FlexBench: A Flexible XML Query Benchmark

Maroš Vranec

Irena Mlýnková

Department of Software Engineering
Faculty of Mathematics and Physics
Charles University
Prague, Czech Republic

maros.vranec@gmail.com
mlynkova@ksi.mff.cuni.cz

Introduction

- XML = a standard for data representation and manipulation ⇒ huge amount of XMLMSs
  - User: Which XMLMS is most sufficient for my application?
  - Vendor: I need to test correctness and efficiency of my application.
  - Analyst: I need to test and analyze various applications from various points of view.

  ⇒ Solution: benchmarking
- Benchmark/test suite = set of testing scenarios/test cases = data + operations + metrics
  - Aim: compare versatility, efficiency or behavior of SUT
  - XMLMS:
    - Data =XML documents (+ XML schema)
    - Operations = XML queries (updates, transformations, …)

Related Work

- Existing benchmarks: XMark, XOO7, XMach-1, MBench, XBench, XPathMark, TPoX
  - Application-level vs. micro (MBench)
  - Number of users (> 1 ... XMach-1, TPoX), applications (> 1 ... XBench), ...
  - Characteristics of data generator
    - Size of the data
  - Operation set
    - Queries, updates (XMach-1, MBench, TPoX), less XML-like operations (XMach-1, TPoX), ...
- Why do we need another one?

Motivation

- Problem: in all cases the sets of data and operations are fixed
  - Data characteristic: size (trivially solved)
  - Advantage: a benchmark should be simple
    - XMark – most popular
  - Disadvantage: we test only a specific XML application
    - Basic testing: sufficient
    - Real-world data: various types of applications

⇒ FlexBench = flexible benchmark

  - Support of huge amount of characteristics
  - Preservation of simplicity

Discussion of Solution

- We do not want to fix anything, we want to synthesize
  - User provides characteristics of data/operations
- Possible approaches:
  1. data → schema → queries
  2. schema → data → queries
  3. queries → schema → data
- Existing benchmarks: schema + queries are fixed, data are "synthesized"
  - Size of data ⇒ modified
- FlexBench: approach 1
  - data → schema → queries
    - Data generator → schema generator → query generator
  - Statistical analyses: describe mostly data, schemas are usually missing

Data Characteristics (1)

!!

Type	Parameter	Conflict with	Data type
Basic	Output directory	None	Constant
	Number of documents	None	Constant
Structural	Size (in Bytes)	Number of elements, fan-out, depth, percentage of text, attribute Values	Statistical distribution
	Fan-out	Depth, size	Statistical distribution
	Depth	Fan-out, size	Statistical distribution
	Number of attributes	Size, percentage of text	Statistical distribution
Textual	Percentage of text	Size	Constant
	Percentage of mixed-content elements	Percentage of text	Constant
	Depth of mixed-content	Depth	Statistical distribution
	Percentage of simple mixed-content elements	Percentage of text	Constant
...			

Data Characteristics (2)

Type	Parameter	Conflict with	Data type
…			
Patterns	Percentage of pure recursions	Other recursions	Constant
	Percentage of trivial recursion	Other recursions	Constant
	Percentage of linear recursion	Other recursions	Constant
	Percentage of general recursion	Other recursions	Constant
	Percentage of DNA patterns	None	Constant
	Percentage of relational patterns	None	Constant
	Percentage of shallow relational patterns	None	Constant
Schema	Percentage of DTDs	None	Constant
	Percentage of XSDs	None	Constant

Schema Generator

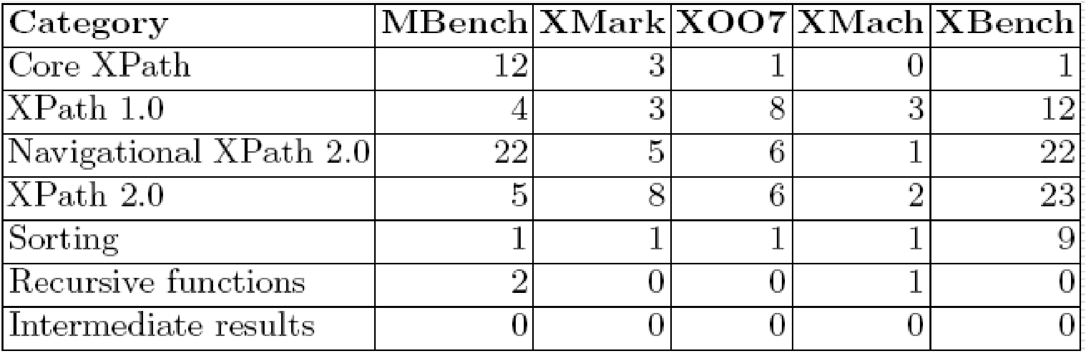
- Motivation: XML schema inference
- Fact: the data are described precisely ⇒ synthesised precisely
  - ⇒ we do not need other schema characteristics
  - ⇒ we can infer the schema from data automatically
- We exploit a third-party implementation

Query Generator

```
for $a in doc("input.xml")//elem
order by $a
return <result>{$a}</result>
```

- Aim: to provide a set of queries over the synthesised data
- Idea: a set of XQuery templates
  - Filled in with document + element/attribute names
- Problem: Which elements/attributes should be used in the templates?
- Possibilities:
  - All possible
    - Too many options vs. analysis of all cases
  - Interesting ones: Mixed-content elements, recursive elements, most common element, elements at particular levels, ...
  - Selected elements
    - User must know the data

Types of Queries



Category	MBench	XMark	XOO7	XMach	XBench
Core XPath	12	3	1	0	1
XPath 1.0	4	3	8	3	12
Navigational XPath 2.0	22	5	6	1	22
XPath 2.0	5	8	6	2	23
Sorting	1	1	1	1	9
Recursive functions	2	0	0	1	0
Intermediate results	0	0	0	0	0

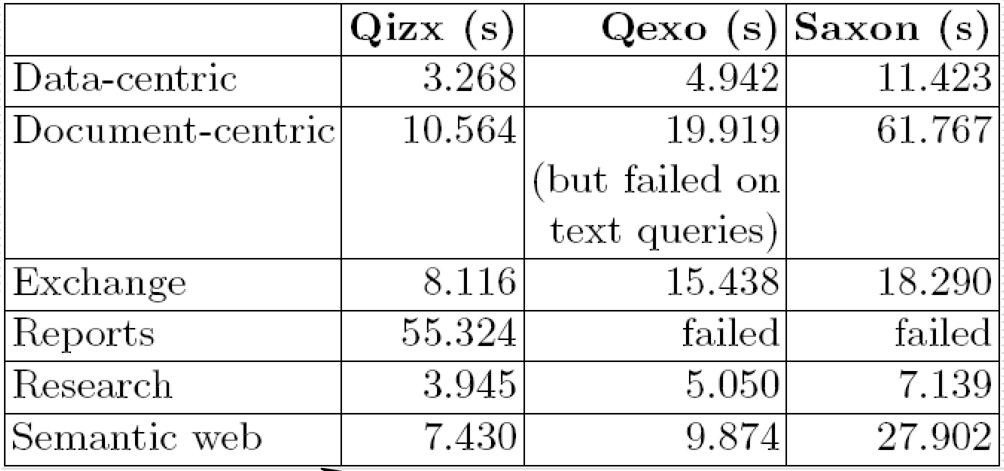
1. Core XPath Queries
   - Navigational part of XPath
2. Text Queries
   - Test preserving the order of a text
3. XPath 1.0 Queries
   - Absolute and relative order of elements
4. Navigational XPath 2.0 Queries
   - XPath 2.0 + some and every
5. XPath 2.0 Queries
   - Position information, aggregation and arithmetic functions
6. Sorting Queries
7. Queries with Recursive Functions
8. Queries with Intermediate Results

⇒ Cover all the existing benchmarks

Pre-Defined Sets of Parameters

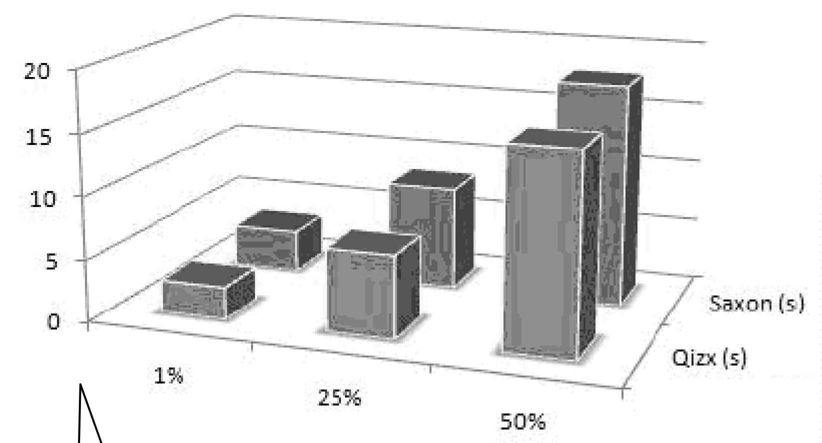
- Key requirement for a benchmark: simplicity
- FlexBench: huge amount of parameters
  - Unfriendly for most users
- Solution: pre-defined settings
  - Analysis of real-world XML data ⇒ realistic settings
- Categories of data:
  - data-centric
  - document-centric
  - exchange
  - report
  - research
  - semantic web

Preliminary Results



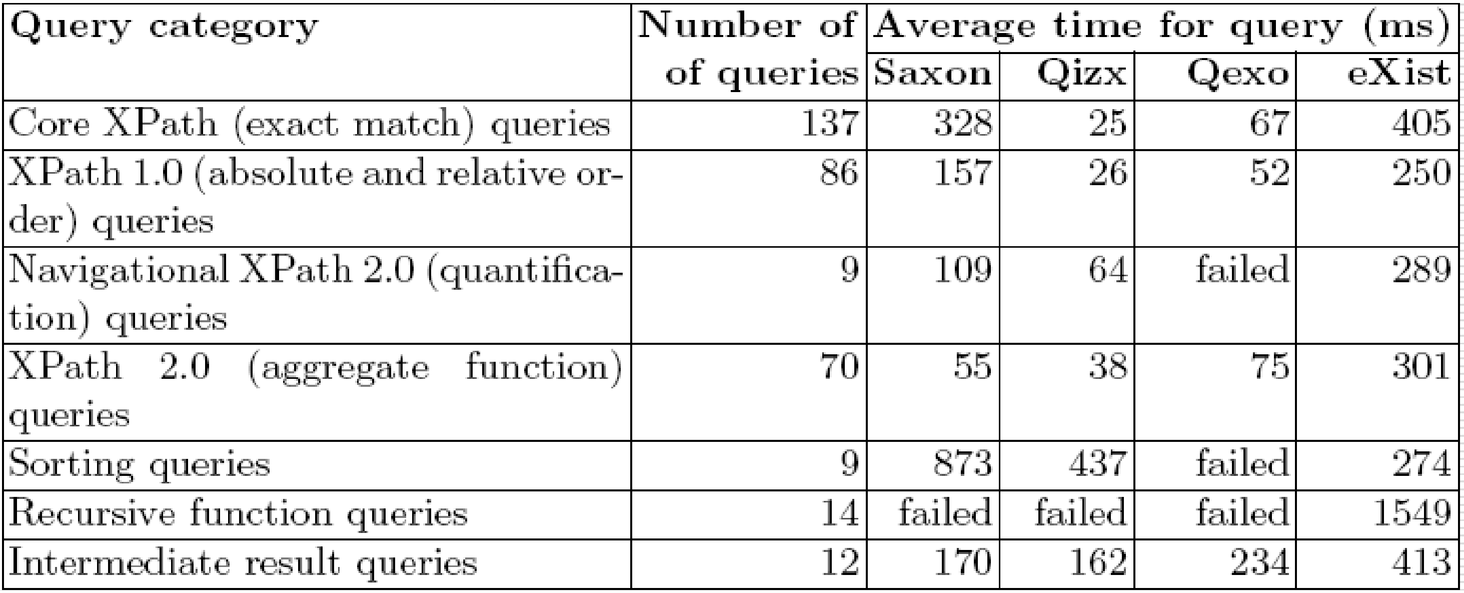
	Qizx (s)	Qexo (s)	Saxon (s)
Data-centric	3.268	4.942	11.423
Document-centric	10.564	19.919 (but failed on text queries)	61.767
Exchange	8.116	15.438	18.290
Reports	55.324	failed	failed
Research	3.945	5.050	7.139
Semantic web	7.430	9.874	27.902

Total execution time



Influence of % of recursion on text queries

Document-centric category



Query category	Number of of queries	Average time for query (ms)			
		Saxon	Qizx	Qexo	eXist
Core XPath (exact match) queries	137	328	25	67	405
XPath 1.0 (absolute and relative order) queries	86	157	26	52	250
Navigational XPath 2.0 (quantification) queries	9	109	64	failed	289
XPath 2.0 (aggregate function) queries	70	55	38	75	301
Sorting queries	9	873	437	failed	274
Recursive function queries	14	failed	failed	failed	1549
Intermediate result queries	12	170	162	234	413

Conclusion

- Achievements:
  - Benchmark with huge number of parameters = multiple applications
  - Query templates cover all the existing benchmarks
  - Analysis of real-world XML data: data characteristics + pre-defined settings ⇒ realistic
- Current aim:
  - More elaborate experiments
  - More user-friendly implementation
  - Repository of pre-defined settings
- Open issues and future work (currently: only simple, straightforward solutions):
  - More sophisticated data generator
  - More complex query templates
  - Better approaches to filtering of the generated queries

Thank you