Fast Parallel Algorithm For Unfolding Of Communities In Large Graphs

Charith Wickramaarachchi, Marc Frincu, Patrick Small and Viktor Prasanna
University of Southern California
Los Angeles, California 90089
Outline

• Big data and graph processing
• Community detection
• Louvain method for community detection
• Our approach
• Results and discussion
• Conclusion
Big data and graph processing

• Complex Systems -> Complex data
  • Graph structured
  • Irregular memory access
  • Fast data rates
  • Large Volume

• Over 700M Daily Facebook users
• Over 1B websites
• Over 100 hours of Youtube videos uploaded per minute
• 143,199 Tweets per second
Community detection in graphs

- Approaches
  - Graph partitioning
  - Community structure detection

- Different goals
  - Graph partitioning
    - Division of tasks between processors
    - Known number of partitions/partition size
  - Community structure detection
    - Task is to find partitions of network
Community detection in graphs

- Problem: Given a graph check whether there exist any “natural division” of vertices into “non overlapping groups.
- Minimize edge cuts?
- More than expected number of edges within the group (compared to a random graph)
  - Statistically surprising arrangement of edges
Modularity

- A quantitative measure to determine quality of communities.

\[
Q = \sum_{c \in C} \left[ \frac{m_c}{M} - \frac{d_c^2}{4M^2} \right]
\]

\( Q \in [-1, 1] \)

- \( m_c \) – number of edges inside community \( c \)
- \( d_c \) – sum of degrees of vertices inside community \( c \)
- \( M \) – total number of edges
- \( C \) – set of all communities in graph
Louvain method for community detection(1)

• A greedy modularity maximization approach.
• Two main steps repeated iteratively

while(improvement) {
    detect-communities() //move one vertex at a time
    mod = modularity()
    if( (mod – prev_mod) > T)
        improvement = true
    else
        improvement = false
    prev_mod = mod
    collapse-graph() //create a new graph
}
Louvain method for community detection (2)

- Scan through all the nodes in a given order.
- Nodes adopts its neighbors community by joining to which gives a maximum increase in modularity.
- This processed repeated iteratively until local maximum modularity is reached.

Source: https://sites.google.com/site/findcommunities/
Louvain method for community detection (3)

• New network is built collapsing communities into single nodes

Source: https://sites.google.com/site/findcommunities/
Observations

- First iteration is the costliest iteration - 79.53% of total time on average
- Graph reduced to a much smaller graph after the first iteration.
- First iteration community structures are smaller - less chance for cross partition communities
Our approach(1)

• Partition the graph into N partitions
• Execute first iteration of Louvain locally in each partition
• Merge the community graph in to a single worker
• Execute rest of the iterations of Louvain
• MPI implementation
Our approach (2)
Our approach (3)

• Vertex ordering strategies
  – High degree nodes first - HDF
  – Low degree nodes first - LDF
  – High degree node neighbors first - HNF

HNF = \{3,1,5,6,8,7,2,4\}
Experimental setup

• Benchmarking conducted at USC HPCC:
  – All benchmarking executed as single batch job of 16 compute nodes w/ 8 cores per node

• Automated testing:
  – Synthetic community graphs 250K -> 16M nodes
  – MPI Task sizes from 2 -> 128 processes
  – Performed strong scaling tests – 49 configurations
Results

Runtime vs Graph size

Speedup vs number of processors
Results (2)

Error = abs($\frac{q_p - q_s}{q_s}$)

- $q_p$ = parallel modularity
- $q_s$ = sequential modularity

Runtime behavior for graph size: 1M
Results(3)

- Optimistic predicted speedup

Super linear speedup
Conclusion

• Community detection in large graphs becoming increasingly popular
• Presented a simple way to speedup Louvain algorithm by using making its first iteration embarrassingly parallel using graph partitioning.
• Minimal impact on final quality
• Evaluate/Extend for much larger real world graphs with known ground truth.
Thank you!
Backup Slides
Cost of graph partitioning

- Question: It is fair to ignore graph partitioning cost
- Answer: Yes 😊
- Why? Think of big data analytic pipeline

One time → Data loading → Multiple times 

Wait time for results → Analytics