VCSR: Mutable CSR Graph Format Using Vertex-Centric Packed Memory Array

Abdullah Al Raqibul Islam, Dong Dai
{aislam6, ddai}@uncc.edu
University of North Carolina at Charlotte
Charlotte, NC, USA

Dazhao Cheng
dcheng@whu.edu.cn
Wuhan University
Wuhan, China
Background: Graph

Graphs are everywhere!

- Social networks
- Economic networks
- Biomedical networks
- Information networks: Web & citations
- Internet
- Networks of neurons

Image Source: https://towardsdatascience.com/graph-theory-132122ac38f2
Background: Dynamic Graph

Graphs are dynamic!

Image Source: https://towardsdatascience.com/animate-dynamic-graphs-with-gephi-d6bd9fa5aec
Background: Graph Processing Systems

- Dynamic graph storage data structures

Background: Graph Processing Systems

● Dynamic graph storage data structures
● Expectations
  ○ Expectation-1: Efficient graph construction
  ○ Expectation-2: Efficient graph analysis
Dynamic In-Memory Graph Storage Data Structures
Common Graph Data-structures

Figure: Three basic graph storage data structures.
Common Graph Data-structures

Figure: Three basic graph storage data structures.
Common Graph Data-structures

Finding all neighbors of a vertex \( v \)

Figure: Three basic graph storage data structures.
Common Graph Data-structures

Figure: Three basic graph storage data structures.
Common Graph Data-structures

- **Storage cost / scanning whole graph**
  - $O(2*e/B)$
  - $O(n/B + e)$
  - $O((n+e)/B)$

- **Finding all neighbors of a vertex v**
  - $O(2*e/B)$
  - $O(deg(v))$
  - $O(deg(v)/B)$

- **Add edge**
  - $O(1)$
  - $O(1)$
  - $O((m + n)/B)$
Common Graph Data-structures

- **O(2*e/B)**
- **O(n/B + e)**
- **O((n+e)/B)**

**Storage cost / scanning whole graph**

- **O(2*e/B)**
- **O(deg(v))**
- **O(deg(v)/B)**

**Finding all neighbors of a vertex v**

- **O(1)**
- **O(1)**
- **O((m + n)/B)**

**Add edge**
Common Graph Data-structures

What is the basic intuition to adopt CSR for the dynamic graphs?

Figure: Three basic graph storage data structures.
CSR Variants for Dynamic Graphs

- Delta Map
  - LLAMA [Macko et. al., ICDE’15]
  - Graphchi [Kyrola et. al., OSDI’12]

- Log-Based
  - Grace [Vijayan et. al., ATC’12]
  - Connectivity Server [Bharat et. al., Elsevier’1998]

- Dynamic Block Linked Lists
  - PowerGraph [Gonzalez et. al., OSDI’12]
  - Graphlab [Low et. al., UAI’10]

- Packed Memory Array (PMA) Based Extensions
  - RMA [De Leo et. al., ICDE’19]
  - PCSR [Wheatman et. al., HPEC’18]
  - GPMA [Sha et. al., VLDB’17]

- Other Common Extensions
  - CSR++ [Firmli et. al., OPODIS’20]
  - Dynamic-CSR [King et. al., Springer’16]
  - …
CSR Variants for Dynamic Graphs

- Delta Map
  - LLAMA [Macko et. al., ICDE’15]
  - Graphchi [Kyrola et. al., OSDI’12]

- Log-Based
  - Grace [Vijayan et. al., ATC’12]
  - Connectivity Server [Bharat et. al., Elsevier’1998]

- Dynamic Block Linked Lists
  - PowerGraph [Gonzalez et. al., OSDI’12]
  - Graphlab [Low et. al., UAI’10]

- Packed Memory Array (PMA) Based Extensions
  - RMA [De Leo et. al., ICDE’19]
  - PCSR [Wheatman et. al., HPEC’18]
  - GPMA [Sha et. al., VLDB’17]

- Other Common Extensions
  - CSR++ [Firmli et. al., OPODIS’20]
  - Dynamic-CSR [King et. al., Springer’16]
  - …
Figure: PMA data structure and one insertion example.
Packed Memory Array (PMA)

- Proposed by Itai et. al. in “A sparse table implementation of priority queues” [ICALP'1981]

Figure: PMA data structure and one insertion example.
Packed Memory Array (PMA)

- Proposed by Itai et. al. in “A sparse table implementation of priority queues” [ICALP'1981]
- PMA is an array data structure, size N

Figure: PMA data structure and one insertion example.
Packed Memory Array (PMA)

- Proposed by Itai et. al. in “A sparse table implementation of priority queues” [ICALP'1981]
- PMA is an array data structure, size $N$
- Insert $N$ items

Figure: PMA data structure and one insertion example.
Packed Memory Array (PMA)

- Proposed by Itai et. al. in “A sparse table implementation of priority queues” [ICALP'1981]
- PMA is an array data structure, size N
- Insert N items
- The array will be left sorted after each insertion

Figure: PMA data structure and one insertion example.
**Packed Memory Array (PMA)**

- Proposed by **Itai et. al.** in “A sparse table implementation of priority queues” [ICALP'1981]
- PMA is an array data structure, size N
- Insert N items
- The array will be left sorted after each insertion
- Asymptotic computational complexity of each insertion: $O(\log^2 n)$

![Array Diagram](image)

Figure: PMA data structure and one insertion example.
Packed Memory Array (PMA)

Figure: PMA data structure and one insertion example.
Packed Memory Array (PMA)

Figure: PMA data structure and one insertion example.
Packed Memory Array (PMA)

Figure: PMA data structure and one insertion example.
Packed Memory Array (PMA)

Level 2: [0.5, 0.8]
Level 1: [0.375, 0.85]
Leaf: [0.25, 0.9]

Figure: PMA data structure and one insertion example.
Packed Memory Array (PMA)

Figure: PMA data structure and one insertion example.
Packed Memory Array (PMA)

Level 2: [0.5, 0.8]
Level 1: [0.375, 0.85]
Leaf: [0.25, 0.9]

Figure: PMA data structure and one insertion example.
Packed Memory Array (PMA)

Level 2: [0.5, 0.8]
Level 1: [0.375, 0.85]
Leaf: [0.25, 0.9]

Figure: PMA data structure and one insertion example.
Packed Memory Array (PMA)

Figure: PMA data structure and one insertion example.
Packed Memory Array (PMA)

Level 2: [0.5, 0.8]

Level 1: [0.375, 0.85]

Leaf: [0.25, 0.9]

Figure: PMA data structure and one insertion example.
Packed Memory Array (PMA)

Level 2: [0.5, 0.8]
Level 1: [0.375, 0.85]
Leaf: [0.25, 0.9]

Figure: PMA data structure and one insertion example.
Packed Memory Array (PMA)

Figure: PMA data structure and one insertion example.
Packed Memory Array (PMA)

Level 2: [0.5, 0.8]
Level 1: [0.375, 0.85]
Leaf: [0.25, 0.9]

Figure: PMA data structure and one insertion example.
Packed Memory Array (PMA)

Figure: PMA data structure and one insertion example.
PMA-Based CSR Extension: PCSR

Figure: PMA-based CSR extension, PCSR.

Graph Skewness

Name: Orkut Graph
Category: Online social network
Number of nodes: 3,072,441
Number of edges: 117,184,899

Image Source: http://konect.cc/networks/orkut-links/
Graph's Mutation Pattern

Figure: The relationship between vertex’ base degree in both the 10% and 30% pre-initialization cases and their expected new edge insertions. x-axis is the degree; y-axis is the expected insertions; both are in log-scale.
Figure: PMA-based CSR extension, PCSR.

PCSRR: Edge Insertion

Figure: PMA-based CSR extension, PCSRR.

Figure: PMA-based CSR extension, PCSR.

PCSR: Edge Insertion

Figure: PMA-based CSR extension, PCSR.

**PCSR: Edge Insertion**

![Diagram of PMA-based CSR extension, PCSR](image)

Figure: PMA-based CSR extension, PCSR.

Packed Compressed Sparse Row (PCSR): Edge Insertion

Figure: PMA-based CSR extension, PCSR.

Figure: PMA-based CSR extension, PCSR.
Figure: PMA-based CSR extension, PCSR.
**PCSR: Edge Insertion**

Figure: PMA-based CSR extension, PCSR.

PCSR: Edge Insertion

Figure: PMA-based CSR extension, PCSR.

Our Contributions

- Design a novel vertex-centric CSR extension, VCSR, to solve the fundamental limitations in handling graph imbalances
  - 1.41x-3.81x better performance in graph insertions
  - 1.22x-2.05x better performance in running typical graph analytic algorithms

- We release our code at github
  
  https://github.com/DIR-LAB/VCSR
Outlines

Novel Mutable CSR Design

Evaluation and Results
VCSR: Design

Figure: PMA-based Vertex-Centric Mutable CSR Extension, VCSR.
VCSR: Design

(a) Example Graph

Figure: PMA-based Vertex-Centric Mutable CSR Extension, VCSR.
VCSR: Design

(a) Example Graph

Figure: PMA-based Vertex-Centric Mutable CSR Extension, VCSR.
VCSR: Design

Figure: PMA-based Vertex-Centric Mutable CSR Extension, VCSR.
VCSR: Design

(a) Example Graph

Fig. PMA-based Vertex-Centric Mutable CSR Extension, VCSR.
VCSR: Design

(a) Example Graph

Figure: PMA-based Vertex-Centric Mutable CSR Extension, VCSR.
VCSR: Design

(a) Example Graph

Figure: PMA-based Vertex-Centric Mutable CSR Extension, VCSR.
Outlines

- Novel Mutable CSR Design
- Evaluation and Results
Evaluation: Platform

- Dell R740 rack server with two sockets
- Each socket installs a 2nd generation Intel Xeon Scalable Processor (Gold 6254 @ 3.10G) with 18 physical (36 virtual) cores
- Ubuntu 18.04, Linux kernel version 4.15.0
- 6 DRAM DIMMS with 32GB each (192GB in total)
Evaluation: System Implementation

- Compressed Sparse Row (CSR)
- Blocked Adjacency-List (BAL)
- PMA-based CSR extension, PCSR
- Proposed vertex-centric PMA-based CSR extension, VCSR
Evaluation: System Implementation

- Compressed Sparse Row (CSR)
- Blocked Adjacency-List (BAL)
- PMA-based CSR extension, PCSR
- Proposed vertex-centric PMA-based CSR extension, VCSR

We used GAPBS for benchmark

https://github.com/sbeamer/gapbs
## Evaluation: Input Graphs

| Datasets       | Domain    | $|V|$     | $|E|$     | $|E|/|V|$ |
|----------------|-----------|----------|----------|---------|
| Amazon         | purchase  | 403393   | 4886816  | 12      |
| Orkut          | social    | 3072626  | 234370166| 76      |
| Live-journal   | social    | 4847570  | 85702474 | 18      |
| Cit-Patents    | citation  | 6009554  | 33037894 | 6       |
| Road           | geo       | 1971280  | 5533214  | 3       |
| as-Skitter     | network   | 1696414  | 22190596 | 13      |
| sx-stackoverflow| temporal | 6024270  | 57724802 | 10      |
| enron          | temporal  | 87273    | 594912   | 7       |
| sx-mathoverflow| temporal | 88580    | 375972   | 4       |
| fb-wall        | temporal  | 63891    | 366824   | 6       |

Table: Graph inputs and their key properties.
# Evaluation: Input Graphs

| Datasets     | Domain   | $|V|$     | $|E|$         | $|E|/|V|$ |
|--------------|----------|---------|--------------|---------|
| Amazon       | purchase | 403393  | 4886816      | 12      |
| Orkut        | social   | 3072626 | 234370166    | 76      |
| Live-journal | social   | 4847570 | 85702474     | 18      |
| Cit-Patents  | citation | 6009554 | 33037894     | 6       |
| Road         | geo      | 1971280 | 5533214      | 3       |
| as-Skitter   | network  | 1696414 | 22190596     | 13      |
| sx-stackoverflow | temporal | 6024270 | 57724802     | 10      |
| enron        | temporal | 87273   | 594912       | 7       |
| sx-mathoverflow | temporal | 88580   | 375972       | 4       |
| fb-wall      | temporal | 63891   | 366824       | 6       |

Table: Graph inputs and their key properties.
# Evaluation: Input Graphs

| Datasets       | Domain   | $|V|$   | $|E|$   | $|E|/|V|$ |
|----------------|----------|--------|--------|---------|
| Amazon         | purchase | 403393 | 4886816| 12      |
| Orkut          | social   | 3072626| 234370166| 76     |
| Live-journal   | social   | 4847570| 85702474| 18      |
| Cit-Patents    | citation | 6009554| 33037894| 6       |
| Road           | geo      | 1971280| 5533214| 3       |
| as-Skitter     | network  | 1696414| 22190596| 13      |
| sx-stackoverflow| temporal | 6024270| 57724802| 10      |
| enron          | temporal | 87273  | 594912 | 7       |
| sx-mathoverflow| temporal | 88580  | 375972 | 4       |
| fb-wall        | temporal | 63891  | 366824 | 6       |

Table: Graph inputs and their key properties.
# Evaluation: Input Graphs

| Datasets       | Domain         | $|V|$   | $|E|$  | $|E|/|V|$ |
|----------------|----------------|-------|-------|---------|
| Amazon         | purchase       | 403393| 4886816 | 12      |
| Orkut          | social         | 3072626| 234370166 | 76      |
| Live-journal   | social         | 4847570| 85702474 | 18      |
| Cit-Patents    | citation       | 6009554| 33037894 | 6       |
| Road           | geo            | 1971280| 5533214 | 3       |
| as-Skitter     | network        | 1696414| 22190596 | 13      |
| sx-stackoverflow| temporal      | 6024270| 57724802 | 10      |
| enron          | temporal       | 87273 | 594912 | 7       |
| sx-mathoverflow| temporal       | 88580 | 375972 | 4       |
| fb-wall        | temporal       | 63891 | 366824 | 6       |

Table: Graph inputs and their key properties.
Evaluation: Input Graphs

| Datasets         | Domain      | $|V|$  | $|E|$      | $|E|/|V|$ |
|------------------|-------------|------|-----------|---------|
| Amazon           | purchase    | 403393 | 4886816   | 12      |
| Orkut            | social      | 3072626 | 234370166 | 76      |
| Live-journal     | social      | 4847570 | 85702474  | 18      |
| Cit-Patents      | citation    | 6009554 | 33037894  | 6       |
| Road             | geo         | 1971280 | 5533214   | 3       |
| as-Skitter       | network     | 1696414 | 22190596  | 13      |
| sx-stackoverflow | temporal    | 6024270 | 57724802  | 10      |
| enron            | temporal    | 87273  | 594912    | 7       |
| sx-mathoverflow  | temporal    | 88580  | 375972    | 4       |
| fb-wall          | temporal    | 63891  | 366824    | 6       |

Table: Graph inputs and their key properties.
**Evaluation: Graph Algorithm Kernels**

<table>
<thead>
<tr>
<th>Graph kernel</th>
<th>Kernel Type</th>
<th>Input</th>
<th>Output</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PageRank (PR)</td>
<td>Link Analysis</td>
<td>-</td>
<td>$</td>
<td>V</td>
</tr>
<tr>
<td>Breadth-First Search (BFS)</td>
<td>Graph Traversal</td>
<td>Source vertex</td>
<td>$</td>
<td>V</td>
</tr>
<tr>
<td>Single-Source Shortest Paths (SSSP)</td>
<td>Shortest Path</td>
<td>Source vertex</td>
<td>$</td>
<td>V</td>
</tr>
<tr>
<td>Connected Components (CC)</td>
<td>Connectivity</td>
<td>-</td>
<td>$</td>
<td>V</td>
</tr>
</tbody>
</table>

Table: A list of graph kernels and inputs and outputs used to evaluate graph data-structures.
Evaluation: Graph Insertion Performance

Figure: Comparing VCSR’s dynamic graph insertion performance (in seconds) for 10% pre-initialization.
Evaluation: Graph Insertion Performance

Figure: Comparing VCSR’s dynamic graph insertion performance (in seconds) for 10% pre-initialization.
Evaluation: Graph Insertion Performance

Figure: Comparing VCSR’s dynamic graph insertion performance (in seconds) for 10% pre-initialization.
Evaluation: Graph Insertion Performance

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Tree Levels</th>
<th>VCSR (Mem. Acc.)</th>
<th>PCSR (Mem. Acc.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon (shuffled)</td>
<td>[1-3]</td>
<td>710 (0.84 M)</td>
<td>251077 (35.50 M)</td>
</tr>
<tr>
<td></td>
<td>[3-7]</td>
<td>4 (0.02 M)</td>
<td>12964 (8.49 M)</td>
</tr>
<tr>
<td></td>
<td>≥ 7</td>
<td>0 (0.00 M)</td>
<td>45 (0.41 M)</td>
</tr>
<tr>
<td>sx-Stack overflow</td>
<td>[1-3]</td>
<td>904363 (457.23 M)</td>
<td>13876752 (2262.72 M)</td>
</tr>
<tr>
<td></td>
<td>[3-7]</td>
<td>357575 (1056.19 M)</td>
<td>3053307 (2720.56 M)</td>
</tr>
<tr>
<td></td>
<td>[7-15]</td>
<td>26530 (1594.56 M)</td>
<td>70307 (1704.70 M)</td>
</tr>
<tr>
<td></td>
<td>≥ 15</td>
<td>21 (438.37 M)</td>
<td>288 (2531.68 M)</td>
</tr>
</tbody>
</table>

Table: Number of re-balancing operations triggered by edge insertions in VCSR and PCSR on two graphs.
# Evaluation: Graph Insertion Performance

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Tree Levels</th>
<th>VCSR (Mem. Acc.)</th>
<th>PCSR (Mem. Acc.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon (shuffled)</td>
<td>[1-3]</td>
<td>710 (0.84 M)</td>
<td>251077 (35.50 M)</td>
</tr>
<tr>
<td></td>
<td>[3-7]</td>
<td>4 (0.02 M)</td>
<td>12964 (8.49 M)</td>
</tr>
<tr>
<td></td>
<td>≥ 7</td>
<td>0 (0.00 M)</td>
<td>45 (0.41 M)</td>
</tr>
<tr>
<td>sx-Stack overflow</td>
<td>[1-3]</td>
<td>904363 (457.23 M)</td>
<td>13876752 (2262.72 M)</td>
</tr>
<tr>
<td></td>
<td>[3-7]</td>
<td>357575 (1056.19 M)</td>
<td>3053307 (2720.56 M)</td>
</tr>
<tr>
<td></td>
<td>[7-15]</td>
<td>26530 (1594.56 M)</td>
<td>70307 (1704.70 M)</td>
</tr>
<tr>
<td></td>
<td>≥ 15</td>
<td>21 (438.37 M)</td>
<td>288 (2531.68 M)</td>
</tr>
</tbody>
</table>

Table: Number of re-balancing operations triggered by edge insertions in VCSR and PCSR on two graphs.
## Evaluation: Graph Insertion Performance

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Tree Levels</th>
<th>VCSR (Mem. Acc.)</th>
<th>PCSR (Mem. Acc.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon (shuffled)</td>
<td>1-3</td>
<td>710 (0.84 M)</td>
<td>251077 (35.50 M)</td>
</tr>
<tr>
<td></td>
<td>3-7</td>
<td>4 (0.02 M)</td>
<td>12964 (8.49 M)</td>
</tr>
<tr>
<td></td>
<td>≥ 7</td>
<td>0 (0.00 M)</td>
<td>45 (0.41 M)</td>
</tr>
<tr>
<td>sx-Stack overflow</td>
<td>1-3</td>
<td>904363 (457.23 M)</td>
<td>13876752 (2262.72 M)</td>
</tr>
<tr>
<td></td>
<td>3-7</td>
<td>357575 (1056.19 M)</td>
<td>3053307 (2720.56 M)</td>
</tr>
<tr>
<td></td>
<td>7-15</td>
<td>26530 (1594.56 M)</td>
<td>70307 (1704.70 M)</td>
</tr>
<tr>
<td></td>
<td>≥ 15</td>
<td>21 (438.37 M)</td>
<td>288 (2531.68 M)</td>
</tr>
</tbody>
</table>

Table: Number of re-balancing operations triggered by edge insertions in VCSR and PCSR on two graphs.
# Evaluation: Graph Insertion Performance

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Tree Levels</th>
<th>VCSR (Mem. Acc.)</th>
<th>PCSR (Mem. Acc.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon (shuffled)</td>
<td>1-3</td>
<td>710 0.84 M</td>
<td>251077 35.50 M</td>
</tr>
<tr>
<td></td>
<td>3-7</td>
<td>4 (0.02 M)</td>
<td>12964 (8.49 M)</td>
</tr>
<tr>
<td></td>
<td>≥ 7</td>
<td>0 (0.00 M)</td>
<td>45 (0.41 M)</td>
</tr>
<tr>
<td>sx-Stack overflow</td>
<td>1-3</td>
<td>904363 (457.23 M)</td>
<td>13876752 (2262.72 M)</td>
</tr>
<tr>
<td></td>
<td>3-7</td>
<td>357575 (1056.19 M)</td>
<td>3053307 (2720.56 M)</td>
</tr>
<tr>
<td></td>
<td>7-15</td>
<td>26530 (1594.56 M)</td>
<td>70307 (1704.70 M)</td>
</tr>
<tr>
<td></td>
<td>≥ 15</td>
<td>21 (438.37 M)</td>
<td>288 (2531.68 M)</td>
</tr>
</tbody>
</table>

Table: Number of re-balancing operations triggered by edge insertions in VCSR and PCSR on two graphs.
# Evaluation: Graph Insertion Performance

## Table: Number of re-balancing operations triggered by edge insertions in VCSR and PCSR on two graphs.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Tree Levels</th>
<th>VCSR (Mem. Acc.)</th>
<th>PCSR (Mem. Acc.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon (shuffled)</td>
<td>[1-3]</td>
<td>710 (0.84 M)</td>
<td>251077 (35.50 M)</td>
</tr>
<tr>
<td></td>
<td>[3-7]</td>
<td>4 (0.02 M)</td>
<td>12964 (8.49 M)</td>
</tr>
<tr>
<td></td>
<td>$\geq 7$</td>
<td>0 (0.00 M)</td>
<td>45 (0.41 M)</td>
</tr>
<tr>
<td>sx-Stack overflow</td>
<td>[1-3]</td>
<td>904363 (457.23 M)</td>
<td>13876752 (2262.72 M)</td>
</tr>
<tr>
<td></td>
<td>[3-7]</td>
<td>357575 (1056.19 M)</td>
<td>3053307 (2720.56 M)</td>
</tr>
<tr>
<td></td>
<td>[7-15]</td>
<td>26530 (1594.56 M)</td>
<td>70307 (1704.70 M)</td>
</tr>
<tr>
<td></td>
<td>$\geq 15$</td>
<td>21 (438.37 M)</td>
<td>288 (2531.68 M)</td>
</tr>
</tbody>
</table>

Table: Number of re-balancing operations triggered by edge insertions in VCSR and PCSR on two graphs.
Evaluation: Graph Insertion Performance

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Tree Levels</th>
<th>VCSR (Mem. Acc.)</th>
<th>PCSR (Mem. Acc.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon (shuffled)</td>
<td>[1-3]</td>
<td>710 0.84 M</td>
<td>251077 35.50 M</td>
</tr>
<tr>
<td></td>
<td>[3-7]</td>
<td>4 0.02 M</td>
<td>12964 8.49 M</td>
</tr>
<tr>
<td></td>
<td>≥ 7</td>
<td>0 0.00 M</td>
<td>45 0.41 M</td>
</tr>
<tr>
<td>sx-Stack overflow</td>
<td>[1-3]</td>
<td>904363 457.23 M</td>
<td>13876752 2262.72 M</td>
</tr>
<tr>
<td></td>
<td>[3-7]</td>
<td>357575 1056.19 M</td>
<td>3053307 2720.56 M</td>
</tr>
<tr>
<td></td>
<td>[7-15]</td>
<td>26530 1594.56 M</td>
<td>70307 1704.70 M</td>
</tr>
<tr>
<td></td>
<td>≥ 15</td>
<td>21 438.37 M</td>
<td>288 2531.68 M</td>
</tr>
</tbody>
</table>

Table: Number of re-balancing operations triggered by edge insertions in VCSR and PCSR on two graphs.
Evaluation: Graph Insertion Performance

Figure: Comparing VCSR’s dynamic graph insertion performance (in seconds) for 10% pre-initialization.

Expectation-1: Efficient graph construction

(a) Random Workload  (b) Temporal Graphs
Evaluation: Graph Analytic Algorithms Performance

Figure: Comparing graph analysis runtime normalized to CSR.
Evaluation: Graph Analytic Algorithms Performance

Figure: Comparing graph analysis runtime normalized to CSR.
Evaluation: Graph Analytic Algorithms Performance

Figure: Comparing graph analysis runtime normalized to CSR.
Evaluation: Graph Analytic Algorithms Performance

Figure: Comparing graph analysis runtime normalized to CSR.
Evaluation: Graph Analytic Algorithms Performance

Expectation-2: Efficient graph analysis

Figure: Comparing graph analysis runtime normalized to CSR.
Summary

- Identify fundamental limitations in existing PMA based mutable CSR extension
- Demonstrate graph’s power-law also exist while it evolves
- Design a novel vertex-centric CSR extension: VCSR
  - Solves the fundamental limitations in handling graph imbalances
- Evaluation results
  - 1.41x-3.81x better performance in graph insertions
  - 1.22x-2.05x better performance in running typical graph analytic algorithms
Thank You

Question?