SGX: A 16nm 1.92TFLOPS Spatial GNN Accelerator Using Subgraph-Parallel Processing

Po-Shao Chen, Mao-Chi Weng, Wei Tang, Zhengya Zhang
University of Michigan, Ann Arbor, Michigan, USA

***Abstract*—SGX is an accelerator designed for spatial graph neural networks (GNNs). It partitions sparse graphs into dense subgraphs via graph traversal and processes them with a subgraph-parallel architecture to improve locality and parallelism. SGX leverages bounded local neighborhoods to cache edge weights and uses local edge sorting to increase weight reuse and utilization. An 8mm^2 SGX chip delivers 1.92TFLOPS (FP16) performance, achieving at least 4.8× higher throughput and 3.1× higher compute density than state-of-the-art GNN accelerators.**

I. Introduction

Graph Neural Networks (GNNs) are essential for graph-structured data. However, real-world graphs are often large, sparse, and irregular, which presents hardware challenges (Fig. 1) including inefficient memory usage, irregular access patterns, low arithmetic intensity, and poor parallel load balancing. Spatial GNNs are a class of GNNs (Fig. 2) that aggregate information over local neighborhoods and naturally handle irregular graphs, making them well suited to event-driven sensors in robots, drones, and autonomous systems [1], [2]. For example, event-driven vision sensors produce asynchronous, sparse events rather than dense frames. These events can be modeled as nodes, with edges linking spatially nearby events to define neighborhoods for aggregation.

In spatial GNNs, edges carry geometric attributes (e.g., relative coordinates) that control how neighbor messages are transformed and weighted during aggregation. For instance, SplineCNN, a notable spatial GNN, employs continuous B-spline kernels [3] (Fig. 2) to generate edge-specific weights as a function of relative node positions. This adds complexity beyond conventional GNNs because the effective weight matrix vary across edges rather than being shared across all neighbors of a node.

We present SGX, a spatial GNN accelerator that clusters a large sparse graph into dense subgraphs and assigns each subgraph to a subgraph-parallel processor (SPP). With multiple SPPs running in parallel, SGX improves efficiency through mostly local memory accesses and higher utilization from denser local connectivity. SGX is enabled by two innovations (Fig. 1): 1) locality-aware graph partitioning via speculative breadth-first search (SBFS), which targets dense subgraphs and maximize subgraph-parallel speedup; and 2) weight caching with local edge sorting to improve weight reuse and sustain high utilization in the subgraph-parallel architecture. Together, these techniques allow SGX to achieve 4.8× higher performance and 3.1× higher compute density than the state of the art [4].

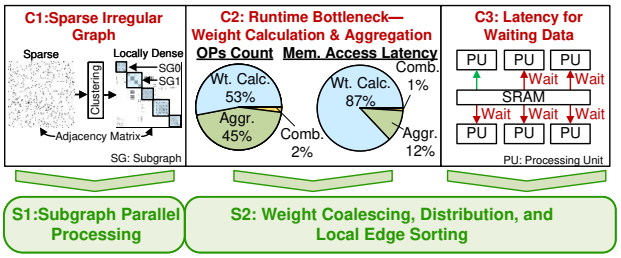


Fig. 1: Typical spatial GNN challenges and the proposed solutions.

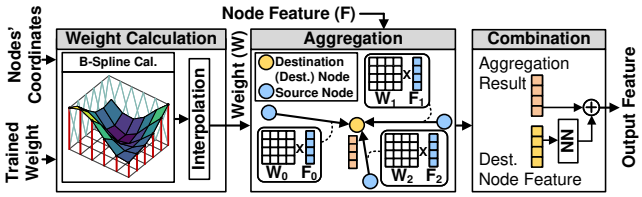


Fig. 2: Flowchart of spatial GNN.

II. SGX Workflow and Architecture

SGX performs spatial GNN inference in four stages (Fig. 1, Fig. 2): 1) clustering, which partitions a large sparse graph into denser subgraphs for subgraph-parallel processing; 2) edge-weight calculation, which computes edge-specific weight matrices from relative coordinates (e.g., via basis-function evaluation and kernel interpolation as in SplineCNN [3]); 3) aggregation, which transforms each neighbor's features with the corresponding edge-specific weights and sums them into a neighborhood message; and 4) combination (update), which combines the node's current features with the aggregated message to produce updated features that capture higher-level semantics for downstream tasks.

Dense subgraphs in real-world graphs vary in size. SGX integrates 16 SPPs, each designed to process a unit subgraph of up to 64 nodes – large enough to benefit from subgraph processing while limiting underutilization for small subgraphs. SGX implements the spatial GNN inference pipeline with four main components (Fig. 3). 1) The clustering engine (CLST) traverses the input graph to reorder nodes, form subgraphs, record local indices in mapping tables, and assign subgraphs to 16 SPPs organized into four quadrants (Quads). 2) The weight precomputation unit (WPU) evaluates edge functions and builds a weight look-up table (LUT) to avoid recomputing edge-specific weights at runtime. 3) The weight distribution

979-8-3195-0631-3/26/$31.00 ©2026 IEEE

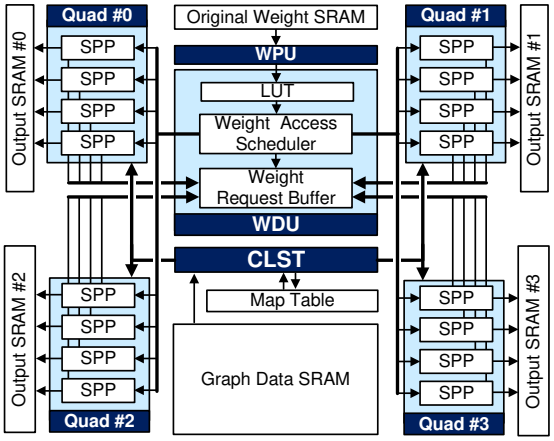


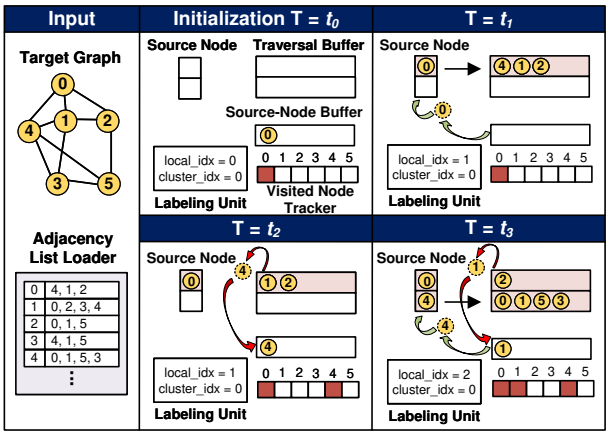
Fig. 3: Overall architecture of SGX.

unit (WDU) broadcasts weights to quadrants and SPPs. Each SPP locally sorts edges to increase weight reuse and reduce contention. 4) Each SPP, consisting of an array of 128 FP16 MACs, runs aggregation (aggr.) and combination (comb.) on its assigned subgraph. We find FP16 is required to maintain accuracy. We benchmark SGX using SplineCNN [3].

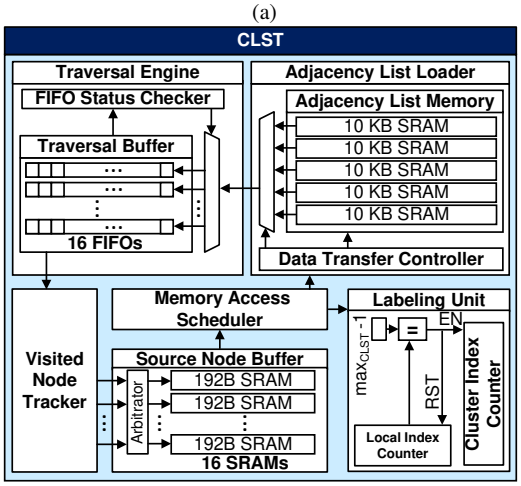
III. Clustering for Subgraph-Parallel Processing

Large, sparse, and irregular graphs pose major memory and processing challenges. We address this by using breadth-first search (BFS) to traverse the graph level by level from a starting node. Grouping nodes discovered at the same depth, or within a small depth window, yields clusters with stronger local connectivity. Classic BFS expands neighbors one frontier node at a time, which is too slow at scale. SGX's CLST instead uses speculative BFS (SBFS) [5], which expands from the next frontier in parallel without waiting for the current level to finish. Because this expansion is speculative, conflicts can occur when multiple threads attempt to visit the same node. SBFS resolves these conflicts by allowing only one thread to claim a node.

SGX's CLST operates as follows (Fig. 4(a)). During initialization (t_0), the source-node buffer is initialized with a starting node and marked visited. To begin a traversal (t_1), CLST allocates an available lane in the traversal buffer and fills it with the neighbors of a node dequeued from the source-node buffer. Once allocated (t_2), the lane processes one neighbor per cycle: unvisited nodes are marked visited and enqueued into the source-node buffer for later traversal (t_3). Increasing the number of lanes enables more parallelism, speeding up traversal and cluster formation. Finally, the labeling unit forms subgraphs by assigning nodes to clusters {local_idx, cluster_idx} in dequeue order from the source-node buffer. SGX's CLST comprises 16 parallel lanes (Fig.4(b)). The source-node buffer includes 16 SRAM-based FIFOs and arbitration logic that remaps unvisited nodes to available write ports to avoid deadlock. CLST occupies 0.32mm^2 in Intel16, a modest area cost for producing denser subgraphs for parallel processing.



(a)



(b)

Fig. 4: SGX clustering engine (CLST) (a) operation and (b) architecture.

We evaluate SGX's CLST on the ModelNet40 (MN) [6], N-Caltech101 (NC) [7], and KITTI (KT) [8] point-cloud datasets. SBFS with 16 lanes (used in CLST) delivers an 11.7× geometric-mean speedup over BFS (Fig. 5(a)). We measure clustering quality using the edge-cut ratio, defined as the fraction of edges that cross cluster boundaries (inter-cluster edges) out of all edges; lower values indicate better locality. SBFS reduces the geometric-mean edge-cut ratio to 0.54 compared to above 0.9 for sequential clustering by node index (Fig. 5(b)). This improved locality enables efficient subgraph-parallel processing and storage: each SPP caches its subgraph and local connectivity. SGX handles cross-cluster edges by replicating required external neighbors within an SPP, so a lower edge-cut ratio reduces replication. With all needed neighbors cached locally, aggregation and combination proceed independently within each SPP.

IV. Weight Storage and Efficient Distribution

In spatial GNNs, edge-specific weights are continuous functions of relative node positions and are typically computed on the fly in software. In a software SplineCNN benchmark

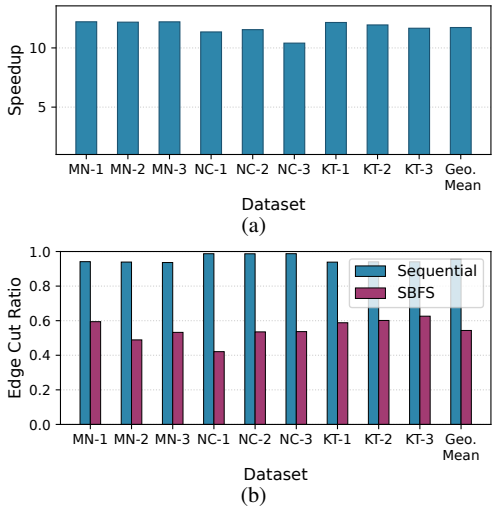


Fig. 5: (a) Speedup of clustering using SBFS (16 lanes) over BFS, and (b) edge-cut ratio of graphs with sequential clustering and SBFS clustering.

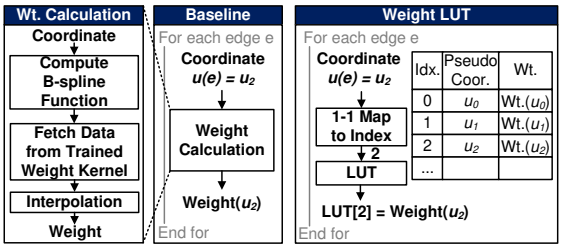


Fig. 6: Weight recomputation and caching.

for event-based object detection in ADAS [2], this on-the-fly computation accounts for 53% of total operations and 87% of memory-access latency (Fig. 1). Because relative positions are bounded within a local neighborhood, the number of unique weights is limited. SGX therefore uses its WPU to precompute and store weights in a LUT (Fig. 6), together with a weight-index buffer that maps relative positions to LUT entries, eliminating runtime computation and enabling reuse. SGX provisions a 128-entry LUT, with each entry storing a 32×16 weight matrix; larger weights can be tiled and cached with external-memory backup. For workloads such as AEGNN [2], SGX's on-chip LUT alone is sufficient to store the required weights.

A centralized weight LUT is storage-efficient but can become a bandwidth bottleneck. During aggregation, an SPP requests a weight from the WDU, which fetches it from the LUT and broadcasts it to all SPPs via a pipelined H-tree (Fig. 7). This works well when many SPPs request the same index; otherwise, the WDU must serialize requests, causing stalls. SGX addresses this with a 2KB double-buffered weight cache per SPP, supporting broadcast snooping and prefetching. On a hit, the SPP reads the weight locally and bypasses the WDU. Each SPP also includes a sorter that reorders local edges by weight index (wt. idx.), aligning requests across SPPs and improving locality (Fig. 8(a), (b)). Together, caching and

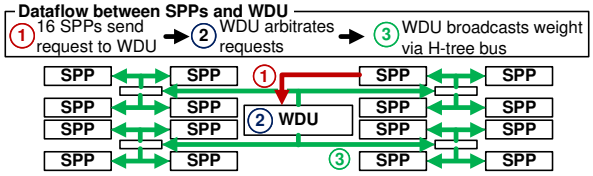


Fig. 7: Pipelined H-tree for weight distribution.

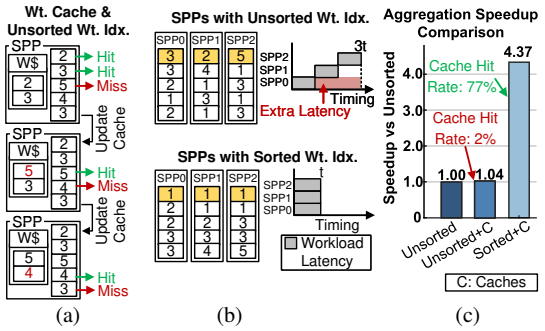


Fig. 8: (a) Weight caching in an SPP, (b) local edge index sorting and access alignment to enable more reuse of broadcast weights and speedup, and (c) speedup due to local edge index sorting and weight caching in SPPs.

sorting raise the hit rate from 2% to 77% (Fig. 8(c)) and deliver a 4.37× aggregation speedup.

Implementing a per-SPP insertion sorter that stores and reorders all edge-associated fields (7-bit weight index, 20-bit edge index, and 4-bit local index) in registers would incur high area and power overhead. Instead, we use a hybrid-storage sorter (Fig. 9): it sorts only address pointers by weight index, while edge and local indices remain in a 2KB SRAM. Compared with a full-register design [9], this hybrid design cuts area by 69% and power by 67%.

V. Chip Measurement Results and Comparison

An SGX test chip was fabricated in Intel16 technology and occupies 8mm^2 (Fig. 10). Measurements show 1.92TFLOPS (FP16) at 0.9V and 850MHz, with 621mW power consumption. With the supply scaled to 0.6V, energy efficiency reaches 5.94TFLOPS/W. SGX is the first to leverage on-chip graph clustering to extract dense subgraphs. The improved edge-cut ratio (Fig. 5(b)) highlights the potential to boost processing efficiency over prior work. Local edge sorting within SPPs further improves performance by 2.33× (Fig. 11), while hybrid-storage sorting adds only 12% area overhead.

We compare SGX with state-of-the-art GNN accelerators as the closest related work, while noting fundamental workload differences (Fig. 12). The comparisons include a multi-chip CIM-based GCN accelerator using message passing [10] and an SpMM-based GNN accelerator for 3D path planning [4]. SGX targets spatial GNNs with edge-specific weight matrices, whereas both prior designs assume a shared edge-weight matrix. In spatial GNNs, unique edge weights limit reuse and reduce the effectiveness of CIM-based approaches [10]. Moreover, spatial-GNN aggregation cannot be implemented

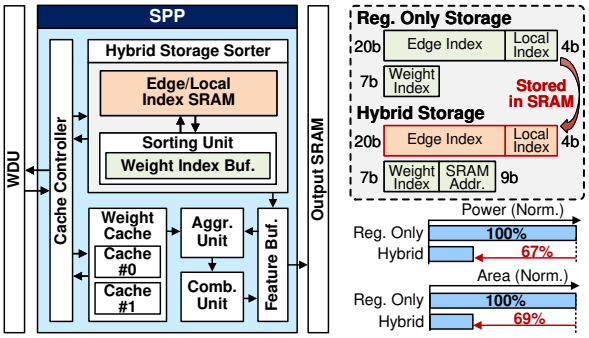
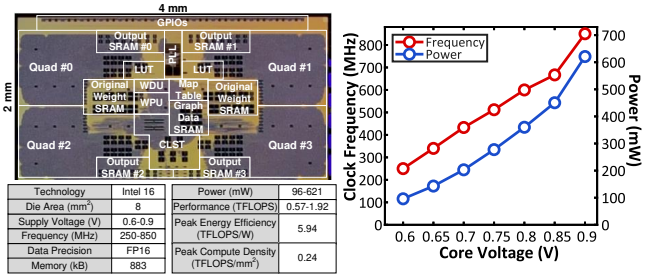


Fig. 9: SPP design and hybrid storage sorter for sorting local edge indices.



Technology	Intel 16
Die Area (mm^2)	8
Supply Voltage (V)	0.6-0.9
Frequency (MHz)	250-850
Data Precision	FP16
Memory (kB)	883

Power (mW)	96-621
Performance (TFLOPS)	0.57-1.92
Peak Energy Efficiency (TFLOPS/W)	5.94
Peak Compute Density (TFLOPS/mm^2)	0.24

Fig. 10: Chip micrograph, measurement results, and voltage scaling measurement.

as a straightforward SpMM between adjacency and feature matrices [4]. SGX addresses these spatial-GNN design challenges, and achieve competitive energy efficiency and compute density, especially since prior work reports performance including skipped zero operations [10], while SGX reports actually executed operations.

VI. Conclusion

SGX is a spatial GNN accelerator that uses 16-lane SBFS-based graph traversal to cluster large, sparse graphs into denser subgraphs, increasing local connectivity for efficient subgraph-parallel processing. SGX precomputes and caches edge-specific weights in an LUT, and improves weight distribution by aligning weight requests via per-SPP local edge sorting. An 8mm^2 Intel16 test chip achieves 3.09TFLOPS/W (FP16) at 0.9V and 850MHz, with a compute density of 0.24TFLOPS/mm^2, demonstrating effective hardware support for spatial GNNs.

Acknowledgment

This work was supported in part by the ACE Center, sponsored by SRC and DARPA under JUMP 2.0. Chip fabrication was provided by the Intel University Shuttle Program.

References

[1] L. Kong, D. Lu, X. Xu, L. X. Ng, W. T. Ooi, and B. R. Cottereau, "Eventfly: Event camera perception from ground to the sky," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 1472–1484.

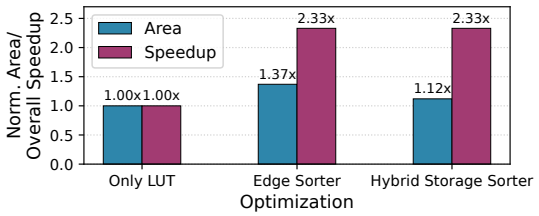


Fig. 11: Evaluation of area and speedup of edge index sorting in SPP.

	JSSC'25 [10]	VLSI'23 [4]	This Work	
Network	GCN	GNN	Spatial GNN	
Technology	28 nm	28 nm	16 nm	
Continuous Convolution Kernel Support	X	X	V	
Unique Weight Per-Edge	X	X	V	
Precision	INT8,INT16,FP32	N/A	FP16	
Area (mm^2)	12.42	5.07	8	
Memory Size (kB)	512 (SRAM)+ 128 (CIM)	886	883	
Supply Voltage (V)	0.6-1.0	0.9	**0.6**	**0.9**
Frequency (MHz)	115-290	50-200	**250**	**850**
Power (mW)	29.1-166.8	34.94-134.8	**96**	**621**
Performance	0.09 TFLOPS*	0.396 TOPS**	**0.57 TFLOPS††**	**1.92 TFLOPS††**
Energy Efficiency	1.3-8.3 TFLOPS/W*†	2.94 TOPS/W**	**5.94 TFLOPS/W**	**3.09 TFLOPS/W**
Compute Density	0.0072 TFLOPS/mm^2*	0.0781 TOPS/mm^2**	**0.07 TFLOPS/mm^2**	**0.24 TFLOPS/mm^2**

*FP32 **@0.9V, 200 MHz
† Zero-data Ops included (Pubmed dataset's feature/adjacency sparsity = 90%/99%)
†† N-Caltech101 dataset [7] is used

Fig. 12: Comparison with state-of-the-art GCN and GNN accelerators.

[2] S. Schaefer, D. Gehrig, and D. Scaramuzza, "Aegnn: Asynchronous event-based graph neural networks," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 12 361–12 371.

[3] M. Fey, J. E. Lenssen, F. Weichert, and H. Müller, "Splinecnn: Fast geometric deep learning with continuous b-spline kernels," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 869–877.

[4] S. Song, D. Han, S. Kim, S. Kim, G. Park, and H.-J. Yoo, "Gppu: A 330.4-j/ task neural path planning processor with hybrid gnn acceleration for autonomous 3d navigation," in *Proc. IEEE Symp. VLSI Technol. Circuits (VLSI Technol. Circuits)*, 2023, pp. 1–2.

[5] J.-F. Zhang and Z. Zhang, "Point-x: A spatial-locality-aware architecture for energy-efficient graph-based point-cloud deep learning," in *Proc. 54th Annu. IEEE/ACM Int. Symp. Microarchitecture, (MICRO)*, 2021, pp. 1078–1090.

[6] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[7] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in Neuroscience*, vol. 9, Nov. 2015.

[8] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[9] Y.-C. Wu, J.-H. Hung, and C.-H. Yang, "A 135mw fully integrated data processor for next-generation sequencing," in *Proc. Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2017, pp. 252–253.

[10] Y. Wang, Z. Wu, W. Wu, L. Liu, Y. Hu, S. Wei, F. Tu, and S. Yin, "Tensorcim: Digital computing-in-memory tensor processor with multichip-module-based architecture for beyond-nn acceleration," *IEEE J. Solid-State Circuits*, vol. 60, no. 2, pp. 734–747, 2025.