Dependency-Driven Disk-based Graph Processing

Keval Vora
Simon Fraser University

USENIX ATC’19 - Renton, WA
July 11, 2019

https://github.com/pdclab/lumos
Graph Processing
Iterative Graph Processing

\[ \text{PR}(v^t) = 1 - d + d \times \sum_{\text{inedges}(v)} \frac{\text{PR}(u^{t-1})}{|\text{outedges}(u)|} \]
Iterative Graph Processing

\[
PR(v^t) = 1 - d + d \times \sum_{\text{in} \text{edges}(v)} \frac{PR(u^{t-1})}{|\text{out} \text{edges}(u)|}
\]
Out-of-Core Graph Processing

\[ V = \begin{array}{cccccc}
  a & b & c & d & e & f \\
\end{array} \]

iteration t

\[ \begin{array}{ccc}
  e & \rightarrow & a \\
  b & \rightarrow & c \\
  d & \rightarrow & e \\
\end{array} \]

\[ \begin{array}{ccc}
  c & \rightarrow & f \\
  b & \rightarrow & d \\
  a & \rightarrow & e \\
\end{array} \]

\[ \text{partitions reside on disk} \]

iteration t+1

\[ \begin{array}{ccc}
  e & \rightarrow & a \\
  d & \rightarrow & c \\
  a & \rightarrow & e \\
\end{array} \]

\[ \begin{array}{ccc}
  c & \rightarrow & f \\
  b & \rightarrow & d \\
  b & \rightarrow & d \\
\end{array} \]
Out-of-Core Graph Processing

\[ V = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix} \]

\[ V = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix} \]

iteration t

\[ \text{iteration t+1} \]
Out-of-Core Graph Processing

\[ V = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix} \]

\[ V = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix} \]

\[ P_0 \quad P_1 \]

iteration t

\[ V = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix} \]

\[ V = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix} \]

iteration t+1
Out-of-Core Graph Processing

\[ V = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix} \]

Iteration t

P0

\[ \begin{array}{c}
  e \\
  b \\
  d \\
\end{array} \rightarrow \begin{array}{c}
  a \\
  c \\
  e \\
\end{array} \]

P1

\[ \begin{array}{c}
  c \\
  b \\
  a \\
\end{array} \rightarrow \begin{array}{c}
  f \\
  d \\
  e \\
\end{array} \]

Iteration t+1

P0

\[ \begin{array}{c}
  e \\
  b \\
  d \\
\end{array} \rightarrow \begin{array}{c}
  a \\
  c \\
  e \\
\end{array} \]

P1

\[ \begin{array}{c}
  c \\
  b \\
  a \\
\end{array} \rightarrow \begin{array}{c}
  f \\
  d \\
  e \\
\end{array} \]
Out-of-Core Graph Processing

\[ V = \{a, b, c, d, e, f\} \]

Iteration \( t \):

- \( V = \{a, b, c, d, e, f\} \)
- \( V = \{a, b, c, d, e, f\} \)

Compute

Iteration \( t+1 \):

- \( V = \{a, b, c, d, e, f\} \)
- \( V = \{a, b, c, d, e, f\} \)

\[ V = \{a, b, c, d, e, f\} \]
Out-of-Core Graph Processing

\[ V = \{ a, b, c, d, e, f \} \]

Iteration t:
- \( P_0 \):
  - e \rightarrow a
  - b
  - d \rightarrow c

- \( P_1 \):
  - c \rightarrow f
  - b \rightarrow d \rightarrow e

Iteration t+1:
- \( P_0 \):
  - e \rightarrow a
  - b
  - d \rightarrow c

- \( P_1 \):
  - c \rightarrow f
  - b \rightarrow d \rightarrow e

I/O = 69-90%
Lumos

V = \{a, b, c, d, e, f\}

- **Out-of-Order** processing
  - Amortize I/O across multiple iterations
- Guarantee **Bulk Synchronous Parallel** Semantics
- **Dependency-Driven Cross-Iteration** value propagation
  - *Safely* compute future values
Computing Future Values

\[ V^F = [a, b, c, d, e, f] \]

\[ V = [a, b, c, d, e, f] \]

Future vertex values

I/O C

iteration t

iteration t+1
Computing Future Values

\[ V^F = \begin{array}{cccccc}
  a & b & c & d & e & f \\
\end{array} \]

\[ V = \begin{array}{cccccc}
  a & b & c & d & e & f \\
\end{array} \]

\[ V = \begin{array}{cccccc}
  a & b & c & d & e & f \\
\end{array} \]
Computing Future Values

\[ V^F = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix} \]

\[ V = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix} \]

Iteration t

Iteration t+1
Computing Future Values

\[ V^F = \begin{array}{cccccc}
    a & b & c & d & e & f \\
\end{array} \]

\[ V = \begin{array}{cccccc}
    a & b & c & d & e & f \\
\end{array} \]

Iteration t

\[ \begin{array}{c}
P0 \\
\end{array} \]

\[ \begin{array}{c}
e \\
b \\
d \\
a \\
f \\
\end{array} \]

\[ \begin{array}{c}
a \\
c \\
d \\
e \\
f \\
\end{array} \]

\[ \begin{array}{c}
a \\
c \\
d \\
e \\
f \\
\end{array} \]

Iteration t+1

\[ \begin{array}{c}
P0 \\
\end{array} \]

\[ \begin{array}{c}
e \\
b \\
d \\
a \\
f \\
\end{array} \]

\[ \begin{array}{c}
a \\
c \\
d \\
e \\
f \\
\end{array} \]

\[ \begin{array}{c}
b \\
d \\
\end{array} \]
Computing Future Values

\[ V^F = \begin{array}{cccccc}
  a & b & c & d & e & f \\
\end{array} \]

\[ V = \begin{array}{cccccc}
  a & b & c & d & e & f \\
\end{array} \]

Future dependencies not met
30-45% vertices compute future values, and they contribute for less than 4% edges.
Graph Computation

\[ PR(v^t) = 1 - d + d \times \sum_{\text{in\text{edges}(v)}} \frac{PR(u^{t-1})}{|\text{out\text{edges}(u)}|} \]

\[ v^t = f( \bigoplus_{\text{in\text{edges}(v)}} (u^{t-1}) ) \]

\[ \text{in\text{edges}(v)} = X \cup Y \quad \rightarrow \quad \bigoplus_{\text{in\text{edges}(v)}} (u^t) = \bigoplus_X (\bigoplus_{u^t} , \bigoplus_Y (u^t) ) \]

Partial Aggregation
Graph Computation

\[ \text{PR}(v^t) = 1 - d \]

\[ v^t = f \left( \bigoplus_{\text{inedges}(v)} (u^t) \right) / |\text{outedges}(u)| \]

\[ \text{inedges}(v) = X \cup Y \quad \Rightarrow \quad \bigoplus_{\text{inedges}(v)} (u^t) = \bigoplus_{X} \left( \bigoplus_{Y} (u^t) \bigoplus (u^t) \right) \]

Partial Aggregation
Graph Computation

\[ PR(v^t) = 1 - d \]

\[ v^t = f( \bigoplus_{u \in \text{inedges}(v)} u^t ) / |\text{outedges}(u)| \]

\[ \text{inedges}(v) = X \cup Y \quad \rightarrow \quad \bigoplus_{\text{inedges}(v)} (u^t) = \bigoplus_{X} (\bigoplus_{X} (u^t), \bigoplus_{Y} (u^t)) \]

Partial Aggregation
Graph Computation

\[ \text{PR}(v^t) = 1 - d \]

\[
\begin{array}{c}
\oplus \\
\text{PR}(v^t) = f \left( \bigoplus_{\text{inedges}(v)} (u^t) \right) / |\text{outedges}(u)|
\end{array}
\]

\[ v^t = f \left( \bigoplus_{\text{inedges}(v)} (u^t) \right) \]

**Partial Aggregation**

\[ \text{inedges}(v) = X \cup Y \]

\[
\bigoplus_{\text{inedges}(v)} (u^t) = \bigoplus_{X} \left( \bigoplus_{X} (u^t) , \bigoplus_{Y} (u^t) \right)
\]
Cross-Iteration Value Propagation

Partial Aggregation

\[ \bigoplus = \begin{array}{cccccc}
    a & b & c & d & e & f \\
\end{array} \]

\[ V = \begin{array}{cccccc}
    a & b & c & d & e & f \\
\end{array} \]

\[ V = \begin{array}{cccccc}
    a & b & c & d & e & f \\
\end{array} \]

Iteration t

\[ P_0 \]

\[ P_1 \]

Iteration t+1

\[ P_0 \]

\[ P_1 \]

I/O C

I/O C

I/O C

I/O C
Cross-Iteration Value Propagation

Partial Aggregation

$$\bigoplus = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix}$$

$$V = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix}$$

iteration $t$

\[ \text{I/O C} \]

iteration $t+1$

\[ \text{I/O C} \]
Cross-Iteration Value Propagation

Partial Aggregation

\[ \oplus = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix} \]

\[ V = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix} \]

\[ V = \begin{bmatrix} a & b & c & d & e & f \end{bmatrix} \]

\[ \begin{array}{c}
\text{iteration t} \\
\hline
P0 \\
\begin{array}{c}
\begin{array}{c}
\text{e} \\
\text{b} \\
\text{d} \\
\text{a} \\
\text{c} \\
\end{array}
\end{array}
\end{array} \]

\[ \begin{array}{c}
\text{P0} \\
\begin{array}{c}
\begin{array}{c}
\text{c} \\
\text{b} \\
\text{d} \\
\text{a} \\
\text{e} \\
\end{array}
\end{array}
\end{array} \]

\[ \begin{array}{c}
\text{iteration t+1} \\
\hline
P0 \\
\begin{array}{c}
\begin{array}{c}
\text{c} \\
\text{b} \\
\text{d} \\
\text{a} \\
\text{e} \\
\end{array}
\end{array}
\end{array} \]

\[ \begin{array}{c}
\text{P1} \\
\begin{array}{c}
\begin{array}{c}
\text{f} \\
\text{d} \\
\text{c} \\
\text{a} \\
\text{e} \\
\end{array}
\end{array}
\end{array} \]
Cross-Iteration Value Propagation

Partial Aggregation

\[ \bigoplus = a \ b \ c \ d \ e \ f \]

\[ V = a \ b \ c \ d \ e \ f \]

Iteration t

\[ P_0 \]

\[ \begin{array}{c}
e \\
b \\
d \end{array} \rightarrow \begin{array}{c}a \\
c \\
e \end{array} \]

\[ P_1 \]

\[ \begin{array}{c}
c \\
\ b \\
\ d \end{array} \rightarrow \begin{array}{c}f \\
a \ e \ d \ b \ c \ f \end{array} \]

Iteration t+1

\[ P_0 \]

\[ \begin{array}{c}
e \\
\ b \\
\ d \end{array} \rightarrow \begin{array}{c}a \\
c \ e \ d \ b \ a \ e \ f \end{array} \]

\[ P_1 \]

\[ \begin{array}{c}
c \\
\ b \\
\ d \end{array} \rightarrow \begin{array}{c}f \\
b \ d \ b \ c \ a \ e \ f \end{array} \]
Cross-Iteration Value Propagation

future values computed across >45% edges
Intra-Partition Propagation

\[ \bigodot = \begin{array}{cccccc}
  a & b & c & d & e & f \\
\end{array} \]

\[ V = \begin{array}{cccccc}
  a & b & c & d & e & f \\
\end{array} \]

Iteration \( t \) vs. Iteration \( t + 1 \):

- **Iteration \( t \)**:
  - \( P_0 \)
    - \( e \rightarrow a \)
    - \( b \rightarrow c \)
    - \( d \rightarrow c \)
  - \( P_1 \)
    - \( c \rightarrow f \)
    - \( b \rightarrow d \)
    - \( a \rightarrow e \)

- **Iteration \( t + 1 \)**:
  - \( P_0 \)
    - \( e \rightarrow a \)
    - \( b \rightarrow d \)
    - \( d \rightarrow c \)
  - \( P_1 \)
    - \( f \rightarrow d \)
    - \( f \rightarrow c \)
    - \( d \rightarrow e \)
Intra-Partition Propagation

\[ \oplus = \begin{array}{cccccc}
  a & b & c & d & e & f \\
\end{array} \]

\[ V = \begin{array}{cccccc}
  a & b & c & d & e & f \\
\end{array} \]

\[ V = \begin{array}{cccccc}
  a & b & c & d & e & f \\
\end{array} \]

\[ \begin{array}{c}
  P_0 \\
  P_1 \\
\end{array} \]

iteration \( t \)

iteration \( t+1 \)

\[ V = \begin{array}{cccccc}
  a & b & c & d & e & f \\
\end{array} \]

\[ V = \begin{array}{cccccc}
  a & b & c & d & e & f \\
\end{array} \]

\[ \begin{array}{c}
  P_0 \\
  P_1 \\
\end{array} \]

I/O

I/O

I/O
Lumos: Cross-Iteration Value Propagation

• Propagation across multiple future iterations
• Synergy with Dynamic Shards [ATC’16]
• Detailed comparison with out-of-order techniques
• Generalized graph layout & partitioning strategies
• Cross-iteration propagation across different partition sizes
• Interplay with selective scheduling

• Lumos for Asynchronous Algorithms
Lumos for Asynchronous Algorithms

\[
SSSP(v^{t+1}) = \min_{\text{inedges}(v)} SSSP(u^t) + \text{weight}(u, v)
\]

\[\oplus = \begin{array}{cccccc}
a & b & c & d & e & f \\
\end{array}\]

\[V = \begin{array}{cccccc}
a & b & c & d & e & f \\
\end{array}\]

\[
\text{iteration } t
\]

\[
\text{iteration } t+1
\]
Lumos for Asynchronous Algorithms

$$\text{SSSP}(v^{t+1}) = \min_{\text{in}} \text{SSSP}(u^t) + \text{weight}(u, v)$$

$$\bigoplus = \begin{array}{cccccc}
    a & b & c & d & e & f \\
\end{array}$$

$$V = \begin{array}{cccccc}
    a & b & c & d & e & f \\
\end{array}$$
Lumos for Asynchronous Algorithms

\[
SSSP(v^{t+1}) = \min_{\text{inedges}(v)} SSSP(u^t) + \text{weight}(u, v)
\]

\[\bigoplus = \begin{array}{cccccc}
a & b & c & d & e & f \\
\end{array}\]

\[V = \begin{array}{cccccc}
a & b & c & d & e & f \\
\end{array}\]

- Monotonic selection (KickStarter [ASPLOS’17])
Lumos for Asynchronous Algorithms

\[
SSSP(v^+) = \min_{\text{inedges}(v)} SSSP(u^v) + \text{weight}(u, v)
\]

- Monotonic selection (KickStarter [ASPLOS’17])
- Partial aggregations merged directly
Lumos for Asynchronous Algorithms

\[ \text{SSSP}(v) = \min_{\text{inedges}(v)} \text{SSSP}(u) + \text{weight}(u, v) \]

- Monotonic selection (KickStarter [ASPLOS’17])
- Partial aggregations merged directly
- Partial values safe to be propagated
  - Any change can be propagated immediately
Lumos for Asynchronous Algorithms

$$SSSP(v) = \min_{\text{in-edges}(v)} SSSP(u) + \text{weight}(u, v)$$

- Monotonic selection (KickStarter [ASPLOS’17])
- Partial aggregations merged directly

- Partial values safe to be propagated
  - Any change can be propagated immediately
Lumos for Asynchronous Algorithms

\[
V = \begin{array}{cccccc}
a & b & c & d & e & f \\
\end{array}
\]

\[
V = \begin{array}{cccccc}
e & a & c & b & d & f \\
\end{array}
\]

\[
V = \begin{array}{cccccc}
a & b & c & d & e & f \\
\end{array}
\]

\[
V = \begin{array}{cccccc}
e & a & c & b & d & f \\
\end{array}
\]

SSP(v) = \min_{\text{in degrees}(v)} \text{SSP}(u) + \text{weight}(u, v)

- Monotonic selection (KickStarter [ASPLOS’17])
- Partial aggregations merged directly
- Partial values safe to be propagated
  - Any change can be propagated immediately
- Intra-partition propagation based on degree of asynchrony (ASPIRE [OOPSLA’14])
Lumos System

• Grid layout based on GridGraph [ATC’15]
Lumos System
future values propagated for upper triangle + diagonal edges
Lumos System

\[ \oplus = \begin{array}{ccccccc} & & & & & & \\
& & & & & & \\
& & & & & & \\
& & & & & & \\
& & & & & & \\
& & & & & & \\
\end{array} \]

\[ V = \begin{array}{ccccccc} & & & & & & \\
& & & & & & \\
& & & & & & \\
& & & & & & \\
& & & & & & \\
& & & & & & \\
\end{array} \]

\[ V = \begin{array}{ccccccc} & & & & & & \\
& & & & & & \\
& & & & & & \\
& & & & & & \\
& & & & & & \\
& & & & & & \\
\end{array} \]

only lower triangle edges loaded

iteration \( t \)

iteration \( t+1 \)
Lumos System

function processPrimary(propagate, crossPropagate, compute)
function processSecondary(propagate, compute)
Light-weight Degree-Aware Partitioning

HOF: Highest Out-Degree First
HIL: Highest In-Degree Last
HRF: Highest Out-Degree to In-Degree Ratio First

maximize edges in upper triangle
Light-weight Degree-Aware Partitioning

HOF: Highest Out-Degree First
HIL: Highest In-Degree Last
HRF: Highest Out-Degree to In-Degree Ratio First

cross-iteration propagation across 72-93% edges

<table>
<thead>
<tr>
<th></th>
<th>Edges</th>
<th>Vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>TT</td>
<td>2B</td>
<td>52.6M</td>
</tr>
<tr>
<td>FT</td>
<td>2.5B</td>
<td>68.3M</td>
</tr>
<tr>
<td>YH</td>
<td>6.6B</td>
<td>1.4B</td>
</tr>
</tbody>
</table>
Experimental Setup

• Graph algorithms
  • PageRank (weighted and unweighted), Belief Propagation, Co-Training Expectation Maximization, Dispersion, Label Propagation

• Performance on single disk
  • h1.2xlarge: 278MB/sec HDD read bandwidth

• Scaling I/O
  • d2.4xlarge: 195-768MB/sec read bandwidth over 1-4 HDDs
  • i3.8xlarge: 1.2-4.1GB/sec read bandwidth over 1-4 SSDs
Performance

h1.2xlarge: 278MB/sec HDD read bandwidth

<table>
<thead>
<tr>
<th></th>
<th>TT</th>
<th>YH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edges</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2B</td>
<td>6.6B</td>
</tr>
<tr>
<td>Vertices</td>
<td>52.6M</td>
<td>1.4B</td>
</tr>
</tbody>
</table>
Performance

h1.2xlarge: 278MB/sec HDD read bandwidth

<table>
<thead>
<tr>
<th></th>
<th>Edges</th>
<th>Vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>TT</td>
<td>2B</td>
<td>52.6M</td>
</tr>
<tr>
<td>YH</td>
<td>6.6B</td>
<td>1.4B</td>
</tr>
</tbody>
</table>
Scaling I/O

<table>
<thead>
<tr>
<th>Device Type</th>
<th>Single Drive</th>
<th>RAID-0 with k drives</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>k = 2</td>
</tr>
<tr>
<td>d2.4xlarge</td>
<td>HDD</td>
<td>195MB/s</td>
</tr>
<tr>
<td>i3.8xlarge</td>
<td>SSD</td>
<td>1.2GB/s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Edges</th>
<th>Vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMAT29</td>
<td></td>
</tr>
<tr>
<td>8.6B</td>
<td>537M</td>
</tr>
</tbody>
</table>
Detailed Evaluation of Lumos

- **Read** and **Compute** performance metrics are compared across different versions of Lumos.

<table>
<thead>
<tr>
<th>Version</th>
<th>TT</th>
<th>FT</th>
<th>YH</th>
</tr>
</thead>
<tbody>
<tr>
<td>GridGraph</td>
<td>737</td>
<td>1008</td>
<td>3223</td>
</tr>
<tr>
<td>LUMOS-BASE</td>
<td>563</td>
<td>659</td>
<td>2027</td>
</tr>
<tr>
<td><em>×</em> LUMOS</td>
<td>439</td>
<td>583</td>
<td>1885</td>
</tr>
<tr>
<td>CoEM</td>
<td>119</td>
<td>1554</td>
<td>5082</td>
</tr>
<tr>
<td>LUMOS-BASE</td>
<td>861</td>
<td>1029</td>
<td>3216</td>
</tr>
<tr>
<td><em>×</em> LUMOS</td>
<td>651</td>
<td>914</td>
<td>3043</td>
</tr>
<tr>
<td>DP</td>
<td>846</td>
<td>1032</td>
<td>3484</td>
</tr>
<tr>
<td>LUMOS-BASE</td>
<td>656</td>
<td>675</td>
<td>2219</td>
</tr>
<tr>
<td><em>×</em> LUMOS</td>
<td>498</td>
<td>611</td>
<td>2111</td>
</tr>
<tr>
<td>BP</td>
<td>2498</td>
<td>3782</td>
<td>13769</td>
</tr>
<tr>
<td>LUMOS-BASE</td>
<td>1921</td>
<td>2456</td>
<td>8660</td>
</tr>
<tr>
<td><em>×</em> LUMOS</td>
<td>1487</td>
<td>2212</td>
<td>7913</td>
</tr>
<tr>
<td>WPR</td>
<td>984</td>
<td>1302</td>
<td>4330</td>
</tr>
<tr>
<td>LUMOS-BASE</td>
<td>769</td>
<td>874</td>
<td>2758</td>
</tr>
<tr>
<td><em>×</em> LUMOS</td>
<td>569</td>
<td>770</td>
<td>2547</td>
</tr>
<tr>
<td>LP</td>
<td>1054</td>
<td>1421</td>
<td>4583</td>
</tr>
<tr>
<td>LUMOS-BASE</td>
<td>805</td>
<td>935</td>
<td>2976</td>
</tr>
<tr>
<td><em>×</em> LUMOS</td>
<td>624</td>
<td>826</td>
<td>2728</td>
</tr>
</tbody>
</table>

- Normalized Time and Space metrics are shown for different layouts and partitions.

- **TW**, **TT**, **FT**, and **YH** layouts are compared for various performance metrics.

- Tables and graphs illustrate performance improvements across different versions and layouts.
Conclusion

- **Dependency-Driven Cross-iteration** value propagation
  - **Out-of-Order** processing to reduce I/O
  - Guarantees **Bulk Synchronous Parallel** semantics
- Generic technique that fundamentally eliminates the performance barrier