Efficient external-memory bisimulation on DAGs

Jelle Hellings

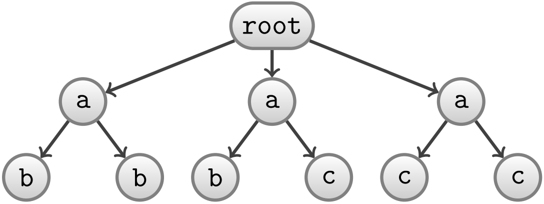
Database and Theoretical Computer Sciences Research Group
Hasselt University

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

November 29, 2011

Example: Querying XML data

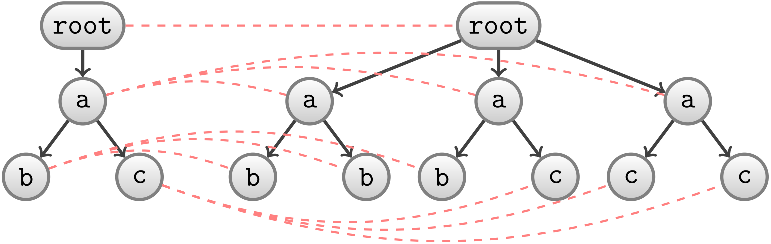
```
<root>
   <a>
      <b/> <b/>
   </a>
   <a>
      <b/> <c/>
   </a>
   <a>
      <c/> <c/>
   </a>
</root>
```



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?

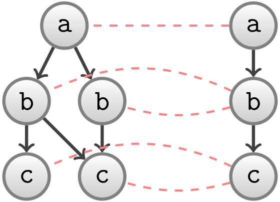
Query 2 give all nodes reachable by path `root/a/b`.

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.



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

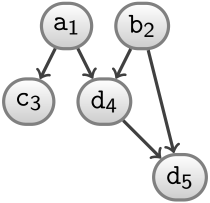
Cost for reading or writing data from hard disk drive:

Seek move to the correct position on the hard disk drive
Slow: 15ms

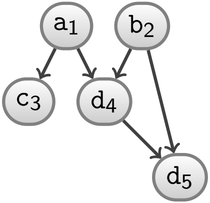
Transfer from the correct position read or write the data
'Fast', small blocks in less than 1μs

Goal: minimize seeks by transferring large blocks

Challenge: graphs



Challenge: graphs



Paige and Tarjan: many edge traversals

Input layout

- Nodes have rank
  Nodes are topologically ordered
- Nodes have identifier
- Nodes has 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.

```
for all increasing rank r; starting at r = 0 do
    Determine bisimilarity decision value of nodes with rank r
    Group on equivalent bisimilarity decision values
    Assign bisimilarity identifiers to each group
    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.

for all increasing rank r; starting at $r = 0$ **do**
    Determine bisimilarity decision value of nodes with rank r
    Group on equivalent bisimilarity decision values
    Assign bisimilarity identifiers to each group
    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

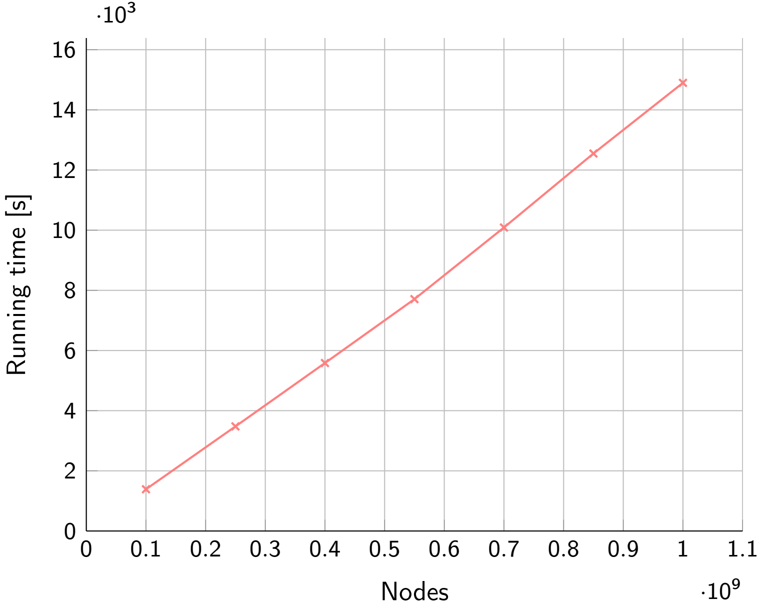
Bisimulation partitioning

- General purpose algorithm
- Topological ordered directed acyclic graphics
- Worst case complexity $O(\mathrm{Sort}(|N| + |E|))$

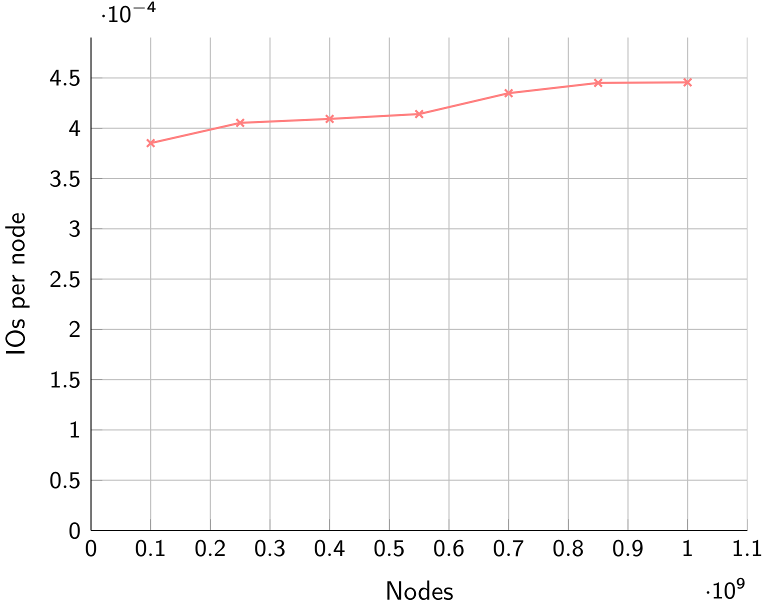
Bisimulation partitioning

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

Runtime performance



IO Complexity



Application: XML index construction

- 1-Index in $O(\mathrm{Sort}(|N|))$
- A(k)-Index in $O(\mathrm{Sort}(k|N|))$
- 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