Differential Dataflow

McSherry, Frank D., Murray, Derek G., Isaacs, Rebecca, Isard, Michael
Outline

- Motivation for Differential Dataflow
- Key Concepts
- Differential Dataflow in practice
- Discussion
Motivation
Traditional data parallel processing

- Take input data in batches.
- Process and output.
- Highly evolved - Hadoop, Spark.
- Mostly stateless.
Interactive - Twitter Mention Graph

- Used to find *trending* #hashtags.
- Billions of vertices and edges.
- Millions of updates per second (storm).
- Needs low latency of streaming and throughput of spark.
- Similar issue with interactive analytics
Loop Processing

- Some algorithms require iterations
  - Pagerank
  - Connected components

- Usually requires transferring entire state between iterations

- Spark, Hadoop etc execution times ~ stateless
Incremental Dataflow

- Stateful.
- Get the differences of collections.
- Only calculate changes.
- Example
  - Loops
  - New Data
  - But NOT both!!
Concepts
Total vs Partial Ordering

- Traditional dataflow systems expect total ordering
  - Multiple variables are a problem
- A partial ordering uses a time vector for ordering
  - Deals well with multiple variables
- Partial because ordering by variable x gives only a partial ordering
Total vs Partial Ordering

- Traditional dataflow systems expect total ordering
  - Multiple variables are a problem
- A partial ordering uses a time vector for ordering
  - Deals well with multiple variables
- Partial because ordering by variable x gives only a partial ordering for x
Total vs Partial Ordering

- Traditional dataflow systems expect total ordering
  - Multiple variables are a problem
- A partial ordering uses a time vector for ordering
  - Deals well with multiple variables
- Partial because ordering by variable x gives only a partial ordering
Differential Dataflow

- **Computational Model**
  - Defines how to process partially ordered data.
  - Defines state between iterations

- **Goals**
  - Do less calculation per change
  - Converge quicker per iteration
Timely Dataflow

- Performs Iterative Calculations
- Computational model with directed graph
- Vertices exchange messages
- Logical Timestamps for messages

\[ (e \in \mathbb{N}, (c_1, \ldots, c_k) \in \mathbb{N}^k) \]
Timely Dataflow

- Loops denoted by,
  - Ingress - adds a counter
  - Feedback - increments a counter
  - Egress - removes a counter

- Pointstamps - events at location and time

\[
(t \in \text{Timestamp}, \ell \in \text{Edge} \cup \text{Vertex})
\]
Differential Dataflow in practise
The Connected Graph Problem
The Connected Graph Problem
Connected Graph with Relational Algebra

Labels

Edges

Min

O
Connected Graph with Relational Algebra
Connected Graph with Relational Algebra

Neighbor Labels:
- 3
- 1
- 3
- 4
- 4
- 4
- 2
- 4
- 2
- 5
- 5
- 1
- 2
- 2
- 3
- 3
- 4
- 4
- 5
- 5

Self Labels:
- 5
- 2
- 2
- 1
- 1
- 1
- 2
- 2
- 3
- 3
- 4
- 4

Min

U

O

Labels

Edges
Connected Graph with Relational Algebra

Result after 1st Iteration

1 1
3 1
4 2
2 2
5 2
Connected Graph in Timely

- Edges are available constantly
- Add counter at Ingress
- Remove Counter at egress
- Increment counter at feedback
- Map converts joined tuples into node/label tuples
- Concat performs the union
Maintaining State in Differential Dataflow

\[\delta B_t = \text{Op}\left(\sum_{s \leq t} \delta A_s\right) \rightarrow \sum_{s \leq t} \delta B_s\]

- Change in state at node \(b\) at \(t\)
- Cumulative state at \(b\) upto \(t\)
- Sum of all states at \(b\) before \(t\)
Connected Graph
Connected Graph in Differential

- **Labels**: Ingress, Concat, Join, Map, GroupBy + Min, Feedback
- **Edges**: Concat, Join, Map, GroupBy + Min, Feedback
- **Egress**:

```
\begin{align*}
\text{t= (0)} &: 1 & 3 \\
&: 3 & 1 \\
&: 4 & 3 \\
&: 4 & 2 \\
&: 5 & 2 \\
\end{align*}
```

```
\begin{align*}
\text{t= (0)} &: 1 & 1 \\
&: 2 & 2 \\
&: 3 & 3 \\
&: 4 & 4 \\
&: 5 & 5 \\
\end{align*}
```
Connected Graph in Differential

1.3.3.4.3.2.4.2.5.5.2.4.3.1

\( t = (0) \)

\( \text{Concat} \) \( \text{Join} \) \( \text{Map} \) \( \text{Concat} \) \( \text{GroupBy} + \text{Min} \) \( \text{Egress} \)
Connected Graph in Differential

$E = (0, 0)$

![Diagram of connected graph in differential]

- **Labels**
  - Ingress
  - Concat
  - Feedback
  - GroupBy + Min
  - Egress

- **Edges**
  - Join
  - Map
  - Concat

$t = (0, 0)$

Nodes:
- 1
- 2
- 3
- 4
- 5
Connected Graph in Differential
Connected Graph in Differential

Graph Diagram:
- **Labels**
  - Ingress
- **Edges**
  - Concat
  - Join
  - Map
  - Concat
  - GroupBy + Min
  - Egress

Feedback connections:
- Feedback from **Labels** to **Ingress**
- Feedback from **Edges** to **Map**
- Feedback from **GroupBy + Min** to **Egress**

Timepoint: $t = (0, 0)$

Node Connections:
- Node 3 connected to Node 1
- Node 1 connected to Node 3
- Node 3 connected to Node 4
- Node 4 connected to Node 3
- Node 2 connected to Node 4
- Node 4 connected to Node 2
- Node 2 connected to Node 5
- Node 5 connected to Node 2
- Node 2 connected to Node 5

Note: The diagram includes a rectangular box with numbers indicating the connections between different nodes.
Connected Graph in Differential

t = (0, 0)

![Diagram of connected graph with labels and edges, showing operations like Ingress, Concat, Join, Map, GroupBy + Min, and Egress.](image-url)
Connected Graph in Differential

t= (0, 1)

1 1
3 1
4 2
2 2
5 2
Connected Graph in Differential

$t = (0, 1)$

Labels
- Concat
- Join
- Map
- GroupBy + Min
- Egress

Feedback
- Concat

Edges
- Ingress

Ingress

Labels

Edges

Feedback

Connected Graph in Differential

$t = (0, 1)$

1 1
2 2
3 3
4 4
5 5

1 1
2 2
3 3
4 4
5 5

1 1
2 2
3 3
4 4
5 5

1 1
2 2
3 3
4 4
5 5

1 1
2 2
3 3
4 4
5 5

1 1
2 2
3 3
4 4
5 5
Connected Graph in Differential

t = (0, 1)

1 1
3 1
4 2
2 2
5 2

1 1
2 2
3 3
4 4
5 5

Labels

Edges

Ingress

Join

Map

Concat

GroupBy

+Min

Feedback

Egress
Connected Graph in Differential

\[ t = (0, 1) \]
Connected Graph in Differential

Labels

Ingress

Concat

Join

Map

Concat

GroupBy + Min

Egress

Feedback

3 1 3
3 1 1
3 4 3
4 3 4
4 3 2
4 2 4
4 2 2
5 2 5
5 2 2
Connected Graph in Differential

- Connected Graph
- Concat
- Join
- Concat
- GroupBy + Min
- Feedback
- Ingress
- Labels
- Edges
- Map
- Concat
- GroupBy + Min
- Egress

$t = (0, 1)$

Values:

- 1
- 3
- 4
- 3
- 1
- 4
- 1
- 3
- 4
- 3
- 2
- 4
- 2
- 2
- 5
Connected Graph in Differential

Labels

Ingress

Concat

Join

Map

Feedback

GroupBy + Min

Edges

Egress

t = (0, 1)

1 3
4 3
3 4
2 4
2 5
4 1
3 2
3 3
3 3
4 4
4 4
Connected Graph in Differential

Cumulative Input from concat
Groupby + Min

\[ \sum_{s \leq t} \delta A_s \]

\[ \text{Op} \left( \sum_{s \leq t} \delta A_s \right) \]

Labels
Edges

Ingress
Concat
Join
Map
Concat
GroupBy + Min
Egress

Feedback

\[ \text{Op} \left( \sum_{s \leq t} \delta A_s \right) - \sum_{s \leq t} \delta B_s \]

\[ t = (0, 1) \]
Connected Graph in Differential

t = (0, 2)
Connected Graph in Differential

- **Ingress**
- **Concat**
- **Join**
- **Map**
- **Concat**
- **GroupBy +Min**
- **Egress**

Feedback:

$t = (0, 2)$
Connected Graph in Differential

t = (0, 3)
Connected Graph in Differential

t = (0, 3)
Connected Graph in Differential

- Concat
- Join
- Map
- Concat
- GroupBy +Min
- Egress

Feedback

$t= (0, 3)$

Nodes: 5 2 5 1
Connected Graph in Differential

t = (0, 4)

Labels

Ingress

Concat

Join

Map

Concat

GroupBy +Min

Egress

Feedback
Connected Graph in Differential

t = (0, 4)
Connected Graph in Differential

t = (0, 4)

Labels

Edges

Ingress

Concat

Join

Map

Concat

GroupBy +Min

Egress

Feedback
Connected Graph in Differential

- **Labels**
  - **Ingress**
  - **Concat**
  - **Join**
    - **Map**
      - **Concat**
        - **GroupBy +Min**
          - **Egress**

- **Feedback**

- **Edges**

- **t = (0, 4)**

- ?
Connected Graph in Differential

t = (0, 4)

Does not increment

t = (0)
Changes to Connected Graph - I

Remove Undirected Edge
Changes to Connected Graph - I

Labels

- Ingress
- Concat
- Join
- Map
- Concat
- GroupBy +Min
- Feedback

Edges

- Egress

$t= (1)$

4 2

2 4
Changes to Connected Graph - I

- Concat
- Join
- Concat
- GroupBy +Min
- Feedback
- Ingress
- Labels
- Edges
- Map
- Egress

$t= (1, 0)$

$t= (1)$
Changes to Connected Graph - I

- Concat
- Join
- Concat
- GroupBy
- +Min
- Feedback
- Egress

- t= (1, 0)
- t= (0, 0)
- t= (1)
Changes to Connected Graph - I

- Labels
  - Ingress
  - Concat
  - Join
  - Feedback
  - Map
  - Concat
  - GroupBy +Min
  - Egress

- Edges
  - t = (1, 0)
  - 4 → 2, 4
  - 2 → 4, 2
Changes to Connected Graph - I

t = (1, 0)

Labels

Edges

Ingress

Concat

Join

Map

GroupBy +Min

Egress

Feedback

Labels: Concat, Join, Concat, GroupBy +Min

Edges: Ingress, Concat, Join, Map, GroupBy +Min, Egress

Feedback: Concat, GroupBy +Min
Changes to Connected Graph - I

- Ingress
- Concat
- Join
- Map
- GroupBy + Min
- Concat
- Feedback
- Egress

$t = (1, 0)$

2 4
4 2
### Changes to Connected Graph - I

**Cumulative Input from `concat`**

<table>
<thead>
<tr>
<th>t=(0, 0)</th>
<th>t=(1, 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 1</td>
<td>2 4</td>
</tr>
<tr>
<td>1 3 1 1</td>
<td>4 2</td>
</tr>
<tr>
<td>3 4 2 2</td>
<td></td>
</tr>
<tr>
<td>4 3 3 3</td>
<td></td>
</tr>
<tr>
<td>2 4 4 4</td>
<td></td>
</tr>
<tr>
<td>4 2 5 5</td>
<td></td>
</tr>
<tr>
<td>5 2</td>
<td></td>
</tr>
<tr>
<td>2 5</td>
<td></td>
</tr>
</tbody>
</table>

**Groupby + Min**

<table>
<thead>
<tr>
<th>t=(0, 0)</th>
<th>t=(1, 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1</td>
<td>2 2</td>
</tr>
<tr>
<td>2 2</td>
<td></td>
</tr>
<tr>
<td>3 1</td>
<td></td>
</tr>
<tr>
<td>4 3</td>
<td>4 2</td>
</tr>
<tr>
<td>5 2</td>
<td>5 2</td>
</tr>
</tbody>
</table>

**Diagram**

```
<table>
<thead>
<tr>
<th>Labels</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingress</td>
<td>Concat</td>
</tr>
<tr>
<td>Join</td>
<td>Map</td>
</tr>
<tr>
<td>Concat</td>
<td>GroupBy + Min</td>
</tr>
<tr>
<td>Egress</td>
<td></td>
</tr>
</tbody>
</table>
```

**Feedback**

- t=(0, 0)
  - 1 1
  - 2 2
  - 3 1
  - 4 3
  - 5 2

- t=(1, 0)
  - 4 2
  - 4 3
```
Changes to Connected Graph - 1

t = (1, 1)
Changes to Connected Graph - I

$\mathbf{t = (1, 1)}$
Changes to Connected Graph - I

![Diagram of connected graph with nodes and edges labeled with numbers 1, 2, 3, and 4. The graph includes operations such as Concat, Join, GroupBy, +Min, Ingress, and Egress. The feedback loop is indicated with a dashed line. The node labels include 't = (1, 1)'.]
Changes to Connected Graph - 1

**Ingress**

**Concat**

**Join**

**Map**

**GroupBy + Min**

**Feedback**

**Edges**

**Labels**

**Egress**

$t = (1, 1)$

Nodes with node weights:
- 4
- 2
- 4
- 3
Changes to Connected Graph - 1

```
<table>
<thead>
<tr>
<th>Labels</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingress</td>
<td>Join</td>
</tr>
<tr>
<td>Concat</td>
<td>Map</td>
</tr>
<tr>
<td>Feedback</td>
<td>Concat</td>
</tr>
<tr>
<td></td>
<td>GroupBy+Min</td>
</tr>
<tr>
<td></td>
<td>Egress</td>
</tr>
</tbody>
</table>
```

$t = (1, 2)$
Changes to Connected Graph - II

Add Undirected Edge
Changes to Connected Graph - II

Labels
- Ingress
- Concat
- Join
- Map
- Concat
- GroupBy
- +Min
- Egress

Edges
- Feedback

Diagram:
- t = (2)
- Nodes:
  1. 1
  2. 4
  3. 1

Connections:
- Ingress to Concat
- Concat to Join
- Join to Map
- Map to Concat
- Concat to GroupBy
- GroupBy to +Min
- +Min to Egress
- Feedback to Concat
- Feedback to GroupBy
- Feedback to +Min
Changes to Connected Graph - II

\[ t = (1, 0) \]

\[ \text{ Concat } \]

\[ \text{ Join } \]

\[ \text{ Concat } \]

\[ \text{ GroupBy } \]

\[ +\text{Min} \]

\[ \text{ Feedback } \]

\[ \text{ Egress } \]

\[ t = (2, 0) \]

\[ t = (0, 0) \]

\[ 1 \]

\[ 2 \]

\[ 3 \]

\[ 4 \]

\[ 5 \]

\[ t = (1) \]

\[ 1 \]

\[ 4 \]

\[ 4 \]

\[ 1 \]
Changes to Connected Graph - II

Labels

Edges

Concat

Join

Map

GroupBy +Min

Egress

Feedback

Ingress

Labels

Edges

Concat

Join

Map

GroupBy +Min

Egress

Feedback

Ingress

t = (2, 0)
Changes to Connected Graph - II

```
<table>
<thead>
<tr>
<th>Ingress</th>
<th>Concat</th>
<th>Join</th>
<th>Map</th>
<th>Concat</th>
<th>GroupBy</th>
<th>+Min</th>
<th>Egress</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

t= (2, 0)

```
[4, 1]
[1, 4]
```
Changes to Connected Graph - II

Labels

Ingress

Concat

Join

Map

GroupBy +Min

Feedback

Edges

Egress

t = (2, 0)

4 1

1 4
Changes to Connected Graph - II

Labels

Ingress

Concat

Join

Map

Concat

GroupBy +Min

Feedback

Egress

Feedback

\(t = (2, 0)\)

4

1

4

3
Changes to Connected Graph - II

t = (2, 1)

Labels → Edges
- Ingress
- Concat
- Join
- Map
- Concat
- GroupBy + Min
- Feedback
- Egress
Changes to Connected Graph - II

t= (2, 1)
Changes to Connected Graph - II

Diagram with nodes labeled as follows:
- Labels
- Edges
- Ingress
- Concat
- Join
- Feedback
- Concat
- Map
- GroupBy +Min
- Egress

The diagram includes a set of numbers indicating connections: t= (2, 1)
Changes to Connected Graph - II

Diagram:

- Labels
  - Ingress
    - Concat
    - Join
      - Map
      - GroupBy +Min
      - Egress
  - Feedback
- Edges
Changes to Connected Graph - II

Labels

- Ingress
- Concat

Edges

- Join
- Map
- Concat
- GroupBy + Min

Feedback

Egress

$t = (2, 1)$

Nodes: 1, 1, 1, 3, 4, 3
Changes to Connected Graph - II

t = (2, 1)
Discussion
How do you deal with new data when you are iterating with old data?

- How much state to keep (Memory)
  - Do you keep all data in memory
  - What about intermediate calculations
  - Incremental View Maintenance

- How much work to do
  - Do everything from the beginning
  - Do work only in the nodes where new data came in
Iterative vs Differential Dataflow

● The flexibility of partial Ordering.
  ○ Iterative Ordering - \((i_1, j_1) \leq (i_2, j_2)\) iff \(i_1 \leq i_2\) and \(j_1 \leq j_2\).
  ○ Lexicographic Ordering - \((i_1, j_1) \leq (i_2, j_2)\) if \(i_1 < i_2\) or \(i_1 = i_2\) and \(j_1 \leq j_2\).
  ○ Programmer can choose the partial ordering.

● Communication via Diffs
  ○ Only the diffs are sent around as messages
  ○ Nodes on both sides know the previous calculations
Discussion

● Is the memory for performance tradeoff justified?
Thank You