Diameter Approximation

We want to approximate the diameter of large graphs. The diameter is the length of the longest shortest path between any two vertices in a graph.

The diameter of this graph is determined by the red path.

One application of the diameter approximation is to select suitable algorithms for BFS computation. Knowing the diameter, the number of I/Os can be reduced in some cases (in particular for real-world graphs like web or social network graphs which usually have a small diameter).

Social networks usually have a small diameter in \(O(\log(N))\).

Some important facts:
- Computation of the exact diameter in main memory is very expensive with \(O(N^2)\).
- Approximation using a single BFS with approximation factor \(1/2\).
- The I/O complexity of external-memory BFS is \(O(N/B)\) I/Os.
- Trade-off: Diameter approximation using Parallel Cluster Growing Algorithm [1] with \(O(k^2 \cdot \text{sort}(N) + \text{sort}(N) + N \cdot \sqrt{\log(k)/(k^2 - B)})\) I/Os.

Further Improvements & Results

For the hierarchical diameter approximation, we have developed several heuristics to improve the result:
- Break ties that the path length is kept small in the shrunk graph.
- Move masters to the cluster centre to counterbalance weights.

Results

<table>
<thead>
<tr>
<th>Size [GB]</th>
<th>HIER-time</th>
<th>ratio</th>
<th>DSLB-BFS-time (single)</th>
<th>ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>web graph</td>
<td>27</td>
<td>0.6 h</td>
<td>3.3</td>
<td>5.6 h (3.2 h)</td>
</tr>
<tr>
<td>n-level graph</td>
<td>128</td>
<td>9.0 h</td>
<td>~1.0</td>
<td>56.6 h (37.3 h)</td>
</tr>
<tr>
<td>n-level graph</td>
<td>83.8</td>
<td>3.6 h</td>
<td>~1.0</td>
<td>27.8 h (19.0 h)</td>
</tr>
<tr>
<td>worse 2step</td>
<td>31.9</td>
<td>1.4 h</td>
<td>3.1</td>
<td>15.3 h (11.6 h)</td>
</tr>
</tbody>
</table>

Hierarchical Diameter Approximation

- Uses main idea from parallel cluster growing but shrinks the graph several times (in our experiment, two shrinking steps were enough even for 128 GB graph data).
- The worst case guarantee is now \(\Omega(k^{4/3-\epsilon})\) instead of \(O(k^{1/2} \log(k))\).
- Experimental results are much better than the pessimistic bound predicts [3].
- To improve our results, we developed some heuristics.

Main memory M is too small for the big data. Therefore, shrunk the data until it fits into the main memory for fast computation.

References