Crescent: Taming Memory Irregularities for Accelerating Deep Point Cloud Analytics

🔗 https://github.com/horizon-research/crescent

Yu Feng
with Gunnar Hammonds, Yiming Gan and Yuhao Zhu

Department of Computer Science
University of Rochester
http://horizon-lab.org
Autonomous Driving
Computation in Image-based DNNs

Inputs
Computation in Image-based DNNs

Inputs \rightarrow \text{Conv} \rightarrow \text{Weights} \rightarrow \text{Features}
Regular Memory Accesses in Image-based DNNs
Regular Memory Accesses in Image-based DNNs
Regular Memory Accesses in Image-based DNNs
Regular Memory Accesses in Image-based DNNs
Regular Memory Accesses in Image-based DNNs
Regular Memory Accesses in Image-based DNNs

DRAM: Streaming memory accesses

Weights

Inputs

On-chip Buffer
Regular Memory Accesses in Image-based DNNs

DRAM: Streaming memory accesses
SRAM: No bank conflicts with careful data layout
Main Computation in Point Cloud DNN
Main Computation in Point Cloud DNN

Neighbor Search
Main Computation in Point Cloud DNN

Neighbor Search

Feature Computation

P1 Feature Matrix

P8 Feature Matrix

MLP
Canonical KD-Tree Neighbor Search

Requests: R1, R2, R3, R4

KD-Tree
Canonical KD-Tree Neighbor Search

Requests:  R1  R2  R3  R4

KD-Tree
Canonical KD-Tree Neighbor Search

Requests: R1 R2 R3 R4

KD-Tree

- Irregular Memory Access
  - Non-streaming DRAM access
  - Frequent bank conflicts in SRAM

- Redundant Memory Access
  - DRAM access overhead
Fully-Streaming KD-Tree Neighbor Search

Requests:  

```
  9
 / \\  /
1 5
| |  |
3 11 14
| |  |
0 8 2 23
| |  |
30 13 17 6 10 12 16 20
| |  |
```

R1  R2  R3  R4
Fully-Streaming KD-Tree Neighbor Search

Requests: R1 R2 R3 R4

Top Tree

Subtrees

Ref:
[1] QuickNN: Memory and Performance Optimization of k-d Tree Based Nearest Neighbor Search for 3D Point Clouds
[2] Tigris: Architecture and Algorithms for 3D Perception in Point Clouds
Fully-Streaming KD-Tree Neighbor Search

Requests:  R1  R2  R3  R4
Fully-Streaming KD-Tree Neighbor Search

Requests:

Subtree 0

Subtree 1

Subtree 2

Subtree 3
Fully-Streaming KD-Tree Neighbor Search

Requests:

Subtree 0
Subtree 1
Subtree 2
Subtree 3

Streaming DRAM access
Fully-Streaming KD-Tree Neighbor Search

Requests:

- Subtree 0
- Subtree 1
- Subtree 2
- Subtree 3

- Streaming DRAM access
- Exhaustive search introduces redundant computations

Exhaustive search
Bank Conflicts in Feature Computation

Neighbor Index Table

**P1:**
{ P0, P1, P2, P4, P5, P6 }

Feature Matrix

Bank 0

- P0 Feature
- P2 Feature
- P4 Feature
- P6 Feature
- P8 Feature
- P10 Feature
- P12 Feature
- P14 Feature

Bank 1

- P1 Feature
- P3 Feature
- P5 Feature
- P7 Feature
- P9 Feature
- P11 Feature
- P13 Feature
- P15 Feature
Bank Conflicts in Feature Computation

Neighbor Index Table

**P1:**
{ P0, P1, P2, P4, P5, P6 }

Feature Matrix

Bank 0
- P0 Feature
- P2 Feature
- P4 Feature
- P6 Feature
- P8 Feature
- P10 Feature
- P12 Feature
- P14 Feature

Bank 1
- P1 Feature
- P3 Feature
- P5 Feature
- P7 Feature
- P9 Feature
- P11 Feature
- P13 Feature
- P15 Feature
Bank Conflicts in Feature Computation

Neighbor Index Table

P1: {P0, P1, P2, P4, P5, P6}

Feature Matrix

P0 Feature | P1 Feature | P2 Feature | P4 Feature

Bank 0:
- P0 Feature
- P4 Feature
- P6 Feature
- P8 Feature
- P10 Feature
- P12 Feature
- P14 Feature

Bank 1:
- P1 Feature
- P5 Feature
- P7 Feature
- P9 Feature
- P11 Feature
- P13 Feature
- P15 Feature
Bank Conflicts in Feature Computation

Neighbor Index Table

P1: { P0, P1, P2, P4, P5, P6 }

Feature Matrix

P0 Feature  P1 Feature
P2 Feature  P4 Feature

Bank 0
P0 Feature  P2 Feature
P4 Feature  P6 Feature
P8 Feature  P10 Feature
P12 Feature P14 Feature

Bank 1
P1 Feature  P3 Feature
P5 Feature  P7 Feature
P9 Feature  P11 Feature
P13 Feature P15 Feature

Bank Conflict!

▸ Only one can access the data from SRAM.
Bank Conflicts in Feature Computation

Neighbor Index Table

P1:
{ P0, P1, P2, P4, P5, P6 }

Feature Matrix

Bank Conflict!

▸ Only one can access the data from SRAM.
Crescent: HW-Algorithm Co-Design Framework

Algorithm

Hardware
Crescent: HW-Algorithm Co-Design Framework

Algorithm

Work Efficient Neighbor Search

Selective Bank Conflict Elision

Hardware
Crescent: HW-Algorithm Co-Design Framework

Algorithm

Work Efficient Neighbor Search

Selective Bank Conflict Elision

Hardware

Processing Element (PE)

Slice Stack (RS)

Fetch Node (FN)

Calc. Dist. (CD)

Store Result (SR)

Update Stack (US)

From Tree Buffer

From Query Buffer

Bypass

Slack Buffer

Result Buffer

Neighbor Index Table

Feature Matrix

Bank 0

P0 Feature

P1 Feature

P4 Feature

P5 Feature

Bank 1

P2 Feature

P2 Feature

P6 Feature

P7 Feature

Arbitration & Crossbar

Mode

Elide?

Bank 0

Bank 1

Data

Conflict?
Work Efficient KD-Tree Neighbor Search

Requests: R1 R2 R3 R4
Work Efficient KD-Tree Neighbor Search

Requests:

Bucket 0
Bucket 1
Bucket 2
Bucket 3
Work Efficient KD-Tree Neighbor Search

Requests: R1 R2 R3 R4
Work Efficient KD-Tree Neighbor Search

Requests: R1 R2 R3 R4

Subtree 1

DRAM

SRAM
Work Efficient KD-Tree Neighbor Search

Requests: R1, R2, R3, R4

Subtree 2

Requests:
- DRAM
- SRAM
Work Efficient KD-Tree Neighbor Search

Requests: R1 R2 R3 R4

Subtree 3

DRAM

SRAM
Work Efficient KD-Tree Neighbor Search

Requests: R1 R2 R3 R4

Subtree 3

DRAM

SRAM
Work Efficient KD-Tree Neighbor Search

Requests: R1 R2 R3 R4

Subtree 3
Work Efficient KD-Tree Neighbor Search

Requests: R1 R2 R3 R4

Subtree 3

- Reduce redundant computation
Work Efficient KD-Tree Neighbor Search

Requests: R1 R2 R3 R4

- Reduce redundant computation
- Introduce bank conflict in SRAM
Selective Bank Conflict Elision in Neighbor Search

Requests: R1, R2, R3, R4

Subtree 1

DRAM

SRAM

Bank 1

8, 9, 10, 11
12, 13, 14, 15
Selective Bank Conflict Elision in Neighbor Search

Requests: R1 R2 R3 R4

Two search requests access different tree nodes that resides in the same bank.
Selective Bank Conflict Elision in Neighbor Search

Requests: R1, R2, R3, R4

Two search requests access different tree nodes that resides in the same bank

Approach:
- Allow only one access to proceed
Selective Bank Conflict Elision in Neighbor Search

Requests: R1  R2  R3  R4

Approach:
- Allow only one access to proceed
- Return NULL for rest of requests.

Two search requests access different tree nodes that resides in the same bank.
Selective Bank Conflict Elision in Neighbor Search

Requests: R1 R2 R3 R4

Subtree 1

Approach:
- Allow only one access to proceed
- Return NULL for rest of requests.
- Those NULL requests will ignore the nodes beneath the lost node during tree traversal.

Two search requests access different tree nodes that resides in the same bank.
Selective Bank Conflict Elision in Feature Computation

Neighbor Index Table

P1: {P0, P1, P2, P4, P5, P6}

Feature Matrix

Bank 0
- P0 Feature
- P4 Feature
- P8 Feature
- P12 Feature
- P2 Feature
- P6 Feature
- P10 Feature
- P14 Feature

Bank 1
- P1 Feature
- P5 Feature
- P9 Feature
- P13 Feature
- P3 Feature
- P7 Feature
- P11 Feature
- P15 Feature
Selective Bank Conflict Elision in Feature Computation

Neighbor Index Table

**P1:**
{ P0, P1, P2, P4, P5, P6 }

Feature Matrix

- Return replication when having bank conflict.

Bank 0
- P0 Feature
- P2 Feature
- P4 Feature
- P6 Feature
- P8 Feature
- P10 Feature
- P12 Feature
- P14 Feature

Bank 1
- P1 Feature
- P3 Feature
- P5 Feature
- P7 Feature
- P9 Feature
- P11 Feature
- P13 Feature
- P15 Feature
Selective Bank Conflict Elision in Feature Computation

Neighbor Index Table

P1: { P0, P1, P2, P4, P5, P6 }

Feature Matrix

▸ Return replication when having bank conflict.
▸ Resemble replications in point cloud network design.
Selective Bank Conflict Elision in Feature Computation

Neighbor Index Table

P1: { P0, P1, P2, P4, P5, P6 }

Feature Matrix

Bank 0
- P0 Feature
- P4 Feature
- P6 Feature
- P8 Feature
- P10 Feature
- P12 Feature
- P14 Feature

Bank 1
- P1 Feature
- P5 Feature
- P7 Feature
- P9 Feature
- P11 Feature
- P13 Feature
- P15 Feature
Selective Bank Conflict Elision in Feature Computation

Neighbor Index Table

P1: { P0, P1, P2, P4, P5, P6 }

Feature Matrix

P0 Feature  P1 Feature  P2 Feature  P4 Feature

Bank 0

P0 Feature  P2 Feature
P4 Feature  P6 Feature
P8 Feature  P10 Feature
P12 Feature  P14 Feature

Bank 1

P1 Feature  P3 Feature
P5 Feature  P7 Feature
P9 Feature  P11 Feature
P13 Feature  P15 Feature
Selective Bank Conflict Elision in Feature Computation

Neighbor Index Table

P1: { P0, P1, P2, P4, P5, P6 }

Feature Matrix

Bank 0
- P0 Feature
- P4 Feature
- P6 Feature
- P8 Feature
- P10 Feature
- P12 Feature
- P14 Feature

Bank 1
- P1 Feature
- P3 Feature
- P5 Feature
- P7 Feature
- P9 Feature
- P11 Feature
- P13 Feature
- P15 Feature

Bank Conflict!
Selective Bank Conflict Elision in Feature Computation

Neighbor Index Table

P1: { P0, P1, P2, P4, P5, P6 }

Feature Matrix

No Bank Conflict!

Bank 0

P0 Feature  P2 Feature
P4 Feature  P6 Feature
P8 Feature  P10 Feature
P12 Feature P14 Feature

Bank 1

P1 Feature  P3 Feature
P5 Feature  P7 Feature
P9 Feature  P11 Feature
P13 Feature P15 Feature
Selective Bank Conflict Elision in Feature Computation

**Neighbor Index Table**

**P1:** { P0, P1, P2, P4, P5, P6 }

**Feature Matrix**

- **Bank 0**
  - P0 Feature
  - P4 Feature
  - P6 Feature
  - P8 Feature
  - P10 Feature
  - P12 Feature
  - P14 Feature

- **Bank 1**
  - P1 Feature
  - P3 Feature
  - P7 Feature
  - P9 Feature
  - P11 Feature
  - P13 Feature
  - P15 Feature
Baseline Hardware Design

Point Cloud DNN Accelerator

DRAM

Ref: Mesorasi: Architecture Support for Point Cloud Analytics via Delayed-Aggregation
Baseline Hardware Design

Point Cloud DNN Accelerator

- DRAM
  - Input Point Cloud
  - MLP Kernel Weights
  - MLP Intermediate Activations
  - Neighbor Index Matrix

Ref: Mesorasi: Architecture Support for Point Cloud Analytics via Delayed-Aggregation
Baseline Hardware Design

Point Cloud DNN Accelerator

Systolic MAC Unit Array

Global Buffer (Weights / FMaps)

BN/ReLU/MaxPool

MCU

Input Point Cloud
MLP Kernel Weights
MLP Intermediate Activations
Neighbor Index Matrix

Ref: Mesorasi: Architecture Support for Point Cloud Analytics via Delayed-Aggregation
Baseline Hardware Design

Ref: Mesarasi: Architecture Support for Point Cloud Analytics via Delayed-Aggregation
Baseline Hardware Design

Point Cloud DNN Accelerator

- Read Stack
- Fetch Node
- Calc. Dist.
- Query
- Store Result
- Update Stack

Neighbor Search Engine
Neighbor Search Buffer
Peer Index Table
Point Buffer
Aggregation Logic (Mesorasi)

Systolic MAC Unit Array
Global Buffer (Weights / FMaps)

BN/ReLU/MaxPool
MCU

Input Point Cloud
MLP Kernel Weights
MLP Intermediate Activations
Neighbor Index Matrix

DRAM

Ref: Mesorasi: Architecture Support for Point Cloud Analytics via Delayed-Aggregation
Baseline Hardware Design

Point Cloud DNN Accelerator

- Read Stack
- Fetch Node
- Calc. Dist.
- Query
- Store Result
- Update Stack

- Neighbor Search Engine
- Neighbor Search Buffer
- Neighbor Index Table
- Aggregation Logic (Mesorasi)
- Point Buffer
- Global Buffer (Weights /FMaps)
- BN/ReLU/MaxPool
- MCU

- Systolic MAC Unit Array

DRAM
- Input Point Cloud
- MLP Kernel Weights
- MLP Intermediate Activations
- Neighbor Index Matrix

Ref: Mesorasi: Architecture Support for Point Cloud Analytics via Delayed-Aggregation
Hardware Support for Bank Conflict Elision

Port 0

Port 1

Bank Conflict Detection

MUX

Bank 0

Bank 1

Data

Conflict?
Hardware Support for Bank Conflict Elision

- **Conflict?**
- **Mode**
- **Elide?**
- **Bank Conflict Detection**
- **Data**
- **Port 0**
- **Port 1**
- **Bank 0**
- **Bank 1**
- **MUX**
- **AND**
- **Arbitration & Crossbar**
Approximation-Aware Network Training
Approximation-Aware Network Training

Accuracy (%)

- Baseline
- Ours w/o retraining

Accuracy

- 100
- 85
- 70
- 55
- 40

Networks
- PointNet++(c)
- PointNet++(s)
- DenseNet
- F-PointNet

PointNet++(c)
Approximation-Aware Network Training

Accuracy (%)

- Baseline
- Ours w/o retraining

Accuracy

- PointNet++(c)
- PointNet++(s)
- DenseNet
- F-PointNet
Approximation-Aware Network Training

Accuracy (%)

Baseline  Ours w/o retraining

85
70
60
50
40

PointNet++(c), PointNet++(s), DenseNet, F-PointNet

Neighbor Search

Aggregation

MLP

Output Features
Approximation-Aware Network Training

Accuracy (%)

Baseline  Ours w/o retraining

PointNet++(c)  PointNet++(s)  DenseNet  F-PointNet

Input

Neighbor Search

Aggregation

MLP

Output Features

Bank Conflict Model
Approximation-Aware Network Training

Accuracy (%)

- Baseline
- Ours w/o retraining

Input
- Neighbor Search
- Aggregation
- MLP
- Output Features

Do not propagate bank conflict model
Approximation-Aware Network Training

Accuracy (%)

| Method              | Baseline | Ours w/ retraining |
|---------------------|----------|--------------------|
| PointNet++(c)       | 85       | 90                 |
| PointNet++(s)       | 80       | 95                 |
| DenseNet            | 75       | 85                 |
| F-PointNet          | 70       | 80                 |

Do not propagate bank conflict model

Input

Neighbor Search

Aggregation

MLP

Output Features
Experimental Setup
Experimental Setup

Three Point Cloud Applications:
- Object Classification, Object Segmentation, and Object Detection
Experimental Setup

Three Point Cloud Applications:
- Object Classification, Object Segmentation, and Object Detection

Datasets:
- ModelNet40, ShapeNet, and KITTI dataset
Experimental Setup

Three Point Cloud Applications:
⊲ Object Classification, Object Segmentation, and Object Detection

Datasets:
⊲ ModelNet40, ShapeNet, and KITTI dataset

Models:
⊲ Classification: PointNet++ (c), DensePoint
⊲ Segmentation: PointNet++ (s)
⊲ Detection: F-PointNet
Experimental Setup

Three Point Cloud Applications:
- Object Classification, Object Segmentation, and Object Detection

Datasets:
- ModelNet40, ShapeNet, and KITTI dataset

Models:
- Classification: PointNet++ (c), DensePoint
- Segmentation: PointNet++ (s)
- Detection: F-PointNet

Github:
https://github.com/horizon-research/crescent
Hardware Simulation Setup

Hardware Baseline:

- GPU: a mobile Pascal GPU on Nvidia SoC.
- Tigris+GPU: a dedicated neighbor search engine with mobile GPU
- Mesorasi: tigris neighbor search engine with systolic array accelerator.
Hardware Simulation Setup

Hardware Baseline:
› GPU: a mobile Pascal GPU on Nvidia SoC.
› Tigris+GPU: a dedicated neighbor search engine with mobile GPU
› Mesorasi: tigris neighbor search engine with systolic array accelerator.

Variants:
› ANS: approximated neighbor search w/o bank conflict elision.
› ANS+BCE: approximated neighbor search w/ bank conflict elision.
Hardware Simulation Setup

Hardware Baseline:
- GPU: a mobile Pascal GPU on Nvidia SoC.
- Tigris+GPU: a dedicated neighbor search engine with mobile GPU
- Mesorasi: tigris neighbor search engine with systolic array accelerator.

Variants:
- ANS: approximated neighbor search w/o bank conflict elision.
- ANS+BCE: approximated neighbor search w/ bank conflict elision.

Implementation:
- 16x16 Systolic Array.
- Aggregation unit adopted from Mesorasi.
- Neighbor search engine w/ 4 PEs.
- TSMC 16nm FinFET technology.
Speedup

| Method       | Speedup |
|--------------|---------|
| PointNet++ (c) | 2.78   |
| PointNet++ (s) | 3.09   |
| DensePoint   |         |
| F-PointNet   |         |
| AVG.         |         |

Legend:
- ANS
- ANS+BCE
- Mesorasi
- Tigris+GPU
- GPU
Energy Savings

- ANS
- ANS+BCE
- Mesorasi
- Tigris+GPU
- GPU
Energy Savings

The diagram illustrates the energy savings for different models and configurations, measured by normalized energy consumption. The x-axis represents various models and configurations: PointNet++ (C), PointNet++ (S), DensePoint, F-PointNet, and AVG. The y-axis shows the normalized energy with a logarithmic scale ranging from 0.01 to 100.

The models are compared in terms of their energy efficiency, with the following configurations:
- **ANS**
- **ANS+BCE**
- **Mesorasi**
- **Tigris+GPU**
- **GPU**

The diagram visually compares these models across different configurations, highlighting the energy savings for each.
Conclusion

https://github.com/horizon-research/Crescent
Conclusion

Approximative point cloud DNN algorithms tame the memory irregularities

https://github.com/horizon-research/Crescent
Conclusion

Approximative point cloud DNN algorithms tame the memory irregularities

Effective incorporation of memory simulation helps DNN model learning the inexact neighbor search in during the training process.

https://github.com/horizon-research/Crescent