Seminar: Massive-Scale Graph Analysis
Summer Semester 2015

Graph Analysis Using Map/Reduce

Laurent Linden
s9lalind@stud.uni-saarland.de

May 14, 2015
Introduction

Peta-Scale Graph
+ Graph mining
+ MapReduce

= PEGASUS
Related Work

- Page Rank (Google)
- Random Walk w/ Restart (Pan et al.)
- Diameter Estimation (Kang et al.)
- Connected Components (Kang et al.)
Problems

- No unified solution
- Individual optimizations
- Hard to understand
Ideas

- Find common primitive
- Translate mining tasks
- Solve generalized problem
- Optimize solution
PeGaSus

• Implements common primitive
• Users specify:
  – stopping criterion (e.g. $y^{t+1} = y^t$)
  – 3 operations (combine2, combineAll, assign)
• Provides MR Jobs for
  – PR, RWR, DE, CC
Generalized Iterative Matrix-Vector mult.

- Graph = Adjacency matrix
  - Graph Mining
    - Generalized Matrix-Vector multiplication
    - Iterative solution finding

\[ y_{i} = \sum_{j=1}^{n} e_{i,j} \cdot y_{j} \]
\[ y^{t+1} = E \times_{G} y^{t} \]
GIM-V

- Combine2($e_{i,j}$, $y_j^t$)
- CombineAll$_i$( {... })
- Assign($y_i^t$, $y_i^{t+1}$)

- In general:

$$y_i^{t+1} = \sum_{j=1}^{n} e_{i,j} \cdot y_j^t$$

$$y_i^{t+1} = \text{assign}(y_i^t, \text{combineAll}_i(\{ z_j | j = 1..n, \text{ and } z_j = \text{combine } 2(e_{i,j}, y_j^t) \}))$$
Example: Page Rank

- Rank pages by relevance
- Start out with normal distribution
- Compute next step via Power Method

Source: wikipedia.org/wiki/PageRank
Example: Page Rank

- **E** column-normalized adjacency matrix
- **U** uniform matrix
- **c** damping factor
- **p** page rank vector
  - Satisfies the equation
- Iterative scheme

\[
E = (e_{i,j}) \\
U = \left( \frac{1}{n} \right)_{i,j} \\
0 \leq c \leq 1 \\
p = (c \cdot E + (1-c) \cdot U) \cdot p \\
p_i^0 = \frac{1}{n} \\
p^{t+1} = M \times_G p^t
\]
Example: Page Rank

\[ p^{t+1} = (c \cdot E^T + (1-c) \cdot U) \times_G p^t \]

\[ \text{combine} 2(e_{i,j}, p^t_j) = c \times e_{i,j} \times p^t_j \]

\[ \text{assign}(p^t_i, p^{t+1}_i) = p^{t+1}_i \]

\[ \text{combineAll}_{i}(\{ x_1, \ldots, x_n \}) = \frac{(1-c)}{n} + \sum_{j=1}^{n} x_j \]

\[ p^{t+1}_i = \text{assign}(p^t_i, \text{combineAll}_{i}(\{ x_j \mid j=1..n \text{, and } x_j = \text{combine} 2(m_{i,j}, p^t_j) \})) \]
Example: Random Walks w/ Restart

- Measures proximity of nodes
- Start with arbitrary distribution
- Refine distribution until results converge

Source: Pan et al, “Fast Random Walk with Restart and its Application”
Example: Random Walks w/ Restart

- **E column-normalized adjacency matrix**
- **$e_k$ vector with k-th component = 1**
- **c restart probability**
- **$r_k$ proximity vector**
  - Satisfies the equation

**Iterative Scheme**

\[
E = (e_{i,j}) \quad e_k = (0, \ldots, 0, 1, 0, \ldots, 0) \quad 0 \leq c \leq 1
\]

\[
r_k = c \cdot E \cdot r_k + (1-c) \cdot e_k
\]

\[
r_k^0 = \text{initial distribution} \quad r_k^{t+1} = M \times_G r_k^t
\]
Example: Random Walks w/ Restart

\[ r_{k}^{t+1} = \left( c \cdot E \cdot r_{k} + \left( 1 - c \right) \cdot e_{k} \right) \times G r_{k}^{t} \]

\[ \text{combine} \ 2(e_{i,j}, r_{j}^{t}) = c \times e_{i,j} \times r_{j}^{t} \]

\[ \text{assign}(r_{i}^{t}, r_{i}^{t+1}) = r_{i}^{t+1} \]

\[ \text{combineAll}_{i}(\{x_{1}, \ldots, x_{n}\}) = (1 - c) I(i = k) + \sum_{j=1}^{n} x_{j} \]

\[ r_{k}^{t+1} = \text{assign}(r_{k}^{t}, \text{combineAll}_{i}(\{x_{j} | j = 1 \ldots n, \text{ and } x_{j} = \text{combine} \ 2(m_{i,j}, r_{k}^{t}) \})) \]
Example: Diameter Estimation

- Estimate radius and diameter of subgraphs

- Radius of a node
  - #Hops to farthest-away node

- Diameter of a graph
  - Maximal length of shortest path between every pair of nodes

Radius(3) = 2

Diameter = 3
Example: Diameter Estimation

- M adjacency matrix
  \[ M = \left( m_{i,j} \right) \]
- \( b_i \) probabilistic bitstring
  - Solution satisfies equation
- Iterative Scheme

\[ b_i^{h+1} = b_i^h \quad \text{(for all } i) \]
\[ b_i^0 = 0...010...0 \]
\[ b_i^{h+1} = b_i^h \ \text{BITWISE-OR} \{ b_k^h | (i, k) \in E \} \]
Example: Diameter Estimation

\[ b_{i}^{h+1} = M \times_{G} b_{i}^{h} \]

\[ \text{combine } 2(m_{i,j}, b_{j}^{h}) = m_{i,j} \times b_{j}^{h} \]

\[ \text{combineAll}_{i} \left( \{ x_{1}, \ldots, x_{n} \} \right) = \text{BITWISE-OR} \left\{ x_{j} \mid j = 1 \ldots n \right\} \]

\[ \text{assign} \left( b_{i}^{h}, b_{i}^{h+1} \right) = \text{BITWISE-OR} \left( b_{i}^{h}, b_{i}^{h+1} \right) \]

\[ b_{i}^{h+1} = \text{assign} \left( b_{i}^{h}, \text{combineAll}_{i} \left( \{ x_{j} \mid j = 1 \ldots n \text{, and } x_{j} = \text{combine } 2(m_{i,j}, b_{j}^{h}) \} \right) \right) \]
Example: Connected Components

- Find connected components
- Assign weight to nodes
- Smallest weight gets propagated to neighbors
- Terminate when result does not change anymore
Example: Connected Components

- $M$ adjacency matrix
- $c$ minimum node vector
  - Solution satisfies equation
- Iterative scheme

\[
M = (m_{i,j})
\]

\[
c_{i}^{h+1} = c_{i}^{h} \quad \text{for all } i
\]

\[
c_{i}^{0} = i
\]

\[
c_{i}^{h+1} = \text{MIN} \{ c_{j}^{h} \mid j \in E \}
\]
Example: Connected Components

\[ c_i^{h+1} = M \times_G c_i^h \]

\[
\text{combine } 2 (m_{i,j}, c_j^h) = m_{i,j} \times c_j^h
\]

\[
\text{combineAll}_i (\{ x_1, \ldots, x_n \}) = \text{MIN} \{ x_j | j = 1 \ldots n \}
\]

\[
\text{assign} (c_i^h, c_i^{h+1}) = \text{MIN}(c_i^h, c_i^{h+1})
\]

\[
 c_i^{h+1} = \text{assign} (c_i^h, \text{combineAll}_i (\{ x_j | j = 1 \ldots n \), and } x_j = \text{combine } 2 (m_{i,j}, c_j^h) )
\]
GIM-V BASE

- **Input**
  - Edge file $E(id_{src}, id_{dst}, mval)$
  - Vector file $V(id, vval)$
- **Output**
  - Vector file $V(id, vval)$

- **2 Stages (MR Jobs)**
  - Stage 1: Applies combine2
  - Stage 2: Applies combineAll, assign
Input:  \( M = \{ (id_{\text{src}}, (id_{\text{dst}}, mval)) \} \)
\( V = \{ (id, vval) \} \)

Output:  \( V' = \{ (id_{\text{src}}, \text{combine2}(mval, vval)) \} \)

**Mapper**\((k, v)\):
- \((k,v)\) of type \(V\):
  - \((id, vval)\) → \((id, vval)\)
- \((k,v)\) of type \(M\):
  - \((id_{\text{src}}, (id_{\text{dst}}, mval))\) → \((id_{\text{dst}}, (id_{\text{src}}, mval))\)

...
GIM-V BASE (Stage 2)

Input: \( V' = \{(id_{src},vval)\} \)

Output: \( V = \{(id_{src},vval)\} \)

Mapper\((k, v)\):
\[
(k, v) \rightarrow (k, v)
\]

Reducer\((k, v[1..m])\):

Extract vval from \( v \)

Extract Stage1 output from \( v \):
Store values in list \( A \)

\[ c = \text{combineAll}_k(A) \]
\[ a = \text{assign}(vval,c) \]
\[
(k, a)
\]
GIM-V BASE (Stage 1)

Mapper

- (0, 0.15) → (0, 0.15)
- (1, 0.25) → (1, 0.25)
- (0, 1, 0.5) → (1, 0, 0.5)
- (1, n, 0.7) → (n, 1, 0.7)

Reducer for id=0

Reducer for id=1

Reducer for id=n

- (1, [0.25, (0, 0.5), ...]) → (1, ("self", 0.25))
- (1, ("others", combine2(0.5, 0.25)))
GIM-V BASE (Stage 2)

Mapper (identity)

... → (0, (“self”, 0.15))
... → (1, (“self”, 0.25))
... → (1, (“others”, 0.3))
... → (n, (“others”, 0.7))
... → ...

Reducer for id=0

Reducer for id=1

(1, [(“self”, 0.25), (“others”, 0.3),...])

(1, assign(0.25, combineAll_1([0.3,...])))

Reducer for id=n

...
GIM-V BASE Bottlenecks

- Network traffic
- Disk IO
- Number of iterations
GIM-V BL (Block Multiplication)

- Groups elements into blocks
  - $M, V$ can be joined block-wise not element-wise
  - Guarantees co-location of nearby edges
- Groups of 2 x 2 blocks saved in a single line
  - $(\text{row}_{\text{block}}, \text{col}_{\text{block}}, \text{row}_{e1}, \text{col}_{e1}, \text{val}_{e1}, \ldots)$
  - $(\text{id}_{\text{block}}, \text{id}_{e1}, \text{vval}_{e1}, \ldots)$
GIM-V CL (Clustered Edges)

- Clusters connected vertices into the same blocks
  - Move connected vertices close to each other
  - No need to serialize 0 blocks anymore
- Preprocessing step (one time cost)
- Only useful in conjunction with GIM-V BL
GIM-V DI (Diagonal Block Iteration)

- Iterate GIM-V on block level
  - Repeated block multiplication
  - Repeated GIM-V application
- Block multiplication cheaper than GIM-V
- Only useful in conjunction with GIM-V BL
GIM-V NR (Node Renumbering)

- Renumber nodes for better convergence
  - Find center node via heuristics (highest degree)
  - Renumber center node to value of minimum node
- Preprocessing step (one time cost)
Evaluation

- Evaluation of GIM-V Performance & Scalability
- Evaluation of optimizations Relative to GIM-V BASE
Experimental Setup

- Yahoo M45
  - 4,000 processors
  - 3 TB of memory
  - 1.5 PB of disk space
  - 27 teraflops (peak performance)
- Datasets
  - Kronecker (177 K nodes, 1977 M edges)
  - LinkedIn (7.5 M nodes, 58 M edges)
  - Wikipedia (4.4 M nodes, 27 M edges)
  - And many more (not shown here)
PR & GIM-V Evaluation

Notice the impact block multiplication has on performance
CC & GIM-V Evaluation

Notice the big reduction in #iterations
DE & GIM-V Evaluation

- LinkedIn

Notice the reduction in #iterations

- Wikipedia
Conclusion

- Feasability of Peta-Scale Graph analysis
- Identification of common primitive
- Unification of GM algorithms
Q&A