External graph partitioning and clustering algorithms

Yaroslav Akhremtsev – yaroslav.akhremtsev@kit.edu
Peter Sander – sanders@kit.edu
Christian Schulz – christian.schulz@kit.edu
Problem

- **Graph partitioning and clustering** are important problems. They have wide usage for processing following graphs:
  - road networks
  - social networks
  - web graphs
  - networks stemming from finite element methods

- **Problem** - graphs do not fit into internal memory
  - Partition large graphs on expensive
  - Partition middle-sized graphs on cheap machine

- **Solution** - multi level graph partitioning
Graph partitioning and clustering are important problems. They have wide usage for processing following graphs:

- road networks
- social networks
- web graphs
- networks stemming from finite element methods

Problem - graphs do not fit into internal memory

- Partition large graphs on expensive
- Partition middle-sized graphs on cheap machine

Solution - multi level graph partitioning
Problem

- **Graph partitioning and clustering** are important problems. They have wide usage for processing following graphs:
  - road networks
  - social networks
  - web graphs
  - networks stemming from finite element methods

- **Problem** - graphs do not fit into internal memory
  - Partition large graphs on expensive
  - Partition middle-sized graphs on cheap machine

- **Solution** - multi level graph partitioning
Why are Hard Disks such slow?

Components of disk access time:
- Seek time (milliseconds, SLOW) - CPU one million times faster
- Rotational latency (milliseconds, SLOW)
- Read/write access (nanoseconds, FAST)
Computational model

- M - size of the internal memory
- B - size of the disk block
Computational model

- M - size of the internal memory
- B - size of the disk block
- $O\left(\frac{N}{B}\right) = \text{Scan}(N)$ - # of I/Os for read/write of array of size N
- $O\left(\frac{N}{B} \cdot \log_{\frac{M}{B}} \frac{N}{B}\right) = \text{Sort}(N)$ - # of I/Os for sorting array of size N.
Computational model

- M - size of the internal memory
- B - size of the disk block
- $O\left(\frac{N}{B}\right) = \text{Scan}(N)$ - # of I/Os for read/write of array of size N
- $O\left(\frac{N}{B} \cdot \log_{\frac{M}{B}} \frac{N}{B}\right) = \text{Sort}(N)$ - # of I/Os for sorting array of size N.
Computational models

We will consider two following computational models:

- Semi-external model
- External model

<table>
<thead>
<tr>
<th></th>
<th>Nodes fit into internal memory?</th>
<th>Edges fit into internal memory?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semi-external model</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>External model</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Computational models

We will consider two following computational models:

- Semi-external model
- External model

<table>
<thead>
<tr>
<th></th>
<th>Nodes fit into internal memory?</th>
<th>Edges fit into internal memory?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semi-external model</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>External model</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Graph data structure

- Array of edges

\[
\begin{array}{cccccc}
\ldots & t, w(s, t) & \ldots & \text{dummy} & \ldots & \ldots \\
\text{adjacency nodes of } s & & & \text{adjacency nodes of } s + 1 & \\
\end{array}
\]

- Array of weights. Element \(i\) contains weight of node \(i\).
Multi-level graph partitioning

Three phases in multi-level graph partitioning scheme:
- Coarsen phase
- Initial partitioning
- Refinement phase
Multi level graph partitioning
Coarsen phase

- Suppose we have some clustering $V_1, \ldots, V_k$ of graph $G$
- Build quotient graph is $Q = (V_q, E_q)$

Figure: Quotient graph
Multi level graph partitioning

Coarsen phase

- Build clustering.
- Map cluster numbers to \{0, \ldots, n\}, where n is amount of clusters. \(\text{Sort}(|V|)\) I/Os.
- Build array of edges \(E_q: (\text{cluster}[v], \text{cluster}[u], w(v, u))\). \(\text{Sort}(|E| + |V|)\) I/Os.
Multi level graph partitioning
Initial partitioning

Initial partitioner: KaHIP - Karlsruhe High Quality Partitioning

Meyerhenke, Sanders, Schulz. Partitioning Complex Networks via Size-constrained Clustering.
Multi level graph partitioning

Uncoarsen phase

Applying clustering of quotient graph $Q$ to the origin graph $G$:

$$v \in V : \text{cluster}_G[v] := \text{cluster}_Q[u], \quad v \in u, u \in V_q$$

$\text{Sort}(|V|)$ I/Os.

Figure: Applying clustering
Label propagation graph clustering

Choose cluster with most sum weights of edges.
Label propagation graph clustering

Computational complexity

- Semi-external model. Array with cluster number of each node. \textbf{Scan}(|E| + |V|) I/Os.
- External model. Use external priority queue to send adjacency cluster number. \textbf{Sort}(|E| + |V|) I/Os.
Coloring of a graph $C = \{C_1, C_2, \ldots, C_k\}$. For $C_i$ maintain a bucket of tuples with cluster info of the nodes.

Figure: Buckets
Coloring-based Graph Clustering
Bucket initialization

- Add to each edge \((v, u)\) color of \(u\): \((v, u) \rightarrow (v, u, \text{color}[u])\).
  \(\text{Sort}(\mid E \mid + \mid V \mid)\) I/Os.

- \(\forall (v, u) \in E: \text{color}[v] < \text{color}[u]\) put tuple \((v, \text{cluster}[u], u)\) for node \(v\) in bucket \(B_{\text{color}[v]}\).
  \(\text{Scan}(\mid E \mid + \mid V \mid)\) I/Os.
Coloring-based Graph Clustering

Bucket processing

- Process each bucket and calculate new cluster number.
  \[\text{Scan}(|E| + |V|)\]
- For each tuple \((v, \text{cluster}[u], u)\) in bucket \(B_i\) push tuple \((u, \text{cluster}[v], v)\) in bucket \(\text{color}[u]\). 
  \[\text{Sort}(|E| + |V|)\]
Experiments
One iteration of graph clustering algorithm

Types of algorithm:
- LP_SE - label propagation in semi-external memory model
- LP_E - label propagation in external memory model (with or without size constraints)
- BT_E - bucket clustering in external memory model
Experiments
One iteration of graph clustering algorithm

Figure: Time, sec
Experiments

One iteration of graph clustering algorithm

Figure: Memory consumption, Mb
Experiments
One iteration of graph clustering algorithm

Figure: Disk I/O, Gb
## Experiments

**Multi level algorithm**

Partition graph on 16 parts.

Internal memory for external priority queue: 1G.

2 x Intel Xeon X5550 2.66 Ghz (Quad-Core) 8 Kernels, 48G RAM, 8xSATA 1000 GB (105 MB/s R, 120 MB/s W)

<table>
<thead>
<tr>
<th>graph</th>
<th>arabic-2005</th>
<th>uk-2002</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time[s]</td>
<td>cut</td>
</tr>
<tr>
<td>algorithm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>semi-external</td>
<td>399.9</td>
<td>2.2M</td>
</tr>
<tr>
<td>external</td>
<td>2476.7</td>
<td>2.2M</td>
</tr>
<tr>
<td>KaHIP</td>
<td>334.3</td>
<td>1.79M</td>
</tr>
<tr>
<td>kMetis</td>
<td>99.6</td>
<td>3.50M</td>
</tr>
<tr>
<td>graph</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sk-2005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>algorithm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>semi-external</td>
<td>1375.4</td>
<td>20.76M</td>
</tr>
<tr>
<td>external</td>
<td>8644.1</td>
<td>20.76M</td>
</tr>
<tr>
<td>KaHIP</td>
<td>1166.4</td>
<td>18.18M</td>
</tr>
<tr>
<td>kMetis</td>
<td>405.3</td>
<td>18.56M</td>
</tr>
</tbody>
</table>

Table: Performance on huge networks for $k = 16$. 
Partition on 2 parts. Big synthetic graph: 2.147B nodes and 21.925B edges.
Size: 368 G.
Cut: 344K

<table>
<thead>
<tr>
<th>algorithm</th>
<th>time[min]</th>
<th>mem[G]</th>
</tr>
</thead>
<tbody>
<tr>
<td>semi-external</td>
<td>406.8</td>
<td>28.24</td>
</tr>
<tr>
<td>external</td>
<td>4862.4</td>
<td>9.68</td>
</tr>
</tbody>
</table>
Questions
Thank you for attention!