SIGMOD Programming Contest 2014

VIDA Team: Fernando Chirigati, Kien T. Pham, and Tuan-Anh Hoang-Vu
Supervised by Huy T. Vo
Problem

- Given a *synthetic social network*, execute a set of queries *as quickly as possible*
  - Data: LDBC Social Network Benchmark
    - Main dataset: friendship relationship (*Persons Graph*)
      - Other datasets: comments, interest tags, forums, post likes, ...
    - Queries: 4 types of query
  - Different social network sizes are tested – from 1K to 1M persons
Solution Overview

• Implementation in C++ (Standard Library and Boost)

• General optimizations
  
  • An efficient *graph encoding* to minimize dynamic allocation
  
  • A technique to execute multiple BFS concurrently in a single thread: *MS-BFS* (Multiple-Source BFS)
  
  • *Multithreading strategy* to efficiently use the available resources

• Query type-specific optimizations
  
  • *Incremental reduction* of the graph [Query Type 1]
  
  • *Precomputation* of solutions prior to query execution [Query Type 2]
  
  • *Early termination* of queries [Query Types 3 and 4]
Solution Overview

- Implementation in C++ (Standard Library and Boost)
- General optimizations
  - An efficient *graph encoding* to minimize dynamic allocation
  - A technique to execute multiple BFS concurrently in a single thread: *MS-BFS* (Multiple-Source BFS)
  - *Multithreading strategy* to efficiently use the available resources
- Query type-specific optimizations
  - *Incremental reduction* of the graph [Query Type 1]
  - *Precomputation* of solutions prior to query execution [Query Type 2]
  - *Early termination* of queries [Query Types 3 and 4]
Graph Encoding

- Use of *adjacency list*
- Implementation avoids dynamic allocations
MS-BFS

- Stands for *Multiple-Source BFS*
- General idea
  - MS-BFS can perform 64 BFS concurrently
  - There is no need for locking or multiple threads
  - MS-BFS updates queue and visited vertices using bit masks and efficient bit operations
  - Vertices can be *shared* and explored only once for multiple concurrent BFS
MS-BFS

Vertices

BFS

Q₀

V
MS-BFS

$Hop = 0$

\[
\begin{array}{c|c}
0 & 1 \\
\hline
0 & x \\
1 & x \\
2 & \\
3 & \\
\end{array}
\]

$Q_0$

\[
\begin{array}{c|c}
0 & 1 \\
\hline
0 & x \\
1 & x \\
2 & \\
3 & \\
\end{array}
\]

$V$
MS-BFS

Hop = 0

0 1

0 1 2 3

V

Q_0

Hop = 1

0 1

0 1 2 3

V

Q_1
MS-BFS

Hop = 0

Hop = 1
MS-BFS

Hop = 0

\[
\begin{array}{cc}
0 & 1 \\
0 & X \\
1 & X \\
2 & \\
3 & \\
\end{array}
\]

Q_0

Hop = 1

\[
\begin{array}{cc}
0 & 1 \\
0 & \\
1 & X \\
2 & X \\
3 & \\
\end{array}
\]

Q_1

Hop = 2

\[
\begin{array}{cc}
0 & 1 \\
0 & X \\
1 & X \\
2 & X \\
3 & X \\
\end{array}
\]

Q_2
MS-BFS

- **Hop = 0**
  - $Q_0$
  - $V$

- **Hop = 1**
  - $Q_1$

- **Hop = 2**
  - $Q_2$

**Vertex 2 is being explored only once!**
MS-BFS

\[ \begin{align*}
\text{Hop} = 0 & \quad \text{Hop} = 1 & \quad \text{Hop} = 2 \\
0 \quad 1 & \quad 0 \quad 1 & \quad 0 \quad 1 \\
\begin{array}{cc}
0 & X \\
1 & X \\
2 & \\
3 & \\
\end{array} & \begin{array}{cc}
0 & \\
1 & \\
2 & X \times \\
3 & X \\
\end{array} & \begin{array}{cc}
0 & \\
1 & X \\
2 & \\
3 & X \\
\end{array}
\end{align*} \]
**MS-BFS**

**Hop = 0**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Q₀**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**V**

**Hop = 1**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

**Q₁**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

**V**

**Hop = 2**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

**Q₂**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

**V**

**Bit Operations:**

\[ Q_h[v] = Q_{h-1}[u] \& \sim V[v] \]

\[ V[u] = Q_h[v] \]
Multithreading Strategy

- I/O for Query Type 1 is a bottleneck
- Strategy provides an efficient use of resources
Solution Overview

- Implementation in C++ (Standard Library and Boost)
- General optimizations
  - An efficient *graph encoding* to minimize dynamic allocation
  - A technique to execute multiple BFS concurrently in a single thread: *MS-BFS* (Multiple-Source BFS)
  - *Multithreading strategy* to efficiently use the available resources
- Query type-specific optimizations
  - *Incremental reduction* of the graph [Query Type 1]
  - *Precomputation* of solutions prior to query execution [Query Type 2]
  - *Early termination* of queries [Query Types 3 and 4]
**Query Type 1**

\[ \text{query1}(P_1, P_2, x) \] – *Find the shortest path between persons* \( P_1 \) *and* \( P_2 \) *in Persons Graph where all persons have made more than* \( x \) *comments to each other*

- Add number of comments in *Persons Graph*
Query Type 1

\texttt{query1(P_1,P_2,x)} – Find the shortest path between persons $P_1$ and $P_2$ in Persons Graph where all persons have made more than $x$ comments to each other

- Queries are grouped by $x$ and graph is \textit{incrementally reduced}
Query Type 2

query2(k, d) – Find top k interest tags with largest communities of people that know each other and who were born on date d or later

- Precomputation: size of connected components for each interest tag ordered by birthdate
- Use binary search to get the size of the largest component given birthdate $d$

![Diagram showing the query and pre-computation process]
Query Type 3

`query3(k, h, p)` – *Find top k pairs of persons with respect to number of common interest tags; maximum number of hops between persons in Persons Graph is h; pair must be located in p, or study or work in organizations located in p*

- Co-located persons are sorted by number of interest tags
- BFS is executed in *Persons Graph* for each of these persons
- *Early termination*
  - Stop query execution when number of tags of upcoming person is less than the current minimum of top k
**Query Type 4**

**query4(k,t)** – *Find top k persons with highest closeness centrality values in Persons Graph where all persons are members of forums with interest tag t*

- Closeness centrality:
  \[
  cc(P) = \frac{(r(P) - 1) \times (r(P) - 1)}{(n - 1) \times (s(P))}
  \]
- Persons who are not members in these forums are removed from *Persons Graph*
- Persons are sorted by degree
  - BFS is executed for each person
  - *Early termination*
    - Stop BFS when current accumulated \( s(P) \) is greater than the current maximum of top k
Further Optimizations

- We sacrifice memory to boost performance
- Some data structures are shared across different query types
  - E.g.: *Persons Graph, Persons* and *Tag* information, ...
- Persons ids are normalized
- Vectors and arrays are used instead of maps
- Size of data structures are estimated based on file size
- Repeated queries are executed only once
- Memory mapped files (from Boost) are used to improve I/O performance
Statistics

- 2,556 lines of code
- 255 submissions / 45 days (01/03 – 04/14): around 5 submissions / day
- 39 failed submissions; 196 passed submissions
Thank You!

Questions?

VIDA Team: Fernando Chirigati, Kien T. Pham, and Tuan-Anh Hoang-Vu
Supervised by Huy T. Vo