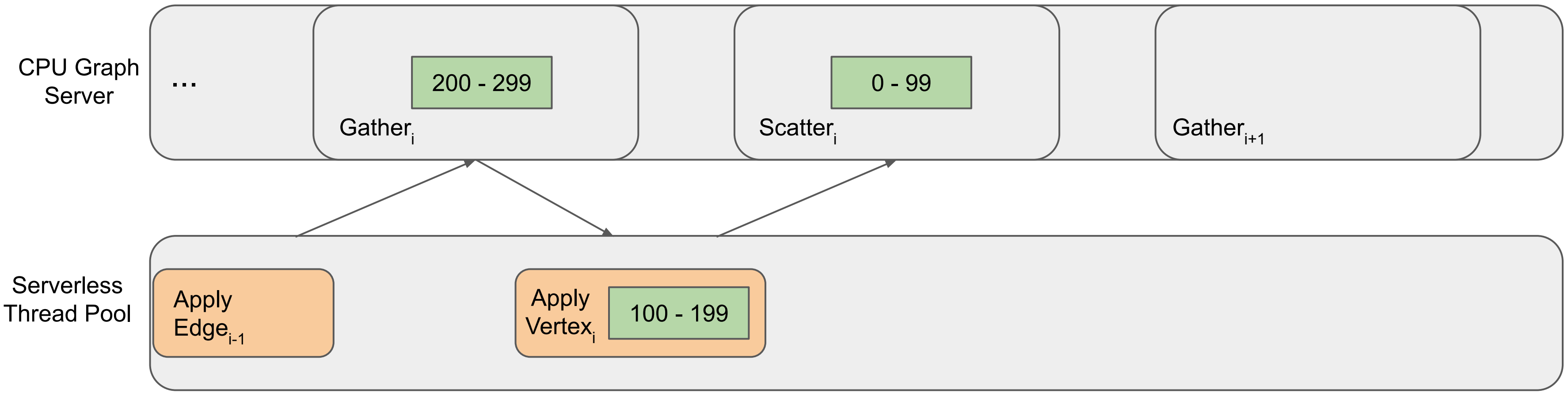


Data Chunks Moving Through Layer of Pipeline

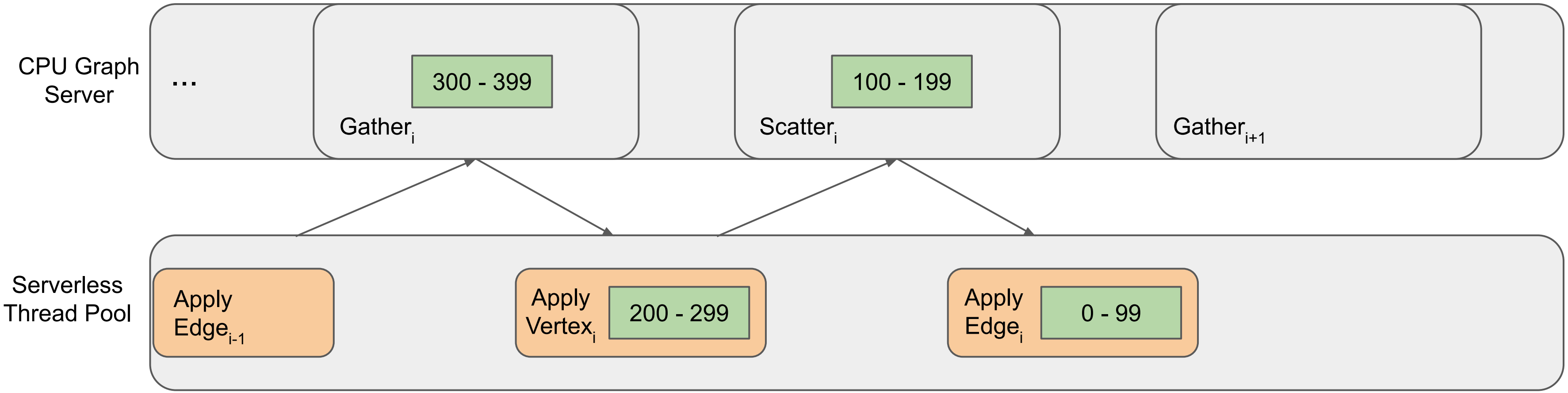


UCLA

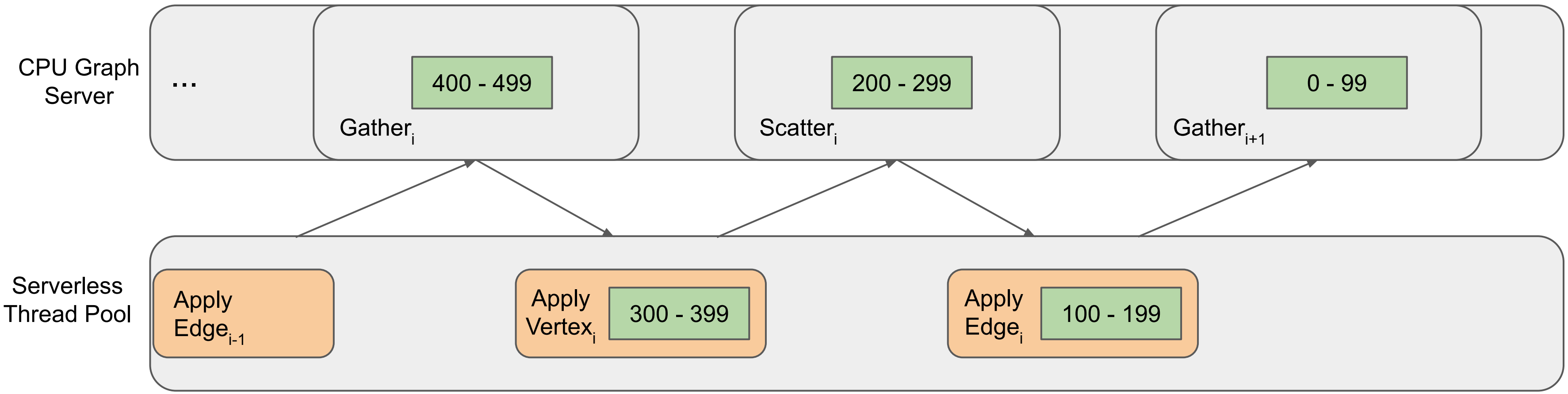
Data Chunks Moving Through Layer of Pipeline



Data Chunks Moving Through Layer of Pipeline

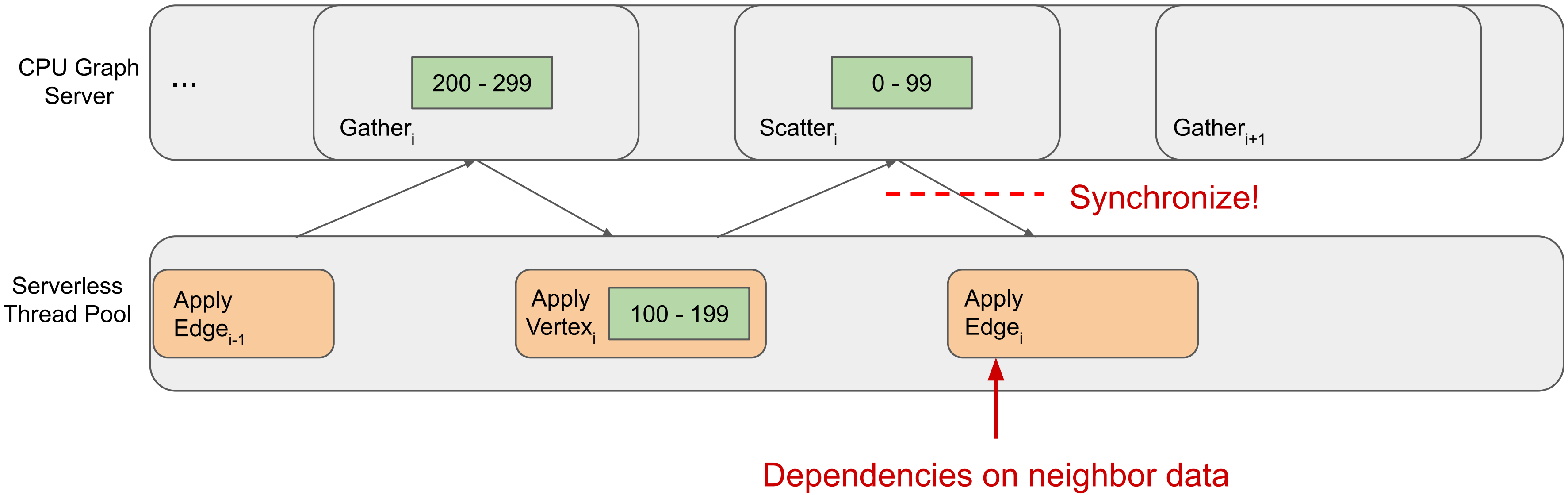


Data Chunks Moving Through Layer of Pipeline



UCLA

Data Chunks Moving Through Layer of Pipeline



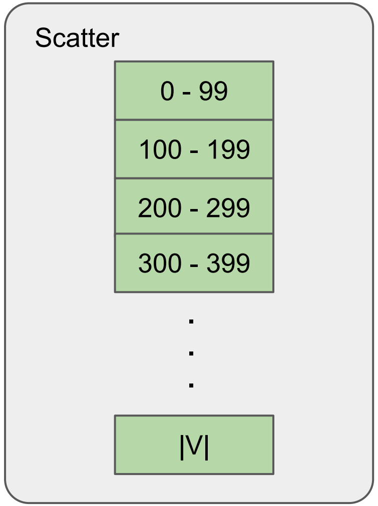
Synchronize after Scatter Hinders Pipeline

Pipeline not fully utilized

- Network latency challenge not resolved!

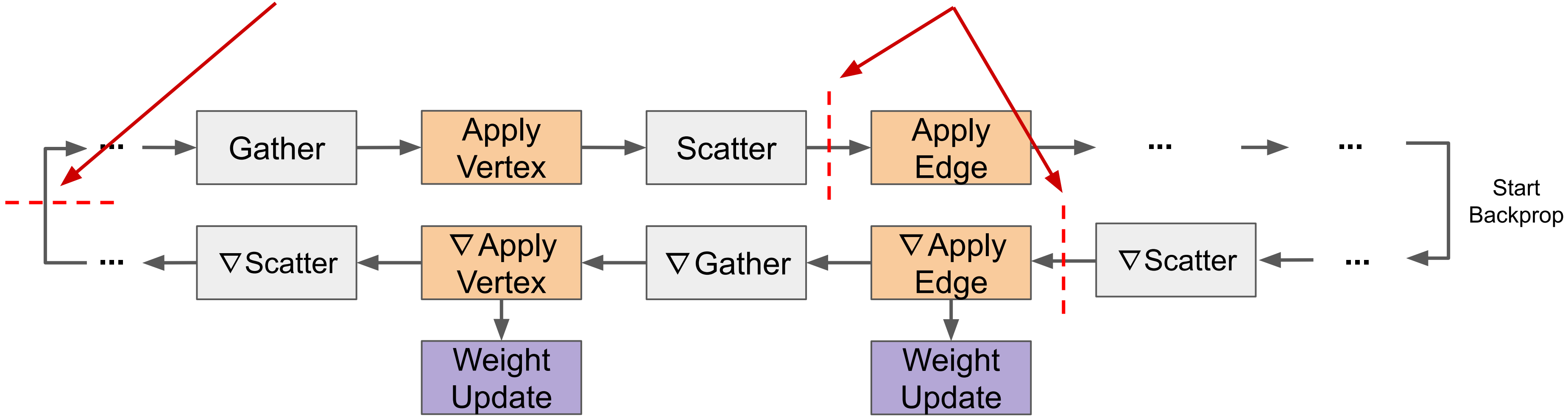
Modified Solution: Introduce asynchrony to pipeline

- Allow pipeline to saturate fully



Two Sync Points Makes Asynchrony Difficult

1. Sync before new epoch
   - Dependency on updated parameters
2. Sync after every Scatter
   - Dependency on neighbors' features



Minimizing Effects of Asynchrony on Convergence

Bounded staleness (graph-parallel path)

- No chunk in the system can get *S* epochs ahead of others
  - *S* is some staleness bound

Weight stashing at weight servers[2] (tensor-parallel path)

- Cache parameters used in forward to use same version in backward

We have formally proved the convergence of our system

[2] D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri,N. R. Devanur, G. R. Ganger, P. B. Gibbons, and M. Zaharia. PipeDream: Generalized pipeline parallelism for DNN training. In SOSP, page 1–15, 2019.

Serverless Optimizations

- Task fusion
- Tensor rematerialization
- Lambda internal streaming

Details in the paper

Data Graphs

	Graph	Size (|V|, |E|)	# features	# labels	Avg. Degree
Dense	Reddit-small	(232.9K, 114.8M)	602	41	492.9
	Reddit-large	(1.1M, 1.3B)	301	50	645.4
Sparse	Amazon	(9.2M, 313.9M)	300	25	35.1
	Friendster	(65.6M, 3.6B)	32	50	27.5

Target metrics:

- Performance
- Cost
- **Value:** Performance-per-dollar

We Evaluated Several Aspects of Dorylus

Compared staleness bounds to determine optimal asynchrony

Evaluated Dorylus variants without serverless
- CPU-only: All stages run on CPUs
- GPU-only: All stages run on GPUs

Compared against existing systems

Effects of scaling out

Breakdown of time/costs per stage

We Evaluated Several Aspects of Dorylus

Compared staleness bounds to determine optimal asynchrony

Evaluated Dorylus variants without serverless

- CPU-only: All stages run on CPUs
- GPU-only: All stages run on GPUs

Compared against existing systems

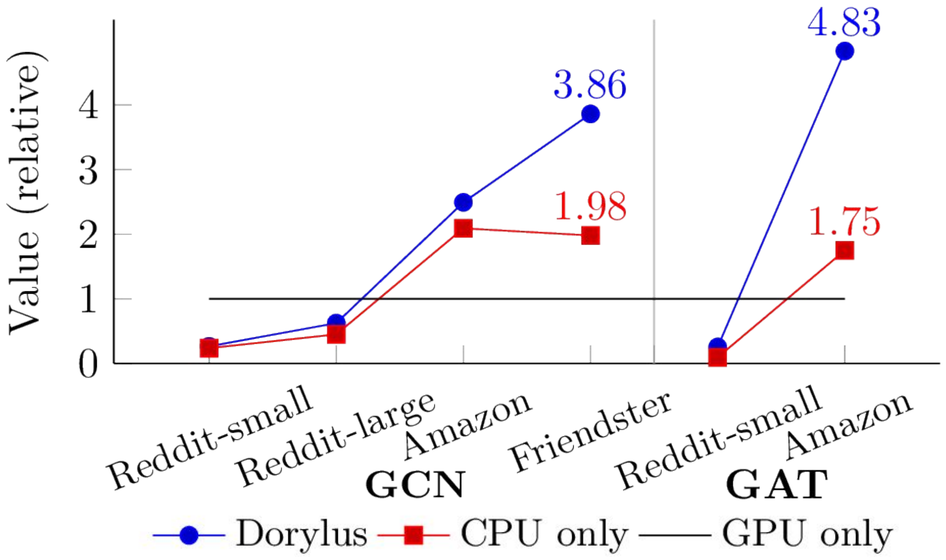
Effects of scaling out

Breakdown of time/costs per stage

High Value on Large-Sparse Graphs

Dorylus provides better value than CPU and GPU-based backends on large sparse graphs

Dorylus outperforms GPU based implementations on very large graphs

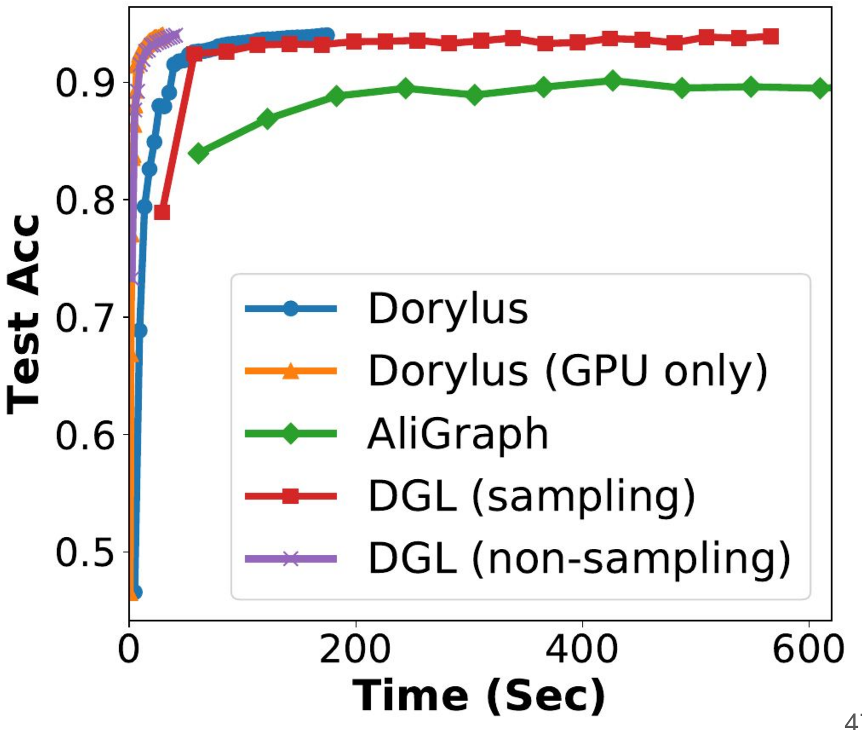


Dorylus Outperforms Existing Systems

Dorylus outperforms sampling based methods
- **3.25x** faster than DGL (sampling)

Slower than GPU-based non-sampling systems
- Whole graph can fit in GPU memory

Time to Target Accuracy on Reddit-small



UCLA

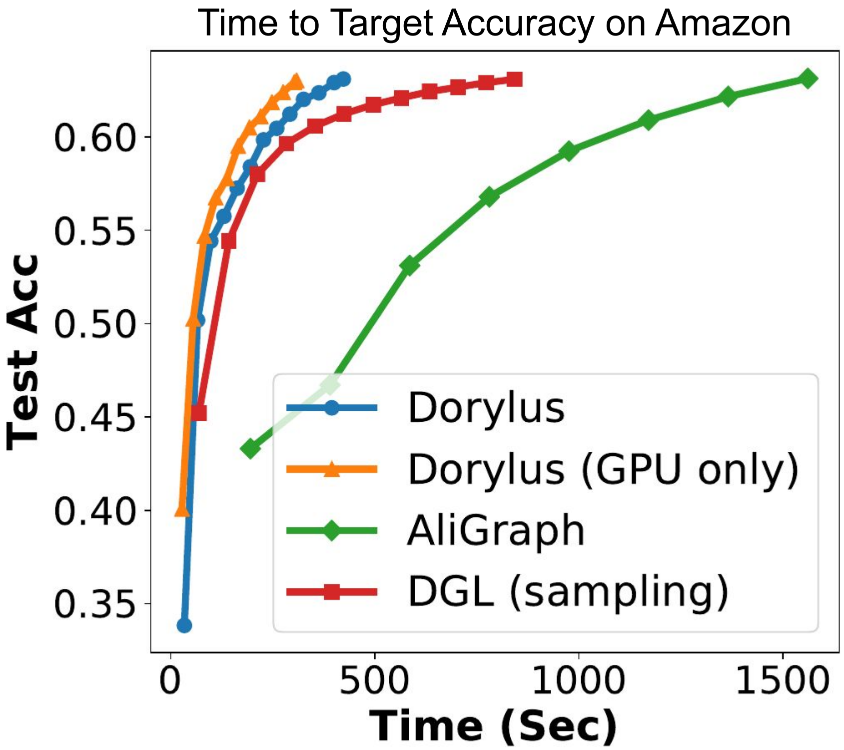
Dorylus Scales Full Graph Training

On a large, sparse graph

- Dorylus **1.99x** faster than DGL (sampling)
- Only **1.37x** slower than Dorylus (GPU only)

Value comparison:

- **17.7x** value of DGL (sampling)
- **8.6x** value of AliGraph



Conclusion: Dorylus Provides Value

Dorylus: Affordably scaling Graph Neural Network training to billion-edge graphs

- Utilize computation separation to specialize resources
- Implement bounded asynchronous pipeline
- Up to 2.75x more performance-per-dollar than CPU-only, 4.83x GPU-only
- Opens possibility to apply our techniques to other models

Thank you! Code at https://github.com/uclasystem/dorylus. For questions email jothor@cs.ucla.edu