Balanced Label Propagation for Partitioning Massive Graphs

Johan Ugander, Cornell University
Lars Backstrom, Facebook
WSDM ‘13
Goal: partition a really big graph
Motivation: distributed computation

- Distributing graph calculations (‘sharding a graph’) makes traversal/aggregation very expensive.
Motivation: distributed computation

- Distributing graph calculations (‘sharding a graph’) makes traversal/aggregation very expensive.

- **Naive sharding**:

  ![Diagram showing naive sharding](image)

  \[ \Pr(\text{colocation}) = \frac{1}{n} \]
Motivation: distributed computation

- Distributing graph calculations (‘sharding a graph’) makes traversal/aggregation very expensive.

- **Naive sharding:**

- **Intelligent sharding:** specify a *shard map* $f()$ that colocates users with friends

Pr( colocation ) = 1 / n
Motivation: distributed computation

- Distributing graph calculations (‘sharding a graph’) makes traversal/aggregation very expensive.

- **Naive sharding:**

  ![Diagram of naive sharding]

  - ID%%n == 0
  - ID%%n == 1
  - ID%%n == 2
  - ID%%n == 3
  - ...  
  - ID%%n == n-4
  - ID%%n == n-3
  - ID%%n == n-2
  - ID%%n == n-1

  Pr(colocation) = 1/n

- **Intelligent sharding:** specify a *shard map* $f()$ that colocalizes users with friends

  ![Diagram of intelligent sharding]

  - f(ID) == 0
  - f(ID) == 1
  - f(ID) == 2
  - f(ID) == 3
  - ...  
  - f(ID) == n-4
  - f(ID) == n-3
  - f(ID) == n-2
  - f(ID) == n-1

Oh... and the algorithm better be FAST.
Partitioning a really big graph: How?

- Garey, Johnson, Stockmeyer 1976: Minimum bisection is NP-hard
- Karypsis and Kumar 1998: METIS
- Feige and Krautgamer 2000: $O(n^{1/2} \log n)$-factor approximation
Partitioning a really big graph: How?

- Garey, Johnson, Stockmeyer 1976: Minimum bisection is NP-hard
- Karypsis and Kumar 1998: METIS
- Feige and Krautgamer 2000: $O(n^{1/2} \log n)$-factor approximation

- METIS does not scale to 100B+ edges.
- Need a principled approach, ideally one that can be Hadoop-ified.
Basic idea: Label propagation

- Iteratively move nodes to be with the plurality of their neighbors:

Basic idea: Label propagation

- Iteratively move nodes to be with the plurality of their neighbors:

Basic idea: Label propagation

- Iteratively move nodes to be with the plurality of their neighbors:
- But how to maintain balance?

Basic idea: Label propagation

- Iteratively move nodes to be with the plurality of their neighbors:
- But how to maintain balance?
- Label = machine
Basic idea: Label propagation

- Iteratively move nodes to be with the plurality of their neighbors:
- But how to maintain balance?
- Label = machine

Modification! Figure out who wants to move. if $P_{13}$ people want to move from 1 to 3, allow only $x_{13}$ people move, s.t. flow balance.
Basic idea: Label propagation

- Iteratively move nodes to be with the plurality of their neighbors:
- But how to maintain balance?
- Label = machine

Modification! Figure out who wants to move. If $P_{13}$ people want to move from 1 to 3, allow only $x_{13}$ people move, s.t. flow balance.
Basic idea: Label propagation

- Iteratively move nodes to be with the plurality of their neighbors:
- But how to maintain balance?
- Label = machine

Modification! Figure out who wants to move. If $P_{13}$ people want to move from 1 to 3, allow only $x_{13}$ people move, s.t. flow balance.

How do we solve for $x_{ij}$?
Balance via Linear Program

- **Greedily** maximize edge locality with constraints (max/min sizes $S_i, T_i$):

  $x_{ij}$  
  $f_{ij}(x)$

  Solution: number of people to move from $i$ to $j$.

  Cumulative gain from moving $x$ people (ordered by co-location gain).

  $f_{ij}(x_{ij})$ (piecewise-linear concave!)
Balance via Linear Program

- **Greedily** maximize edge locality with constraints (max/min sizes $S_i, T_i$):

  $\max \ x \ \sum_{i,j} f_{ij}(x_{ij})$

  s.t. \begin{align*}
  S_i - |V_i| & \leq \sum_{j \neq i} (x_{ij} - x_{ji}) x_{ij} \\
  0 & \leq \frac{\sum_{j \neq i} (x_{ij} - x_{ji})}{x_{ij}} \\
  T_i - |V_i| & \leq \frac{\sum_{j \neq i} (x_{ij} - x_{ji})}{x_{ij}} \\
  0 & \leq x_{ij} \leq P_{ij}, \quad \forall i, j
  \end{align*}

  (Maximize the co-location gain of all machine swaps) (Subject to balance)

  Solution: number of people to move from $i$ to $j$. (and the number of people available to move)
Greedily maximize edge locality with constraints (max/min sizes $S_i, T_i$):

$$x_{ij}$$

$\textbf{Solution:}$ number of people to move from $i$ to $j$.

$$f_{ij}(x)$$

Cumulative gain from moving $x$ people (ordered by co-location gain).

Maximize the co-location gain of all machine swaps

\[
\max_X \sum_{i,j} f_{ij}(x_{ij}) \quad \text{s.t.} \quad \begin{cases} S_i - |V_i| & \leq \sum_{j \neq i} (x_{ij} - x_{ji}) \\ \sum_{j} x_{ij} & \leq T_i - |V_i|, \\ 0 & \leq \sum_{j} x_{ij}, \end{cases} \]

(Subject to balance)

(ordered by co-location gain)

and the number of people available to move)

Balance via Linear Program

**Linear Program:** $n=78$ machines $\Rightarrow 12k$ variables / $400k$ constraints

\[
\max_{X,Z} \sum_{i,j} z_{ij} \quad \text{s.t.} \quad \begin{cases} S_i - |V_i| & \leq \sum_{j \neq i} (x_{ij} - x_{ji}) \\ 0 & \leq \sum_{j} x_{ij} \\ -a_{ijk}x_{ij} + z_{ij} & \leq b_{ijk}, \end{cases} \]

$\forall i, j, k$
Balance via Linear Program

- Summary of algorithm:
  - Step 1: Figure out who wants to move
  - Step 2: Solve LP to decide who can move without breaking balance
  - Step 3: Move those people
Balance via Linear Program

- Summary of algorithm:
  - Step 1: Figure out who wants to move
  - Step 2: Solve LP to decide who can move without breaking balance
  - Step 3: Move those people

Step 2 is the contribution compared to ordinary Label Prop.
What about geography?
Initialization using geography

- Possible to do much better than random with Facebook, using geography.
Initialization using geography

- Possible to do much better than random with Facebook, using **geography**.
- **Spatial model** of small-world networks (for routing): Kleinberg 2000
- **Validation**: Liben-Nowell et al. 2005; Backstrom, Sun, Marlow 2010.
- Friendship probability as a function of rank-distance:

![Number of Friends at Different Ranks](image)

- Backstrom, Sun, Marlow 2010
Initialization using geography

- IP data reveals geographic location of users:
  - 1,000,000,000 users mapped to 700,000 cities
Initialization using geography

- IP data reveals geographic location of users:
  - 1,000,000,000 users mapped to 700,000 cities
- Grow equi-population balloons around population centers.
Initialization using geography

- IP data reveals geographic location of users:
  - 1,000,000,000 users mapped to 700,000 cities
- Grow equi-population balloons around population centers.
Results: Iteration convergence

- Geographic initialization ‘converges’ within 1 step

Facebook ($n=800m$, $|E|=68b$)
Results: Iteration convergence

- Geographic initialization ‘converges’ within 1 step
- Random initialization slow to start when: avg degree > # partitions
  Use ‘restraint’: only move big gainers (*s below)

Facebook ( n=800m, |E|=68b )
Results: Iteration convergence

- Geographic initialization ‘converges’ within 1 step
- Random initialization slow to start when: avg degree > # partitions
  Use ‘restraint’: only move big gainers (*s below)
- LJ partitioning quality not so dependent on # partitions:
  BLP exploiting primarily local structure.

Facebook (n=800m, |E|=68b)

LiveJournal (n=4.8m, |E|=42.8m)
Results: Machine adjacency matrix

- Random initialization + 8 step prop
- Geographic initialization ONLY
- Geographic + 1 step prop

- Targeting n=78 machines: 2 racks of 39, visible as blocks
‘People You May Know’

- PYMK = ‘People You May Know’
- Ranked suggestion of friends-of-friends (FoFs) as friends.
- Average user has 40k FoFs, widely distributed.
- Ranks **145,000,000 suggestions per second**.

- Graph distributed across 78 machines with 72GB RAM each.
‘People You May Know’

- PYMK = ‘People You May Know’
- Ranked suggestion of friends-of-friends (FoFs) as friends.
- Average user has 40k FoFs, widely distributed.
- Ranks **145,000,000 suggestions per second**.

- Graph distributed across 78 machines with 72GB RAM each.

Want to shard so that my friends on same machine as me!
Results: PYMK request concentration

- Median number of machines hit per query reduced from 60 to ?.
Results: PYMK request concentration

- Median number of machines hit per query reduced from 60 to 9.
Results: PYMK request concentration

- Median number of machines hit per query reduced from 60 to 9.

- Query time: What about overhead? Faster or slower?
Results: Query time / network traffic

- Median number of machines hit per query reduced from **60** to **9**.
- **Query time** reduced by **49%**, traffic reduced by **63%**:

![Query time graph](image1)

![Network traffic graph](image2)
Conclusions and Future work

- Label propagation is fast, we show it can be constrained
- Social networks very clustered, making local algorithms very effective
- Geographic metadata very useful

- Sharding greatly improves distributed graph computations such as PYMK