1st International Workshop on
Performance and Evaluation of Data Management Systems
EXPDB 2006, June 30

Pushing XPath Accelerator to its Limits

Christian Grün, Marc Kramis
Alexander Holupirek, Marc H. Scholl, Marcel Waldvogel

Department of Computer and Information Science
University of Konstanz

Overview

…processing large XML documents

…our two prototypes

…a benchmark framework

…performance results

…what we will do next

Motivation

Observation

- sizes of XML instances are continuously growing:
  - Library Data, U Konstanz: 2 GB
  - DBLP: 300 MB
  - Wikipedia: 5 GB up to 500 GB
  - Log files > 10 GB …

Fact

- XML processors needed to handle these documents
- current XML processors usually fail:
  - by design (on-the-fly parsing, 2GB limit, indexing overhead, …)
  - by technical limits (main memory barrier, swapping, …)

Motivation

MonetDB/XQuery

- based on the Pathfinder project, developed in Konstanz
- XPath Accelerator: relational XML encoding
- StairCase Join: very efficient path traversal
- Loop-Lifting: linear execution of nested loops

Identified Bottlenecks (Challenges…)

- main memory limitation
- no content/value indexes

Motivation

Two Approaches

BaseX

shrink main memory representation

→ pure main memory processing

→ compressed representation
of XPath Accelerator encoding

→ introduction of an inherent
value index

Idefix

optimize disk layout

persistent native XML storage ←

constant scalability ←

logarithmic updateability ←

BaseX – Memory Architecture

Node Table Representation

Pre	Par	Tag	Content	Kind	AttName	AttVal
1	0	db		elem		
2	1	address		elem	id	add0
3	2	name		elem	title	Prof.
4	3		Hack Hacklinson	text		
5	2	street		elem		
6	3		Alley Road 43	text		
7	2	city		elem		
8	3		Chicago, IL 60611	text		
9	1	address		elem	id	add1
10	9	name		elem		
11	10		Jack Johnson	text		
12	9	street		elem		
13	10		Pick St. 43	text		
14	9	city		elem		
15	10		Phoenix, AZ 85043	text		

Parent (32 bit)	Kind/Token (1/31 bit)	Attributes (10/22 bit)
...0000	0.....0000	nil
...0001	0.....0001	0000...0000
...0010	0.....0010	0001...0001
...0011	1.....0000	nil
...0010	0.....0011	nil
...0011	1.....0001	nil
...0010	0.....0100	nil
...0011	1.....0010	nil
...0001	0.....0001	0000...0010
...0010	0.....0010	nil
...	... ...	...

- ⇨ index storage
- ⇨ numeric references

ID	Tag
0000	db
0001	address
0010	name
0011	street
0100	city

ID	Text
0000	Hack Hacklinson
0001	Alley Road 43
0010	Chicago, IL 60611
0011	Jack Johnson
0100	Pick St. 43
0101	Phoenix, AZ 85043

ID	AttName
0000	id
0001	title

ID	AttValue
0000	add0
0001	Prof.
0010	add1

BaseX – Querying

Value Indexing

- Text and AttributeValue indexes are extended by references to Pre values (⇨ inverted index)
- small memory overhead (12 – 18%)

Query Optimization:

- predicates are evaluated first (selection pushdown)
- internal `index` axis and `cs()` kind test are added for predicate evaluation
- queries are inverted & rewritten

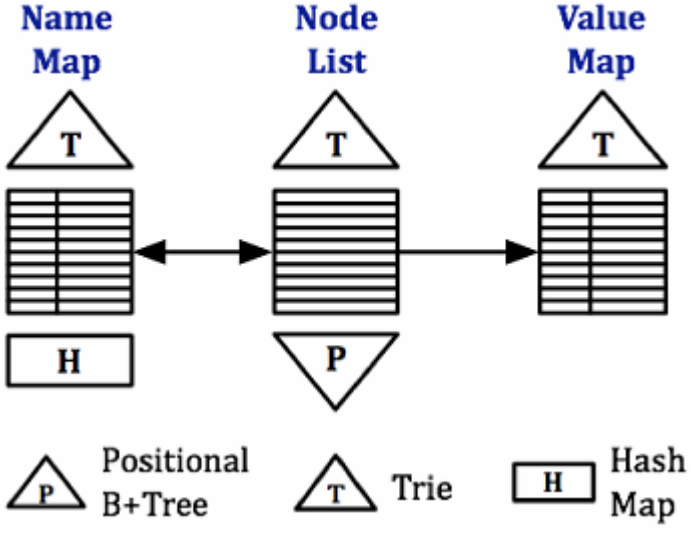
Example:

```
/db/address[@id = "add0"]/name
```

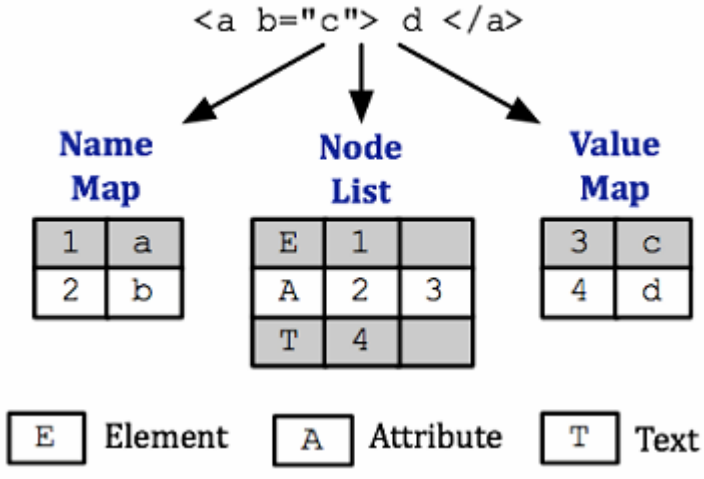
⇨ `index::node()[@id = "add0"]/parent::address[parent::db/parent::cs()]/child::name`

Idefix – Data Structures

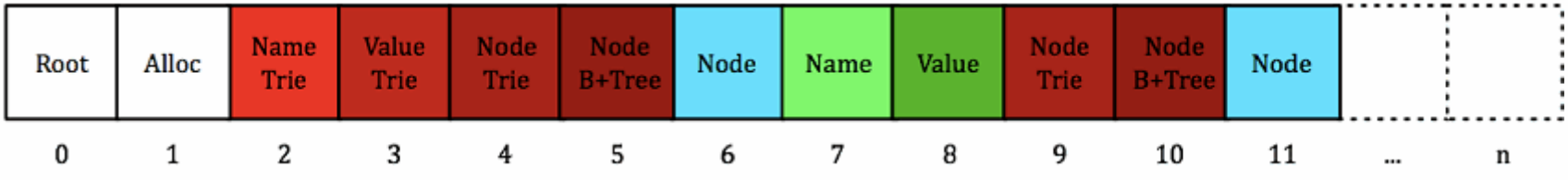
Concept



Shredding



Block Storage



Perfidix – Java Benchmarking Framework

Task

- automate tedious manual benchmarking tasks
  - generic
  - à la JUnit
  - integration (Eclipse, Ant, …)

Output

- console or XML per benchmark (*n* runs)
  - minimum, maximum, average, standard deviation, confidence interval 95

Discussion

- Java memory management
- benchmark history

Perfidix – Java Benchmarking Framework (cont.)

Code Example

```
public class DemoBench extends Benchmarkable {
        public DemoBench() {...}        // one-time initialization
        public setUp() {...}            // per-run & method preparation
        public tearDown() {...}         // per-run & method cleanup
        public benchFoo() {...}         // method Foo to bench
        public benchBar() {...}         // method Bar to bench
}
```

Output Example

```
=============================================================================================================
| -        | unit |      sum |     min |     max |        avg |     stddev |         conf95          | runs |
=============================================================================================================
| benchFoo |   ns |  8023000 |   19000 | 3822000 |   80230.00 |  376167.54 |   [6501.16,153958.84]   |  100 |
| benchBar |   ns |  3951000 |   15000 |  778000 |   39510.00 |   74585.05 |   [24891.33,54128.67]   |  100 |
| _____________________________________________________________________________________________________|
| TOTAL    |   ns | 11974000 | 3951000 | 8023000 | 5987000.00 | 2036000.00 | [3165247.96,8808752.04] |      |
=============================================================================================================
```

Evaluation

Systems

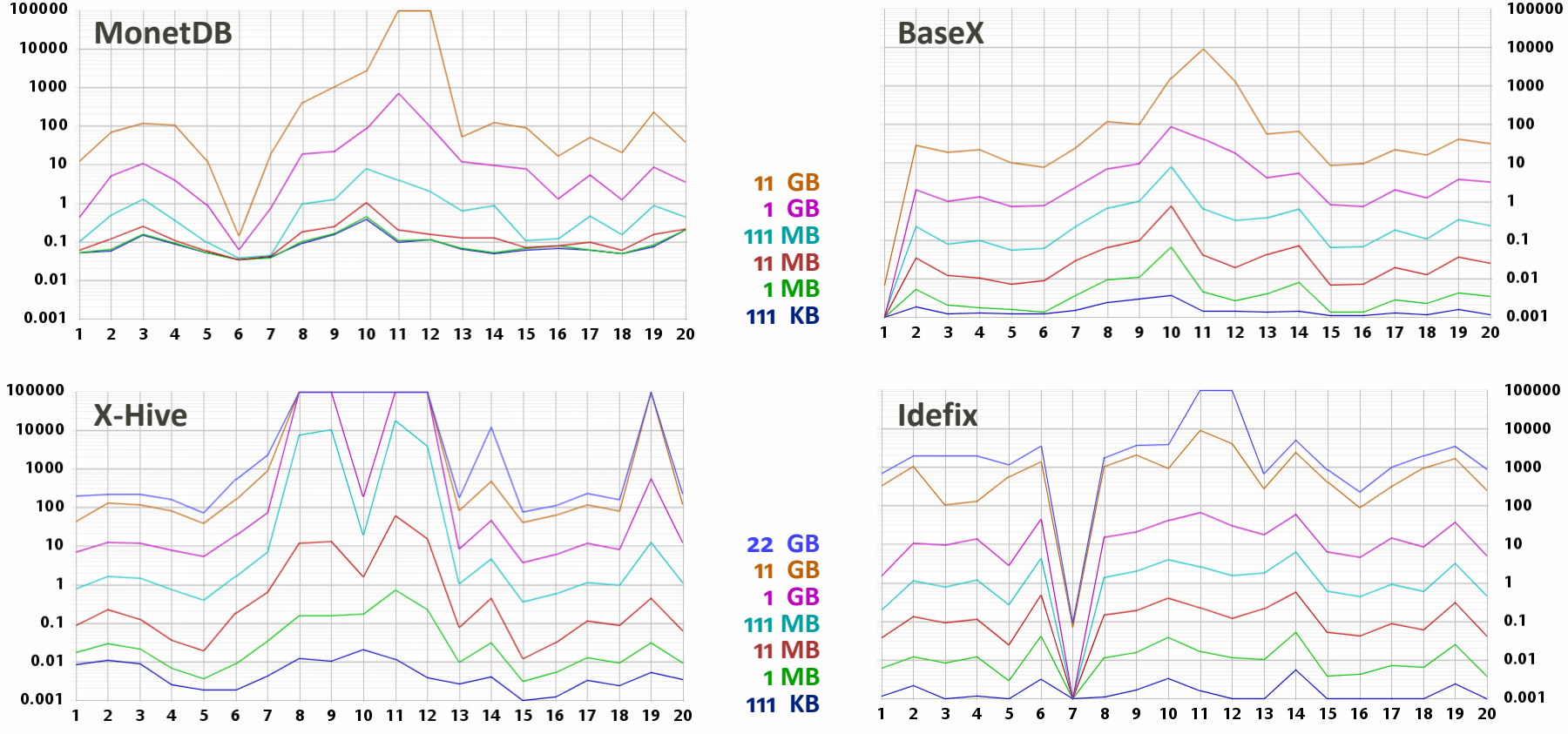
- MonetDB & BaseX
  - main memory based processing
  - similar data structures
- X-Hive & Idefix
  - persistent disk storage
  - comparable scalability

Benchmark Queries

- XMark, 110 KB – 22 GB
- six value-oriented DBLP Queries, 300 MB

Evaluation – Scalability

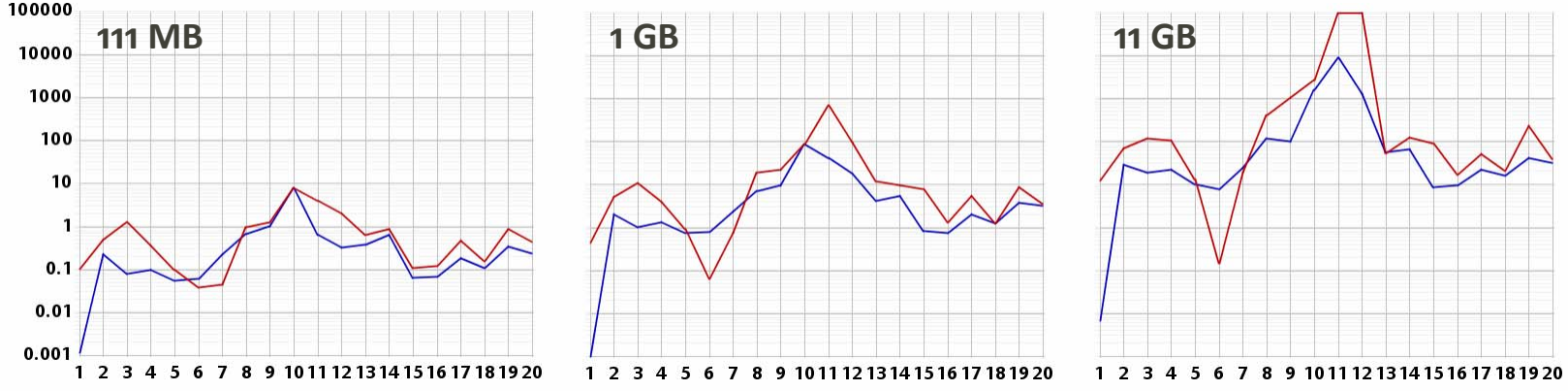
XMark queries (x-axis ⇨ number of query, y-axis ⇨ execution time in sec.)



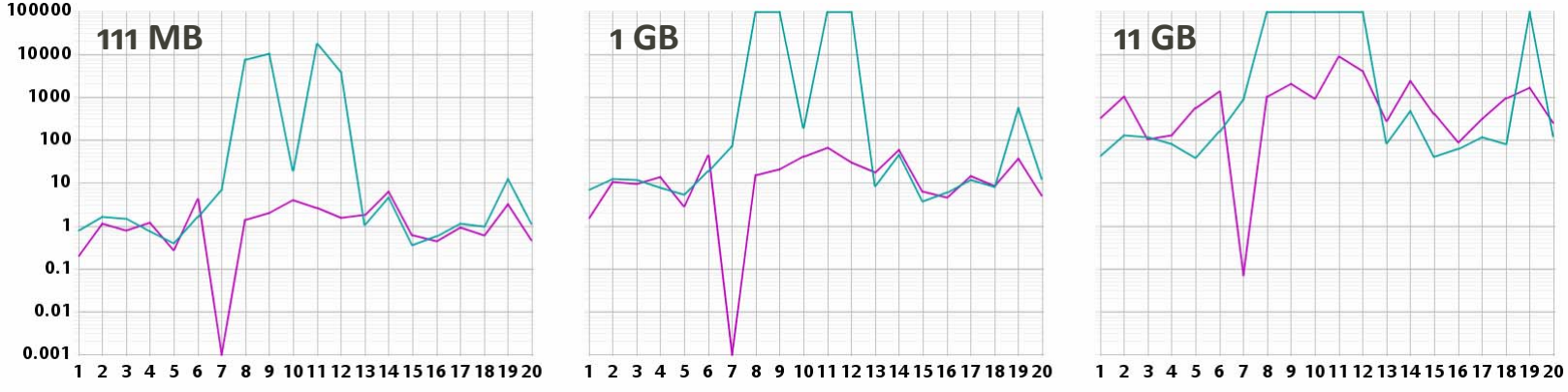
Evaluation – XMark

XMark queries (x-axis ⇨ number of query, y-axis ⇨ execution time in sec.)

MonetDB
BaseX



X-Hive
Idefix



Evaluation – DBLP

DBLP queries (x-axis: number of query, y-axis: execution time in sec.)

contains() function:

[1] /dblp/*[contains(title, 'XPath')]

range query:

[2] /dblp/*[year/text() < 1940]/title

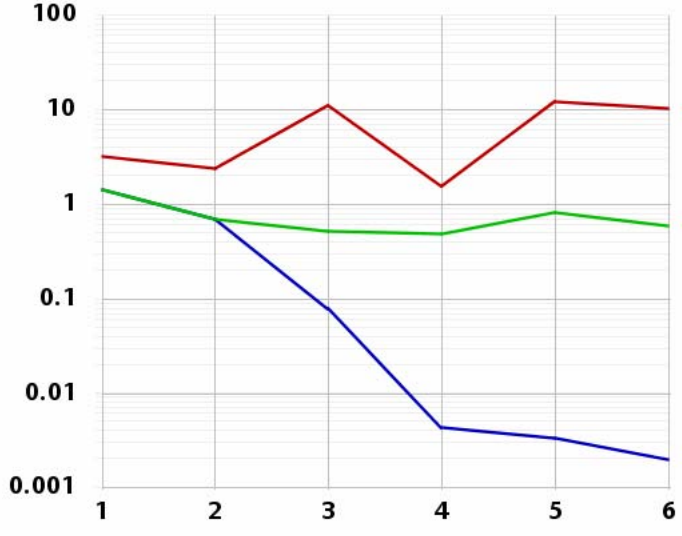
exact predicate match:

[3] /dblp//inproceedings[contains(@key, '/edbt/')]
[year/text() = 2004]

[4] /dblp/article[author/text() = 'Alan M. Turing']

[5] //inproceedings[author/text() = 'Jim Gray']/title

[6] //article[author/text() = 'Donald D. Chamberlin']
[contains(title, 'XQuery')]



MonetDB
BaseX (no index)
BaseX (with value index)

Lessons Learned

- hard-coded queries might blur evaluation results
- comparison troublesome with different systems
  - granularity of measurements (shredding, compilation, serialization, …)
  - impact of different system components (storage, query)
  - availability of different features (updates, complete query implementation)
- handling of serialization output
- assure correctness of large results
- many factors to measure:
  - CPU load
  - memory I/O
  - disk I/O
  - memory consumption

Future Work

Merge BaseX & Idefix

- comprehensive support for value-based queries
- full text queries, including scoring algorithms (like SRA/INEX)
- optimize XML table compression
- optimize disk layout (hybrid, networked, and holographic storage)
- write Pathfinder plugin to support XQuery
- complete update implementation

Benchmarking

- use of virtual machines for benchmark reproducibility
- specify benchmark for XML updates
- application benchmark for eMail storage