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

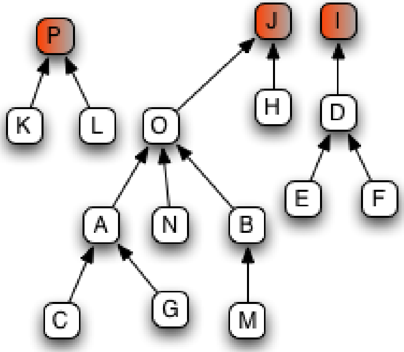
Union Find set

- Union find set: A data structure to keep disjoint subsets.

Union Find set

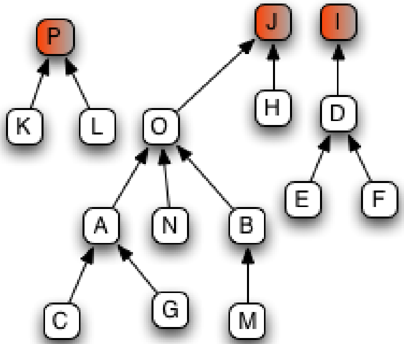
- Union find set: A data structure to keep disjoint subsets.
- Two operations: Union and Find

Union Find set



- Two operations: Find

Union Find set



- Two operations: Find
- Two operations: Union

Acknoledgement:picture from the Internet

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
```

Acknoledgement: 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
```

Acknoledgement: 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
```

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

Acknoledgement: 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 = root(u), r_v = 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 out going 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.

Distributed Algorithm

Find Roots
Can be done with embarasing parallel

Distributed Algorithm

Construct Root Merging graph:

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

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

Distributed Algorithm

Construct Root Merging graph:

- Reduce duplucated 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 out going edge to r_j.

Distributed Algorithm

Merging the Roots

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

Merging the Roots

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

- At least half of the roots are merged to some other root.

Analysis

Scaling

Both number of nodes and number of request can be scaled.

Complexity

number of iterations: O($\log m$)

Time complexity: O($\frac{m}{k} \log m$)

Shuffle size: O(m)