XBird/D: Distributed and Parallel XQuery Processing using Remote Proxy

Graduate School of Information Science, Nara Institute of Science and Technology, Japan.

Makoto Yui (Ph.D. Candidate)
Jun Miyazaki, Shunsuke Uemura, Hirokazu Kato
Outline

- Background & Motivation
- Open Problems
- Our Solution to Open Problems
- XBird/D Implementation
- Experimental Evaluation
- Conclusion
Background

As XML spreads over networks, the need to integrate distributed and dynamic XML documents is increasing.

Examples include,

- Integration of (heterogeneous) biological databases by XML
  e.g., Integrating Genbank and Uniprot by Blast search results

- Display on-the-fly information of Web from thousands of XML feeds for each user
  e.g., aggregating the latest result of football games

Our assumptions

- XML data is primary data source
- Direct access to background databases is not allowed
- XML data is frequently changing

Biological databases receive frequent update/corrections
Motivation

To realize on-the-fly processing of thousands of distributed XML documents, we apply a divide-and-conquer design paradigm.

Motivating example

On-fly-processing of thousands of XML feeds using a PC-cluster

Divide a query into sub-queries recursively

Filter results of sub-queries so as to show the latest 50 news about a recent disaster

Responsible for RSS data in .COM domain

Responsible for ATOM data in .biz domain
Open Problems and Our Solution

The prior approaches [1][2][3] commonly used **pass-by-value** semantics

A) **Limitation of inter-operator parallelism**
   due to lack support for pipelining
   - Pipelining using pass-by-reference

B) **Overhead of encoding/decoding**
   - Direct result forwarding to reduce
     the latency

C) **Poor resource utilization**
   - Remote blocking-queue with which
     processing rates of operators are managed

Explore open problem (A)

• What was meant by the limitation of inter-operator parallelism?

• Why is pipelining indispensable?
Problems in Pass-by-Value Semantics

Pass-by-value semantics involves blocking edges.

Remote Query Invocation

Query Invocation

Blocking edge

Blocking edge

Pipeline chain

Dh1

Dhn

Dv1

Dhm

... 

Pht treats n data sources

Pv treats m data sources

Consider the elapsed time of qa.

Recall the previous example

LT(qa) represents “Elapsed time of local query processing”

Edge involves latency such as encoding/decoding and network latency

Problems

- Inter-operator parallelism is limited
- Depends on the most time-consuming edge
- Non-parallelized portion of queries restricts the theoretical maximum speedups (according to Amdahl’s law)
Explore open problem (B)

How critical is the overhead of encoding/decoding?
We conducted a micro-benchmark, with using the following queries where $\text{doc}$ locates an XML document generated by XMark $\text{SF}=10$, to estimate the costs involved in remote query execution.

```xml
for $a$ in $\text{doc/site/closed_auction/closed_auction}$ where $a/\text{price/text()} \geq 40$ return $a/\text{price}
```

What this experiment showed is ...

- The latency including encoding/decoding is as same as query execution time
- Thus, each **blocking** edge of operators can potentially be a bottleneck
What is the resource utilization problem?
Resource Utilization Problems

Intuitions

- selecting low degrees of an operator parallelism can lead to under-utilization of the system and reduce throughput.

- selecting high degrees of an operator parallelism can spend “too many” resources to one query and lead to high resource contention.

Further details can be obtained in our paper and the following paper,

Our Solution for each Open Problem

A) **Limitation of inter-operator parallelism** due to lack support for pipelining
   - Pipelining using pass-by-reference

B) **Overhead of encoding/decoding**
   - Direct result forwarding to reduce the latency

C) **Poor resource utilization**
   - Remote blocking-queue with which processing rates of operators are managed

Our contribution consists of the above three techniques
A) Pass-by-Reference using a Remote Proxy

Handling a reference to a remote sequence as if it were on a local site

Our Solution for the Problem (C)

A) Limitation of inter-operator parallelism due to lack support for pipelining
   → Pipelining using pass-by-reference

B) Overhead of encoding/decoding
   → Direct result forwarding to reduce the latency

C) Poor resource utilization
   → Remote blocking-queue with which processing rates of operators are managed
B) Asynchronous Production and Queue Management

This queue is a bounded-size blocking-queue

A general consumer and producer problem

(b) consume items

(c) fetch items

Blocked if the queue is full

Blocked if the queue is empty and not reached end

Avoiding oversupply and undersupply
Our Solution for the Problem (B)

A) Limitation of inter-operator parallelism due to lack support for pipelining

- Pipelining using pass-by-reference

B) Overhead of encoding/decoding

- Direct result forwarding to reduce the latency

C) Poor resource utilization

- Remote blocking-queue with which processing rates of operators are managed
C) Direct Result Forwarding

Pass-by-Reference  

Direct result forwarding

Reduce operator does not cause access to the local resources, thus the execution is **re-locatable**.

By forwarding the reduce operator to the upper operator, **redundant encoding and decoding are eliminated**!
Experimental Evaluation - Settings

In order to evaluate the effectiveness of our three enhancements, we conducted performance comparisons to MonetDB/XRPC, one of the state-of-the-art distributed XQuery processors that represents pass-by-value semantics.

```javascript
declare function bdq:select1() {
  execute at $PE1 {
    fn:collection($col)/site/open_auctions/open_auction
  }
};
declare function bdq:select2() {
  execute at $PE4 {
    fn:collection($col)/site/open_auctions/open_auction
  }
};
declare function bdq:reduce() {
  execute at $PE2 {
    ( fn:subsequence(bdq:select1(), 1, 1000)
      | fn:subsequence(bdq:select2(), 1, 1000) )
  }
};
declare function local:filter() {
  for $a in bdq:reduce()
    where $a/seller/@person >= "person10000"
    or $a/buyer/@person >= "person10000"
  return $a
};
local:filter() (: execute at $PE1 :)
```

Experimental Evaluation - Results

Our pass-by-reference implementation using a remote proxy shows significant improvements on the elapsed time.

Combination of “Remote Proxy” and “Direct result forwarding” obtained about 22 times better performance than the competitive method (XRPC).

Moreover, only our system using pass-by-reference semantics could process 100 concurrent requests.

Pass-by-value semantics implementations suffered from frequent swap-in/swap-out due to their poor resource utilization.

Lazy evaluation effects!
Conclusion

We proposed an efficient distributed XML query processing strategy using a remote proxy and the other two techniques. Our experimental results showed up to 22x speedups compared with a competitive method in a certain situation, and demonstrated the importance for distributed XML database systems to take pass-by-reference semantics into consideration.

Future work

- Dynamic execution dispatching of remote query processors taking system resources and utilizations
- Development of a selection model of execution strategies

We have tried other methods. But the competitive system that is currently available and works properly is only MonetDB/XRPC.
Thank you for your attention

XBird will be released as an open source software on
http://db-www.naist.jp/~makoto-y/proj/xbird

I have a demo-video in which XBird/D executed
180 remote queries on Niagara T2 in 5 seconds.
If you are interested in that, please contact me later.

Any questions or suggestions?
declare variable $colname := "/dews2008/xmark10.xml";
declare variable $remote-endpoint := "//niagara:1099/xbird/srv-01";
fn:subsequence(
    execute at $remote-endpoint
    {
        for $a in
            fn:collection($colname)/site/closed_auctions/closed_auction
            where $a/price/text() >= 40
            return $a/price
    }
,1,1000
)
Resource utilization problem

- Oversupply of tuples
  - Too much production wastes system resources

- Undersupply of tuples
  - Consumer tends to be idle

Executed at different PEs

\[ \sigma \]

Select

Join

Consumed tuple

Produced tuple

tuple

tuple