

RAPID: Accelerating Point Cloud Diffusion Models via Space-Aware Mix-Precision Quantization

Qichu Sun^{*†}, Yanan Zhu^{*†}, Linxi Lu^{*†}, Haishuang Fan^{*†}, Jingya Wu^{*}, Huawei Li^{*}, Xiaowei Li^{*‡}, Guihai Yan[§]

^{*}State Key Laboratory of Processors, Institute of Computing Technology, Chinese Academy of Sciences

[†]University of Chinese Academy of Sciences [‡]Zhongguancun Laboratory [§]YUSUR Technology Co., Ltd.

{sunqichu22z, zhuyan24s, lulinx23z, fanhaishuang20z, wujingya, lihuawei, lxw}@ict.ac.cn, yan@yusur.tech

Abstract—Point cloud diffusion models, as an emerging 3D generation method, hold broad prospects in 3D modeling, AR/VR, and so on. However, their reliance on costly full-precision neural network computations during extended denoising process limits their practical application. To address this challenge, we propose RAPID, an accelerator co-designed with a space-aware quantization method. First, RAPID uses K-means to partition points into groups and computes scaling factors in each, mitigating accuracy issues caused by uneven distribution. Second, it employs a mixed-precision quantization scheme that uses low precision for internal point groups and high precision for detail-rich edge groups, ensuring generation quality while minimizing bit-width. Third, it reuses computation results for groups with little change between timesteps, reducing redundant calculations. Moreover, RAPID’s hardware features a mixed-precision PE array for efficient computations at various bit-widths, and a filter for dynamic bit-width allocation and result reuse. Evaluations show that, compared to the NVIDIA RTX A5000 GPU and state-of-the-art accelerators, RAPID achieves average speedups of 9.22×, 4.66×, 3.69×, and 3.01×, and energy savings of 61.74×, 4.30×, 3.94×, and 2.76×, with negligible accuracy loss.

Index Terms—diffusion model, point cloud, mixed-precision, quantization, accelerator

I. INTRODUCTION

Recently, point cloud diffusion models (PCDMs) [1]–[3] have emerged as an effective 3D shape generation method, and find wide applications in 3D modeling, AR/VR, digital twins and others. As shown in Fig. 1, these models treat discrete points in 3D space as independent samples from a geometric distribution while incorporating extra shape constraints, gradually transforming Gaussian noise into the target output during a denoising process. However, despite their superior results compared to previous approaches [4], [5], PCDMs incur high computational and storage costs because they require hundreds of timesteps and execute full-precision neural network computations at each timestep. For example, DPM [1] takes over 80 ms to generate a point cloud frame on an NVIDIA RTX A5000 GPU. This prevents PCDMs from meeting low-latency requirements and hinders deployment on resource-constrained platforms.

For efficient deployment of point cloud models, several accelerators [6]–[9] have been proposed. However, while these works eliminate computation redundancy caused by farthest point sampling and k-nearest neighbor searching in classification [10] and segmentation [11] tasks through localized search [7], approximate substitution [8] and octree

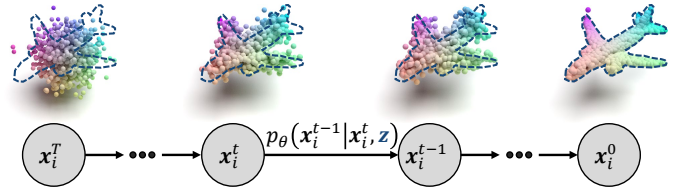


Fig. 1. Point cloud diffusion model.

organization [9], they fail to alleviate the heavy burden in PCDMs because PCDMs employ different operators, leaving the acceleration of these models a major challenge.

To address this challenge, we analyzed PCDMs’ dataflow and identified two main redundancies: 1) **Spatial bit-width redundancy**. Spatial portions of a point cloud vary in their importance. Interior points carry redundant local information with minimal impact on the overall shape, while edge points hold detailed shape information that greatly affects accuracy. Although quantization [12] can significantly reduce inference time and storage, using a uniformly high bit-width (W8A8) for all points is unnecessary. Existing mixed-precision methods [13] assign different bit-widths across network layers, but addressing spatial bit-width redundancy in PCDMs remains unexplored. 2) **Temporal computation redundancy**. At each timestep, all point coordinates are processed by a neural network to compute results for updating themselves, yet many coordinates change only slightly between adjacent timesteps. As shown in Fig. 1, most points in the point cloud exhibit minimal positional differences between timesteps t and $t-1$, rendering much of the computation redundant.

To eliminate redundancies and speed up the denoising process in PCDMs, we propose the following optimizations: 1) **K-means-based point group quantization**. Since point distributions in space are uneven, activations from points along the sequence dimension fluctuate dramatically and cover a wide range, degrading model accuracy even after block quantization [14]. To address this, we leverage that nearby points with similar coordinates share similar features in neural networks. Specifically, we cluster nearby points with K-means [15] to partition them into groups, and then apply block quantization to compute scaling factors independently. Within each group, the activation range is reduced, thereby preserving overall accuracy. 2) **Space-aware mixed-precision quantization**. Different groups in the point cloud can use varying bit-widths for activation quantization. For internal point groups, lowering the bit-width has minimal impact on

accuracy, so activations are quantized to INT4. For edge point groups, a higher bit-width is necessary to preserve accuracy, so activations are quantized to INT8. This approach reduces bit-widths while maintaining generation accuracy. 3) **Inter-timestep point result reuse.** When input coordinates are similar, the neural network produces similar results. Leveraging this, if a group’s point distributions at a timestep are nearly identical to those at the previous timestep, the neural network skips processing that group and reuses its previously computed results to update the coordinates. This approach eliminates redundant computations between timesteps.

To efficiently support our optimizations, we designed a dedicated hardware accelerator, RAPID. It features a processing element (PE) array for matrix multiplications and accumulations (MACs) and a special function unit for other PCDM calculations. To handle activations with varying bit-widths within blocks of the block quantization, the PE array is organized into multiple sub-arrays. Each sub-array performs block-wise MACs with a group of PEs that execute mixed-precision multiplication by shifting and combining the results of 4-bit multiplications. Additionally, a point group filter assigns a quantization bit-width to each point group and omits those groups that vary only slightly from subsequent neural network computations to enable point result reuse.

Generally, our contributions can be summarized as follows:

- We adopt effective quantization for PCDMs by computing scaling factors in groups of nearby points to preserve precision and using different quantization bit-widths for various groups to balance model accuracy and cost.
- We employ inter-timestep point result reuse, reusing computed results for point groups with minimal changes to reduce redundant computations between timesteps.
- We propose an accelerator with a mixed-precision PE array for varying bit-width computations and a point group filter for group bit-width allocation and pruning.

Experimental results show that RAPID achieves a W8A5.2 quantization and 30% sparsity with minimal accuracy loss on ShapeNet [16]. Compared to the NVIDIA RTX A5000 GPU and state-of-the-art point cloud accelerators, PointAcc [6], MARS [7] and MoC [17], our work achieves average 9.22 \times , 4.66 \times , 3.69 \times and 3.01 \times speedup, and 61.74 \times , 4.30 \times , 3.94 \times and 2.76 \times energy savings. This enables fast, low-energy 3D point cloud generation on mobile devices. To our knowledge, RAPID is the first work to accelerate PCDMs.

II. PRELIMINARY

A. Point Cloud Diffusion Models

PCDMs are models that effectively generate 3D point clouds. In PCDMs, each point x_i in the point cloud is considered an independent sample $p(x_i^0|z)$ from a point distribution, where z represents the shape constraint determining the target shape of the point cloud. The generation process is a Markov chain [18] that starts from simple noise $p(x_i^T)$ and iteratively applies the reverse denoising in (1) to produce the final result:

$$p_\theta(x_i^{t-1}|x_i^t, z) = \mathcal{N}(x_i^{t-1}|\mu_\theta(x_i^t, t, z), \beta_t \mathbf{I}) \quad (1)$$

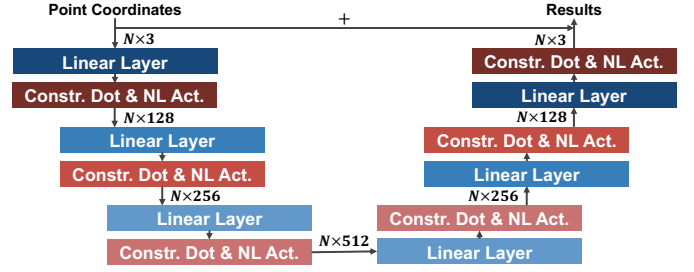


Fig. 2. A typical neural network to update the point coordinates.

where μ_θ is the estimated means, β_t is a variance scheduling hyperparameter that controls the diffusion rate, and $p(x_i^T)$ is set to $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Additionally, μ_θ can be parameterized as:

$$\mu_\theta(x_i^t, t, z) = \frac{1}{\sqrt{\alpha_t}}(x_i^t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}}\epsilon_\theta(x_i^t, t, z)) \quad (2)$$

where ϵ_θ is a neural network implemented via linear [1] or convolutional [2] layers with non-linear activation functions (e.g., Leaky ReLU [19]). For a point cloud with n points, ϵ_θ takes their 3D coordinates x_i^t as the initial activation matrix $\mathbf{X} \in \mathbb{R}^{N \times 3}$, and its results, together with the current point coordinates x_i^t , are used in (2) to get the updated points x_i^{t-1} . Moreover, ϵ_θ is iteratively applied over multiple timesteps until the final shape is obtained. As shown in Fig. 2, a typical ϵ_θ architecture [1] has six linear layers. After each linear layer, the output is dot-multiplied with z to enforce shape constraints and then passed through an activation function.

B. Quantization

Quantization [12] rounds a neural network’s high-precision parameters to a set of discrete values, reducing computational overhead and memory usage. The process of quantizing floating-point numbers into integers is shown in (3):

$$X^{INT} = clip(\lfloor \frac{X^{FP}}{\delta} \rceil + c_0, c_{min}, c_{max}) \quad (3)$$

where X^{FP} and X^{INT} are values before and after quantization, δ is the scaling factor, c_0 is the zero point, and $clip(\cdot)$ clamps the rounded integer to a range $[c_{min}, c_{max}]$ (e.g., $[-128, 127]$ in INT8 quantization). In uniform quantization, $c_0 = 0$, while in non-uniform quantization, $c_0 \neq 0$. Moreover, the scaling factor δ is determined using the minimum and maximum values in the dataset, as shown in (4):

$$\delta = \frac{max(X^{FP}) - min(X^{FP})}{2^N - 1} \quad (4)$$

where N denotes the number of bits in the quantized integer representation (e.g., $N = 8$ for INT8 quantization). Furthermore, quantization can be applied to linear layers [20]. For a linear layer $\mathbf{Y} = \mathbf{X}\mathbf{W}$, quantizing weights \mathbf{W} and activations \mathbf{X} from FP32 to INT8 reduces storage, and converting the full-precision matrix multiplication of \mathbf{X} and \mathbf{W} into integer multiplication speeds up inference. Moreover, block quantization [14] splits $\mathbf{X} \in \mathbb{R}^{N \times C_{in}}$ and $\mathbf{W} \in \mathbb{R}^{C_{in} \times C_{out}}$ into blocks $\mathbf{X}_{ij} \in \mathbb{R}^{B \times B}$ and $\mathbf{W}_{ij} \in \mathbb{R}^{B \times B}$ to compute a scaling factor for each block, better preserving model accuracy.

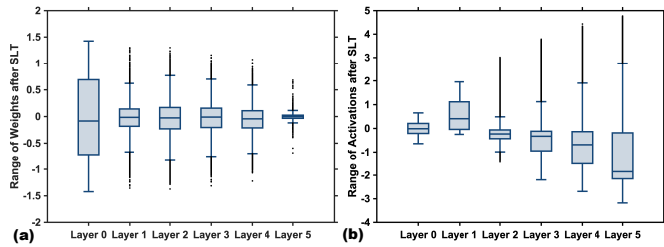


Fig. 3. (a) Weight distribution for each layer of the neural network ϵ_θ . (b) Activation distribution for each layer ($t = 80$).

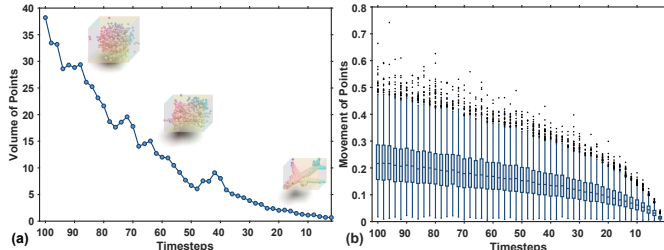


Fig. 4. (a) Overall point cloud range changes in the denoising process. (b) Distances of points' movement per timestep.

III. MOTIVATION

A. Large Activation Ranges Leading to Poor Accuracy

Quantization speeds up inference and reduces memory usage in PCDMs, but its effectiveness depends on weight and activation distributions. Fig. 3 shows the distributions after a signed logarithmic transformation (SLT) [21] in the network ϵ_θ [1]. In each layer, weights are nearly symmetric within a small range. However, due to points' uneven distribution in space, activations are asymmetric and fluctuate dramatically, showing a large range (nearly 60000 in layer 5 in Fig. 3 (b)) far greater than in image diffusion models [22]. Thus, even block quantization struggles with PCDMs' activation ranges, resulting in reduced generation accuracy.

B. Spatial Bit-Width Redundancy

In the PCDMs' denoising process, different parts of the point cloud have varying importance. Fig. 4 (a) and (b) show changes in the overall range of the point cloud and the distribution of point movement distances at each timestep. At timestep t , the point cloud range is defined by V^t , which is the volume of the bounding cuboid, as shown in (5):

$$V^t = (x_{max}^t - x_{min}^t) \cdot (y_{max}^t - y_{min}^t) \cdot (z_{max}^t - z_{min}^t) \quad (5)$$

It can be seen from Fig. 4 that the denoising process gradually aggregates points inward, with each timestep moving points only slightly. Thus, points can be classified as internal or edge. Internal points are uniformly distributed over smooth, continuous surfaces, making their local information highly redundant and less sensitive to shape changes. On the other hand, edge points lie sparsely along contours, so even minor shifts can affect the overall shape and edge features. Therefore, while 8-bit quantization is necessary for the few edge points to ensure accuracy, applying the same bit-width to the majority of internal points leads to significant redundancy.

C. Temporal Computation Redundancy

PCDMs require hundreds of timesteps in the denoising process to generate 3D shapes. At each timestep, a multi-layer

neural network ϵ_θ updates the point coordinates as defined in (2). However, the overall distribution of point coordinates changes very little between consecutive timesteps. As shown in Fig. 4 (b), although a point cloud contains a large number of points (typically 2048), only a limited subset undergoes significant updates at any given timestep. For example, at timestep 40, over 75% of the points exhibit an Euclidean distance of less than 0.22 between their pre- and post-update coordinates. This implies that a significant portion of the computational effort is spent on updating points with minimal changes. Consequently, executing expensive neural network operations for all points at every timestep is both unnecessary and inefficient, leading to substantial computational redundancy.

IV. OPTIMIZATIONS

A. K-Means-Based Point Group Quantization

We adopt the K-means-based point group quantization to resolve precision loss caused by oversized activation ranges. Since nearby points have similar coordinates, their activations across layers are also similar. Leveraging this, we use K-means to group nearby points before denoising process. Specifically, we rearrange the points to cluster around some centers and group every 8 points within each cluster, ensuring that points in each group are nearby. As shown in Fig. 5 (a), simple block quantization causes accuracy loss due to large activation fluctuations between adjacent points. However, after K-means clustering, each group has a compact range with minimal outliers, effectively preserving accuracy. Moreover, Fig. 4 shows that points' movement per timestep is negligible compared to the overall range, so K-means is performed only once during pre-processing, with its results reused across the timesteps.

After grouping the points with K-means, we adopt an asymmetric block quantization for activations. In detail, activations are partitioned into blocks along the sequence and channel dimensions, with scaling factors computed for each block individually. Most blocks yield small factors and retain high precision. Meanwhile, weights are symmetrically quantized and also partitioned into blocks with individual scaling factors. In practice, we use 8×8 blocks for weights and activations of linear layers to exploit the local similarity of points, and apply the im2col method [23] for convolution layers to reshape them into linear layers to implement our point group quantization.

B. Space-Aware Mixed-Precision Quantization

Applying a uniform W8A8 quantization in PCDMs results in bit-width redundancy. To minimize bit-widths while preserving accuracy, mixed-precision quantization can be used. However, existing methods [20], [24] assign high precision to outliers, resulting in complex computational patterns. Other approaches [13] vary precision across layers, but they aren't suitable for PCDMs due to minimal layer sensitivity differences. Instead, by leveraging the varying importance of points, we propose space-aware mixed-precision quantization. As discussed in Sec. III-B, internal points have minimal impact on shape, allowing low-precision quantization, while edge points require higher precision to avoid accuracy loss.

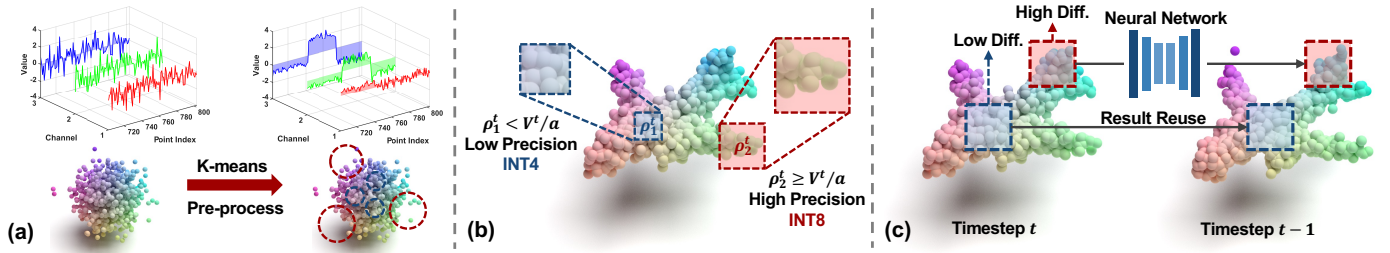


Fig. 5. (a) K-means-based point group quantization. (b) Space-aware mixed-precision quantization. (c) Inter-timestep point result reuse.

Therefore, we use INT4 for internal point groups and INT8 for edge point groups, reducing bit-widths to save computational and storage resources without sacrificing accuracy.

Moreover, we use a lightweight method to identify internal and edge groups. Compared to internal groups, edge groups have sparser distributions along the object’s contours, resulting in a larger range along at least one dimension. As shown in Fig. 5 (b), group 2 exhibits a wider y-axis range than group 1. Thus, we define a coefficient ρ^t for each group to represent its maximum range across the three axes, as defined in (6):

$$\rho^t = \max\{x_{max}^t - x_{min}^t, y_{max}^t - y_{min}^t, z_{max}^t - z_{min}^t\} \quad (6)$$

Specifically, before feeding point coordinates into the neural network ϵ_θ , we use ρ^t to assign different quantization bit-widths. For point group j , if $\rho_j^t \geq V^t/a$, the activations from its points across all layers are quantized to INT8; otherwise, they are quantized to INT4, where V^t is the overall point cloud range in (5) and a is a hyperparameter. This strategy maintains generation quality while minimizing bit operations.

C. Inter-Timestep Point Result Reuse

PCDMs require many denoising timesteps, yet the point cloud’s coordinates change minimally between steps, resulting in wasted computation. To mitigate this inefficiency, we propose an inter-timestep point result reuse method that reduces the number of points processed per timestep. Recognizing that nearly identical input coordinates yield similar computed results in the neural network ϵ_θ , only point groups with significant coordinate changes are processed at each timestep, while results for other groups are reused from the previous timestep, as shown in Fig. 5 (c). Furthermore, ρ^t in (6) is derived from each point group’s coordinate range and can assess coordinate changes without additional computation. Leveraging this, before feeding the point coordinates into the network, we compare the group’s current ρ^t with that of the previous timestep ρ^{t+1} . If the difference exceeds a predefined threshold, the group is processed; otherwise, the previous results are reused for updating in (2). This can significantly reduce computational overhead over multiple timesteps.

V. HARDWARE DESIGN

A. Overview

Fig. 6 shows RAPID’s hardware architecture. Generally, it consists of three components: the matrix multiplication unit (MMU), the special function unit (SFU), and on-chip buffers. The MMU is organized into multiple sub-arrays to support

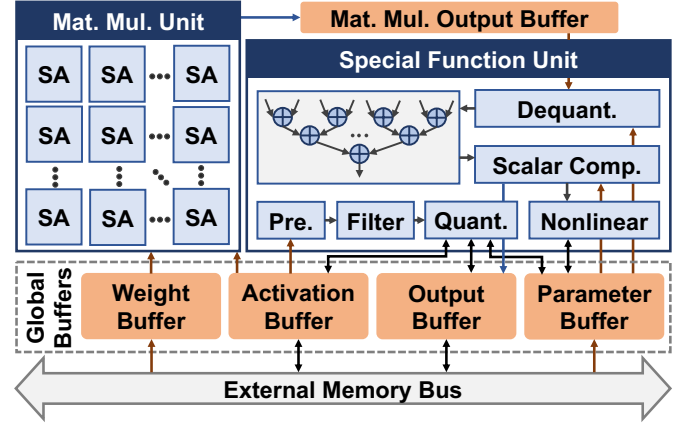


Fig. 6. Overview of RAPID Architecture.

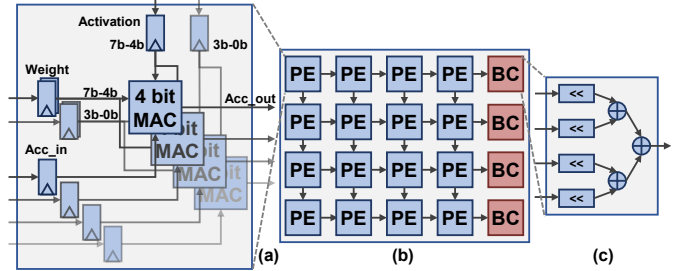


Fig. 7. (a) The mixed-precision processing element. (b) A sub-array. (c) The bit combination module.

block quantization, which adopt different bit-widths for blocks from different point groups. Meanwhile, the SFU handles all remaining tasks in PCDMs. Within the SFU, the point group filter allocates bit-widths for point groups (see Sec. IV-B) and selects groups for the quantization module (see Sec. IV-C), while the de-quantization module converts results to FP16 and compensates the zero points c_0 . Once de-quantized, the results of different blocks are aggregated using an adder tree. In addition, a scalar computation module performs element-wise operations, such as combining the outputs with the shape constraint z in Sec. II-A, and a nonlinear module implements Leaky ReLU [19], BatchNorm [25], and other nonlinear functions from (2) using lookup tables. Moreover, RAPID’s data exchange with the off-chip high-bandwidth memory (HBM) is conducted through an external memory bus.

B. Mixed-Precision PE Array

The MMU is a mixed-precision PE array designed to support matrix multiplications after applying our quantization method. It supports mixed-precision operations by using bitwise decomposition in integer multiplications. As shown in (7), given an INT8 weight $w_{7:0}$ and an INT8 activation

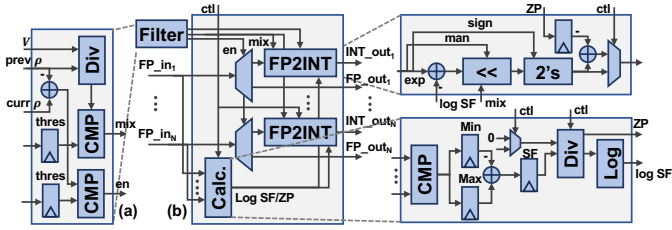


Fig. 8. (a) The point group filter. (b) The quantization module.

$a_{7:0}$, their upper 4 bits and lower 4 bits are separated into two individual operands each. This decomposition allows the original multiplication to be broken down into four INT4 multiplications. Finally, by shifting and summing the results of these four operations, the final result is obtained.

$$\begin{aligned}
 w_{7:0} \times a_{7:0} &= (2^4 \times w_{7:4} + w_{3:0}) \times (2^4 \times a_{7:4} + a_{3:0}) \\
 &= 2^8 \times w_{7:4} \times a_{7:4} + 2^4 \times w_{3:0} \times a_{7:4} \\
 &\quad + 2^4 \times w_{7:4} \times a_{3:0} + w_{3:0} \times a_{3:0}
 \end{aligned} \quad (7)$$

Similarly, the mixed-precision multiplication between an INT8 weight $w_{7:0}$ and an INT4 activation $a_{3:0}$ can be decomposed into two 4-bit multiplications, as shown in (8):

$$w_{7:0} \times a_{3:0} = 2^4 \times w_{7:4} \times a_{3:0} + w_{3:0} \times a_{3:0} \quad (8)$$

Based on this mechanism, the MMU was designed. As shown in Fig. 7, it consists of 16×16 sub-arrays for activations in different groups, and each sub-array contains 4×4 PEs, with a bit-compounding (BC) module following each row of PEs. Each PE houses four 4-bit MAC modules that multiply two INT4 operands and add the result to the corresponding accumulator. When all MAC operations in the array are completed, the BC module shifts and sums the outputs from the four MAC modules using shift bits $\{4, 0, 4, 0\}$ for internal groups and $\{8, 4, 4, 0\}$ for edge groups to produce the final result. Unlike previous works [26] that feature an entire PE array performing shifting and summing within each PE, our design is partitioned into sub-arrays to support fine-grained computations in blocks, and it carries out these operations with separate BC modules, reducing the number of on-chip shifters and adders and achieving higher throughput.

C. Quantization Module with a Point Group Filter

We use a quantization module with a point group filter to perform mixed-precision quantization and reduce redundant computations. As shown in Fig. 8, the filter compares ρ^t and ρ^{t+1} to prune point groups with minimal changes between timesteps in Sec. IV-C, and further leverages ρ^t and V^t to allocate different bit-widths for the remaining groups in Sec. IV-B. Next, before each layer in the neural network ϵ_θ , scaling factors for each block are computed and the necessary floating-point activations are converted to integers in parallel, with different bit-widths applied to activations from various point groups. Moreover, because inter-layer operations incur significant off-chip data transfer and memory overhead, we quantize each layer’s output before writing it back to off-chip memory, thereby reducing the data exchange bit-width from FP16 to INT8 or INT4. This approach reduces memory bandwidth usage and improves overall storage efficiency.

TABLE I
OVERALL COMPARISONS WITH PRIOR WORKS

	PointAcc	MARS	MoC	Ours
Technology (nm)	40	40	40	40
Area (mm ²)	15.7	-	7	7.8
Frequency (MHz)	1024	1024	1024	1024
On-chip SRAM (KB)	776	776	450	416
Peak Throughput (TOPS)	8	8	16.7	16.7
Bandwidth (GB/s)	256	256	256	256
MAC Bit-width	32b	32b	32b&4b	8b&4b
Support Group Pruning	X	X	X	✓

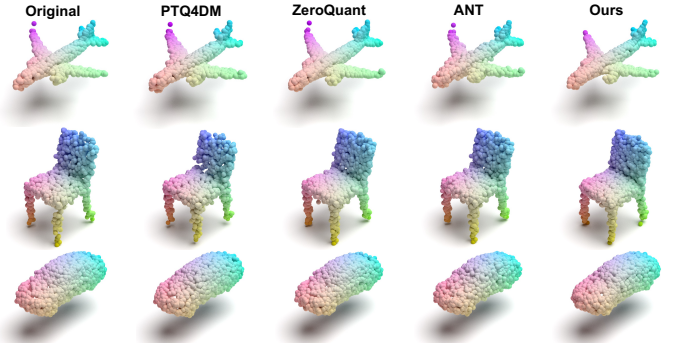


Fig. 9. 3D shape generation results of different quantization methods.

VI. EVALUATION

A. Experimental Setup

Benchmarks. To evaluate RAPID, we selected two representative PCDMs as benchmarks: DPM [1] and ShapeGF [2]. These models, featuring distinct architectures and denoising timesteps, were applied to 3D shape generation and auto-encoding tasks on the ShapeNet [16] dataset.

Hardware Synthesis. We implemented RAPID’s hardware design in Chisel, compiled it to Verilog, and synthesized it with Synopsys DC compiler in TSMC’s 40nm process to generate area and power reports. We also developed a cycle-accurate simulator to model RAPID’s behavior, count cycles, and track read/write operations of on-chip SRAMs. It was then integrated with DRAMsim3 [27] to simulate HBM2. Additionally, we used CACTI [28] to obtain the on-chip SRAMs’ area and power consumption. The parameters for RAPID’s implementation are listed in Tab. I.

Baselines. We compare our work with NVIDIA RTX A5000 GPU and three state-of-the-art point cloud accelerators, including PointAcc [6], MARS [7], and MoC [17], as detailed in Tab. I. For a fair comparison, all platforms were configured with equivalent compute resources.

B. Model Accuracy

To evaluate the effectiveness of our space-aware quantization method, we applied it to several PCDMs for shape generation tasks on ShapeNet. We use 1-NN classifier accuracy (1-NNA) to measure generation accuracy, where lower values indicate higher quality [4]. Experimental results show that our approach achieves a W8A5.2 bit-width and 30% sparsity, with less than a 1% drop in 1-NNA on most shapes. We also compared our method with uniform W8A8 quantization methods PTQ4DM [29] and ZeroQuant [30], and mixed-precision method ANT [24] that uses the same bit-width as ours. As

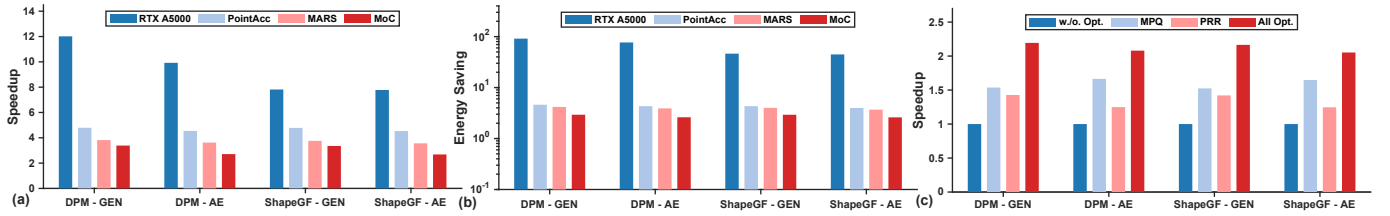


Fig. 10. (a) Speedup results of RAPID over RTX A5000 GPU, PointAcc, MARS, and MoC. (b) Energy saving results. (c) Speedup results of different optimizations over the unoptimized design. (MPQ: mixed-precision quantization, PRR: point result reuse)

TABLE II
EVALUATION OF SHAPE GENERATION ACCURACY

Method	DPM			ShapeGF		
	Airplane	Chair	Car	Airplane	Chair	Car
<i>Original</i>	67.051	59.909	77.652	80.493	57.703	62.926
PTQ4DM	72.652	63.144	90.341	83.086	59.139	64.489
ZeroQuant	67.545	60.414	84.280	81.111	58.459	65.057
ANT	74.547	61.577	87.689	86.049	68.202	75.142
<i>Ours</i>	67.462	60.313	80.682	80.617	58.233	67.330

shown in Tab. II, our approach achieves the best results and significantly exceeds another mixed-precision method. Fig. 9 displays visual generation results of different quantization methods. Compared to others, our approach yields fewer outliers, smoother edges, and more regular, complete shapes, resulting in superior visual quality.

Furthermore, we examined how activation bit-widths and sparsity levels impact the accuracy of PCDMs in our approach. As shown in Fig. 11 (a), while lower activation bit-widths generally cause greater degradation, moderate bit-widths can improve the 1-NNA metric. Meanwhile, Fig. 11 (b) demonstrates that increasing sparsity produces trends similar to those observed with reduced bit-widths. This is because moderately lowering bit-widths or increasing sparsity allows the model to retain only the most essential, robust information while filtering out minor fluctuations, resulting in smoother edges and an overall shape distribution closer to the ground truth.

C. End-to-End Performance

We evaluated RAPID’s end-to-end speedup and energy savings by running generation and auto-encoding tasks on different PCDMs, as shown in Fig. 10 (a) and (b). Compared to the NVIDIA RTX A5000 GPU, PointAcc, MARS, and MoC, RAPID achieved speedups of $9.22\times$, $4.66\times$, $3.69\times$, and $3.01\times$, and energy savings of $61.74\times$, $4.30\times$, $3.94\times$, and $2.76\times$, respectively. This enables low-latency, low-power 3D object generation on resource-constrained platforms. RAPID’s performance gains primarily come from parallel INT4 multiplications in Sec. V-B and support for point group pruning in Sec. V-C. Additionally, compared to the GPU, our work achieves speedups of $12.01\times$ and $9.92\times$ on linear-layer DPM, significantly higher than $7.81\times$ and $7.78\times$ on convolution-based ShapeGF. This difference is due to GPUs’ efficient convolution support via PyTorch’s high-performance libraries, while our work must first convert convolutions into large-scale matrix multiplications, incurring extra overhead.

D. Ablation Study

To analyze the effectiveness of RAPID’s optimizations, ablation experiments were conducted. Fig. 10 (c) breaks down

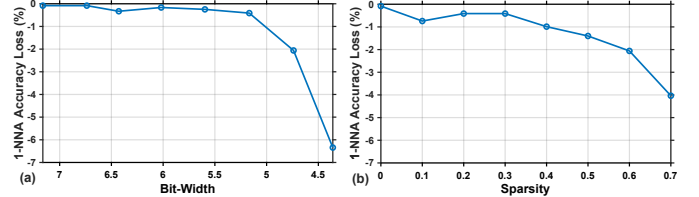


Fig. 11. (a) Accuracy loss at different quantization bit-widths. (b) Accuracy loss at different sparsity levels.

TABLE III
IMPACT OF DIFFERENT OPTIMIZATIONS ON ACCURACY

	Original	MPQ w./o. K-means	MPQ	PRR	All Opt.
1-NNA	67.051	68.698	67.133	67.133	67.462

their individual impacts. Compared to the design without our optimizations, mixed-precision quantization (MPQ) achieves an average speedup of $1.59\times$ by reducing bit operations, while point result reuse (PRR) reduces the number of points processed per timestep, achieving an average speedup of $1.33\times$. When combined, these optimizations deliver an overall speedup of $2.12\times$. Furthermore, while MPQ and PRR offer similar speedups in generation, MPQ significantly outperforms PRR in auto-encoding. This is because models for auto-encoding involve more timesteps, and quantizing a larger proportion to low bit-width has minimal impact on accuracy, allowing MPQ to yield greater benefits. Moreover, Table III shows the effects of optimizations on accuracy and the precision gains from K-means-based point grouping in Sec. IV-A.

VII. CONCLUSION

PCDMs generate detailed 3D shapes but incur high computational and storage costs. To optimize these models without sacrificing quality, we propose RAPID. First, it uses K-means to group nearby points, reducing the effects of activation fluctuations, and then applies mixed-precision quantization to different groups to eliminate spatial bit-width redundancy. Second, it reuses neural network results at the previous timestep for point groups with minimal changes, cutting redundant computations. Moreover, RAPID’s hardware employs a mixed-precision PE array for activations at varying bit-widths and a point group filter for bit-width allocation and pruning. Experimental results show that RAPID enables low-latency, low-power, and high-accuracy point cloud generation.

ACKNOWLEDGMENT

This work is supported in part by the Strategic Priority Research Program of the Chinese Academy of Sciences under grant No. (XDB0660000, XDB0660100, XDB0660102, XDB0660103) and the National Natural Science Foundation of China (NSFC) under grant No. 62090024. The corresponding authors are Xiaowei Li and Guihai Yan.

REFERENCES

- [1] S. Luo and W. Hu, "Diffusion probabilistic models for 3d point cloud generation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 2837–2845.
- [2] R. Cai, G. Yang, H. Averbuch-Elor, Z. Hao, S. Belongie, N. Snavely, and B. Hariharan, "Learning gradient fields for shape generation," in *European Conference on Computer Vision*. Springer, 2020, pp. 364–381.
- [3] G. K. Nakayama, M. A. Uy, J. Huang, S.-M. Hu, K. Li, and L. Guibas, "Diffacto: Controllable part-based 3d point cloud generation with cross diffusion," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 14257–14267.
- [4] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan, "Pointflow: 3d point cloud generation with continuous normalizing flows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 4541–4550.
- [5] P. Achlioptas, O. Diamanti, I. Miliagkas, and L. Guibas, "Learning representations and generative models for 3d point clouds," in *International conference on machine learning*. PMLR, 2018, pp. 40–49.
- [6] Y. Lin, Z. Zhang, H. Tang, H. Wang, and S. Han, "Pointacc: Efficient point cloud accelerator," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021, pp. 449–461.
- [7] X. Yang, T. Fu, G. Dai, S. Zeng, K. Zhong, K. Hong, and Y. Wang, "An efficient accelerator for point-based and voxel-based point cloud neural networks," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.
- [8] Z. Song, N. Jing, and X. Liang, "Prada: Point cloud recognition acceleration via dynamic approximation," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2023, pp. 1–6.
- [9] F. Chen, R. Ying, J. Xue, F. Wen, and P. Liu, "Parallelnn: A parallel octree-based nearest neighbor search accelerator for 3d point clouds," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2023, pp. 403–414.
- [10] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.
- [11] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3075–3084.
- [12] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.
- [13] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "Haq: Hardware-aware automated quantization with mixed precision," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 8612–8620.
- [14] H. Xi, Y. Chen, K. Zhao, K. J. Teh, J. Chen, and J. Zhu, "Jetfire: Efficient and accurate transformer pretraining with int8 data flow and per-block quantization," *arXiv preprint arXiv:2403.12422*, 2024.
- [15] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.
- [16] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.
- [17] X. Liu, Z. Song, H. Chen, X. Li, and X. Liang, "Moc: A morton-code-based fine-grained quantization for accelerating point cloud neural networks," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024, pp. 1–6.
- [18] K. L. Chung, "Markov chains," *Springer-Verlag, New York*, 1967.
- [19] J. Xu, Z. Li, B. Du, M. Zhang, and J. Liu, "Reluplex made more practical: Leaky relu," in *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2020, pp. 1–7.
- [20] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale," *Advances in neural information processing systems*, vol. 35, pp. 30318–30332, 2022.
- [21] E. E. Swartzlander and A. G. Alexopoulos, "The sign/logarithm number system," *IEEE Transactions on Computers*, vol. 100, no. 12, pp. 1238–1242, 1975.
- [22] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10684–10695.
- [23] K. Chellapilla, S. Puri, and P. Simard, "High performance convolutional neural networks for document processing," in *Tenth international workshop on frontiers in handwriting recognition*. Suvisoft, 2006.
- [24] C. Guo, C. Zhang, J. Leng, Z. Liu, F. Yang, Y. Liu, M. Guo, and Y. Zhu, "Ant: Exploiting adaptive numerical data type for low-bit deep neural network quantization," in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2022, pp. 1414–1433.
- [25] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.
- [26] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, J. K. Kim, V. Chandra, and H. Esmailzadeh, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 764–775.
- [27] S. Li, Z. Yang, D. Reddy, A. Srivastava, and B. Jacob, "Dramsim3: A cycle-accurate, thermal-capable dram simulator," *IEEE Computer Architecture Letters*, vol. 19, no. 2, pp. 106–109, 2020.
- [28] R. Balasubramonian, A. B. Kahng, N. Muralimanohar, A. Shafiee, and V. Srinivas, "Cacti 7: New tools for interconnect exploration in innovative off-chip memories," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 14, no. 2, pp. 1–25, 2017.
- [29] Y. Shang, Z. Yuan, B. Xie, B. Wu, and Y. Yan, "Post-training quantization on diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 1972–1981.
- [30] Z. Yao, R. Yazdani Aminabadi, M. Zhang, X. Wu, C. Li, and Y. He, "Zeroquant: Efficient and affordable post-training quantization for large-scale transformers," *Advances in neural information processing systems*, vol. 35, pp. 27168–27183, 2022.