Efficient external-memory bisimulation on DAGs

Jelle Hellings

Database and Theoretical Computer Sciences Research Group
Hasselt University

Work performed at the Eindhoven University of Technology
Supervised by George Fletcher and Herman Haverkort

November 29, 2011
Bisimulation

External memory

Partitioning

Results
Example: Querying XML data

Query 1 does a path root/a/b exist?
Query 2 give all nodes reachable by path root/a/b.
Graph index

Query 1 does a path root/a/b exist?
Query 2 give all nodes reachable by path root/a/b.
**Bisimulation**

**Definition**
Node $n$ bisimulates node $m$; denoted as $n \approx m$; if and only if:

1. The nodes have the same label,
2. Every child $n'$ of $n$ is bisimulated by a child $m'$ of $m$, and
3. Every child $m'$ of $m$ is bisimulated by a child $n'$ of $n$. 

![Graph showing bisimulation relationships]
Bisimulation partitioning

- Already very well studied
- Fast general algorithm by Paige and Tarjan
  - Runtime complexity: $O(|E| \log(|N|))$
  - Memory usage: $O(|N| + |E|)$
Internal vs External

Cost for reading or writing data from hard disk drive:

Seek move to the correct position on the hard disk drive

\[ \text{Slow: 15ms} \]

Transfer from the correct position read or write the data

\[ \text{‘Fast’, small blocks in less than 1μs} \]

Goal: minimize seeks by transferring large blocks
Challenge: graphs

Bisimulation

External memory

Partitioning

Results

Paige and Tarjan: many edge traversals
Challenge: graphs

Paige and Tarjan: many edge traversals
Input layout

- Nodes have rank
  Nodes are topologically ordered
- Nodes have identifier
- Nodes have list of identifiers of parents
- Nodes are ordered on identifier and rank
Algorithm - sketch

**Input:** Directed acyclic graph $G = \langle N, E, l \rangle$.

**Output:** Bisimulation partition of $N$.

1. **for all** increasing rank $r$; starting at $r = 0$ **do**
2. Determine bisimilarity decision value of nodes with rank $r$
3. Group on equivalent bisimilarity decision values
4. Assign bisimilarity identifiers to each group
5. Send bisimilarity identifiers to parent nodes
Bisimilarity decision value

- Every equivalence class: bisimilarity identifier
- Nodes are bisimilar equivalent iff they have
  - The same label
  - Same set of bisimilarity identifiers of children
Algorithm - sketch

**Input:** Directed acyclic graph $G = \langle N, E, l \rangle$.

**Output:** Bisimulation partition of $N$.

1. for all increasing rank $r$; starting at $r = 0$ do
2. Determine bisimilarity decision value of nodes with rank $r$
3. Group on equivalent bisimilarity decision values
4. Assign bisimilarity identifiers to each group
5. Send bisimilarity identifiers to parent nodes
Grouping on bisimilarity decision values

Assume: set of bisimilarity identifiers of children are ordered
- Bisimilarity decision values are ‘strings’
- External memory string sorting
Algorithm - sketch

**Input:** Directed acyclic graph $G = \langle N, E, l \rangle$.

**Output:** Bisimulation partition of $N$.

1. **for all** increasing rank $r$; starting at $r = 0$ do
2. Determine bisimilarity decision value of nodes with rank $r$
3. Group on equivalent bisimilarity decision values
4. Assign bisimilarity identifiers to each group
5. Send bisimilarity identifiers to parent nodes
Send bisimilarity identifiers to parent nodes

Time-forward processing:

- External memory priority queue
- Ordered on (node identifier, bisimilarity identifier)
- Process nodes in order of their node identifier
Bisimulation partitioning

- General purpose algorithm
- Topological ordered directed acyclic graphics
- Worst case complexity $O(Sort(|N| + |E|))$
Bisimulation partitioning

- General purpose algorithm
- Topological ordered directed acyclic graphics
- Worst case complexity $O(\text{Sort}(|N| + |E|))$
- Implementation in C++ on top of STXXL
  - Expected IO complexity of $O(\text{Sort}(|N| + |E|))$
Runtime performance

Running time [s] vs. Nodes
IO Complexity

- Nodes vs. IOs per node
- The graph shows the IO complexity in terms of IOs per node as a function of the number of nodes.
- The x-axis represents the number of nodes, ranging from 0 to 1.1.
- The y-axis represents the IOs per node, ranging from $10^{-4}$ to 100.
- The data points indicate an increasing trend in IO complexity with the number of nodes.
Application: XML index construction

- 1-Index in $O(\text{Sort}(|N|))$
- A(k)-Index in $O(\text{Sort}(k|N|))$
- F&B-Index by using general algorithm
Future Work

- Cyclic graph partitioning
- Incremental partition maintenance
- Applications in query processing
Conclusion

- Bisimulation partitioning
- External memory
- Applications for XML indexing
<table>
<thead>
<tr>
<th>Bisimulation</th>
<th>External memory</th>
<th>Partitioning</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>