Parallelize Union Find Set

Zi Yin and Zhiang Hu (Harvy)

Dept. Electrical Engineering, Stanford University
Institute for Computational and Mathematical Engineering, Stanford University

Jun 1, 2015
Union find set: A data structure to keep disjoint subsets.
- Union find set: A data structure to keep disjoint subsets.

- Two operations: Union and Find

Union Find set
Union Find set

Two operations: Find

Parallelize Union Find Set
Union Find set

- Two operations: Find
- Two operations: Union

Acknowledgement: Picture from the Internet
Optimization: Union by rank

if \( xRoot.\text{rank} < yRoot.\text{rank} \)
   \( xRoot.\text{parent} := yRoot \)
else if \( xRoot.\text{rank} > yRoot.\text{rank} \)
   \( yRoot.\text{parent} := xRoot \)
else
   \( yRoot.\text{parent} := xRoot \)
   \( xRoot.\text{rank} := xRoot.\text{rank} + 1 \)

Acknowledgement: Algorithm from Wikipedia
Union Find set

- **Optimization: Union by rank**
  
  ```
  if xRoot.rank < yRoot.rank
      xRoot.parent := yRoot
  else if xRoot.rank > yRoot.rank
      yRoot.parent := xRoot
  else
      yRoot.parent := xRoot
      xRoot.rank := xRoot.rank + 1
  ```

- **Optimization: Path compression**
  
  ```
  if x.parent != x
      x.parent := Find(x.parent)
  return x.parent
  ```

Acknowledgement: Algorithm from Wikipedia
Optimization: Union by rank

```plaintext
if xRoot.rank < yRoot.rank
    xRoot.parent := yRoot
else if xRoot.rank > yRoot.rank
    yRoot.parent := xRoot
else
    yRoot.parent := xRoot
    xRoot.rank := xRoot.rank + 1
```

Optimization: Path compression

```plaintext
if x.parent != x
    x.parent := Find(x.parent)
    return x.parent
```

Complexity for finding connected components in graph:
Almost $O(m)$.

Acknowledgement: Algorithm from Wikipedia
How to Parallelize it

Settings:

1. $k$ machines, $m$ edges, $n$ nodes
2. $n$ fit in memory but $m$ does not
3. find connected components using union find set
How to Parallelize it

- **Settings:**
  1. $k$ machines, $m$ edges, $n$ nodes
  2. $n$ fit in memory but $m$ does not
  3. find connected components using union find set

- Find root can be done in parallel
How to Parallelize it

- Settings:
  1. $k$ machines, $m$ edges, $n$ nodes
  2. $n$ fit in memory but $m$ does not
  3. find connected components using union find set

- Find root can be done in parallel

- How about union?
Distributed Algorithm

One Iteration:
Step 1: each merge request \((u,v)\) \(r_u = \text{root}(u), r_v = \text{root}(v)\), If \(r_u \neq r_v\), emit root merge request \((r_u, r_v)\)
Step 2: construct root merging graph (directed)
Step 3: for each root \(r\):
if it has at least one outgoing edge, pick up arbitrary one \((r, r_0)\), set \(p(r)\) as \(r_0\)
emit all other unmerged edge \((r, r_1)\) ... \((r, r_k)\) as new input for Step1.
**Find Roots**
Can be done with embarasing parallel
Construct Root Merging graph:

- Reduce duplicated root merging requests into one.
  Got an undirected graph $G_u$
Construct Root Merging graph:

- Reduce duplicated root merging requests into one. Got an undirected graph $G_u$
- Count the degree of each root
Distributed Algorithm

Construct Root Merging graph:

- Reduce duplicated root merging requests into one.
  Got an undirected graph $G_u$

- Count the degree of each root

- For each edge $E=(r_i, r_j)$ suppose $deg(r_i) < deg(r_j)$ in $G_u$, set it to directed edge $< r_i, r_j >$, i.e. $r_i$ has an outgoing edge to $r_j$. 

Zi Yin and Zhiang Hu (Harvy) Parallelize Union Find Set
Merging the Roots

- Each root can change its parent pointer to at most one other root simultaneously.
Merging the Roots

- Each root can change its parent pointer to at most one other root simultaneously.

- At least half of the roots are merged to some other root.
**Scaling**
Both number of nodes and number of request can be scaled.

**Complexity**
- number of iterations: $O(\log m)$
- Time complexity: $O\left(\frac{m}{k} \log m\right)$
- Shuffle size: $O(m)$