Dynamic XML with distribution and replication

Angela Bonifati
mailTo: bonifati@icar.cnr.it

Joint work with S. Abiteboul, T.Milo, G. Cobena and I. Manolescu, Inria Rocquencourt (France)
Agenda

- Motivations
  - Distribution and replication of documents, whether dynamic or plain, is gaining momentum in distributed data management within large-scale P2P networks

- Contributions
  - Data model and query language
  - Cost model and observable performance of a peer
  - Query evaluation strategies and query optimization
  - A customized replication algorithm

- Conclusions
Motivations

Background:
- **Active XML (AXML)** contains already an inherent form of distribution.
- Executable XML is not a novelty: Apache Jelly, Macromedia Mx, Sun JSP, PHP mainly run XML for presentation purposes.
- Previous works in distributed databases and LDAP directories: they do not straightly apply to Active XML.

*Which scenarios of replication and distribution do else arise?*

Active XML on **Mobile** Devices: how to face the varied conditions of the network (bad communication, bottlenecks, small storage devices etc.)?
Dynamic AXML Documents

```xml
<knownAuctions ID="peer10">
  <category name="Toys">
    <sc>eBay.net/getOffers("Toys")</sc>
    <auction id="1254">
      <heldBy>eBay.net</heldBy>
      <item>Pink Panther</item>
    </auction>
    ...
    <sc>babel.org/translate("Czech", "English", <sc>crystal.cz/getToys()</sc>)</sc>
  </category>
  ...
  <sc frequency="once">getMyAuctions()</sc>
</knownAuctions>
```

May contain calls to any SOAP web service

- e-bay.net, google.com, babel.org, etc.
- Web services can be locally or remotely invoked
- Are exposed to the world through WSDL

Are enriched by each service calls results

The returned nodes are inserted as siblings of the corresponding `<sc>` element.

Their call parameters can be XPath expressions or other service calls

```xml
let service Get-Toys( ) be
for $a in document("crystal.cz/Toys.axml")/toys,
  $b in $a//name,
  $c in $a//size,
  $d in $a//price
return <nice-toy tname={ $b/text() }> { $c } { $d } </>
```

AXML Documents are already data integrators, but they need extensions to fully play in a distributed data management setting.
DB and LDAP: a quick glance

- Horizontal and vertical fragmentation in distributed databases issue queries by using different degrees of transparency (fragmentation, location, language transparency, no transparency)
- LDAP foresees a common shared directory for storing ldap entries (like in Yellow Pages). Beneficial query templates are selected, whose results are cached.
- We chose to:
  - Distribute and replicate XML fragments, as well as service calls and service definitions (*data and code shipping*). Our setting is P2P networks.
D&R AXML

- Accessing remote services:
  - Peer1/getOffers("Toys")

- Replicating data fragments with <sc> elements:
  - Data and sc are replicated here
  - Peer1/getOffers("Toys")

- Replicating service definitions:
  - Data, sc and getOffers() def are replicated here
A dynamic document is a labeled tree, where the tree nodes are XML elements and attributes, and may also be special elements called functions.

The function elements are web service calls of two types: declarative or opaque.

To make replication possible, each tree node is identified by:

<ID@P_url>

To make distribution feasible, external children edges and external parent edges have been devised in the model. Consistency: the collapsed version of replicas and distributed fragments must be a regular dynamic XML document.
D&R simulation

Dynamic peer P1

Service Evaluator

Dynamic Document Base

Dynamic peer P2

SOAP

LE

AXML peer P3

SOAP

LE

Replicas

Distributed fragments

Element nodes

Service call nodes

Service call invocation

Service result

External child edge

External parent edge
A user has a PDA, and is in Colorado. We could let her call the ski portal services. We would rather replicate on her PDA the relevant data and services, so that they can be called locally.

Example: a Ski Portal

- A national US ski portal:
  - Contains information about ski resorts, hotels.
  - Provides services to query this information.

- www.UnisysWeather.com:
  - Gives up-to-date temperatures and weather conditions all over the US.

- A user has a PDA, and is in Colorado.
  - We could let her call the ski portal services.
  - We would rather replicate on her PDA the relevant data and services, so that they can be called locally.
D&R AXML for the Ski Portal

US SkiPortal Site

<resorts state='Colorado'>
  <resort>
    <name> Aspen </>
    <scond> SnowConditions('Aspen') </>
  </>
  <hotels ID=AspHotels > .... </>
</>
<OpativeSkiResorts($state)
<HotelsInfo ($state, $resort)

Colorado PDA

<resort>
  <name> Aspen </>
  <scond> SnowConditions('Aspen') </>
</>
<hotels ID=AspHotels>
  <externalURL>
    <externalURL>
      http:// www.HS.com/ColoradoSC </>
    </>
  <resort> .......<>
  <OpativeSkiResorts()
  <HotelsInfo ($resort)

www.UnisysWeather.com

SnowConditions($resort)

Opaque Web Service
Declarative Web Service
Adjusted definition of
Queries in D&R AXML

- **Location-transparency**: one would like queries to act in D&R without really caring of the peer(s) actual locations.

- **Location-awareness**: queries in which locations are explicit in the syntax let the user decide the query locations, or execute locally whenever possible.
XQuery\textsubscript{dr}

- XQuery\textsubscript{dr}: an extension of XQuery, in which
  - the LET/FOR clauses are the same, except for the path qualifiers : \{path1@P1, …, \texttt{pathn@Pn}\}, where P1, …, Pn hold \texttt{local}, \texttt{any} (default), \texttt{all, localORany, s_i, master, masterORlocalORAny}.
    - Example: \{\texttt{document("ColoradoPDA")/resort[name="Aspen"]}/hotels/hotel}@localORany
  - the WHERE clause slightly changes in the assignment of variables.
  - the RETURN clause returns fresh IDs for constructed data and includes the external edges if any.
  - the clause \texttt{REPLICATE $var AT peername WITH pe}.
Query evaluation

- Given a query Q in XQuery\textsubscript{dr}, evaluation starts at the local peer P\textsubscript{1} and proceeds on the next peers: \( Q_{\text{local}@P_{1}} \cup Q_{\text{next}@P_{2}} \cup \ldots \cup Q_{\text{next}@P_{n}} \).
- How to choose P\textsubscript{next}?
  - In case of Q location-unaware: identify the set of candidate peers P\textsubscript{cand} by looking at exit points;
  - In case of Q location-aware, the set of candidate peers is known in advance and hardcoded in the query;
  - Perform a What-if cost-based analysis;
  - Master copies need a special treatment.
Eligible Next Peers ($P_{next}$)

- The chosen $P_{next}$ is the one that minimizes the costs of $Q$ w.r.t. $P_1$, which in turn depends on the following (a priori unknown to $P$):
  - Data statistics at $P_{next}$
  - Cost parameters at $P_{next}$
  - Exit points (external links) of $Q_{next}$

- An Info record is built by the What-if cost analysis:

```
Info  <record peer = P>
   <decompose local=Qlocal next=Qnext />
   <local cost=cost(Qlocal) fanout=N_{qlocal}
      size=size(Qlocal) />
   <bw> P-> Pnext </>
</record>
```
Cost Parameters

- Observable performance of a peer:

  - Inputs:
    - **Subjective Parameters**: communication, space, computation cost relative **weights** w.r.t. the overall cost afforded by each peer.
    - **Objective Parameters**: size of data transfers, cost incurred by the execution of a query on a given peer, frequency of query invocations.

  - Output: *given a peer* $j$ *and a query workload on that peer*, we compute
    - **the observable cost vectors** of computation, space consumption, received data and sent data.
Configuration changes

- A set of peers with data and web services is called a configuration.

- For each configuration and query workload, the cost model measures the observable performance of each peer.

- We have devised a replication algorithm, which enforces a configuration change, meaning that it recommends some local replication steps aimed at improving the single peer’s observable performance.
Replication alternatives

1. Accessing remote services: this is a no-replication choice, which consists in simply adding external links to the data of interest.

2. Replicating data with/without `<sc>` elements: this is a choice to replicate data by either adding links to `<sc>` elements or replicating `<sc>` elements as well.

3. Replicating service definitions: this is a choice to replicate data, service calls and service definitions/variants of service definitions.

ALL THESE CHOICES ARE COST-BASED!
Replication decisions for (III)

- Replication is not trivial for (III) since
  - When replicating the service code, one may want to
    - replicate the data it uses, but only the data that is strictly necessary
    - replicate the code as it is or better adjust it/adapt to the local data
    - make another replication decision for each service call that is included into the replica
  - Our algorithm focuses on (III) and devises an intermediate solution between
    - Naively copy all the data touched by the service code together with the service code (unchanged)
    - Copy only the results of the service code evaluation together with the modified service code (query result caching)
Replicate what?

Q1: dept/professor/name

Q1': ./name

<name>
Jennifer
<NrEnrolled students>
230
</>
</>

The solution may be in the middle!
Replicate what?

Q1: dept/professor/name

The solution may be in the middle!
On Implementation Status

- Technical environment for AXML:
  - SUN’s Java JDK 1.4 (includes XML parser, XPath processor, XSLT engine)
  - Apache Tomcat 4.0 servlet engine
  - Apache Axis SOAP toolkit 1.0 beta 3
  - X-OQL query processor, persistent DOM repository
  - JSP-based user interface, using JSLT 1.0 standard tag library

- AXML D&R-specific (ongoing):
  - A collaborative distributed Workspace Demo (submitted to VLDB03)
  - A light AXML peer for CLDC devices (kXML-RPC, XPath proxy)

- AXML is online: [http://www-rocq.inria.fr/verso/Gemo/Projects/axml/](http://www-rocq.inria.fr/verso/Gemo/Projects/axml/)
Conclusions

- We have devised:
  - AXML D&R data model
  - XQuery_{DR}
  - A cost-wise *query evaluation* strategy
  - A cost-wise *replication* algorithm

- Future directions:
  - Smarten the query evaluation strategies (ad-hoc techniques)
  - Implement a small prototype for query decomposition by using a local XML Schema annotated with XLink and an XLink base
Thanks for your attention
Merci de votre attention
Grazie....