MapReduce-based Distributed $K$-shell Decomposition for Online Social Networks

Katerina Pechlivanidou
Dimitrios Katsaros
Leandros Tassiulas

Contact Information
Katerina Pechlivanidou: kapehliv@uth.gr
Dimitrios Katsaros: dkatsar@uth.gr
Leandros Tassiulas: leandros@uth.gr
Presentation Structure

1. $K$-shell Decomposition
2. Why $k$-shell decomposition?
3. Filling the gap
4. Why Hadoop’s MapReduce?
5. Contributions
6. The MR-SD Algorithm
7. Results and evaluation
8. Future work
If from a given graph we recursively delete all vertices, and lines incident with them, of degree less than \( k \), the remaining graph is the \( k \)-core.

The terms \( k \)-core and \( k \)-shell are used interchangeably.
Why \( k \)-shell decomposition?

\( K \)-shell decomposition is appealing in Social Network Analysis because it

- reveals the hidden hierarchies in large network graphs
- highlights the core structure of network
- characterizes the network nodes beyond classic centrality measures like the degree distribution
- covers a wide range of diverse application scenarios
No suitable algorithm exist for modern datacenter environments

All existing algorithms are doomed to fail eventually due to lack of computational resources.
Why Hadoop’s MapReduce?

- Process and analysis of very large datasets
- Time performance
- Suitable for distributed and parallel algorithms
- Suitable for huge high-programmed datacenters owned by Internet giants (Google, Yahoo!, Facebook, etc.)
We developed a distributed algorithm for the computation of the $k$-shells of a given network graph.

For the first time in the literature we present a parallel and distributed algorithm for the $k$-shell decomposition based on Hadoop’s MapReduce suitable for huge datacenter environments.
Our work assesses the performance of the proposed algorithm proven with experimental results based on real datasets.

Our algorithm is suitable for static and unweighted networks, which are the most encountered and appealing in the majority of today’s online social network algorithms.

Our work analyzes various tradeoffs in our algorithm’s operation.
The MR-SD Algorithm

MapReduce-based Shell Decomposition

Contact Information
Katerina Pechlivanidou: kapehliv@uth.gr
Dimitrios Katsaros: dkatsar@uth.gr
Leandros Tassiulas: leandros@uth.gr
During each MapReduce Job the above scenarios are possible to occur:

- **Scenario 1**
  node *K* was *marked* in preceding Map task

- **Scenario 2**
  node *K* was *not marked* by previous Mapper

- **Scenario 3**
  node *K* comes coupled with *additional information* meaning that one or more neighbors are going to be deleted in this pruning round
Progress and Termination

- **Progress**
  - $\text{Cores}_k$ & $G_{\text{remaining}}$
    - Some nodes were deleted in previous pruning phase
    - Force another round with the same core value
  - Only $G_{\text{remaining}}$
    - No nodes were deleted previously
    - Increase $k$ value and force another round

- **Termination**
  - Only $\text{Cores}_k$
    - No nodes left to prune
    - $k$-cores retrieved

---

Contact Information
Katerina Pechlivanidou: kapehliv@uth.gr
Dimitrios Katsaros: dkatsar@uth.gr
Leandros Tassiulas: leandros@uth.gr
An example running of MR-SD

One (the first) pruning round of MR-SD

Contact Information
Katerina Pechlivanidou: kapehliv@uth.gr
Dimitrios Katsaros: dkatsar@uth.gr
Leandros Tassiulas: leandros@uth.gr
The evaluation platform

Our cluster consists of five nodes: One master and four slaves

Each node:

- 42GB disk space
- 12GB RAM
- 8-core Intel CPU-based blade
- CentOS
- 10-gigabit Ethernet connection

There was no significant interference from other workloads during any experiment

Contact Information

Katerina Pechlivanidou: kapehliv@uth.gr
Dimitrios Katsaros: dkatsar@uth.gr
Leandros Tassiulas: leandros@uth.gr
## The Datasets

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Social Network name</th>
<th>Number of nodes</th>
<th>Number of edges</th>
<th>Number of Jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Autonomous systems AS-733</td>
<td>6474</td>
<td>13895</td>
<td>61</td>
</tr>
<tr>
<td>2</td>
<td>DBLP collaboration network</td>
<td>317080</td>
<td>1049866</td>
<td>360</td>
</tr>
<tr>
<td>3</td>
<td>Autonomous systems by Skitter</td>
<td>1696415</td>
<td>11095298</td>
<td>1305</td>
</tr>
<tr>
<td>4</td>
<td>LiveJournal online Social Network</td>
<td>3997962</td>
<td>34681189</td>
<td>3363</td>
</tr>
<tr>
<td>5</td>
<td>Orkut online social network</td>
<td>3072441</td>
<td>117185083</td>
<td>5918</td>
</tr>
<tr>
<td>6</td>
<td>Amazon product co-purchasing network</td>
<td>334863</td>
<td>925872</td>
<td>87</td>
</tr>
<tr>
<td>7</td>
<td>Deaseasome</td>
<td>7533</td>
<td>22052</td>
<td>118</td>
</tr>
<tr>
<td>8</td>
<td>Protein Interaction Network in budding Yeast</td>
<td>2361</td>
<td>7182</td>
<td>74</td>
</tr>
</tbody>
</table>
The Performance Measures

- **Average CPU time spend (msec)**
  stands for the time spend solely by the CPU to perform the computations

- **Average Total Execution Time (sec)**
  stands for the total execution time for the $k$-shell decomposition of the input network

- **Average Total committed heap usage (Bytes)**
  it shows the amount of heap memory that is required during the experimentation phase

Contact Information

Katerina Pechlivanidou: kapehliv@uth.gr
Dimitrios Katsaros: dkatsar@uth.gr
Leandros Tassiulas: leandros@uth.gr
Experimental results showed that:

The density of the network is directly correlated to number of MR-SD rounds that are required to decompose the graph.

denser network    more rounds
Experimental results showed that:

When operating a small cluster with only few VMs then decomposing very large networks means that each node runs more than one map task resulting a greater time.
Experimental results showed that:

The workload demands for the decomposition of the network graphs require an **increasing processing power** while the heap size remains actually almost stable during this interval.

Reduce tasks tend to be **more time consuming in the first 20% to 45% of the pruning rounds** than in the last ones.
What’s next?

We plan to

- find ways to gain some additional speedup
- improve the way the tasks are distributed over the cluster
- perform experimentation on a MapReduce environment of a major cloud service provider

Contact Information

Katerina Pechlivanidou: kapehliv@uth.gr
Dimitrios Katsaros: dkatsar@uth.gr
Leandros Tassiulas: leandros@uth.gr
Thank you

Contact Information
Katerina Pechlivanidou: kapehliv@uth.gr
Dimitrios Katsaros: dkatsar@uth.gr
Leandros Tassiulas: leandros@uth.gr
MR-SD’s Complexity Analysis

Algorithm 1: computeOneHopNeighborhood(Network Graph G) A Map-Reduce pair to compute the one-hop neighborhood for each node in G

Map: omdo do
for each KV pair do
K = node
V = neighbor; collect(K, V);
end map

Reduce: omdo do
for each KV pair do
collect(K, V);
end reduce

Shuffle & Sort:

Each node has to be examined one time only and each slave reads only a chunk of the actual network graph

\[ O\left(\frac{n}{\text{#slaves}}\right) \times O\left(\max(\text{degree})\right) \]

The pruning procedure has to examine all core values

\[ k \times O\left(\frac{n}{\text{#slaves}} \times \max(\text{degree})\right) \]

Algorithm 2: Driver Routine executed to coordinate the join-decimate termination of the k-core decomposition process

on initialization do
configure(k);
G_{k+1} = computeOneHopNeighborhood(G_k);
k = 1;
end initialization

repeat until each node has k-cores
configure(k);
\textbf{if} Core not found then [k = k + 1;]
\textbf{if} Core not found then [k = k + 1;]
else [Core = merge k-cores(Core_k);
return k-cores;]
end repeat

The MR-SD algorithm: A Map-Reduce pair that implements the pruning phase of the k-shell decomposition process

Mapper:
on map do
\textbf{k} = get(k);
for each KV pair do
\textbf{if} degree = 0 then
node = mark(node);
\textbf{else}
\textbf{if} vV do 
collect(v, attach(v, K));
end collect(node, K);
end if
\textbf{if} degree > 0 then
\textbf{if} vV do 
collect(v, attach(v, K));
degree = degree - 1;
end collect(node, K);
end if
end for
end map

Reducer:
on reduce do
\textbf{k} = get(k);
for each KV pair do
\textbf{if} attach not received from Mapper then
\textbf{if} attach not received do
oneHopNeighborhood \rightarrow \textbf{V} = attach(v, K);
degree = degree - 1;
\textbf{if} degree = 0 then
mark(node);
Core_{k+1} = collect(K, V);
end if
\end if
end if
end for
end reduce

Each slave has to parse a chunk of the network graph and read only the one-hop neighborhood of each node

Contact Information

Katerina Pechlivanidou: kapehliv@uth.gr
Dimitrios Katsaros: dkatsar@uth.gr
Leandros Tassiulas: leandros@uth.gr

University of Thessaly
Some application scenarios

K-shell has been used in Social Network Analysis for

• detecting influential spreaders
• discovering communities
• the Internet structure analysis at the autonomous level
• visualization purposes

Contact Information

Katerina Pechlivanidou: kapehliv@uth.gr
Dimitrios Katsaros: dkatsar@uth.gr
Leandros Tassiulas: leandros@uth.gr
An example of $k$-shell Decomposition

Contact Information
Katerina Pechlivanidou: kapehliv@uth.gr
Dimitrios Katsaros: dkatsar@uth.gr
Leandros Tassiulas: leandros@uth.gr
Algorithm 1: computeOneHopNeighborhood(NetworkGraph G) A Map-Reduce pair to compute the one-hop neighborhood for each node in G

**Mapper:**

```plaintext
on map do
  for each KV pair do
    K <- nodeId
    V <- neighbor;
    collect(K, V);
end map
```

**Reducer:**

```plaintext
on reduce do
  for each KV pair do
    collect(K, V);
end reduce
```
Algorithm 2: Driver Routine executed to coordinate the job and detect termination of the k-core decomposition process

on initialization do
    configure(Job);
    $G_{remaining}$ ← computeOneHopNeighborhood($G_{in}$);
    $k ← 1$;
end initialization

repeat until each node has k-coreness
    configure(Job);
    $<\text{Cores}_k, G_{remaining}> ← \text{MR-SD}(G_{remaining})$;
    if $\text{Cores}_k$ not received then
        $k ← k+1$;
    if $G_{remaining}$ not received then
        $k$-cores ← mergeIntermediate($\text{Cores}_k$);
return $k$-cores;
The MR-SD Algorithm

The MR-SD algorithm: A Map-Reduce pair that implements the pruning phase of the k-shell decomposition process.

**Mapper:**
```
on map do
    k ← get(k);
    for each KV pair do
        degree ← |V|;
        if degree ≤ k then
            node ← mark(node);
            for each v ∈ V do
                collect(v, attachedInfo);
                collect(node, k);
        else
            for each v ∈ V do
                V ← V + v;
                collect(K, V);
    end map
```

**Reducer:**
```
on reduce do
    k ← get(k);
    for each KV pair do
        if attachedInfo received from Mapper then
            for each attachedInfo received do
                oneHopNeighborhood ← {V} - attachedInfo;
                degree ← |V|;
                if degree == 0 then
                    mark(node);
                    Cores_k ← collect(K,k);
                else
                    V ← oneHopNeighborhood;
                    G_remaining ← collect(K,V);
        end reduce
```
## The Results in numbers

<table>
<thead>
<tr>
<th>Dataset Description</th>
<th>Map</th>
<th>Reduce</th>
<th>Total (Job)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autonomous systems AS-733</td>
<td>2061.31</td>
<td>2715.74</td>
<td>4777.05</td>
</tr>
<tr>
<td>DBLP collaboration network</td>
<td>7696.08</td>
<td>5768.33</td>
<td>13464.42</td>
</tr>
<tr>
<td>Autonomous systems by Skitter</td>
<td>61377.72</td>
<td>61982.36</td>
<td>123360.08</td>
</tr>
<tr>
<td>LiveJournal online Social Network</td>
<td>79122.92</td>
<td>28758.76</td>
<td>107881.68</td>
</tr>
<tr>
<td>Orkut online social network</td>
<td>505243.72</td>
<td>304598.03</td>
<td>809841.74</td>
</tr>
<tr>
<td>Amazon product co-purchasing network</td>
<td>9803.56</td>
<td>6339.20</td>
<td>16142.76</td>
</tr>
<tr>
<td>Deaseasome</td>
<td>2223.05</td>
<td>2718.64</td>
<td>4941.69</td>
</tr>
<tr>
<td>Protein Interaction Network in budding Yeast</td>
<td>1559.32</td>
<td>2256.49</td>
<td>3815.81</td>
</tr>
<tr>
<td>Network Description</td>
<td>Map</td>
<td>Reduce</td>
<td>Total (Job)</td>
</tr>
<tr>
<td>--------------------------------------------------------</td>
<td>-------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>Autonomous systems AS-733</td>
<td>99.95</td>
<td>9.13</td>
<td>16.18</td>
</tr>
<tr>
<td>DBLP collaboration network</td>
<td>104.79</td>
<td>10.30</td>
<td>19.33</td>
</tr>
<tr>
<td>Autonomous systems by Skitter</td>
<td>158.55</td>
<td>71.48</td>
<td>90.83</td>
</tr>
<tr>
<td>LiveJournal online Social Network</td>
<td>159.53</td>
<td>33.63</td>
<td>56.77</td>
</tr>
<tr>
<td>Orkut online social network</td>
<td>311.50</td>
<td>299.78</td>
<td>339.79</td>
</tr>
<tr>
<td>Amazon product co-purchasing network</td>
<td>200.13</td>
<td>10.75</td>
<td>20.06</td>
</tr>
<tr>
<td>Deaseasome</td>
<td>175.80</td>
<td>8.87</td>
<td>16.26</td>
</tr>
<tr>
<td>Protein Interaction Network in budding Yeast</td>
<td>156.41</td>
<td>8.82</td>
<td>15.00</td>
</tr>
</tbody>
</table>
## The Results in numbers

### Average Total committed heap usage (Bytes)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Map</th>
<th>Reduce</th>
<th>Total (Job)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autonomous systems AS-733</td>
<td>379584512</td>
<td>189792256</td>
<td>569376768</td>
</tr>
<tr>
<td>DBLP collaboration network</td>
<td>381046693</td>
<td>192004460</td>
<td>573051153</td>
</tr>
<tr>
<td>Autonomous systems by Skitter</td>
<td>397844599</td>
<td>199193282</td>
<td>597037881</td>
</tr>
<tr>
<td>LiveJournal online Social Network</td>
<td>486663048</td>
<td>201195325</td>
<td>687858373</td>
</tr>
<tr>
<td>Orkut online social network</td>
<td>1695512575</td>
<td>157003260</td>
<td>1852515835</td>
</tr>
<tr>
<td>Amazon product co-purchasing network</td>
<td>376372495</td>
<td>191817092</td>
<td>568189587</td>
</tr>
<tr>
<td>Deaseasome</td>
<td>379584512</td>
<td>189792256</td>
<td>569376768</td>
</tr>
<tr>
<td>Protein Interaction Network in budding Yeast</td>
<td>379584512</td>
<td>189792256</td>
<td>569376768</td>
</tr>
</tbody>
</table>