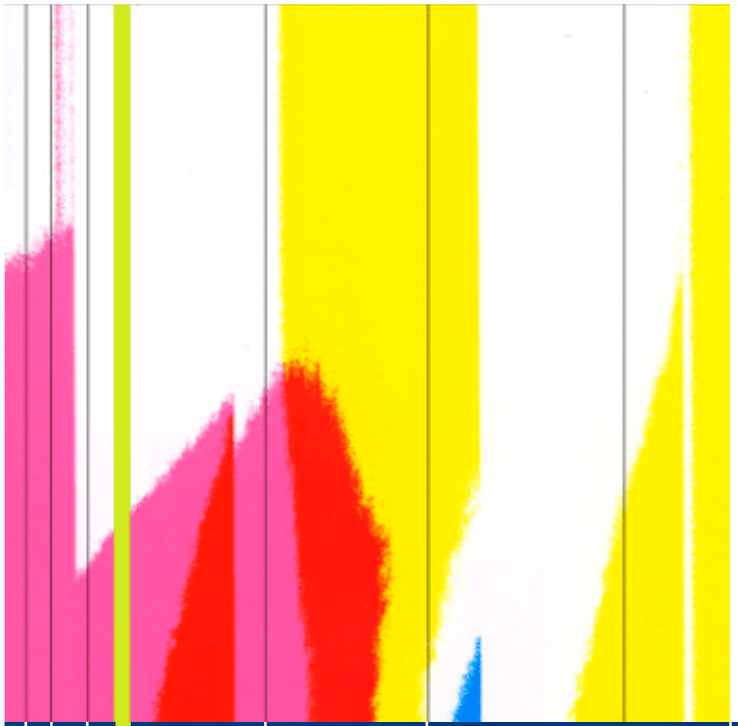


Le Store WebContent

Benjamin NGUYEN UVSQ & INRIA-SMIS
Spyros ZOUPANOS U.Paris Dauphine & INRIA-LEO

Workshop Sources Ouvertes et Service – Caen 2010

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

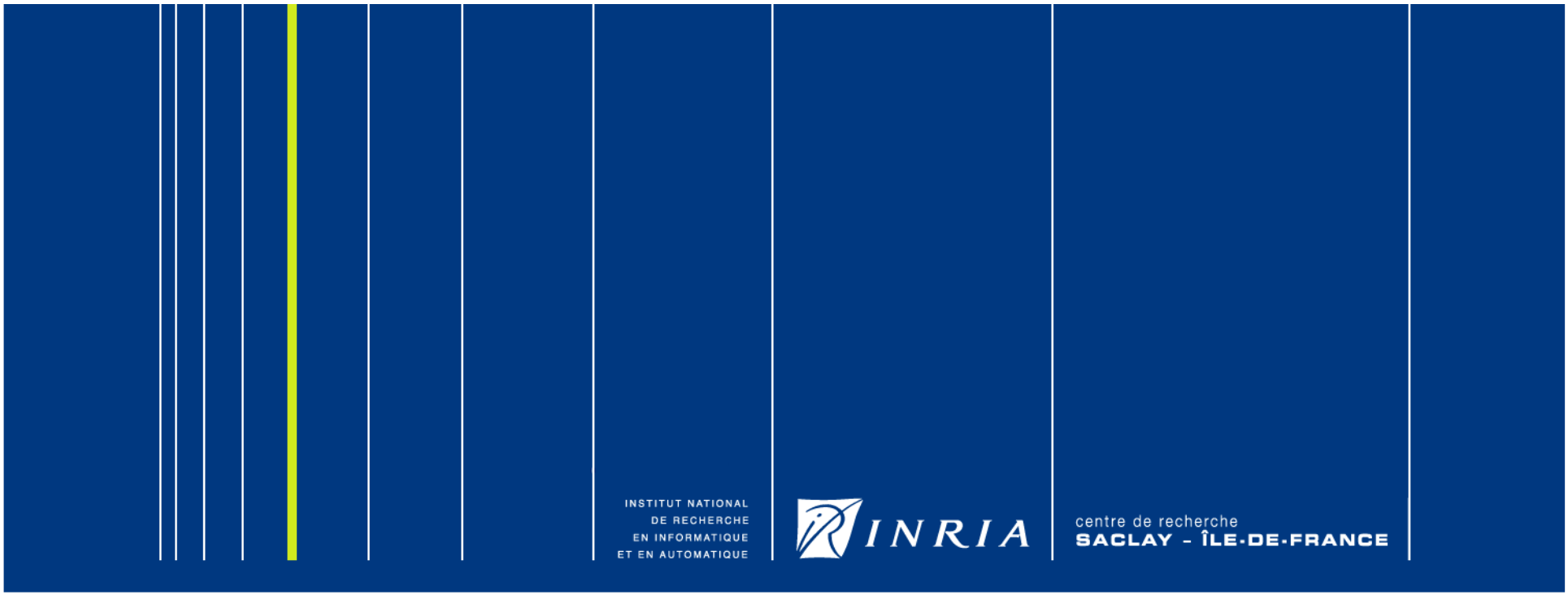
INRIA

centre de recherche
SACLAY - ÎLE-DE-FRANCE

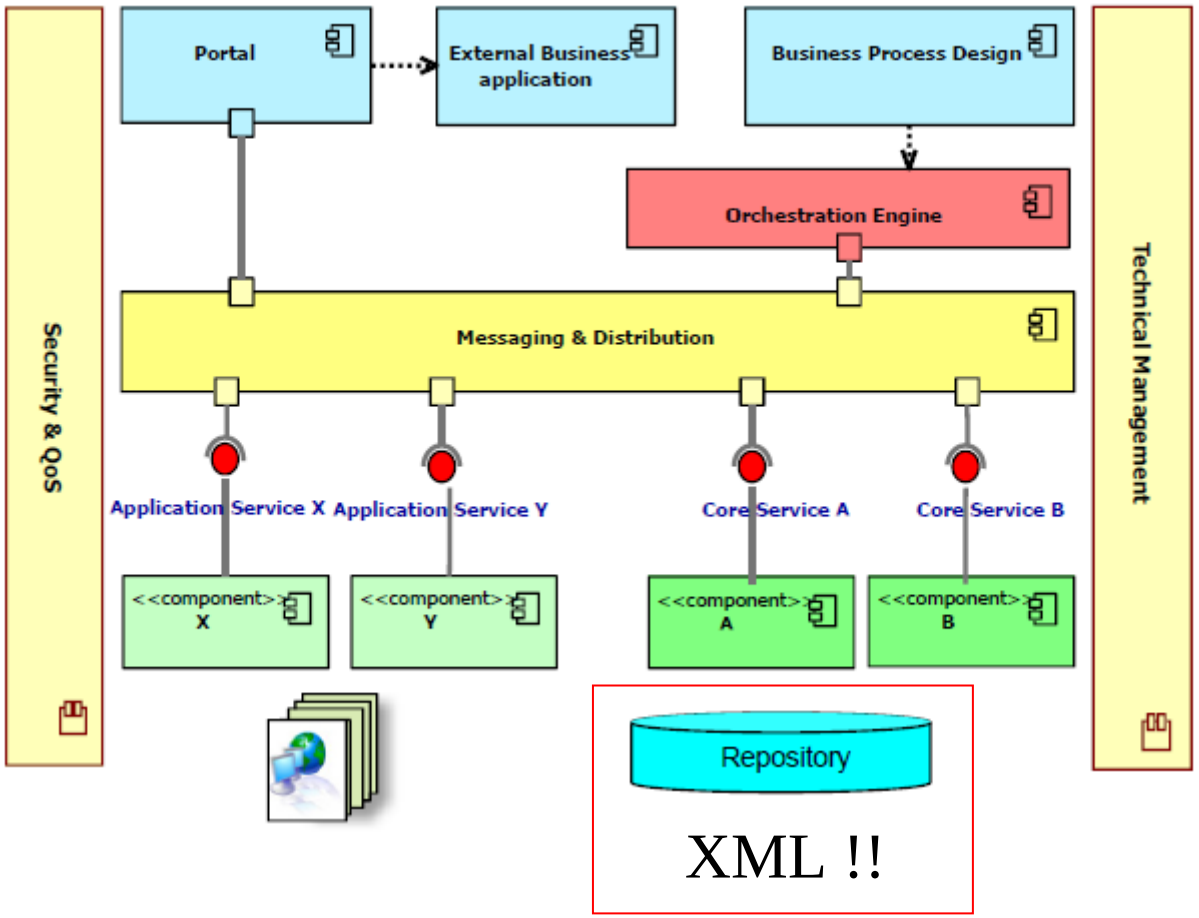
Plan

1. Enjeux de l’utilisation d’un SGBD XML
2. Exemple du store centralisé
3. Store P2P

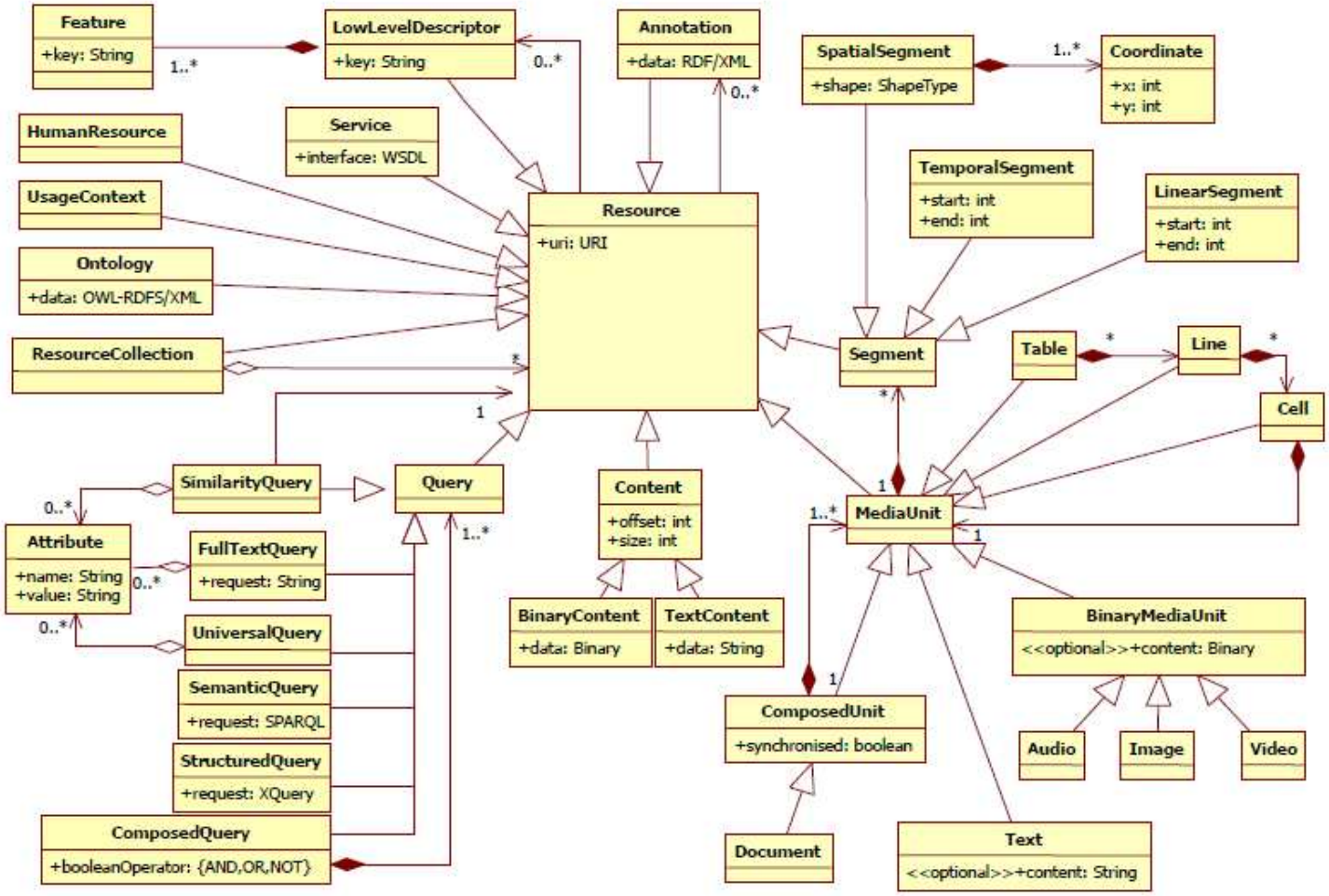
Enjeux de l’utilisation d’un SGBD XML



Architecture de WebContent



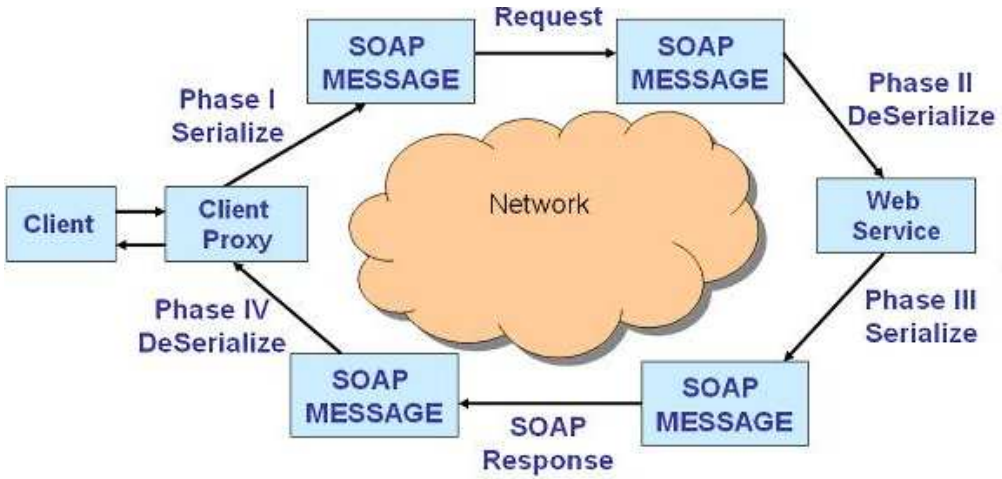
Le format pivot (UML)



Le format pivot (exemple XML)

- Les schémas XML
- Un exemple de document

Paradigme des Services Web



- Un modèle pivot de documents basé sur UML et implémenté par un XML Schema
- Des données échangées via des services web en XML
- Un traitement en interne des informations par des programmes en C, Java, etc…

Quelle utilisation des documents ?

- Stockage
  - Définition d'une URI
  - Sert de référence future
- Récupération
  - Intégralité du document ← indexation des documents
  - Sous partie du document ← indexation structurelle
- Requêtes ← utilisation d'un langage de requête (XQuery)
  - Modification de la structure du document
  - Jointure entre documents
- Modification (possiblement concurrente) des documents lors d'une chaîne de traitement ← problème d'incohérences / redondance

- Il faut utiliser un véritable SGBD et pas simplement des fichiers !

Store centralisé

Construction du StoreService

- Problème :
  - Construire un wrapper de Web Service sur une base de données XML déjà existante (eXist, MonetDB, SQL Server, Quizx, … Sedna ?)
  - Choix d'un SGBD existant performant & fonctionnel (et des interfaces existant par rapport au langage d'interrogation)
- Caractéristiques
  - Passage à l'échelle
    - Nombre de documents (ok?)
    - Nombre de requêtes (pb?)
  - Robustesse
  - Reprise sur panne

Fonctionnalités

- Stockage d'un document du modèle
  - Génération de l'URI
- Récupération du document
  - Dans son intégralité
  - Juste une sous partie
  - Basé sur l'utilisation des URIs WebContent
- Requêtes XQuery complexes sur les documents

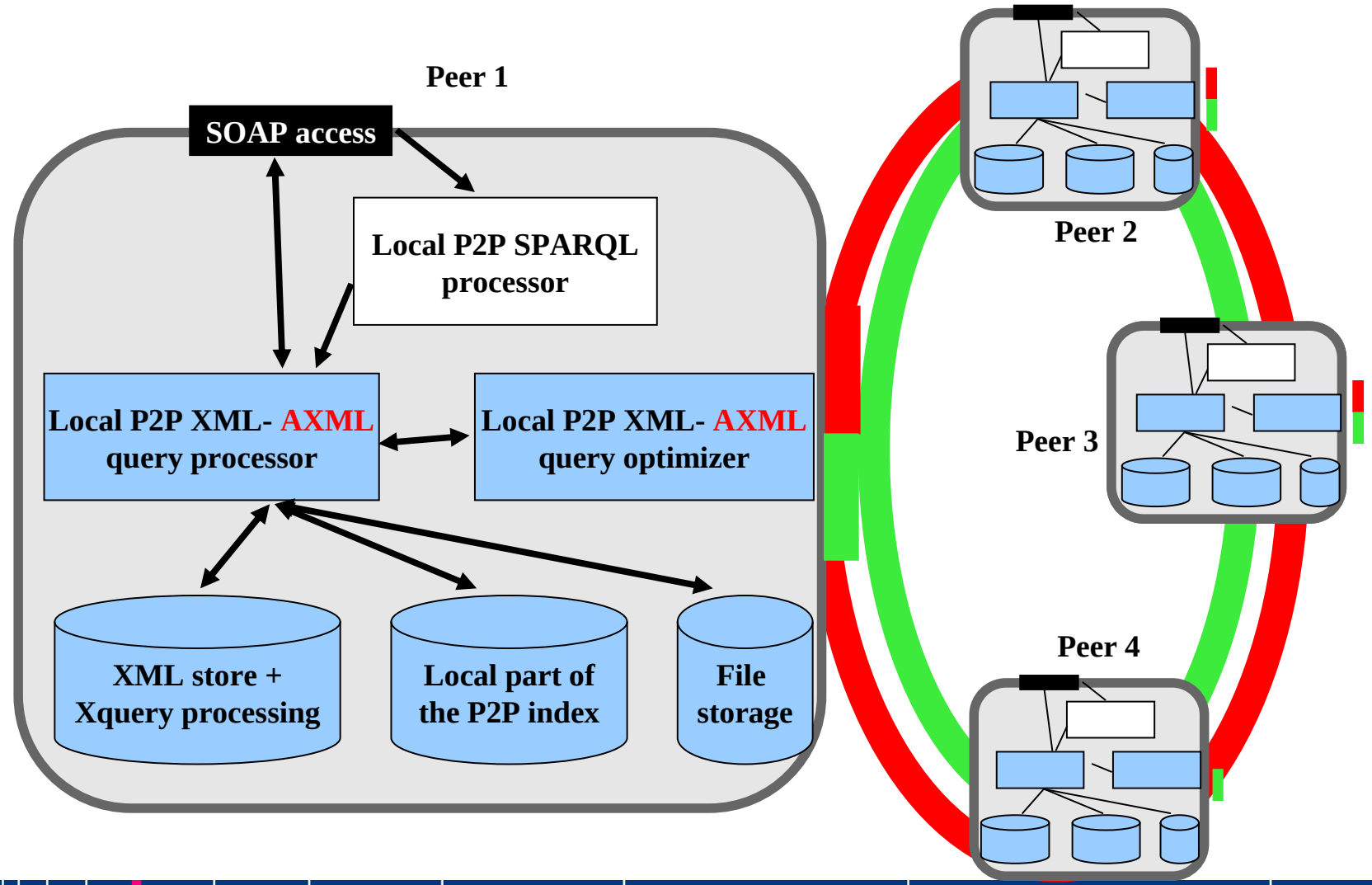
Démo du store centralisé

Enseignements

- L’utilisation simple de fichiers n’aurait pas fonctionné
- Le système passe à l’échelle en terme de documents (plusieurs milliers)
- Un système d’indexation simple aurait pu suffire
- Peu de fonctionnalités véritables du SGBD ont été utilisées
- Il est difficile de séparer le service de stockage du service d’interrogation

3- Store P2P

Outline of P2P services



Peer-to-peer storage service

Implemented jointly by several peers

DHT service is implemented on top of DHTs

Basic functionality of a DHT

- put(k,v)
- get(k)

Different DHTs may have different algorithmic properties

- Goal: combine the advantages of the available DHTs

KadoP

Index keys:

- all element & attribute names

Index values:

- structural identifiers *(docID, start, end)*

Kadop's query language:

- tree pattern language
- each node is labeled with a XML node or word
- edges are / or //

Query evaluation:

- holistic join on the list of identifiers associated to the query node labels

Advantage:

- fast tree patern query execution based on the one and only holistic join

PathFinder

Index keys:

- linear parent-child rooted paths

Index values:

- sorted *docId* list of documents that contain a given path

PathFinder's query language:

- conjunctions of disjunctions of atomic path expressions
- atomic path: XPath with child axis + predicate

Special hash function:

- keys that are lexicographically close are stored to peers that are close between themselves

Advantage:

- inequality queries e.g. `/article[ year > 2005 ]`

AXML language

Data-centric Web service composition

ActiveXML document = XML document including calls to (continuous) Web services

- A service call contains contact info for the Web service
- When the calls is activated, results are added to the document as siblings of the service call.

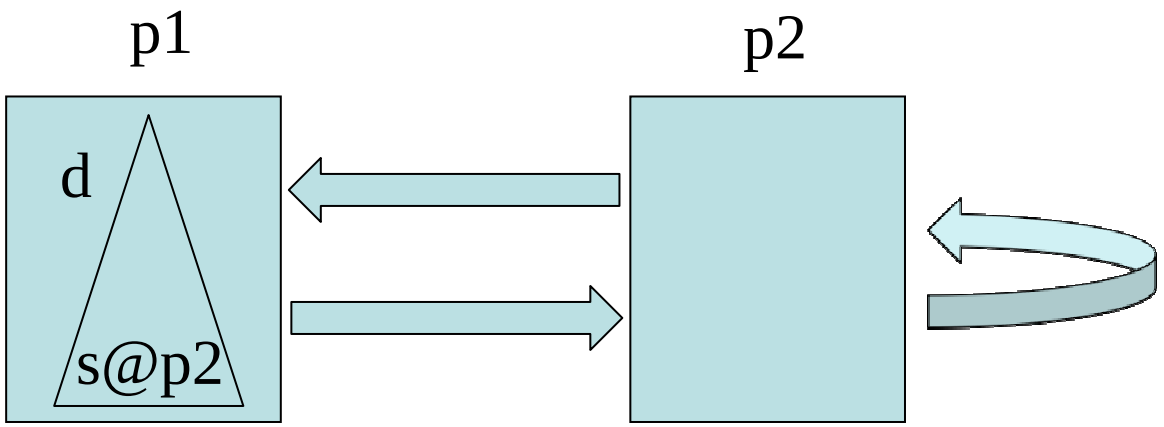
```
<myPage>
        <axml:sc service="getProgram" peer="tvchannel.com">
                <parameter>Movies</parameter>
        </axml:sc>
        <program day="today"><movie>Shrek 3</movie></program>
        <program day="tomorrow"><movie>Persepolis</movie></program>
</myPage>
```

AXML evaluation

AXML document d@p1, sc in d calls s@p2($in)

Activating sc entails:

- stream $in to p2
- evaluate s@p2
- stream the results of s@p2 to p1



OptimAX

A rule based AXML optimizer

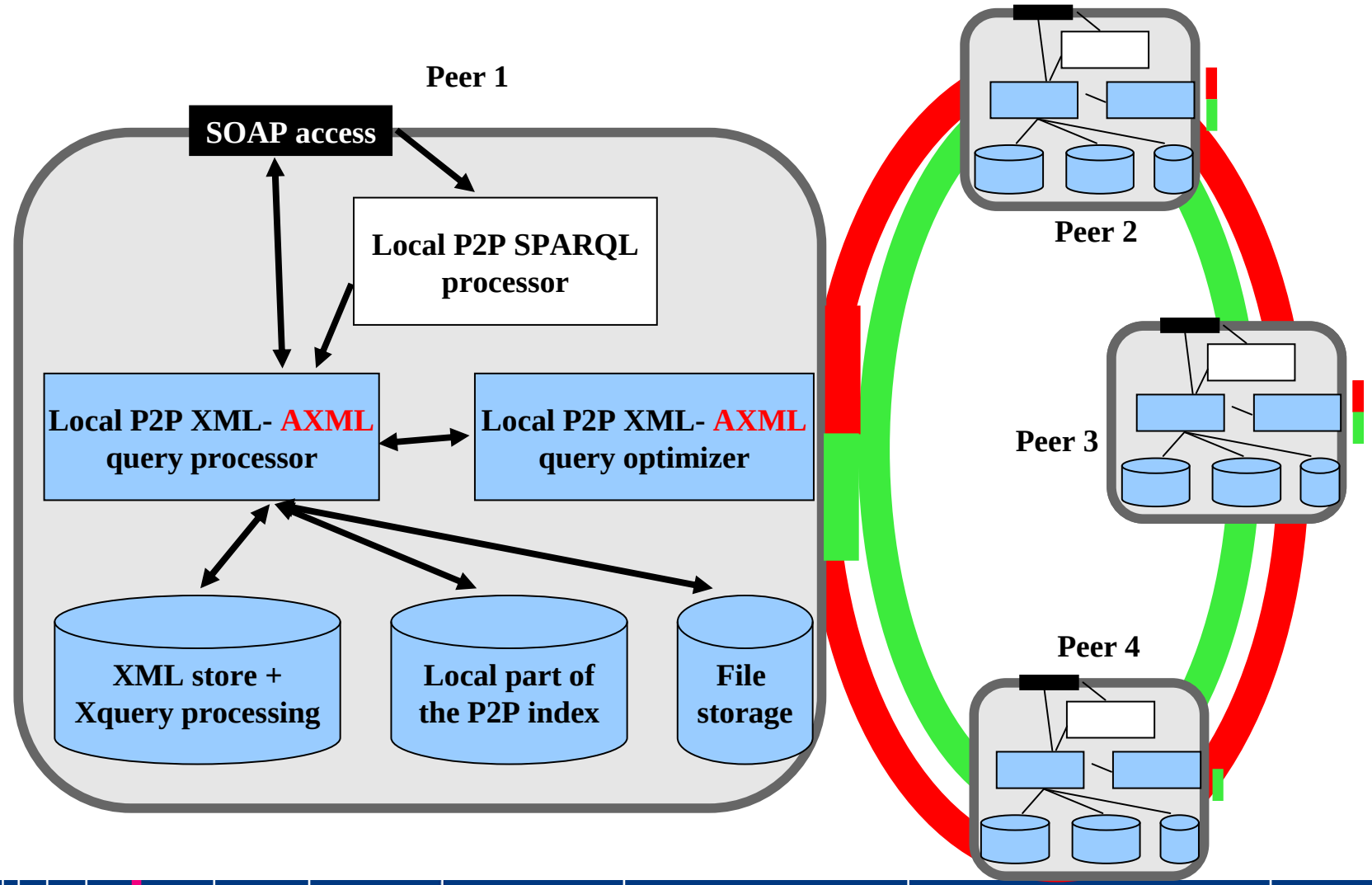
- uses an extensible set of rules (cooperates with other programs to perform better document opimization)
- rewrites an AXML document to another one with smaller overall evaluation cost
- uses statistics for document optimization

Putting everything together

When a query arrives:

- it is introduced into a document d_1
- d_1 is given to optimAX for optimization
- optimAX will cooperate with TGV – an algebraic XQuery compiler – to decompose it into sub-queries
- based on the type of the sub-query, optimAX will create a call to the right DHT
- a compensation query will be added in the new document d_2 above all the calls to the DHTs
- further optimization if needed will be performed
- d_2 will be given to the AXML engine for evaluation

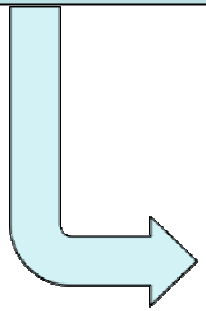
Outline of P2P services



Example

Original query

```
for $d1 in /Col1/Book, $d2 in /Col2/Book
where $d1/PageNo < 400
and $d2/Author = "Ullman"
and $d2/Code = $d1/Code
return $d2/Title
```



OptimAX + TGV

Example

Join query

```
for $x_0 in $in_0//res, $x_1 in $in_1//res
where $x_0//Code= $x_1/Code
return $x_0//Title
```

Tree pattern query

```
<Col2>
  <Book returned="true">
    <Author>Ullman</Author>
    <Code/></Book>
</Col2>
```

Query with inequality

```
for $d1 in /Col1/Book
where $d1/PageNo < 400 return
<res>{$d1/Code}</res>
```

SPARQL processor

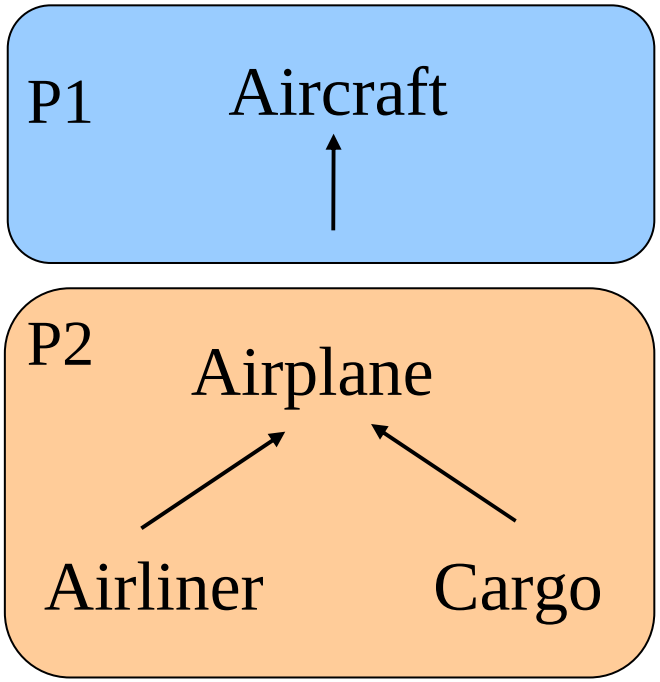
In WebContent there are semantic data to be queried.

A semantic peer marks available resources according to the appropriate ontologies.

Semantic peers interact with each other by means of mapping.

With the use of ontologies and mappings, reasoning can be inferred.

Semantic example



Query:

Q(X) :- P1:Aircraft(X)

Max conjunctive rewritings:

R1(X) :- P1:Aircraft(X)

R2(X) :- P2:Airplane(X)

R3(X) :- P2:Airliner(X)

R4(X) :- P2:Cargo(X)