LSPU: A 20.7ms Low-latency Point Neural Network-based 3D Perception and Semantic LiDAR SLAM System-on-Chip for Autonomous Driving System

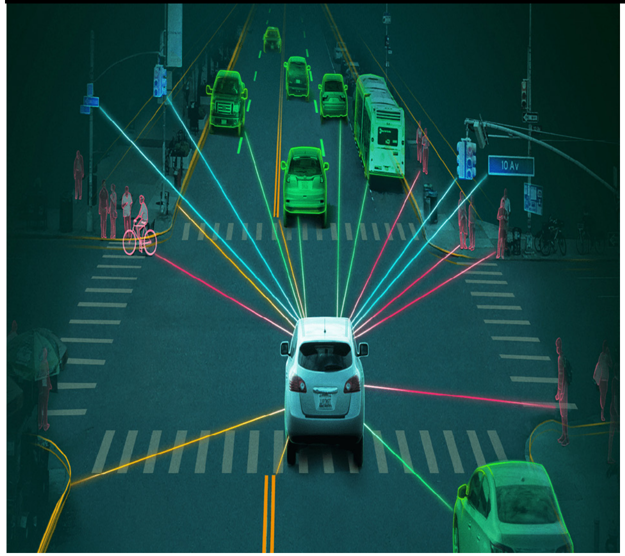
Jueun Jung[1], Seungbin Kim[1], Bokyoung Seo[1], Wuyoung Jang[1],
Sangho Lee[1], Jeongmin Shin[1], Donghyeon Han[2], and Kyuho Jason Lee[1]

[1]Intelligent Systems Lab., Department of EE, UNIST, Korea
[2]Department of EECS, MIT, USA

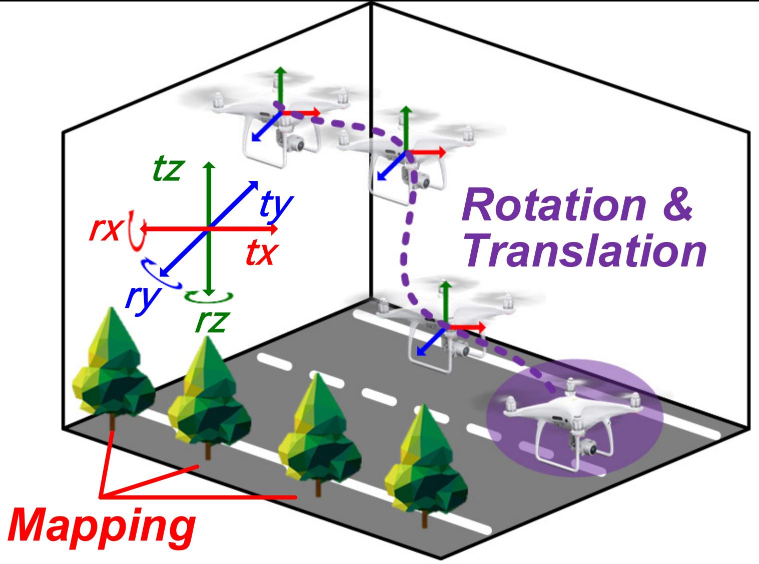
Semantic SLAM for Autonomous Driving

- ❑ **Intelligent 3D Interaction with Wide & Dynamic Surroundings**
  - Accurate & Reliable Mapping with 3D Semantic Information
  - Essential component for advanced autonomous driving systems

Semantic SLAM → 3D Information



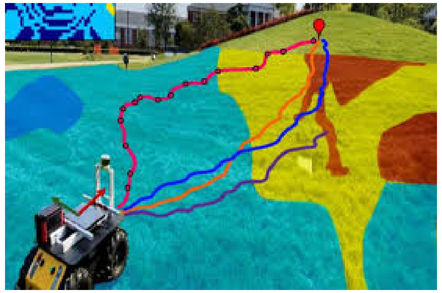
3D Perception (What is that?)



Odometry & Mapping (Where am I?)

Autonomous Mobile Robots

Path Planning

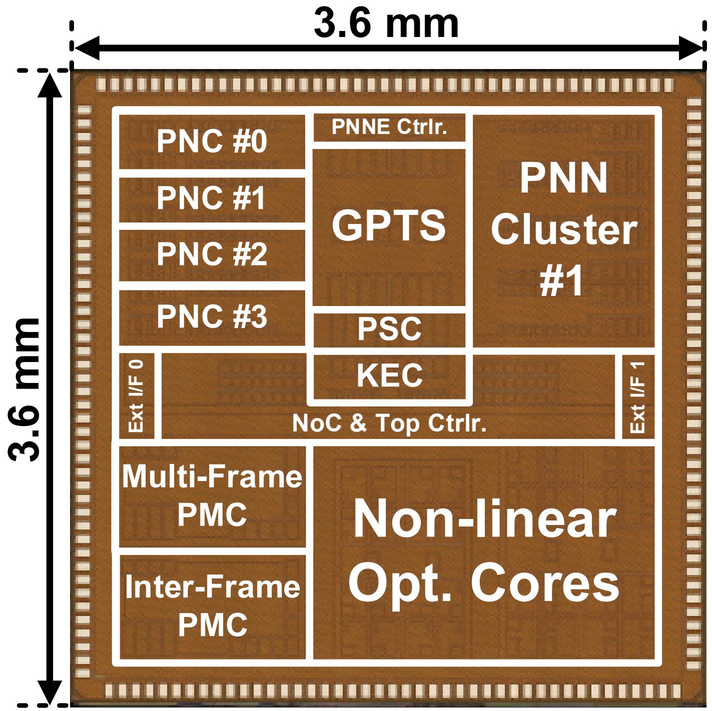


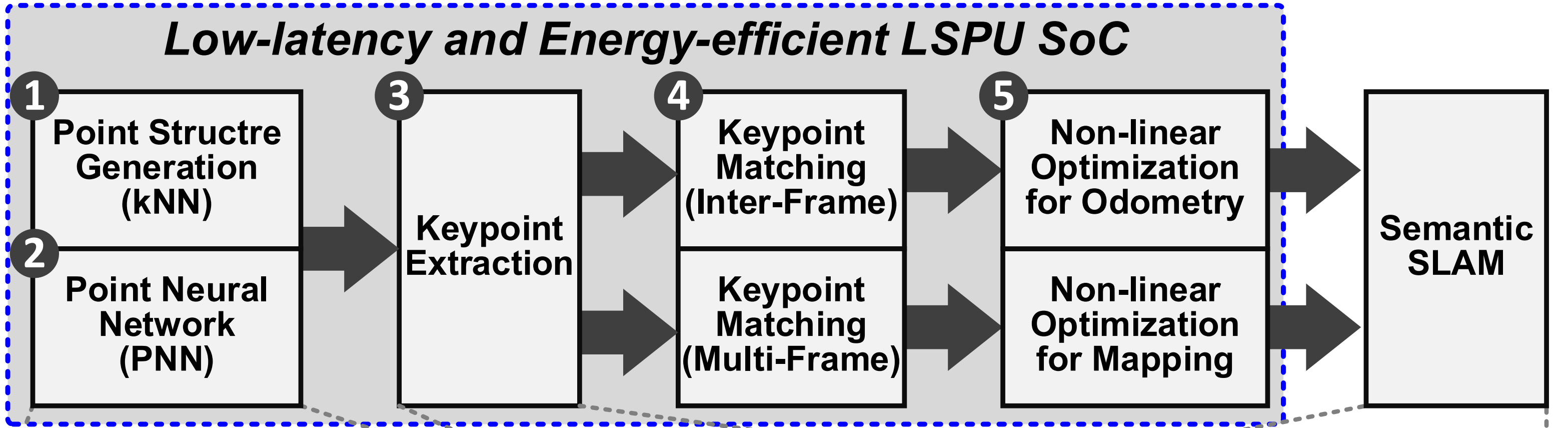
Navigation

LSPU: End-to-end Semantic SLAM SoC

**Point Neural Network*

- ❑ **A 20.7ms and 349.6mw Semantic LiDAR SLAM Processor**
  - – *1) *PNN-based 3D Perception & LiDAR SLAM SW/HW Architecture*
  - – *2) 5 Heterogeneous Architecture with SIMD / Reconfigurable PE*





Low-latency and Energy-efficient LSPU SoC

1 Point Structre Generation (kNN)

2 Point Neural Network (PNN)

3 Keypoint Extraction

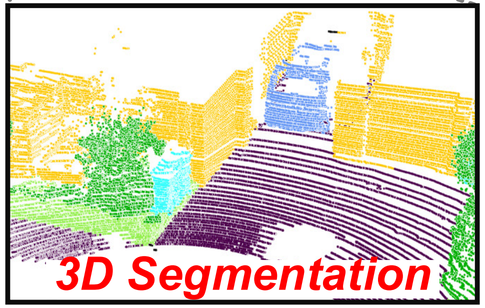
4 Keypoint Matching (Inter-Frame)

Keypoint Matching (Multi-Frame)

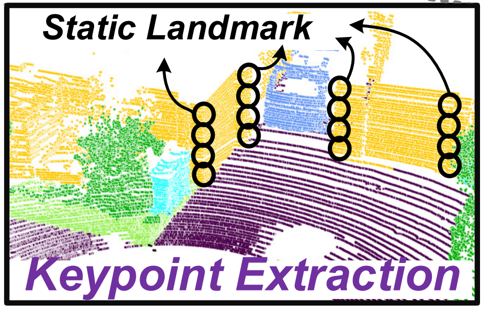
5 Non-linear Optimization for Odometry

Non-linear Optimization for Mapping

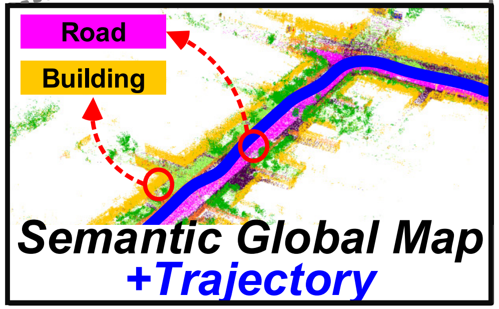
Semantic SLAM



3D Segmentation



Keypoint Extraction

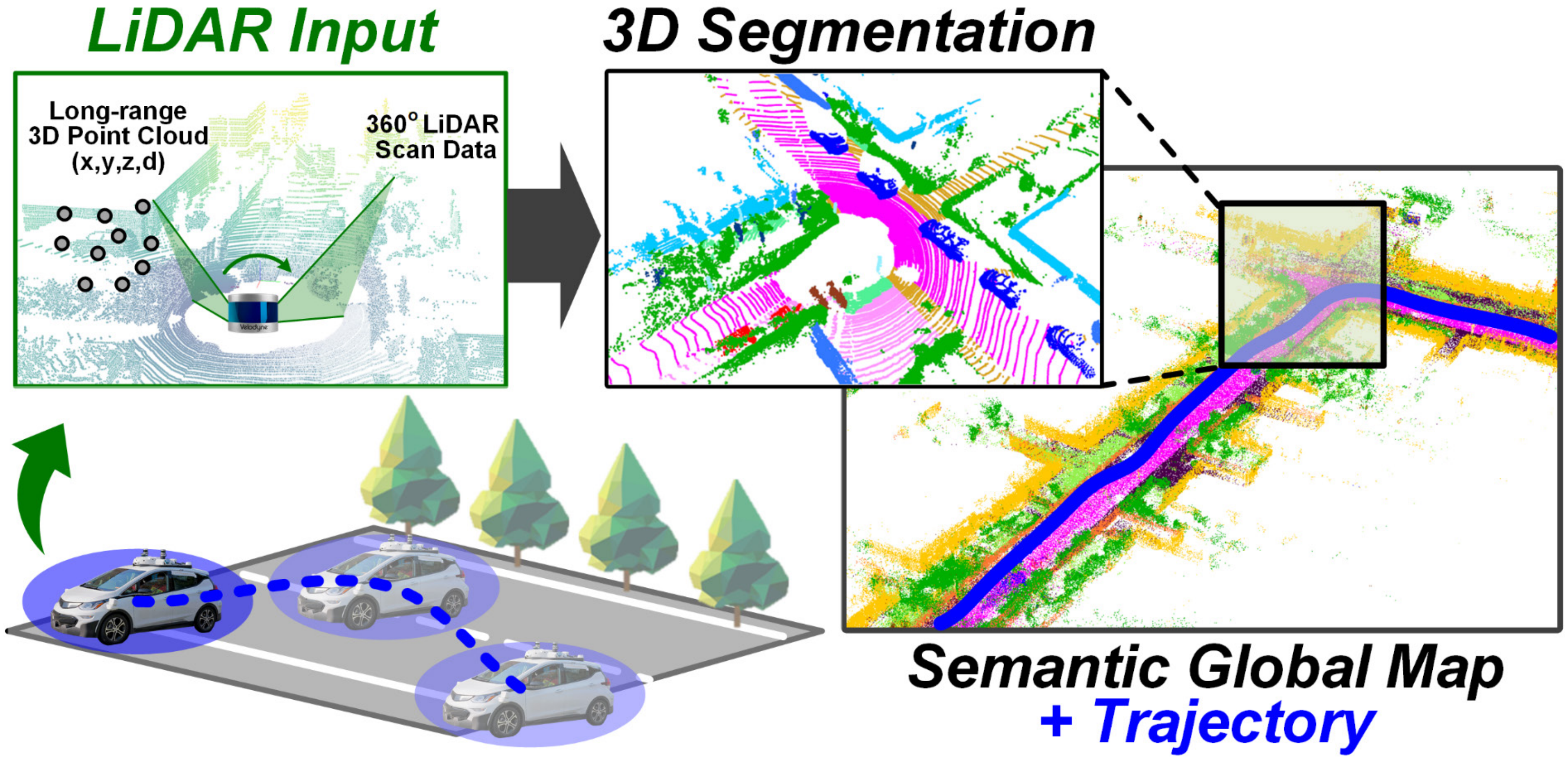


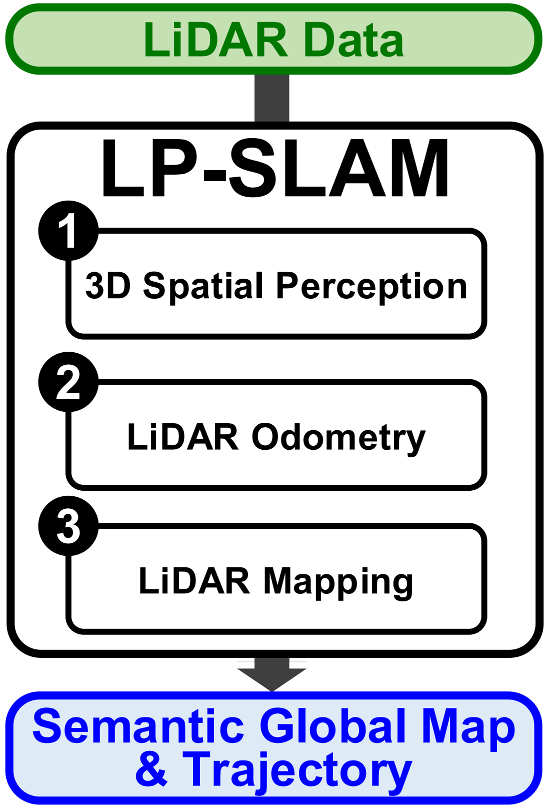
Semantic Global Map
+Trajectory

Proposed Semantic LiDAR SLAM

❑ **3 Stages of LiDAR Point-Neural-Network SLAM (LP-SLAM)**

– ❶ 3D Spatial Perception → ❷ LiDAR Odometry → ❸ LiDAR Mapping





LiDAR Data

LP-SLAM

❶ 3D Spatial Perception

❷ LiDAR Odometry

❸ LiDAR Mapping

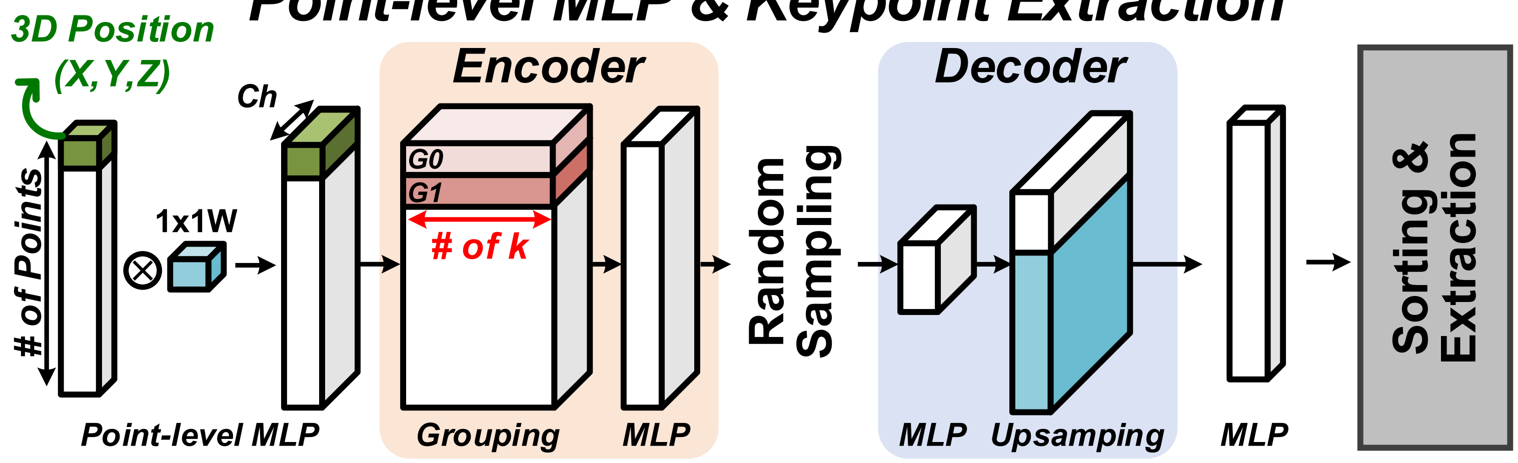
Semantic Global Map & Trajectory

Stage 1: 3D Spatial Perception

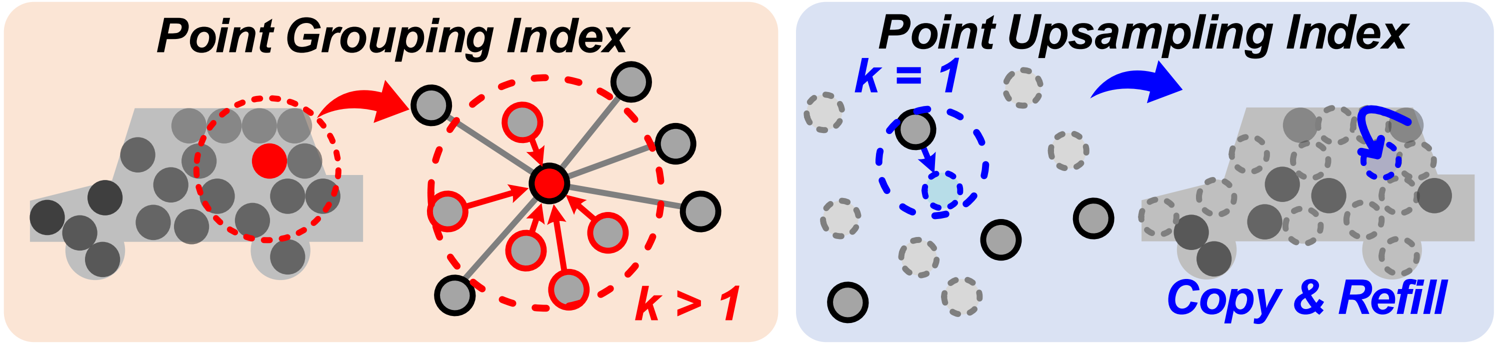
❑ 3D Segmentation & Keypoint Extraction with PNN

– Point-level MLP and intra-frame *kNN for grouping and upsampling

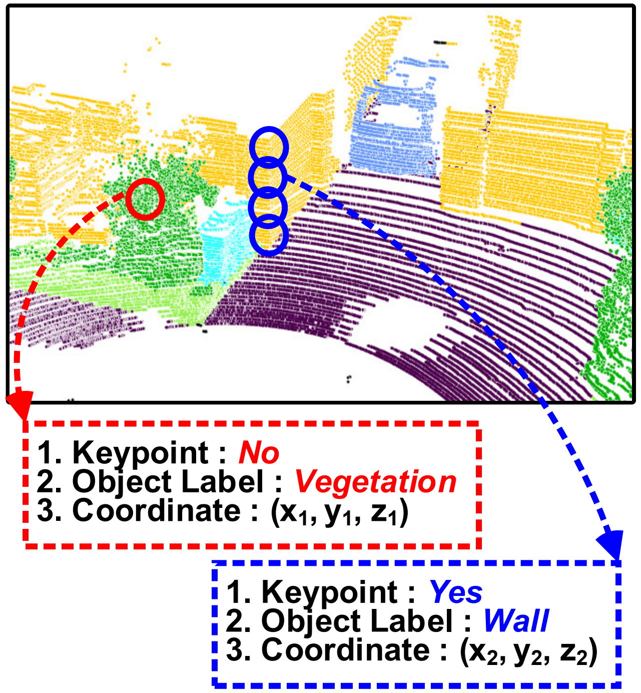
Point-level MLP & Keypoint Extraction



Intra-Frame kNN: Point Structure Generation



Accurate Spatial Perception



1. Keypoint : *No*
2. Object Label : *Vegetation*
3. Coordinate : (x_1, y_1, z_1)

1. Keypoint : *Yes*
2. Object Label : *Wall*
3. Coordinate : (x_2, y_2, z_2)

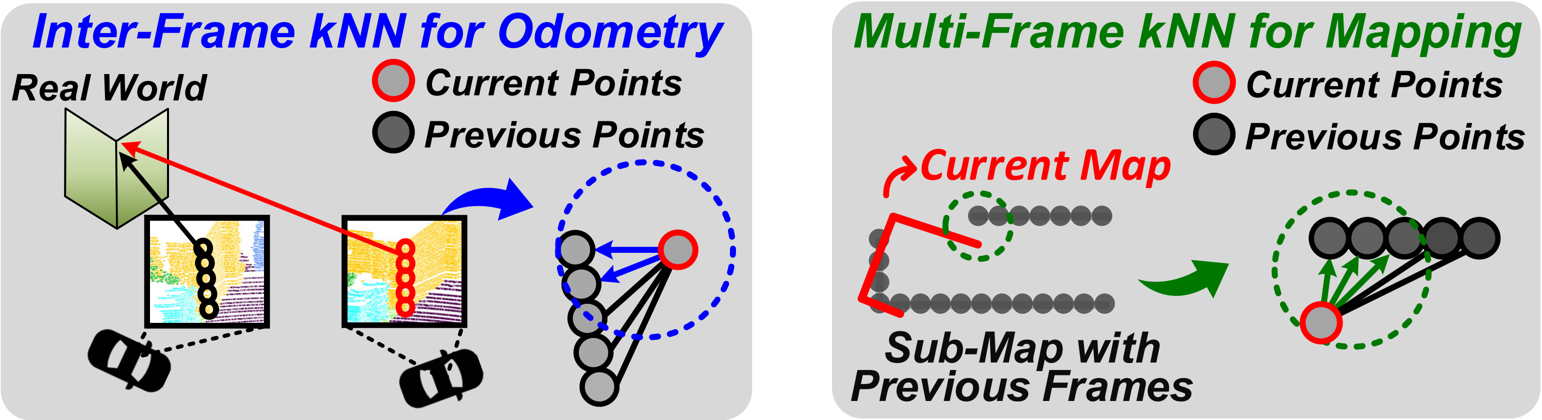
*k-Nearest Neighbor

Stage 2&3: LiDAR Odometry & Mapping

❑ Reconstruction of Global Map & Trajectory

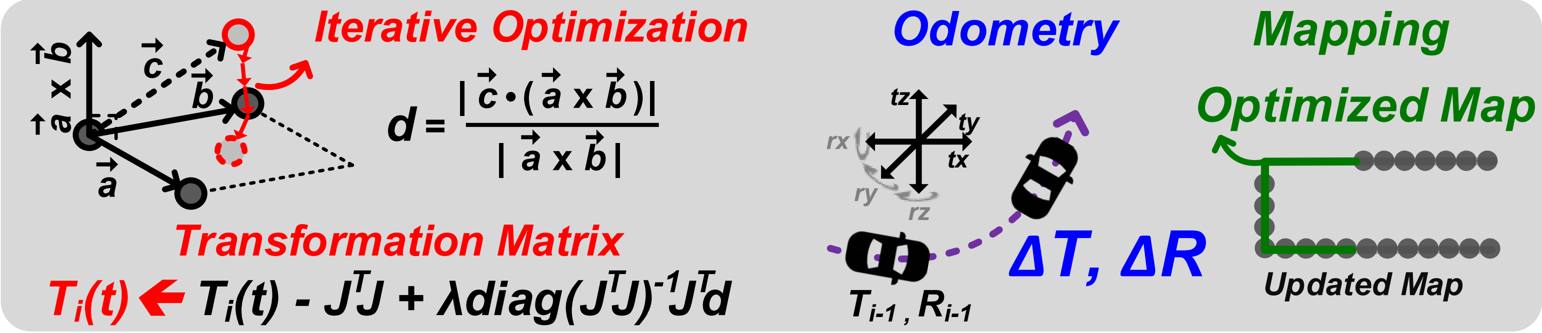
– Iteration of keypoint matching and non-linear optimization (NLO)

❶ *Keypoint Matching with kNN*



Inter-Frame kNN for Odometry
Real World
Current Points
Previous Points
Multi-Frame kNN for Mapping
Current Points
Previous Points
Current Map
Sub-Map with Previous Frames

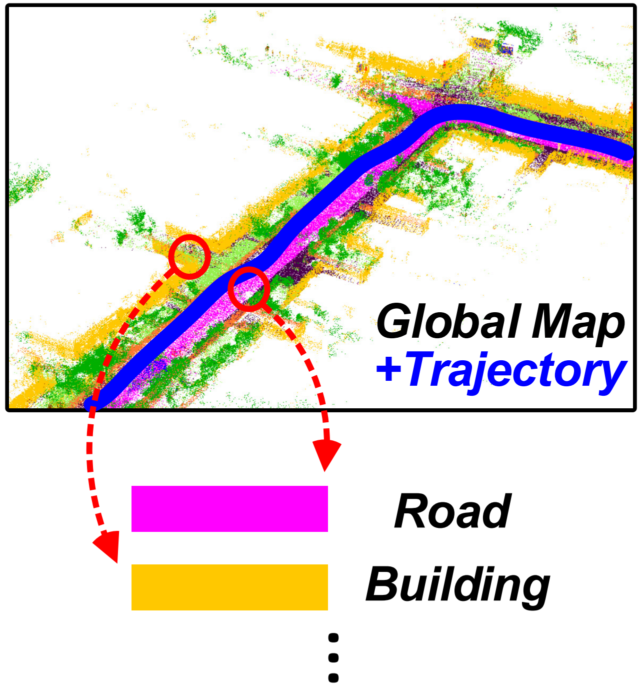
❷ *NLO (Levenberg-Marquardt Opt.)*



$$d = \frac{|\vec{c} \cdot (\vec{a} \times \vec{b})|}{|\vec{a} \times \vec{b}|}$$

$$T_i(t) \leftarrow T_i(t) - J^TJ + \lambda diag(J^TJ)^{-1}J^Td$$

Semantic LP-SLAM Result

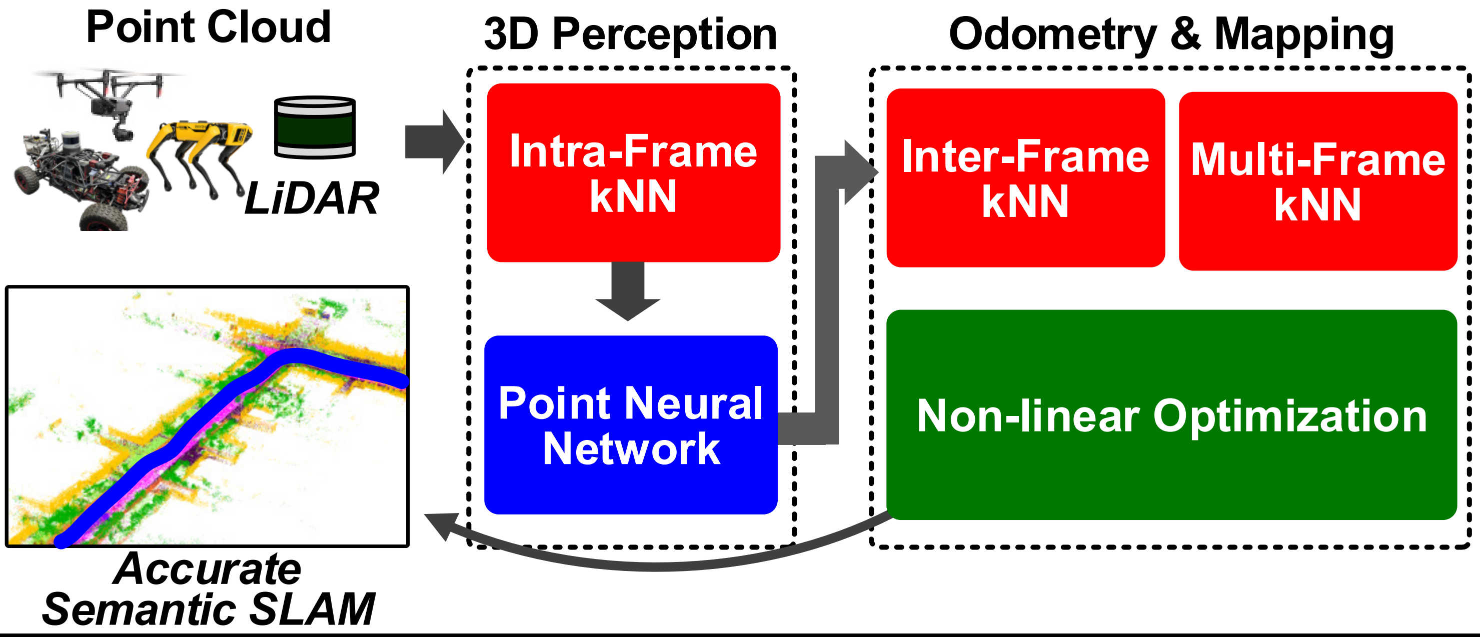


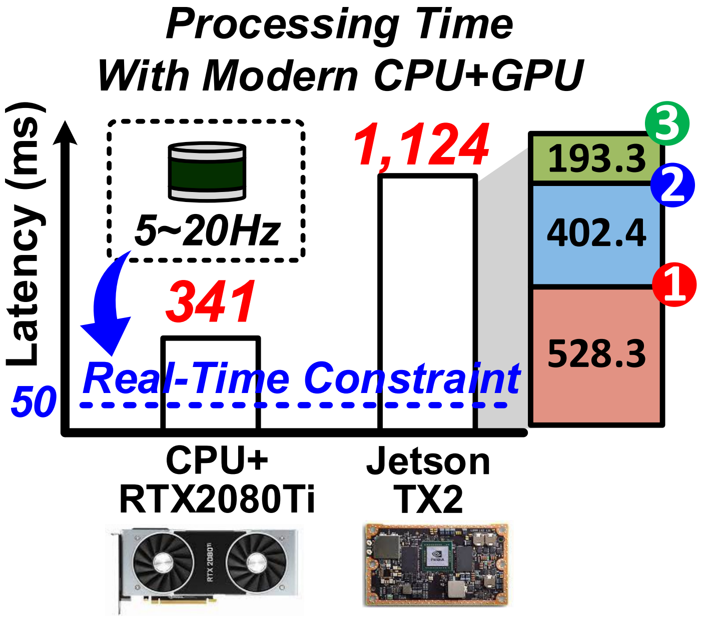
Hardware Requirement of Semantic SLAM

❑ Multiple Algorithms: Extreme Complexity & Memory Demand

– Composed of 3 massive operations: **1) kNN, 2) PNN,** and **3) NLO**

➔ CPU+GPU fails real-time processing of LP-SLAM (<50ms) ☹





A Dedicated SLAM Processor
with Heterogeneous Architecture **is Necessary**

Overall Architecture of LSPU

**Global Point-level Task Scheduler*

1. *Point Structure & Matching Core*

– 2D-SB based Upsampling Prediction

– 3D-SB based Memory Management Unit

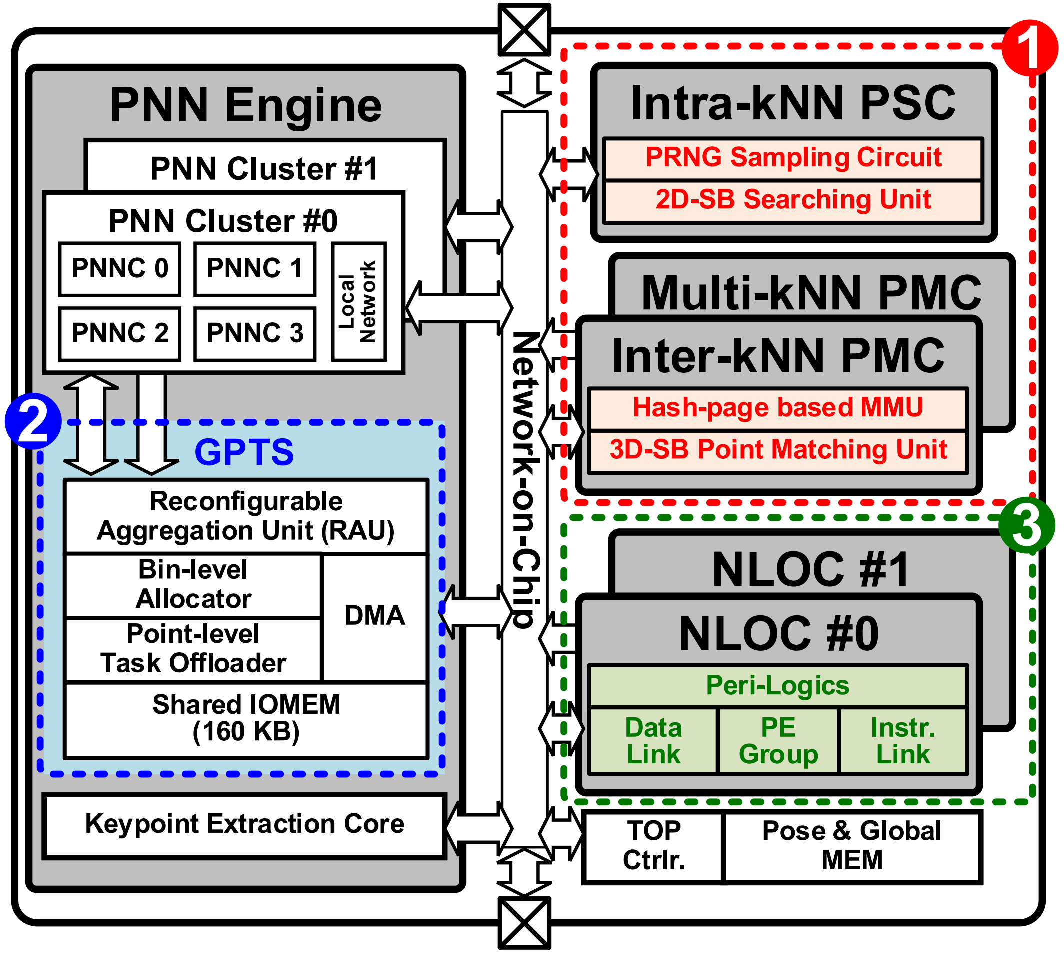
2. *PNN Engine w/ *GPTS*

– 2-step Workload Balancing

– 3-level Reconfigurable Aggregation Unit

3. *Non-linear Optimization Core*

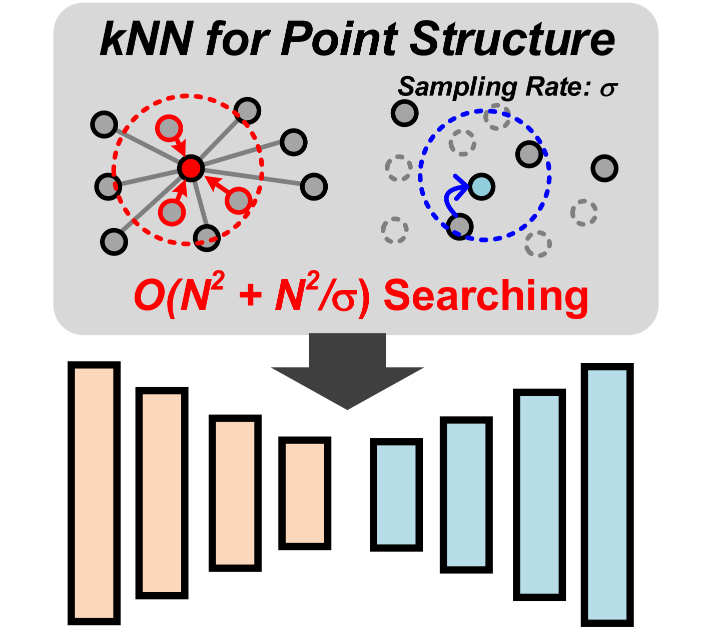
– Reconfigurable Computation Mode



kNN Challenge: Performance Bottleneck

❑ **Memory Intensive Computation with Large-scale LiDAR Points**

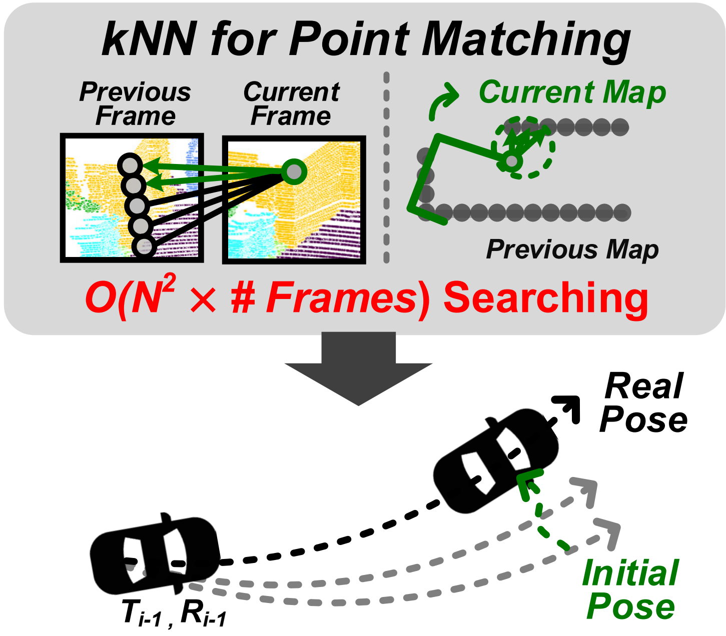
- Massive operation & memory access by kNN of O(N^2)
- **For point structure:** kNN iterations ∝ #(M)
- **For point matching:** kNN iterations ∝ #(P)



kNN for Point Structure

$O(N^2 + N^2/\sigma)$ Searching

M: # of Encoder / Decoder Pairs

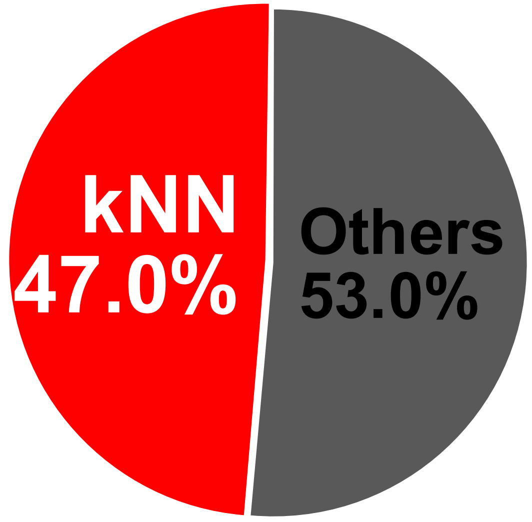


kNN for Point Matching

$O(N^2 \times \#\ Frames)$ Searching

P: Iteration of NLO

Latency Breakdown on Jetson TX2

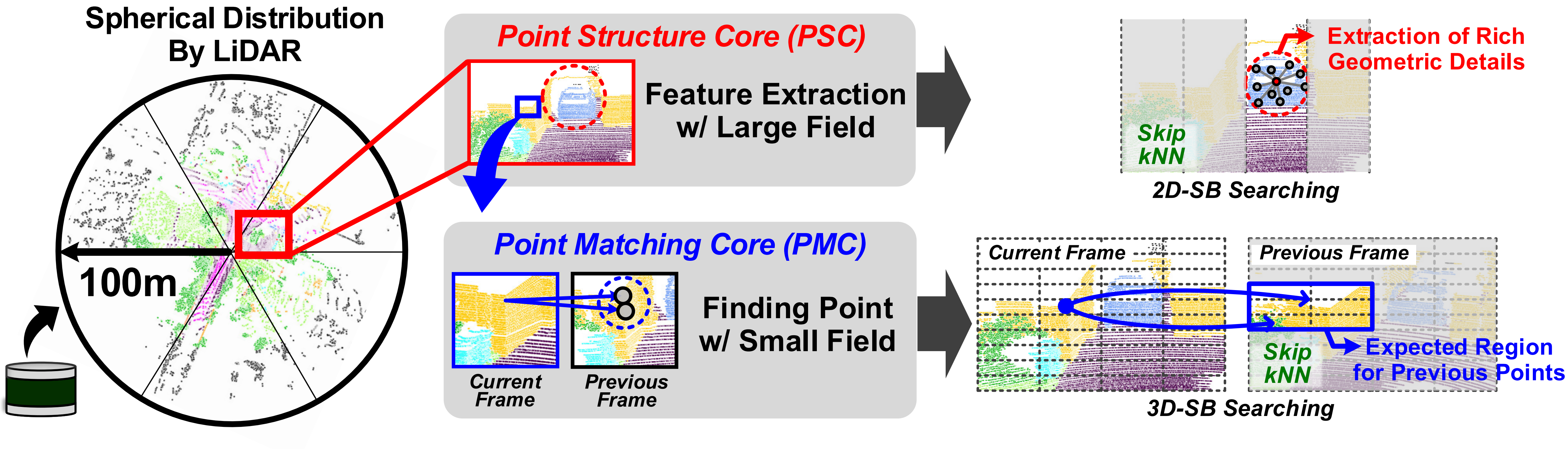


LiDAR-optimized Spherical Binning

❑ Reduction of Meaningless Searching Area

- Point structure core: 2D spherical bin (2D-SB) searching
- Point matching core: 3D spherical *neighbor-bin* (3D-SB) searching

➔ Reducing kNN operations by 97.6~98.5%



Spherical Distribution By LiDAR

100m

Point Structure Core (PSC)

Feature Extraction w/ Large Field

Point Matching Core (PMC)

Current Frame ***Previous Frame***

Finding Point w/ Small Field

Extraction of Rich Geometric Details

Skip kNN

2D-SB Searching

Current Frame ***Previous Frame***

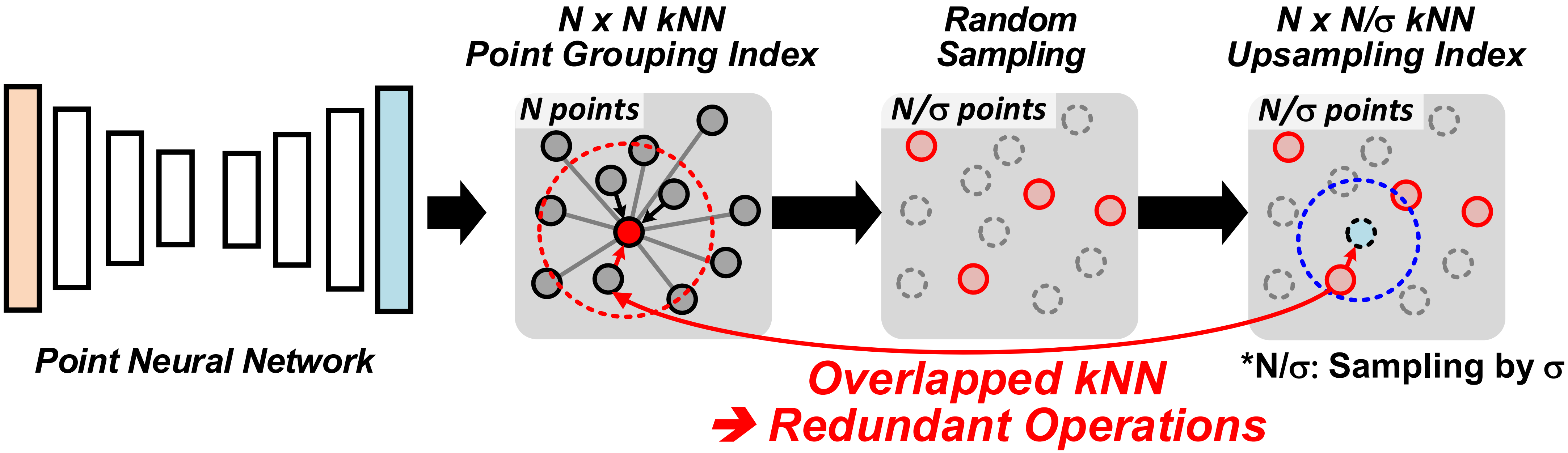
Skip kNN

Expected Region for Previous Points

3D-SB Searching

Problem of Point Structure Generation

- ❑ **Redundant Operations for Upsampling Index Searching**
  - – Point group searching: ‘N-to-N’ points
  - – Upsampling index searching: ‘N-to-N/σ’ points

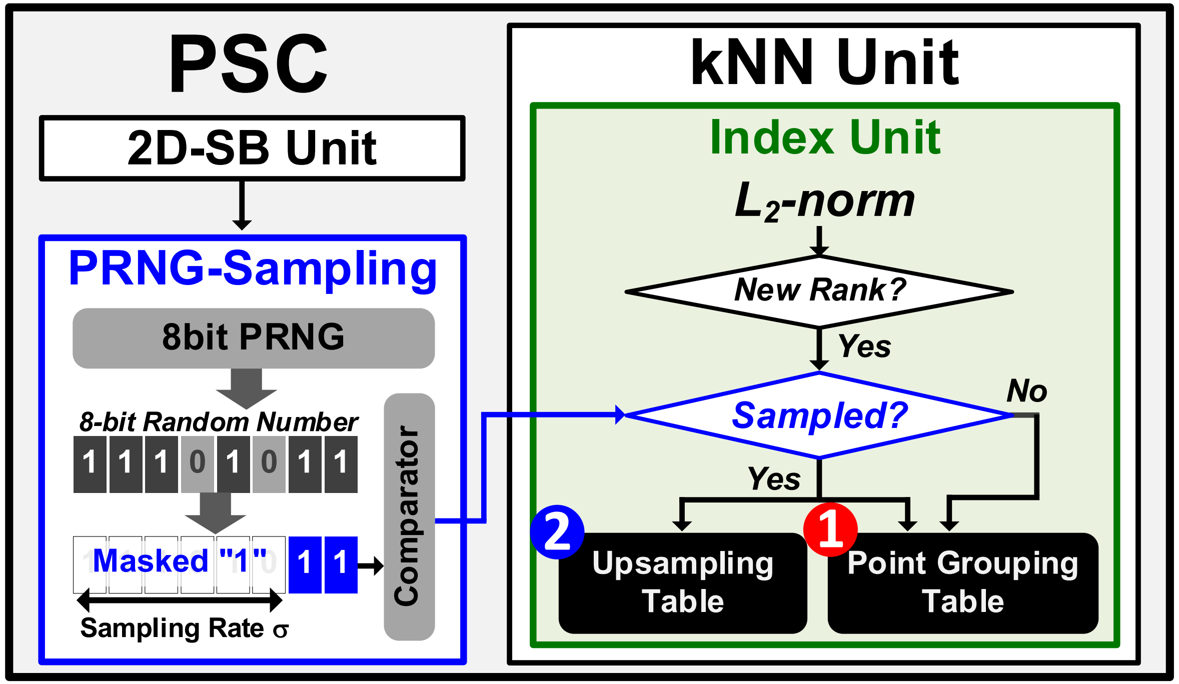


*N/σ: Sampling by σ

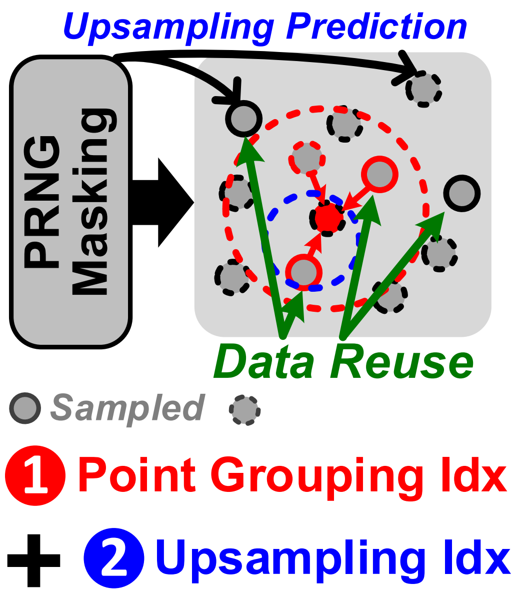
PSC: Upsampling Prediction

❑ In-advance Upsampling Prediction with **PRNG-Sampling Circuit**

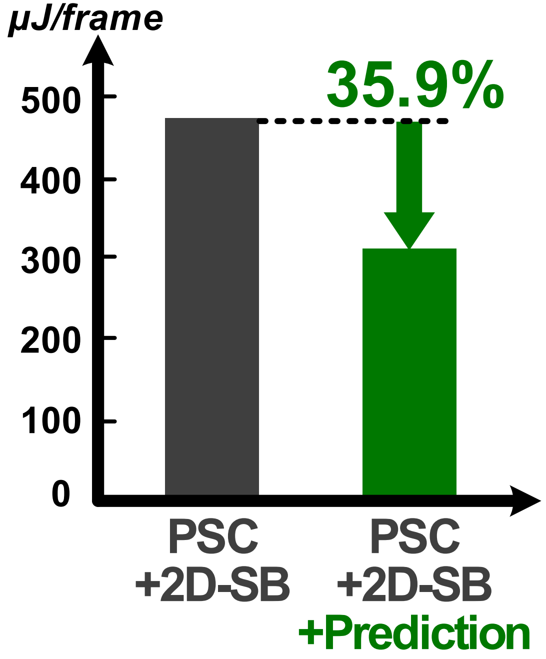
- Data *reuse* in point group searching
- *Unified* point structure generation ➔ **35.9% Energy ▼**



Unified Point Structure Generation (*N* × *N kNN*)



< Energy per Frame>

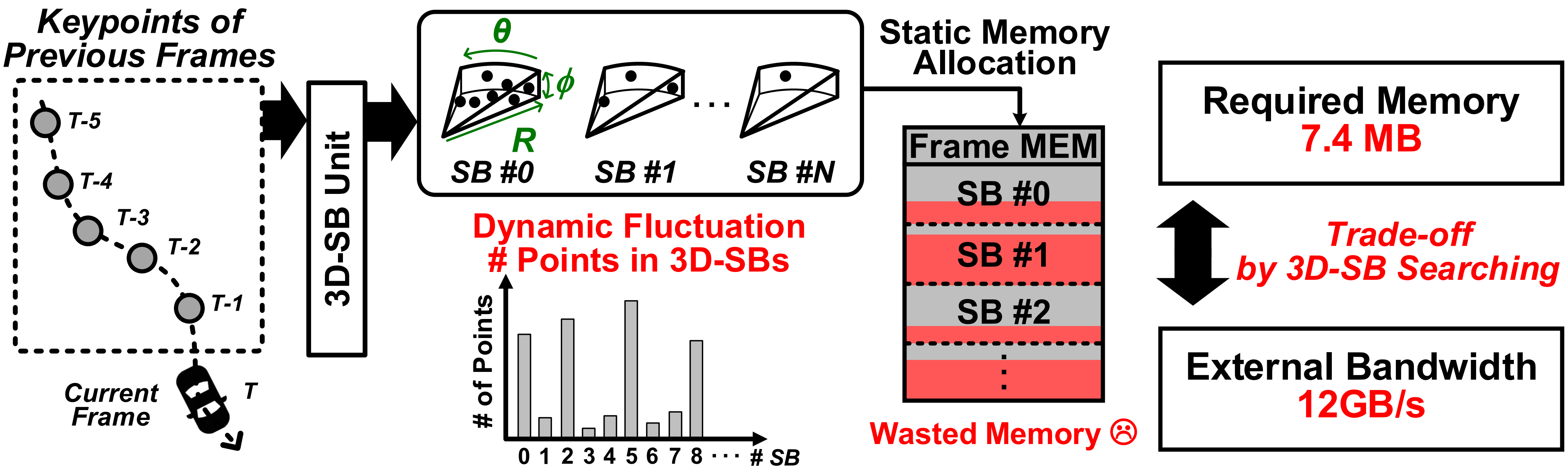


Problem of 3D-SB Searching

❑ Memory Overhead by Fine-grained 3D-SB Searching

– Dynamic fluctuation in 3D-SBs ➔ static memory allocation w/ max. size

– Requires ***7.4MB large on-chip memory* vs. *12GB/s of EMA***



Keypoints of Previous Frames

Static Memory Allocation

Dynamic Fluctuation # Points in 3D-SBs

Required Memory 7.4 MB

Trade-off by 3D-SB Searching

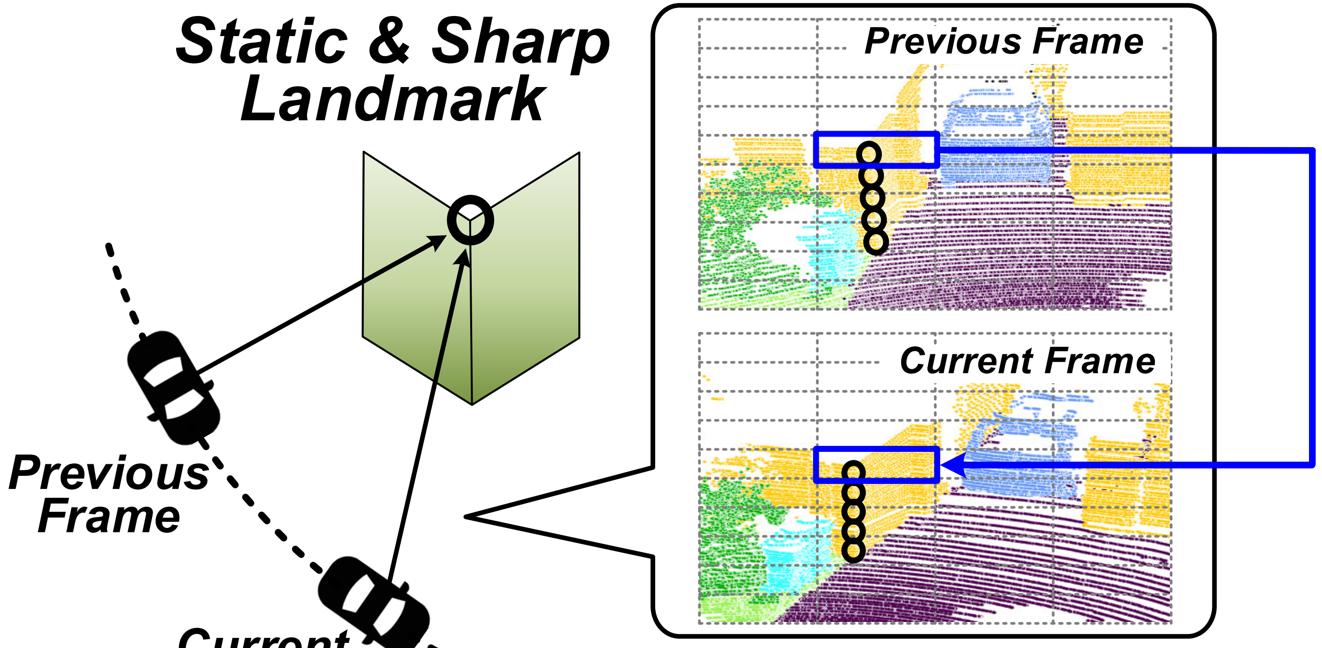
External Bandwidth 12GB/s

Wasted Memory ☹

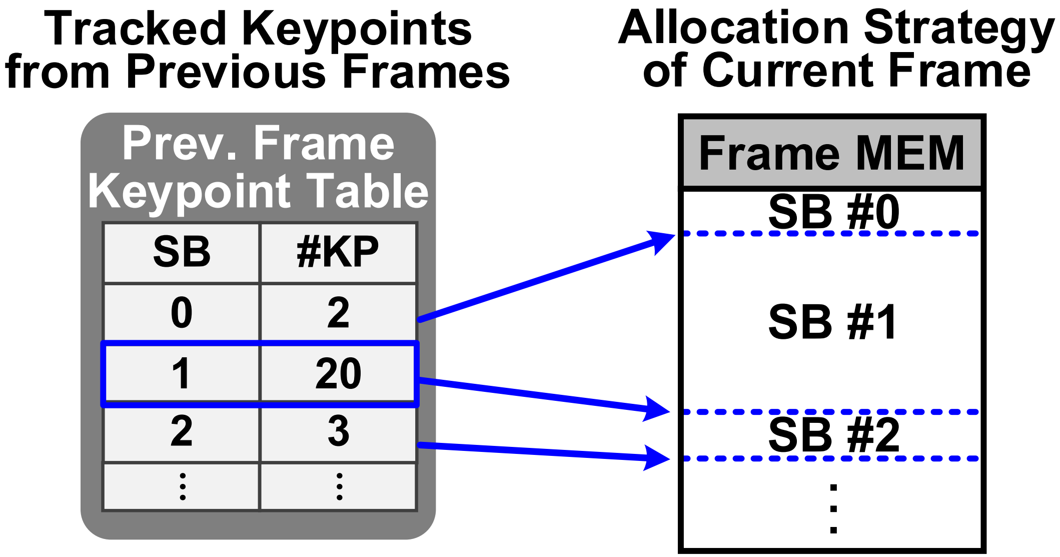
Observation: Bin-wise Coherency

❑ **Predicting # of Keypoints in 3D-SBs using Neighboring Frames**

- Static landmark across multiple frames
- Tracking the bin-level allocation strategy from previous frames



Spatio-temporal Coherency of Keypoints



Tracked Keypoints from Previous Frames
Allocation Strategy of Current Frame
Prev. Frame Keypoint Table
Frame MEM
SB #0
SB #1
SB #2
⋮

SB	#KP
0	2
1	20
2	3
⋮	⋮

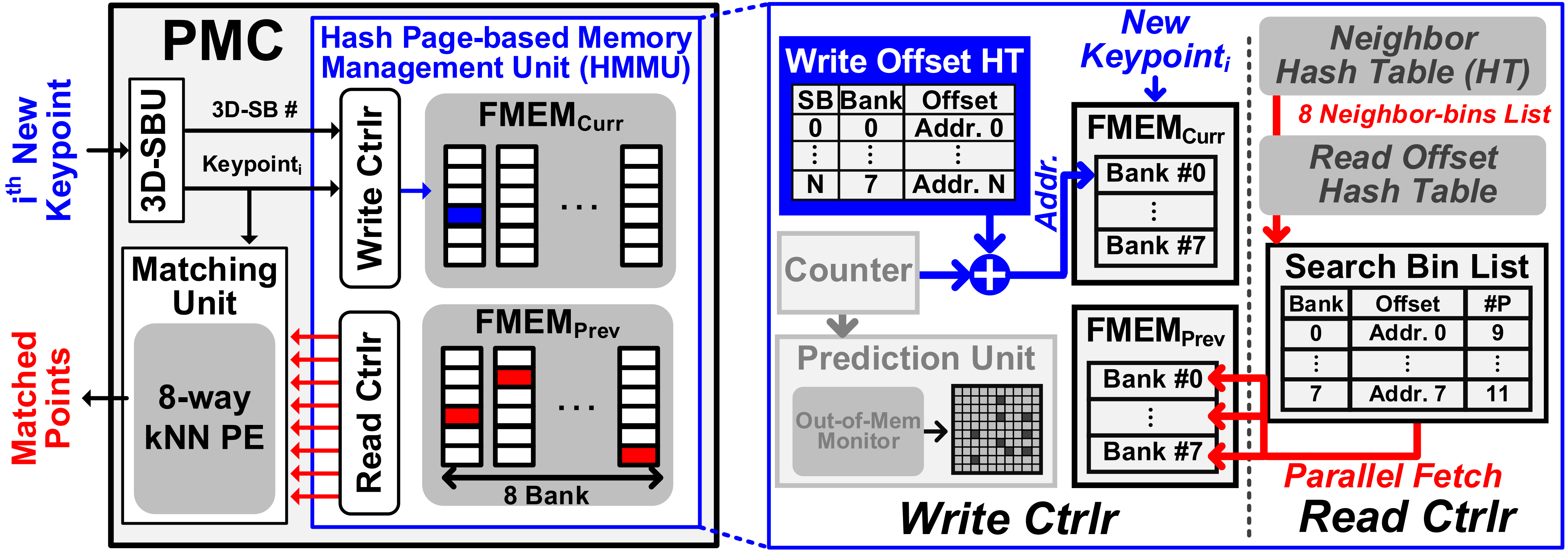
Dynamic Bin-wise Allocation based on Prediction

PMC: Dynamic Memory Allocation

❑ **Hash Page-based Memory Management Unit (HMMU)**

– **Write controller:** storing new keypoint into predicted memory space

– **Read controller:** parallel fetching to search neighbor-bins simultaneously



Write Offset HT

SB	Bank	Offset
0	0	Addr. 0
⋮	⋮	⋮
N	7	Addr. N

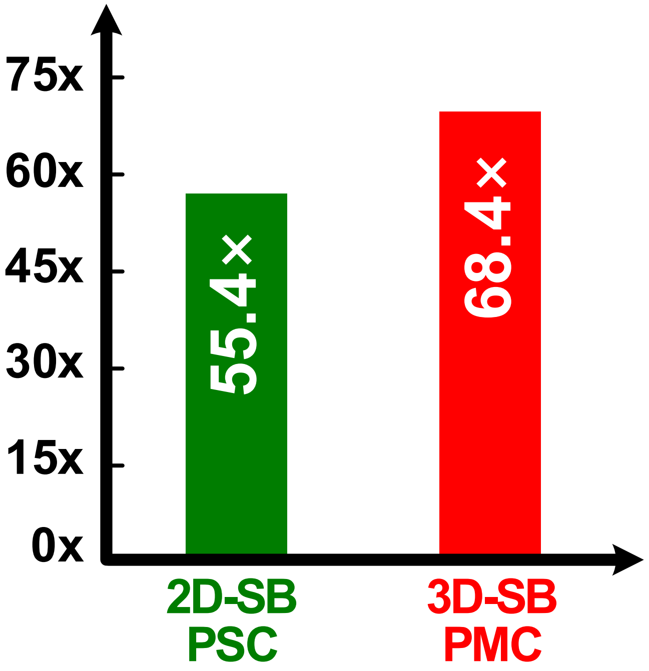
Search Bin List

Bank	Offset	#P
0	Addr. 0	9
⋮	⋮	⋮
7	Addr. 7	11

kNN Performance w/ PSC & PMC

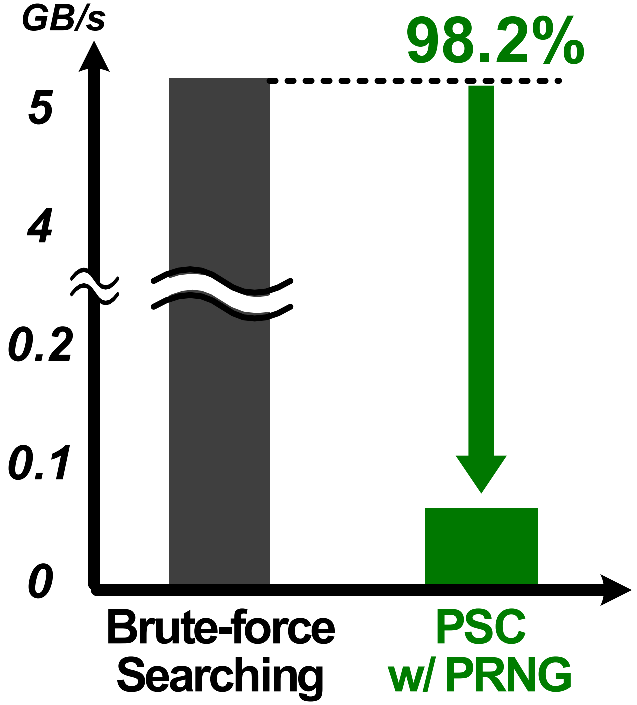
- ❑ 2D-SB PSC: 55.4× Speed-up & 98.2% of Memory Bandwidth ▼
- ❑ 3D-SB PMC: 68.4× Speed-up & 96.9% of Memory Size ▼

< *Speed-up for kNN >

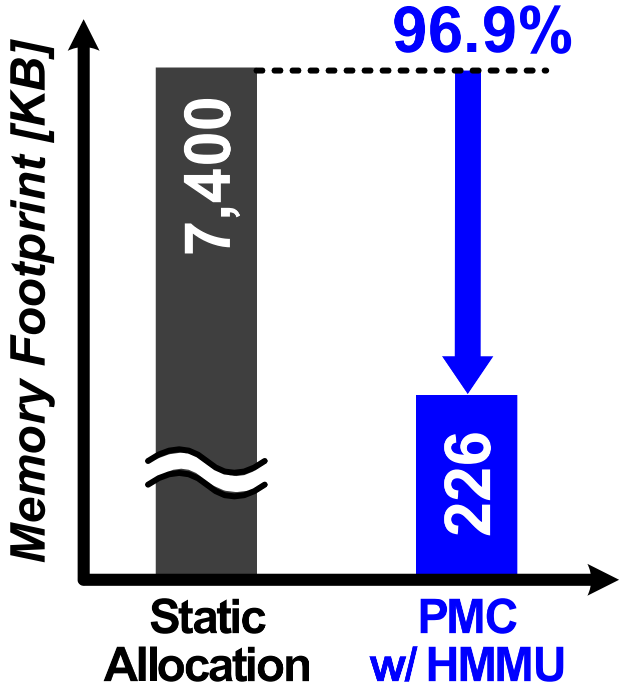


*Baseline: Brute-force Searching

< Memory BW w/ PSC >



< On-chip Memory Size w/ PMC >

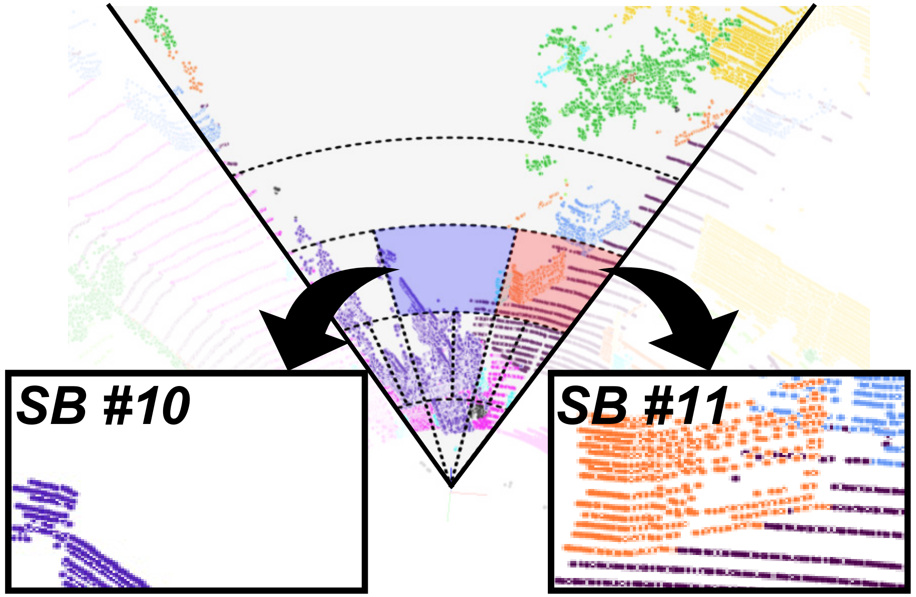


PNN Challenge: Workload Imbalance

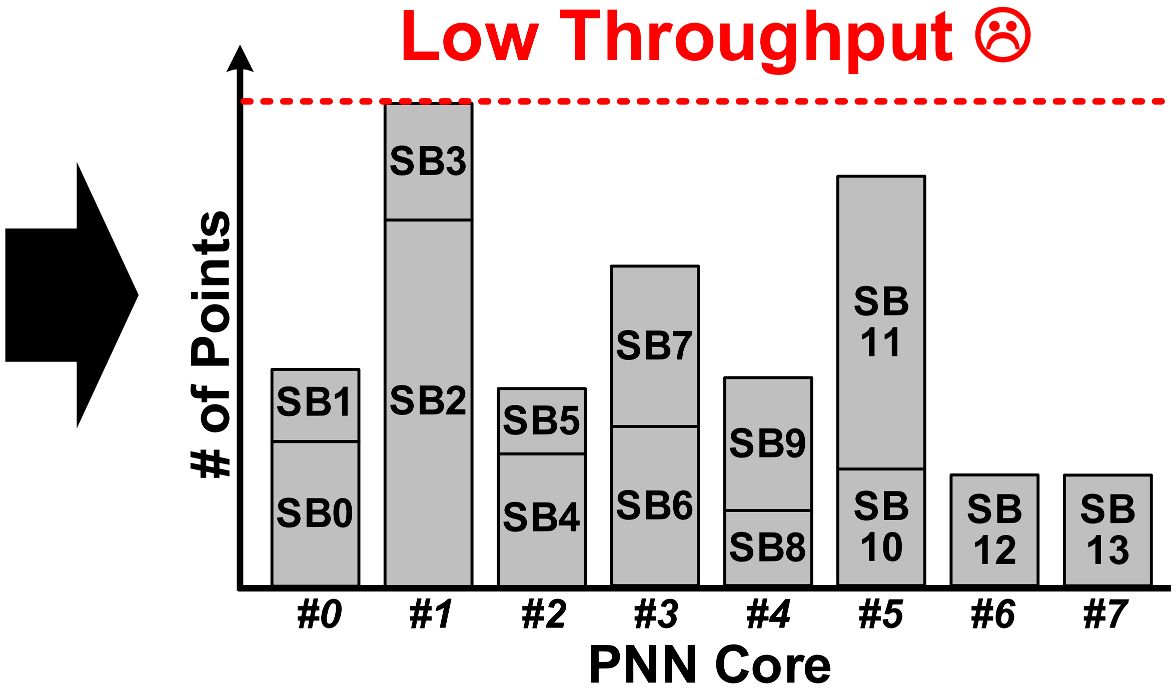
❑ **Bin-level Point-count Variations in Dynamic Environments**

– Points are assigned to PNN cores by static SB-level pre-allocation

– Workload Imbalance b/w each cores



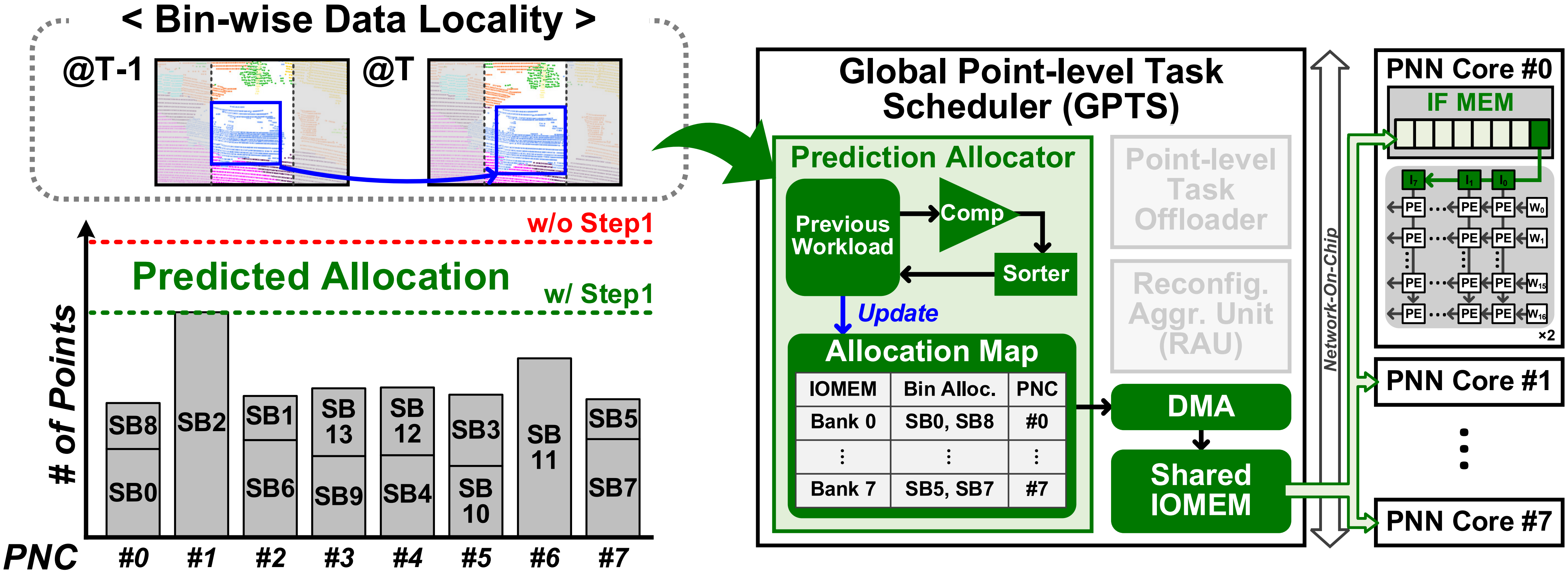
SB-level Imbalanced Workload



GPTS: 2-step Workload Balancing (1/2)

❑ Step 1: SB-level Predicted Workload Allocation

- Prediction of current workload by using data locality within intra-frame
- Workload variation b/w predicted & current workloads

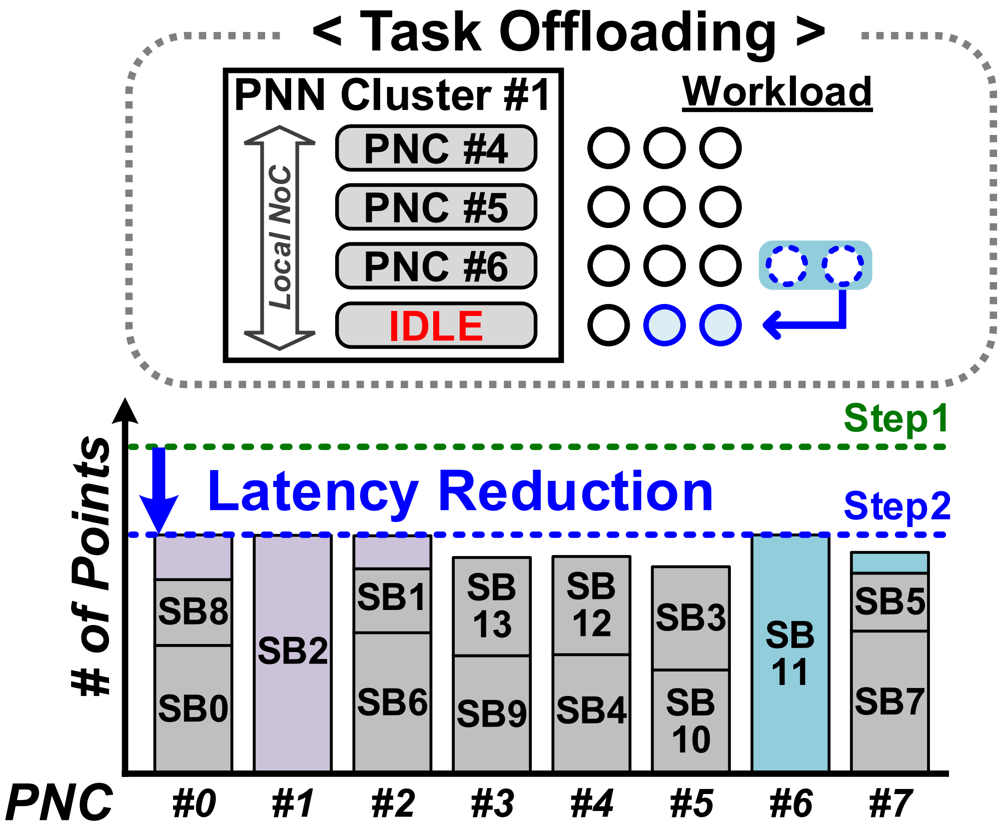


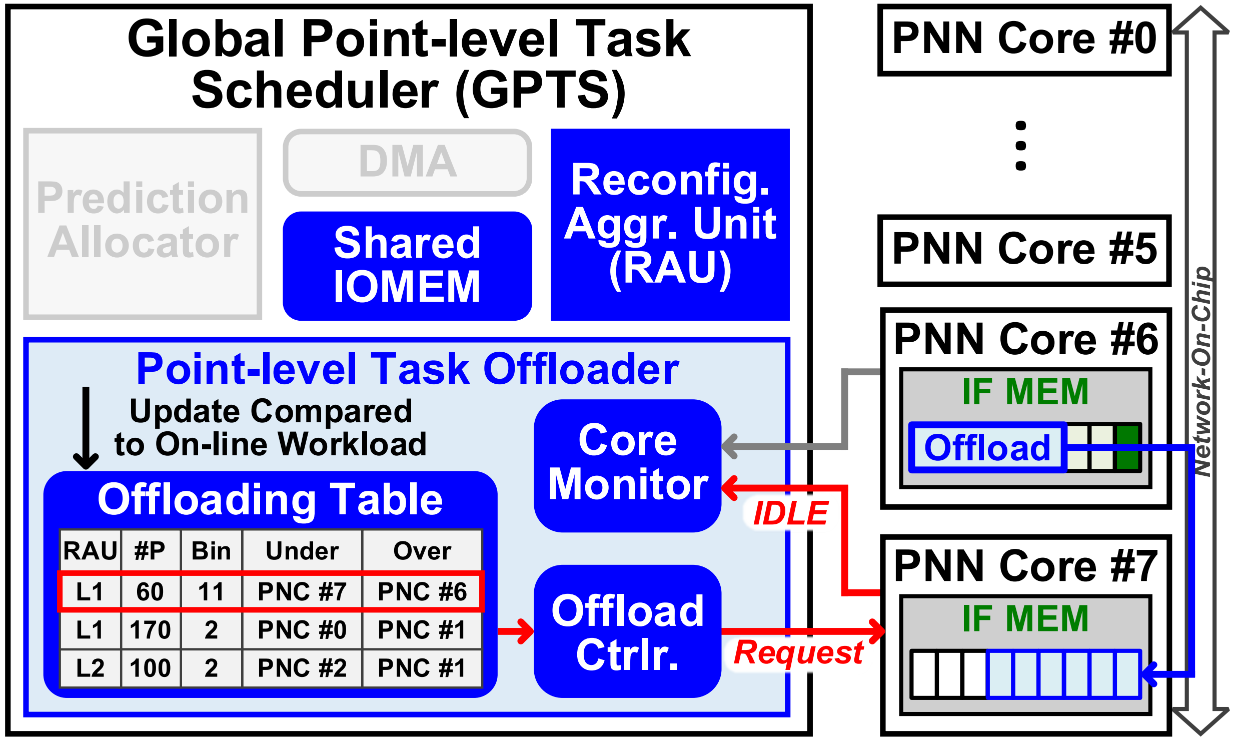
IOMEM	Bin Alloc.	PNC
Bank 0	SB0, SB8	#0
⋮	⋮	⋮
Bank 7	SB5, SB7	#7

GPTS: 2-step Workload Balancing (2/2)

❑ Step 2: Point-level On-line Task Offloading

- Evaluation between predicted and current workload when PSC is finished
- Offloading excessive points to ***IDLE*** cores





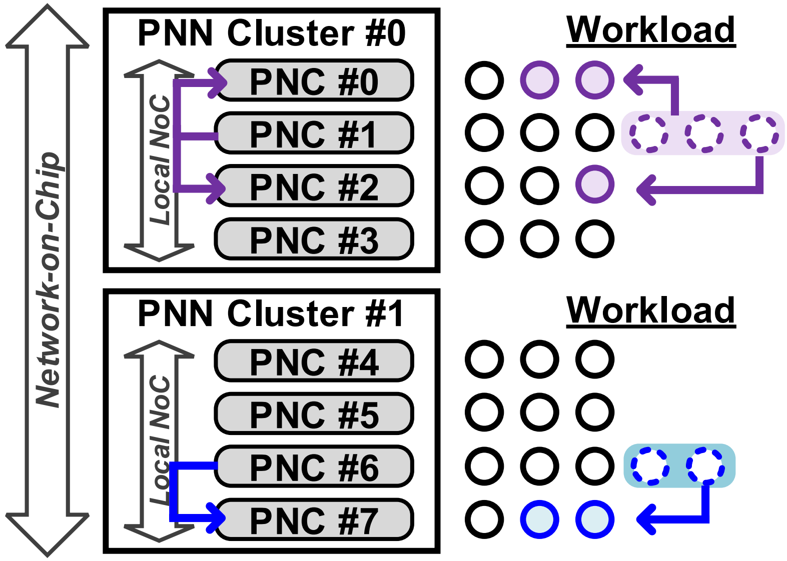
Offloading Table

RAU	#P	Bin	Under	Over
L1	60	11	PNC #7	PNC #6
L1	170	2	PNC #0	PNC #1
L2	100	2	PNC #2	PNC #1

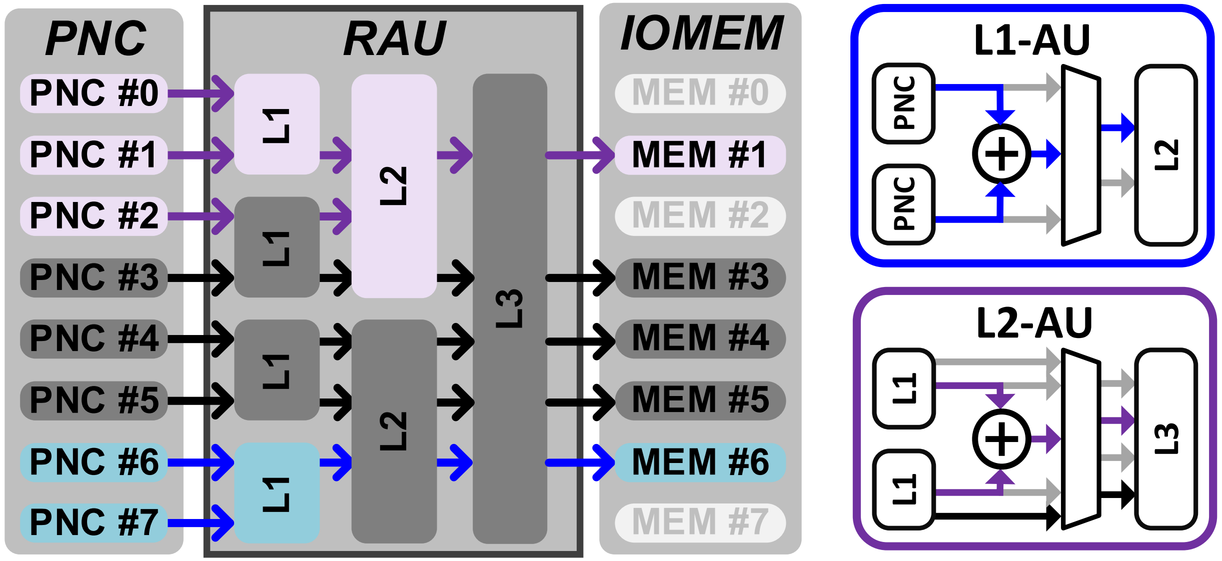
Reconfigurable Aggregation Unit (RAU)

❑ 3-level Workload Allocation & Reconfigurable Aggregation

- Offloading excessive point sequentially according to the level
- Aggregating partial sums of offloaded pair into an IOMEM



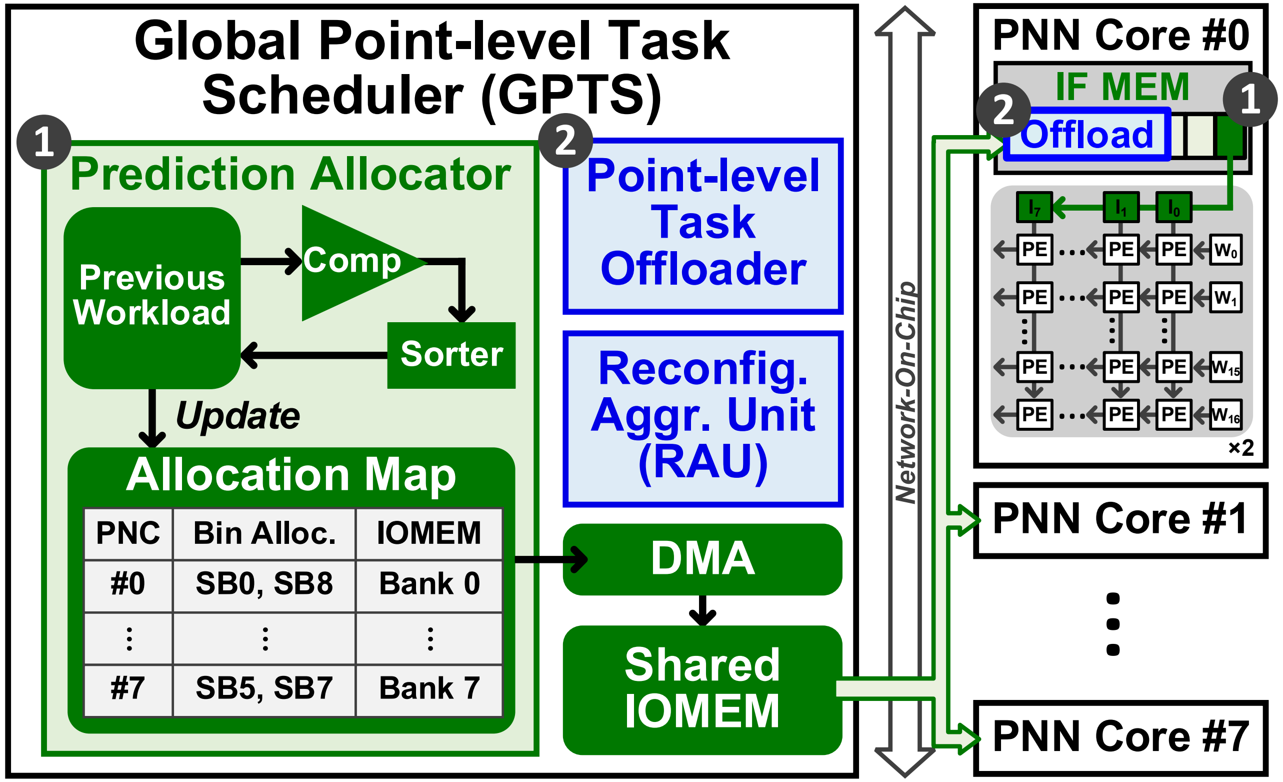
Congestion-aware Task Offloading Priority



Gathering for SB-level Partial SUM

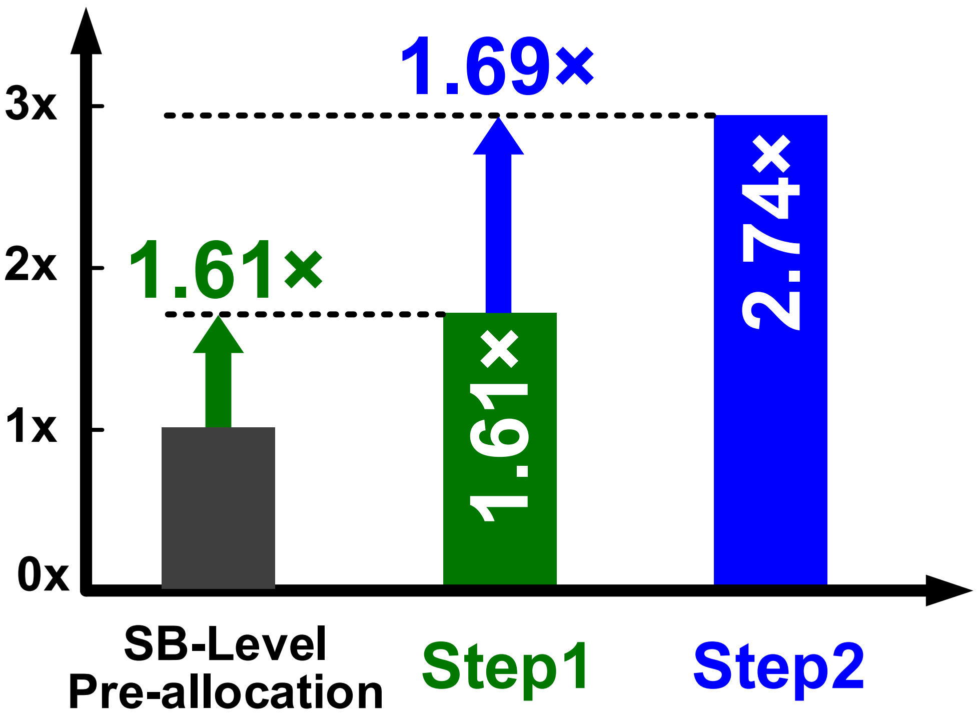
PNN Performance w/ GPTS & RAU

- ❑ Step 1: SB-level Predicted Workload Allocation
- ❑ Step 2: Point-level On-line Task Offloading



PNC	Bin Alloc.	IOMEM
#0	SB0, SB8	Bank 0
⋮	⋮	⋮
#7	SB5, SB7	Bank 7

< Speed-up for PNN >



Challenge of Non-linear Optimization

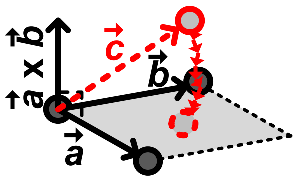
❑ Heterogeneous Characteristics of Iterative Optimization

– KCP: **massive repetition of instruction sequence** for each keypoint

– OTP: **massive matrix operation** of large-sized Jacobian multiplication

Keypoint Matching by PMC

❶ Point-to-Plane Distance



● : *Matched Points*

◌ : *Optimized Point*

$$d = \frac{|\vec{c} \cdot (\vec{a} \times \vec{b})|}{|\vec{a} \times \vec{b}|}$$

❷ Calculating Gradient

$$\frac{\partial d}{\partial T} = \frac{\partial d}{\partial \tilde{P}} \bullet \frac{\partial \tilde{P}}{\partial T} = \left[\frac{\partial d}{\partial rx}\ \frac{\partial d}{\partial ry}\ \frac{\partial d}{\partial rz}\ \frac{\partial d}{\partial tx}\ \frac{\partial d}{\partial ty}\ \frac{\partial d}{\partial tz}\right]$$

-[cos(rz)sin(ry)+sin(rz)sin(rx)cos(ry)(x-tx)
+[-sin(rz)sin(ry)+cos(rz)sin(rx)cos(ry)](y-ty)
-[cos(rx)cos(ry)](z-tz)

Keypoint-level Computation Phase (KCP)

Iterative Optimization

$$J = \begin{bmatrix} \frac{\partial d_1}{\partial rx} & \frac{\partial d_1}{\partial ry} & \frac{\partial d_1}{\partial rz} & \frac{\partial d_1}{\partial tx} & \frac{\partial d_1}{\partial ty} & \frac{\partial d_1}{\partial tz} \\ \frac{\partial d_2}{\partial rx} & \frac{\partial d_2}{\partial ry} & \frac{\partial d_2}{\partial rz} & \frac{\partial d_2}{\partial tx} & \frac{\partial d_2}{\partial ty} & \frac{\partial d_2}{\partial tz} \\ \vdots & & \vdots & & \vdots & \\ \frac{\partial d_N}{\partial rx} & \frac{\partial d_N}{\partial ry} & \frac{\partial d_N}{\partial rz} & \frac{\partial d_N}{\partial tx} & \frac{\partial d_N}{\partial ty} & \frac{\partial d_N}{\partial tz} \end{bmatrix}$$

# of Keypoints (~4K)

❶ LM-Equation

$$T_{K+1}(t) = T_K(t) - J^T J + \lambda diag(J^T J)^{-1} J^T d$$

❷ Non-linear Optimzation

$$\Delta = argmin_T(J^T J\ T_{K+1} + J^T d)$$

❸ QR Decomp.

$$(R^T Q^T)\ QR\ T_{K+1} = R^T Q^T\ d$$

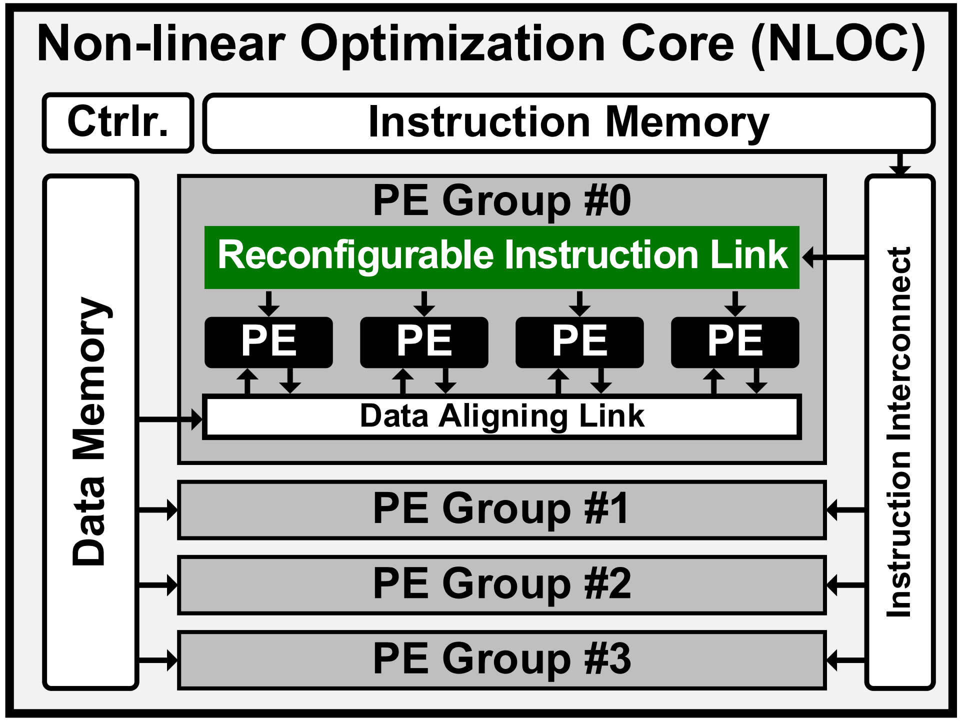
❹ *T* Integration

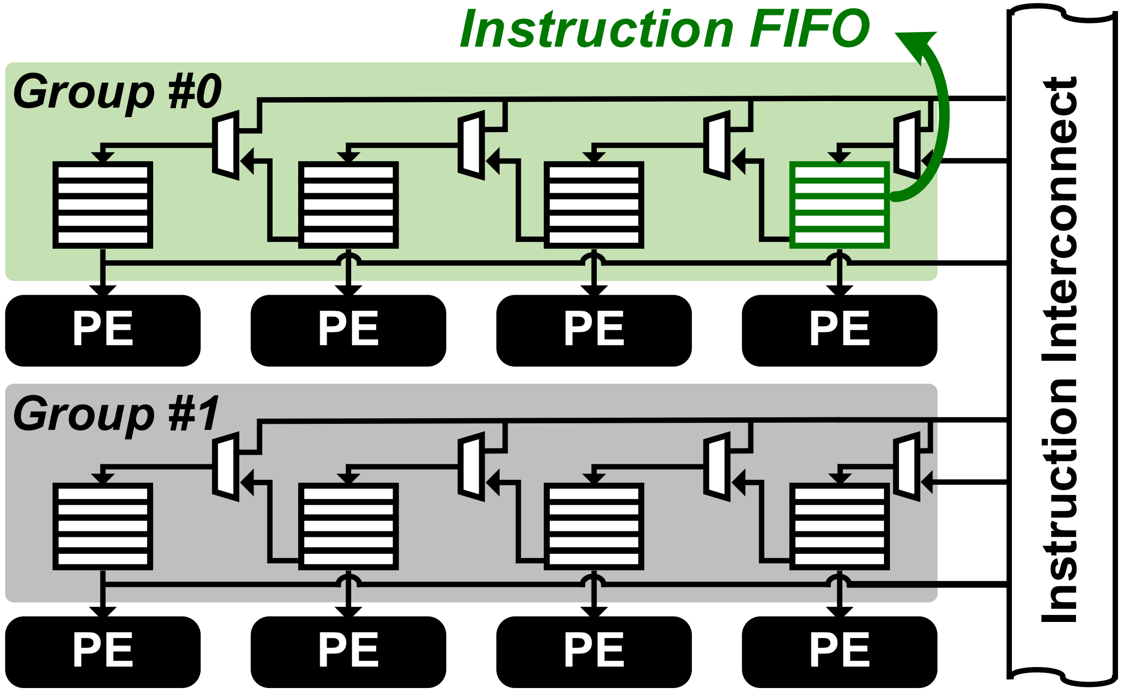
$$T_{(k,\ i)} = \frac{t_{(k,i)} - t_k}{t - t_k} T_k(t)$$

Optimization Phase (OTP)

NLOC: Reconfigurable Mode

- ❑ **Reconfigurable Instruction Link to Support Two PE Modes**
  - – Instruction FIFOs replacing the instruction cache for each PE
  - – Connection to neighboring FIFOs and instruction interconnect



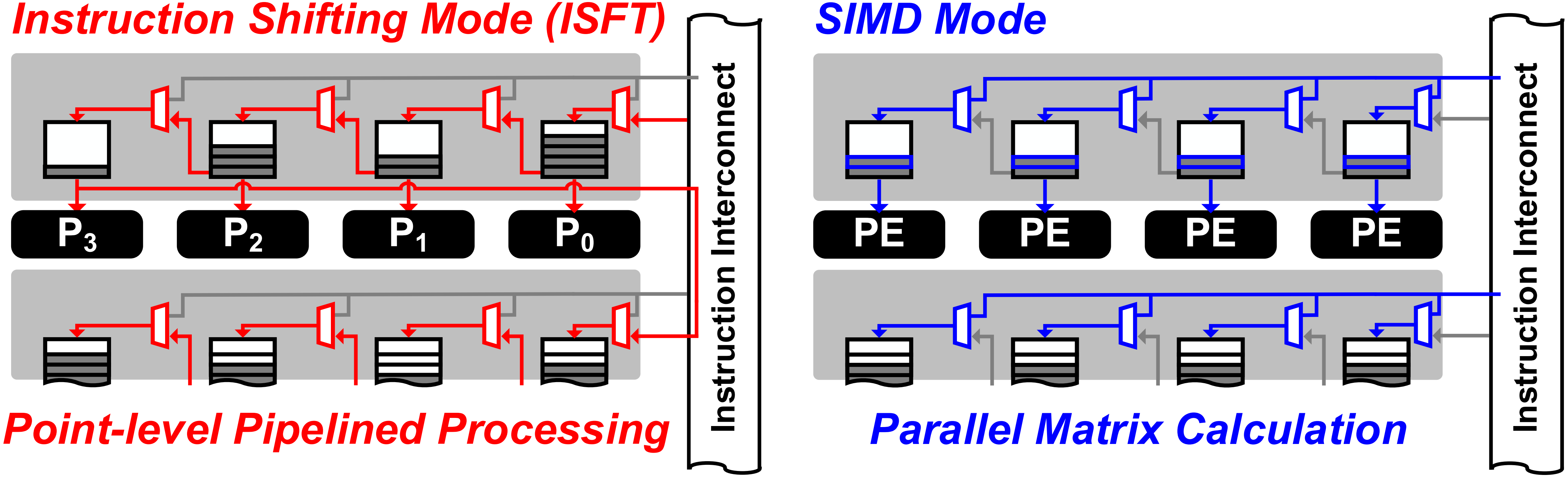


NLOC: Reconfigurable Mode

- ❑ **Reconfigurable Instruction Link to Support Two PE Modes**
  - **Instruction Shifting:** point-level pipelined processing w/ keypoint matching
  - **SIMD**: parallel matrix computation

Instruction Shifting Mode (ISFT)

SIMD Mode



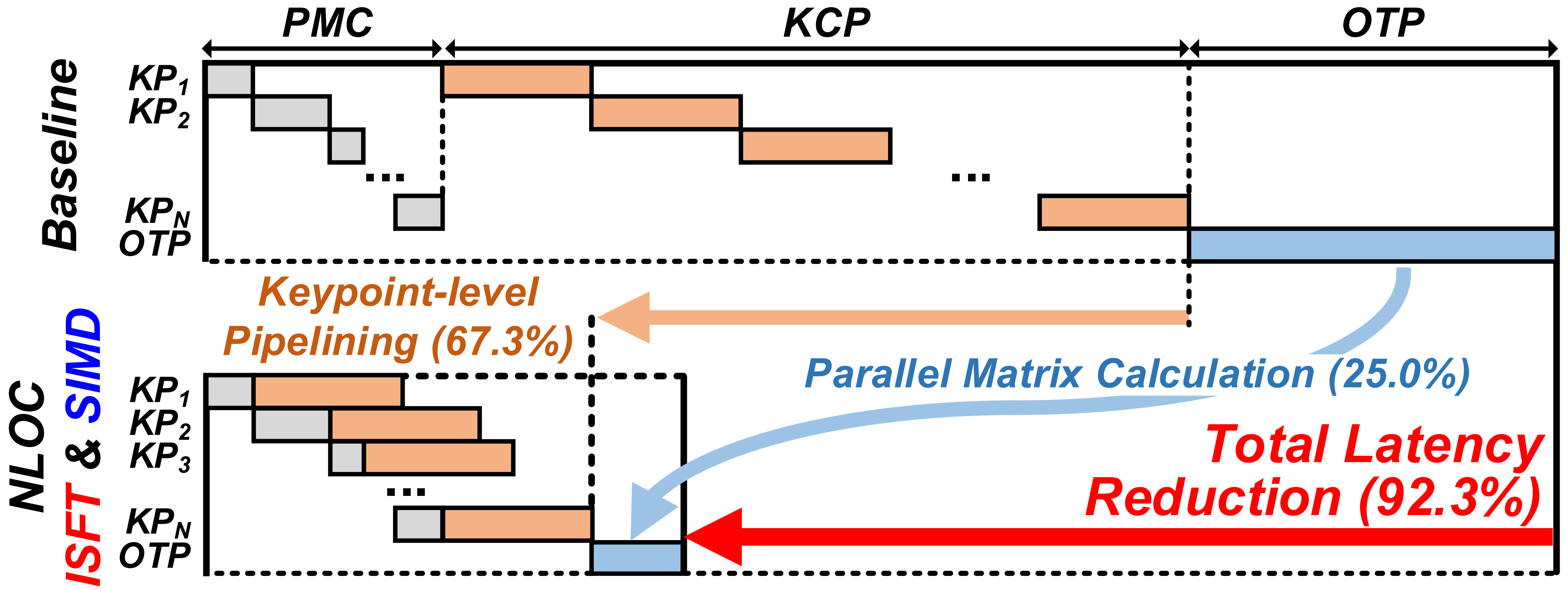
Point-level Pipelined Processing

Parallel Matrix Calculation

➔ **92.3% Processing Time Reduction of NLO**

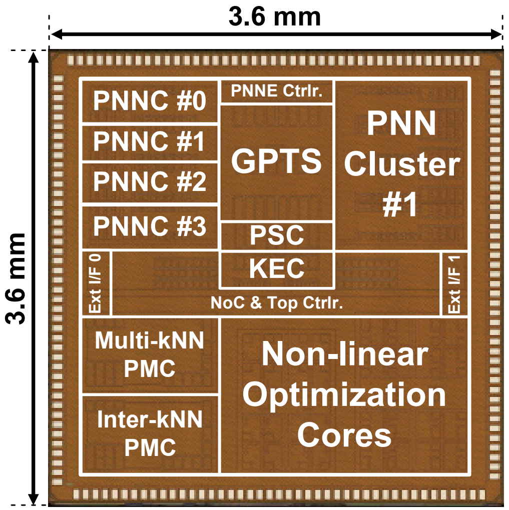
NLO Performance Improvement

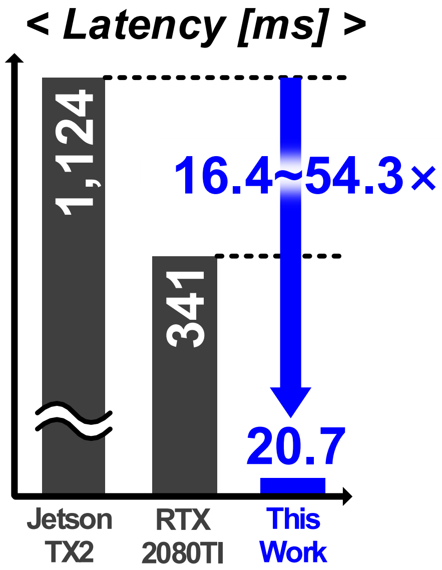
- **Reconfigurable Pipelined & Parallel Computation Mode**
  - Asynchronous and parallelized computation w/ non-uniformly delayed PMC


PMC
KCP
OTP
Baseline
KP1
KP2
…
KPN
OTP
Keypoint-level Pipelining (67.3%)
Parallel Matrix Calculation (25.0%)
NLOC
ISFT & SIMD
KP1
KP2
KP3
…
KPN
OTP
Total Latency Reduction (92.3%)

Chip Photography and Summary

- ❑ **16.4-to-54.3x Faster Latency than Previous System**
- ❑ **17.48mJ/Frame Lowest Energy Consumption**





	ASIC	Process	Power	Latency	mJ/frame
Jetson TX2	X	16 nm	15W	341ms	16,853
RTX 2080Ti	X	12 nm	250W	1124ms	85,324
LSPU	O	28 nm	0.35W	20.7ms	17.48

		Specifications
Technology		28nm CMOS
On-chip SRAM		914 KB
LiDAR Sensor Spec.	Resolution	16 ch / 28,800 Points
	Range	100 m
	Horizontal FoV	360°
	Max. Frequency	50 ms (20 fps*)
Supply Voltage		1.0 V
Max. Frequency		250 MHz
SLAM Latency		20.7 ms
Power Consumption		349.6 mW
Energy per Frame*		17.48 mJ/frame

*Limited to max. operating frequency of conventional LiDAR

Conclusion

- ❑ LSPU is The 1st Implementation of LiDAR SLAM for Real-Time Semantic Mapping on Mobile Robots

- ❑ For Energy-Efficient and Real-Time Semantic LiDAR SLAM
  1) 2D/3D-SB Searching kNN Cores with Dynamic Memory Management
  2) PNN Task Scheduler for 2-step Workload Balancing
  3) Reconfigurable NLO Core for Keypoint-level Pipelining

LSPU: A Mobile Sematic LiDAR SLAM Processor with 17.48mJ/frame and 20.7ms Real-Time Operation

Thank You

❑ Feel Free to Contact Us!

- Email: jueunjung@unist.ac.kr
- LinkedIn: linkedin.com/in/jueunjung
- Lab Homepage: isl-units.ac

- Zoom Meeting: https://unist-ac-kr.zoom.us/j/4403811673?pwd=ZHAC5MGPKc6zMKZ0ajif8swY2apIMc.1 (Password: LSPU)