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

Image

Point Cloud
Computation in Image-based DNNs
Computation in Image-based DNNs

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

![Diagram showing memory accesses in DNNs](image-url)
Regular Memory Accesses in Image-based DNNs
Regular Memory Accesses in Image-based DNNs

- DRAM
- Weights
- Inputs
- On-chip Buffer
- PE
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

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

- **DRAM**: Streaming memory accesses
- **SRAM**: No bank conflicts with careful data layout

Diagram:
- DRAM: Weights and Inputs
- On-chip Buffer
- PE (Processing Element) connections
Main Computation in Point Cloud DNN
Main Computation in Point Cloud DNN
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

이는 Canonical KD-Tree Neighbor Search입니다.

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

Redundant Memory Access
- DRAM access overhead

KD-Tree
Fully-Streaming KD-Tree Neighbor Search

Requests:  

\[ \text{Requests: } R1 \quad \text{R2} \quad \text{R3} \quad \text{R4} \]
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
Fully-Streaming KD-Tree Neighbor Search

Requests:  R1  R2  R3  R4

Subtree 0  Subtree 1  Subtree 2  Subtree 3
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

R1

R3

Subtree 2

R2

Subtree 3

R4

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
- 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
Bank Conflicts 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
- P14 Feature

Bank 1
- P1 Feature
- P5 Feature
- P9 Feature
- P13 Feature

- P2 Feature
- P6 Feature
- P10 Feature
- P11 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
  - 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 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 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.
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
Work Efficient KD-Tree Neighbor Search

Requests: R1 R2 R3 R4

Bucket 0
Bucket 1
Bucket 2
Bucket 3
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

DRAM

SRAM
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
Work Efficient KD-Tree Neighbor Search

Requests: R1 R2 R3 R4

Subtree 3

Requests:

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
Work Efficient KD-Tree Neighbor Search

Requests: R1 R2 R3 R4

- Reduce redundant computation

Subtree 3
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

Approach:
- Allow only one access to proceed

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

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

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
- 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 \}

Return replication when having bank conflict.

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
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.

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**

```plaintext
P0 Feature  P1 Feature
```

**Bank 0**

- P0 Feature
- P4 Feature
- P8 Feature
- P12 Feature
- P14 Feature

**Bank 1**

- P1 Feature
- P5 Feature
- P9 Feature
- P13 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
- 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 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

No Bank Conflict!
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
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

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

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

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

Point Cloud DNN Accelerator

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

Point Buffer
- Neighbor Search Engine
- Neighbor Search Buffer
- Aggregation Logic (Mesorasi)

Systolic MAC Unit Array
- Global Buffer (Weights/FMaps)
- BN/ReLU/MaxPool
- MCU

Global Buffer
- (Weights/FMaps)

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

DRAM

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

Conflict?

Port 0

Port 1

Bank Conflict Detection

MUX

Bank 0

Bank 1

Data

Arbitration & Crossbar
Hardware Support for Bank Conflict Elision

Bank Conflict Detection

MUX

AND

Conflict?

Port 0

Port 1

Data

Bank 0

Bank 1

Mode

Elide?

Arbitration & Crossbar
Approximation-Aware Network Training
Approximation-Aware Network Training

Accuracy (%)

Baseline  Ours w/o retraining

40  55  70  85  100

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

![Accuracy (%)](chart.png)

- Baseline
- Ours w/o retraining

Accuracy (%): 100, 85, 70, 55, 40

Networks: PointNet++, DenseNet, F-PointNet
Approximation-Aware Network Training

Accuracy (%)

- Baseline
- Ours w/o retraining

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

Input
Neighbor Search
Aggregation
MLP
Output
Features
Approximation-Aware Network Training

Accuracy (%)

- Baseline
- Ours w/o retraining

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

85 | 85 | 85 | 85
70 | 70 | 70 | 70
55 | 55 | 55 | 55
40 | 40 | 40 | 40

neighbor search
aggregation
MLP
output features
bank conflict model
Approximation-Aware Network Training

Accuracy (%)

- Baseline
- Ours w/o retraining

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

Input

MLP

Neighbor Search

Aggregation

Output Features

Bank Conflict Model

Do not propagate bank conflict model
Approximation-Aware Network Training

Accuracy (%)

<table>
<thead>
<tr>
<th>Network</th>
<th>Baseline</th>
<th>Ours w/ retraining</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointNet++(c)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PointNet++(s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DenseNet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F-PointNet</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Do not propagate bank conflict model

Input

Neighbor Search

Aggregation

MLP

Output Features

Bank Conflict Model
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

- ANS
- ANS+BCE
- Mesorasi
- Tigris+GPU
- GPU

![Speedup Chart](chart.png)
### Speedup

<table>
<thead>
<tr>
<th>Method</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANS</td>
<td>2.78</td>
</tr>
<tr>
<td>ANS+BCE</td>
<td>3.09</td>
</tr>
<tr>
<td>Mesorasi</td>
<td></td>
</tr>
<tr>
<td>Tigris+GPU</td>
<td></td>
</tr>
<tr>
<td>GPU</td>
<td></td>
</tr>
</tbody>
</table>

**Graph:**
- **ANS**
- **ANS+BCE**
- **Mesorasi**
- **Tigris+GPU**
- **GPU**

<table>
<thead>
<tr>
<th>Method</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointNet++ (c)</td>
<td>1.5</td>
</tr>
<tr>
<td>PointNet++ (s)</td>
<td>1.8</td>
</tr>
<tr>
<td>DensePoint</td>
<td>2.7</td>
</tr>
<tr>
<td>F-PointNet</td>
<td>3.0</td>
</tr>
<tr>
<td>AVG.</td>
<td>2.7</td>
</tr>
</tbody>
</table>
Energy Savings

![Graph showing energy savings comparison for different methods: ANS, ANS+BCE, Mesorasi, Tigris+GPU, and GPU. The x-axis represents different methods: PointNet++ (C), PointNet++ (S), DensePoint, F-PointNet, and AVG., while the y-axis represents normalized energy as a percentage from 0.01 to 100.]}
Energy Savings

![Energy Savings Chart](chart.png)
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