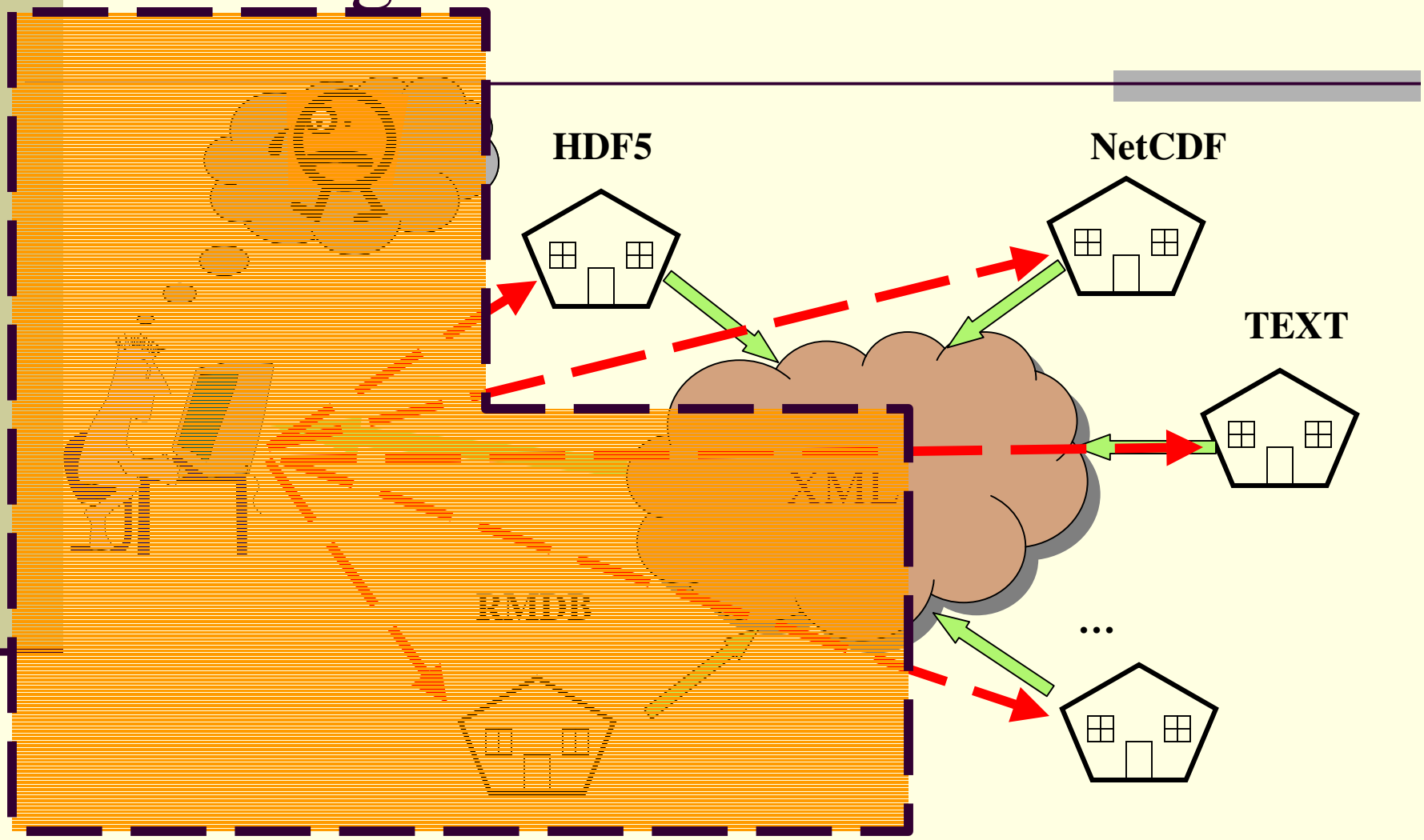
Supporting High-Level Abstractions through XML Technologies

Xiaogang Li
Gagan Agrawal
The Ohio State University

Motivation

- The need
  - Analysis of datasets is becoming crucial for scientific advances
    - Emergence of X-Informatics
  - Complex data formats complicate processing
  - Need for applications that are easily portable - compatibility with web/grid services
- The opportunity
  - The emergence of XML and related technologies developed by W3C
  - XML is already extensively used as part of Grid/Distributed Computing
- Can XML help in scientific data processing?

The Big Picture



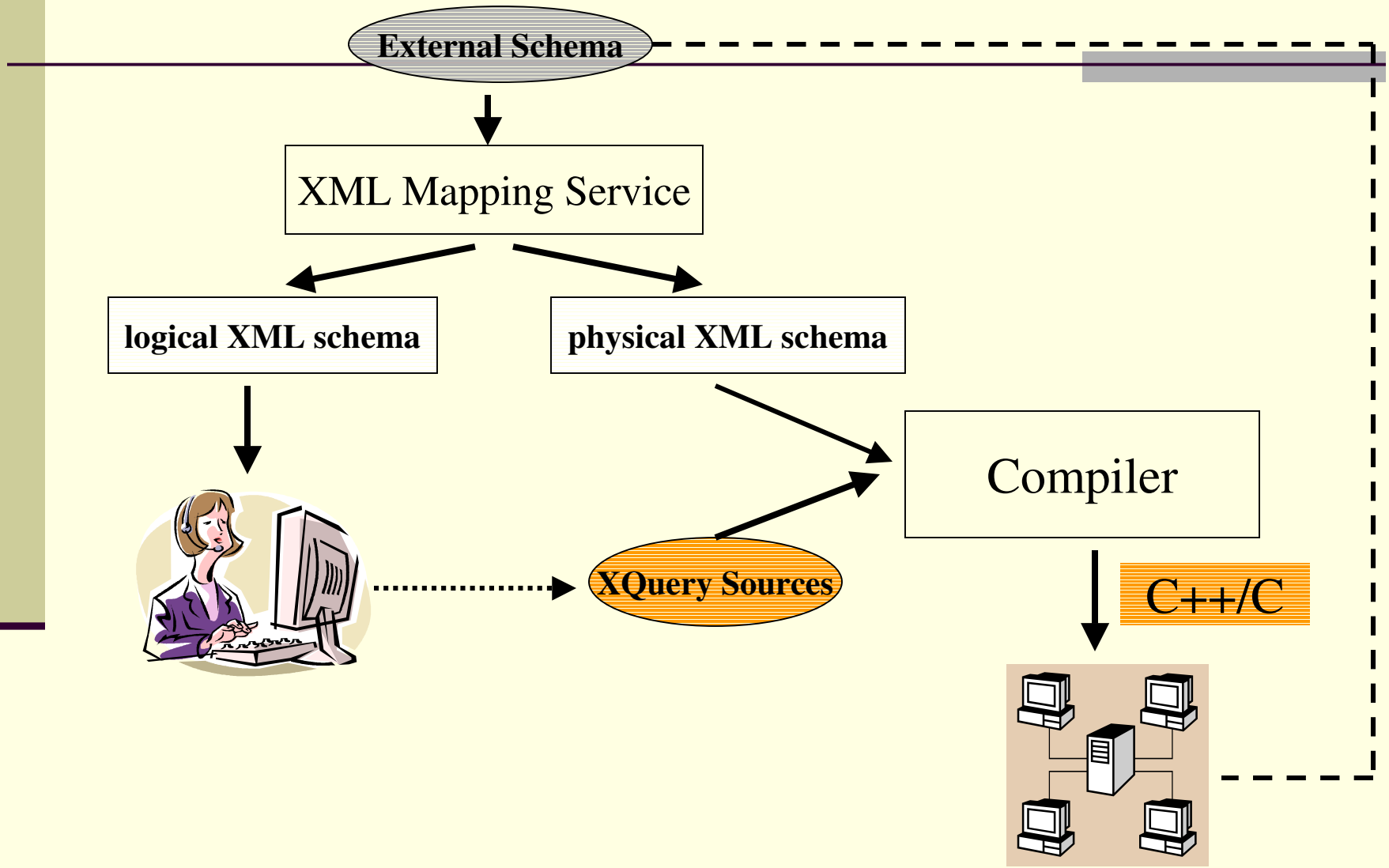
Programming/Query Language

- High-level declarative languages ease application development
  - Popularity of Matlab for scientific computations
- New challenges in compiling them for efficient execution
- XQuery is a high-level language for processing XML datasets
  - Derived from database, declarative, and functional languages !
  - XPath (a subset of XQuery) embedded in an imperative language is another option

Approach / Contributions

- Use of XML Schemas to provide high-level abstractions on complex datasets
- Using XQuery with these Schemas to specify processing
- Issues in Translation
  - High-level to low-level code
  - Data-centric transformations for locality in low-level codes
  - Issues specific to XQuery
    - Recognizing recursive reductions
    - Type inferencing and translation

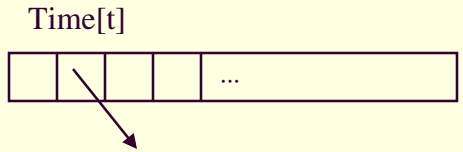
System Architecture

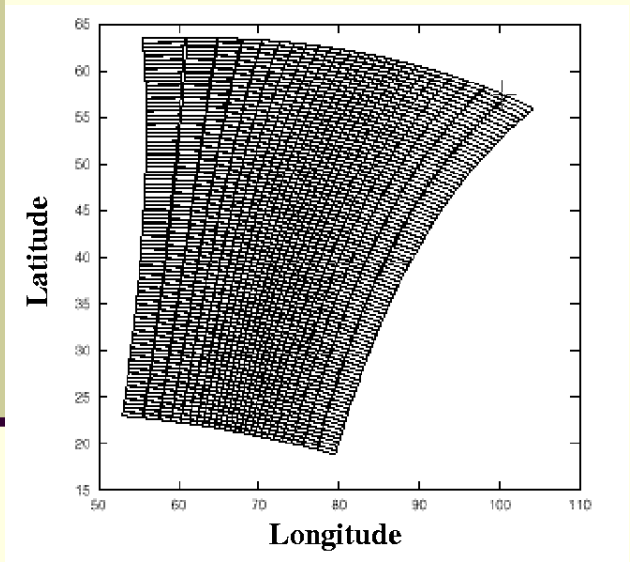


Outline

- Example application and Use of XML Schemas
- XQuery features and example
- Translation Issue
  - High-level to low-level translation
  - Data-centric transformations
  - Analyzing recursive reductions
- Experimental results
- Conclusions

Satellite Data Processing





- Data collected by satellites is a collection of chunks, each of which captures an irregular section of earth captured at time t
- The entire dataset comprises multiples pixels for each point in earth at different times, but not for all times
- Typical processing is a reduction along the time dimension – hard to write on the raw data format

Using a High-level Schema

- High-level view of the dataset – a simple collection of pixels
- Latitude, longitude, and time explicitly stored with each pixel
- Easy to specify processing
  - Don’t care about locality / unnecessary scans
- At least one order of magnitude overhead in storage
  - Suitable as a logical format only

XQuery Overview

- XQuery

-A language for querying and processing XML document

- Functional language
- Single Assignment
- Strongly typed

- XQuery Expression

- for let where return (FLWR)
- unordered
- path expression

```
Unordered(
For $d in document("depts.xml")//deptno
  let $e:=document("emps.xml")//emp
              [Deptno= $d]
                where count($e)>=10
        return
          <big-dept>
            {$d,
              <count> {count($e) }</count>
              <avg> {avg($e/salary)}<avg>
              }
          </big-dept>  )
```

Satellite- XQuery Code

```
Unordered (
  for $i in ( $minx to $maxx)
   for $j in ($miny to $maxy)
    let p:=document("sate.xml")
    /data/pixel  where
        lat = i  and long = j
    return
        <pixel>
            <latitude> {$i} </latitude>
            <longitude> {$j} <longitude>
             <sum>{accumulate($p)}</sum>
        </pixel>
    )
```

```
Define function accumulate ($p)
      as double
{   let $inp := item-at($p,1)
  let $NVDI := (( $inp/band1 -
    $inp/band0)div($inp/band1+$inp/band0)+
    1)*512
   return
     if (empty( $p) )
       then  0
     else
       { max($NVDI, accumulate(subsequence
    ($p, 2 ))) }
```

Challenges

- Need to translate to low-level schema
  - Focus on correctness and avoiding unnecessary reads
- Enhancing locality
  - Data-centric execution on XQuery constructs
  - Use information on low-level data layout
- Issues specific to XQuery
  - Reductions expressed as recursive functions
  - Generating code in an imperative language
    - For either direct compilation or use a part of a runtime system
    - Requires type conversion

Mapping to Low-level Schema

- A number of getData functions to access elements(s) of required types
- getData functions written in XQuery
  - allow analysis and transformations
- Want to insert getData functions automatically
  - preserve correctness and avoid unnecessary scans

getData(lat x, long y)

getData(lat x)

getData(long y)

getData(lat x, long y, time t)

….

Low-level Code

```
Unordered (
   for $i in ( $minx to $maxx)
     for $j in ($miny to $maxy)
       let p:=  getData(i,j)
       return
            <pixel>
               <latitude> {$i}
</latitude>
               <longitude> {$j}
<longitude>

<sum>{accumulate($p)}</sum>
            </pixel>
      )
```

- Generate correct low-level code
- Insert getData function which reads the smallest superset of the values read in high-level code
- Use relational algebra
  - Reduce to canonical form
  - Compare canonical forms
- Resulting code
  - Can require unnecessary scans
  - May have very poor locality

Data Centric Transformation

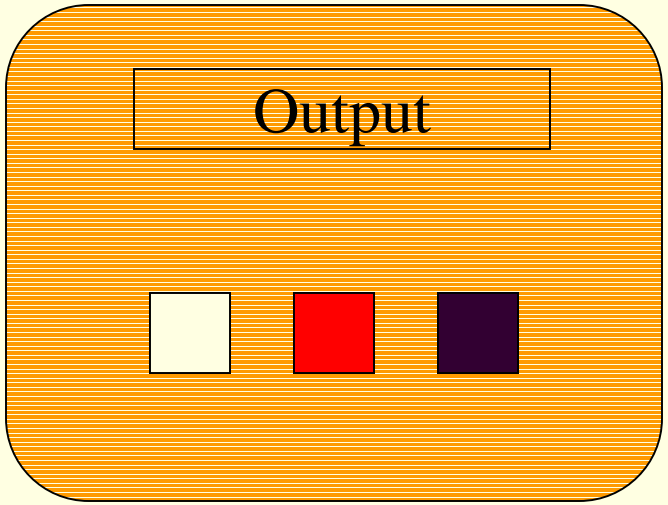
- Objective

  -Reconstruct unordered loops of the query so that only one scan of the entire dataset is sufficient

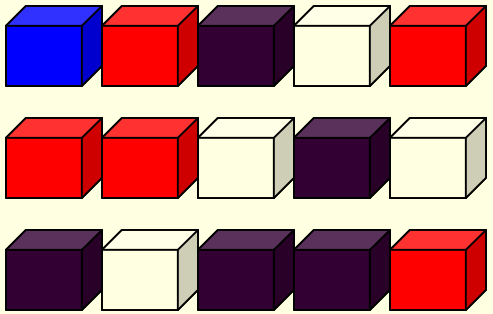
- Algorithm

1. Perform loop fusion that is necessary
2. Generate abstract iteration space
3. Extracting necessary and sufficient conditions that maps an element in the dataset to the iteration space

Naïve Strategy

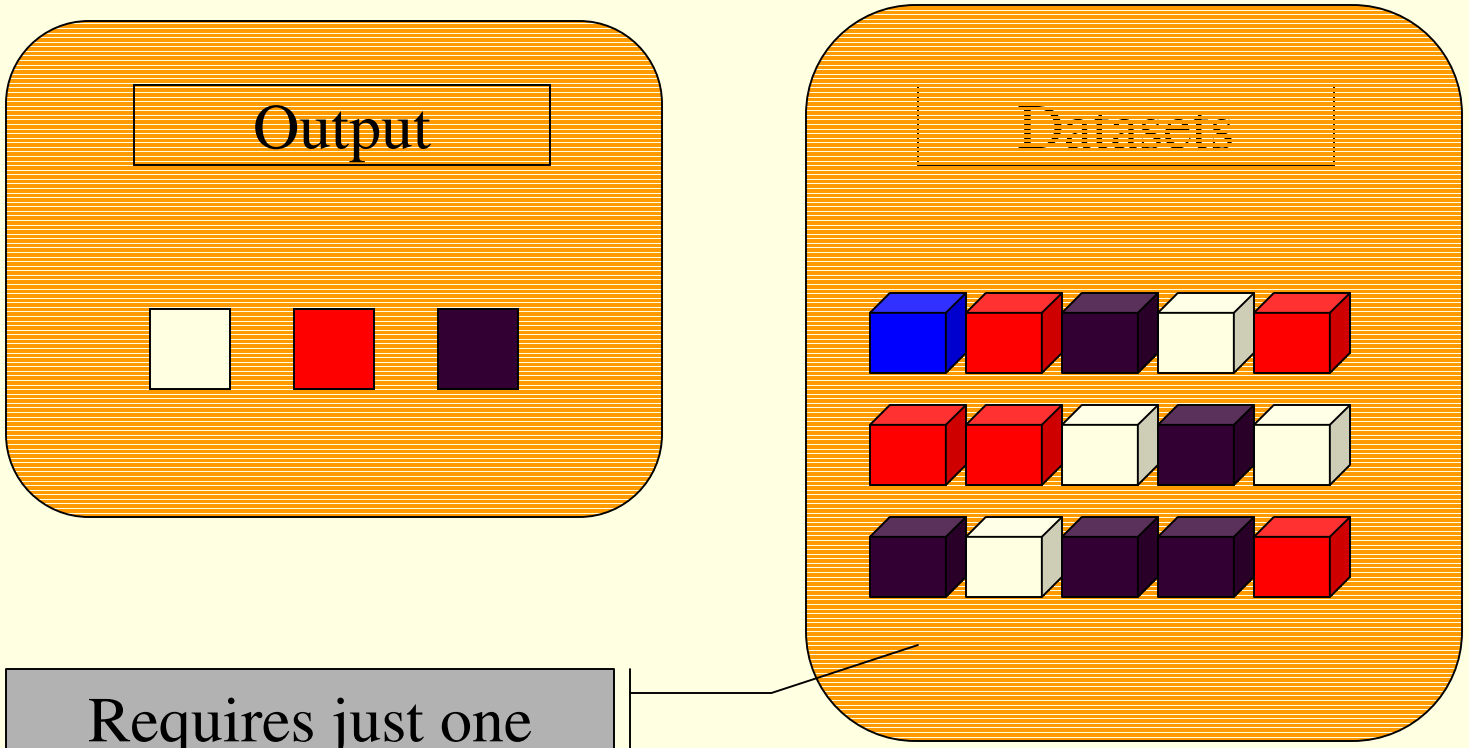


Dataset



Requires 3 Scans

Data Centric Strategy



Recursion Analysis

- Assumption
  - -Expressed in Canonical form
    1) Linear recursive function
    2) Operation is associative and commutative
- Objective
  - - Extracting associative and commutative operations
  - - Extracting initialization conditions
  - - Transform into iterative operations
  - - Generate a global reduction function

- **Canonical Form**

```
Define function F($t) {
 if (p1) then F1 ($t)
Else F2(F3($t),
      F4(F(F5($t))))
}
```

Recursion analysis -Algorithm

- Algorithm

1. Add leaf nodes represent or are defined by recursive function to Set S.

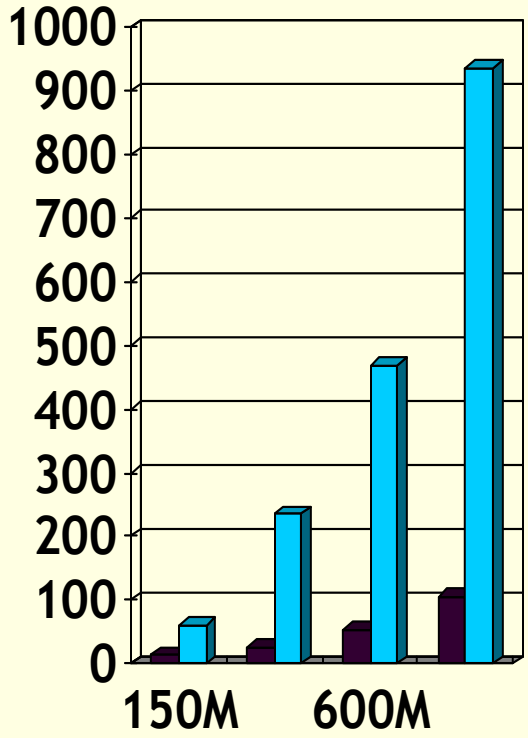
2. Keep only nodes that may be returned as the final value to Set S.

3. Recursively find a least common ancestor A of all nodes in S.
4. Return the subtree with A as Root.
5. Examine if the subtree represents an associative and communicative operation

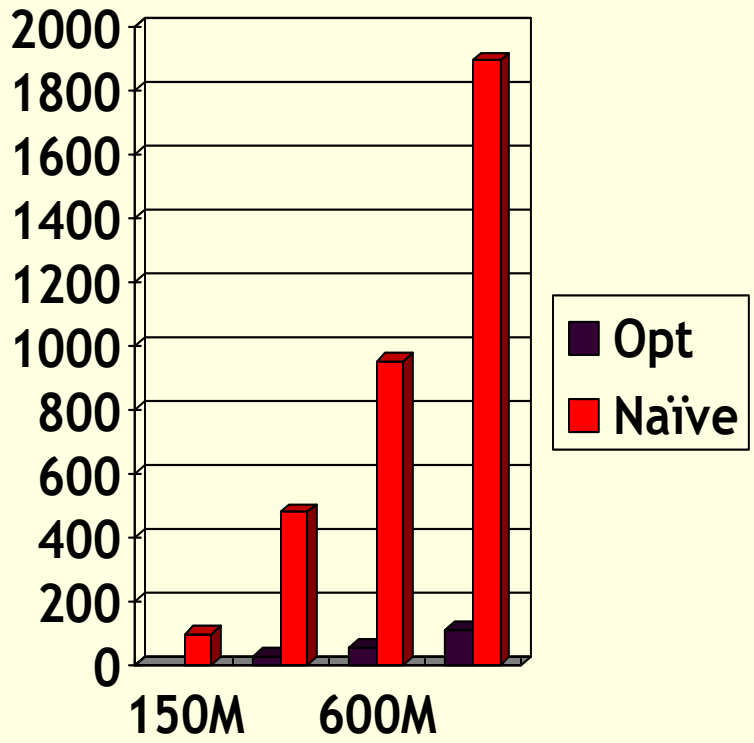
Example

```
define function accumulate ($p)
return double
 if (empty($p) ) then 0
     else let $val :=
     accumulate(subsequence($p,2))
           let $q := item-at($p,1)
             return
               If ($q >= $val) then $val
                  else $q
```

Evaluating Data centric Transformation

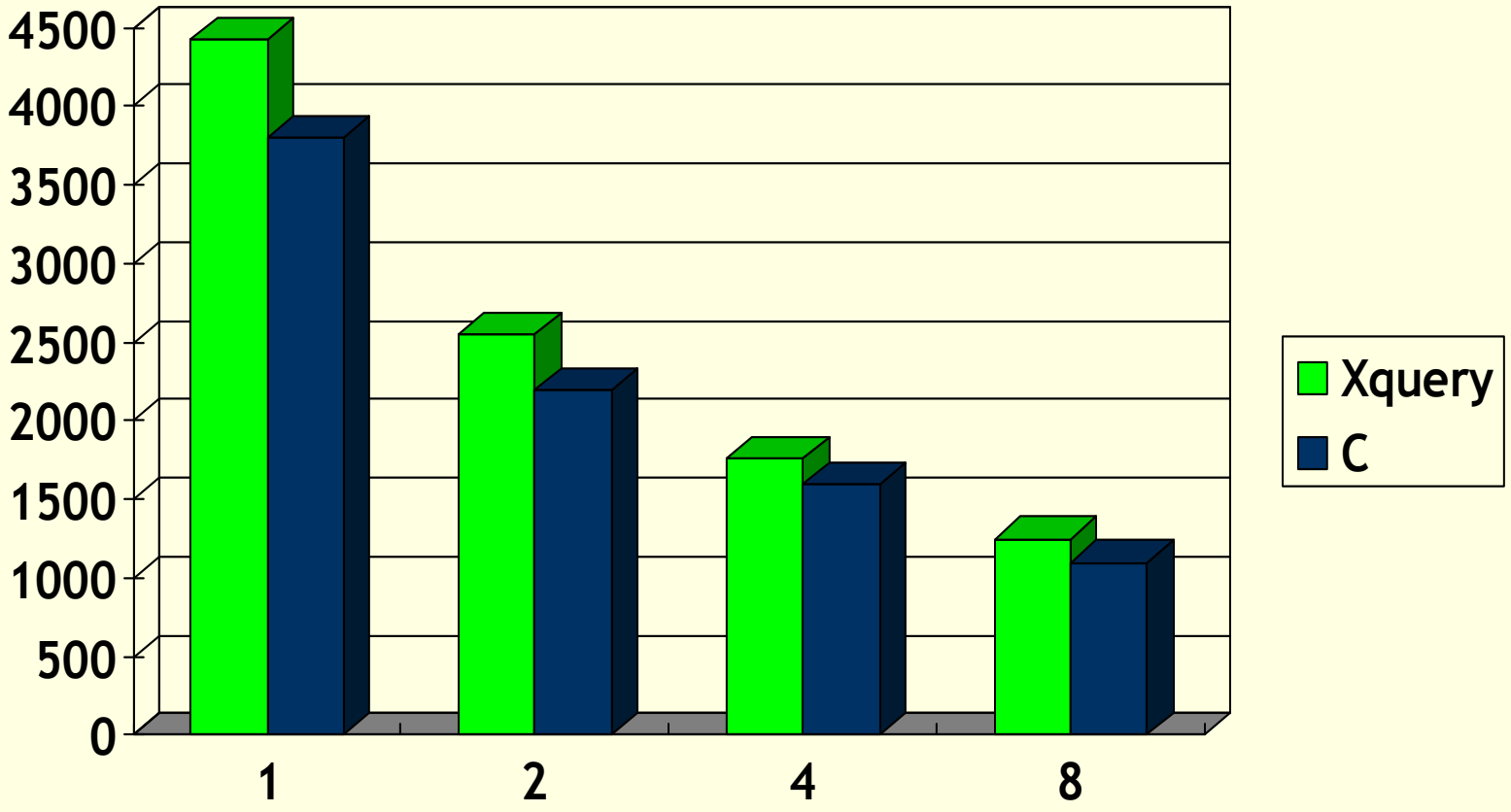


Virtual Microscope

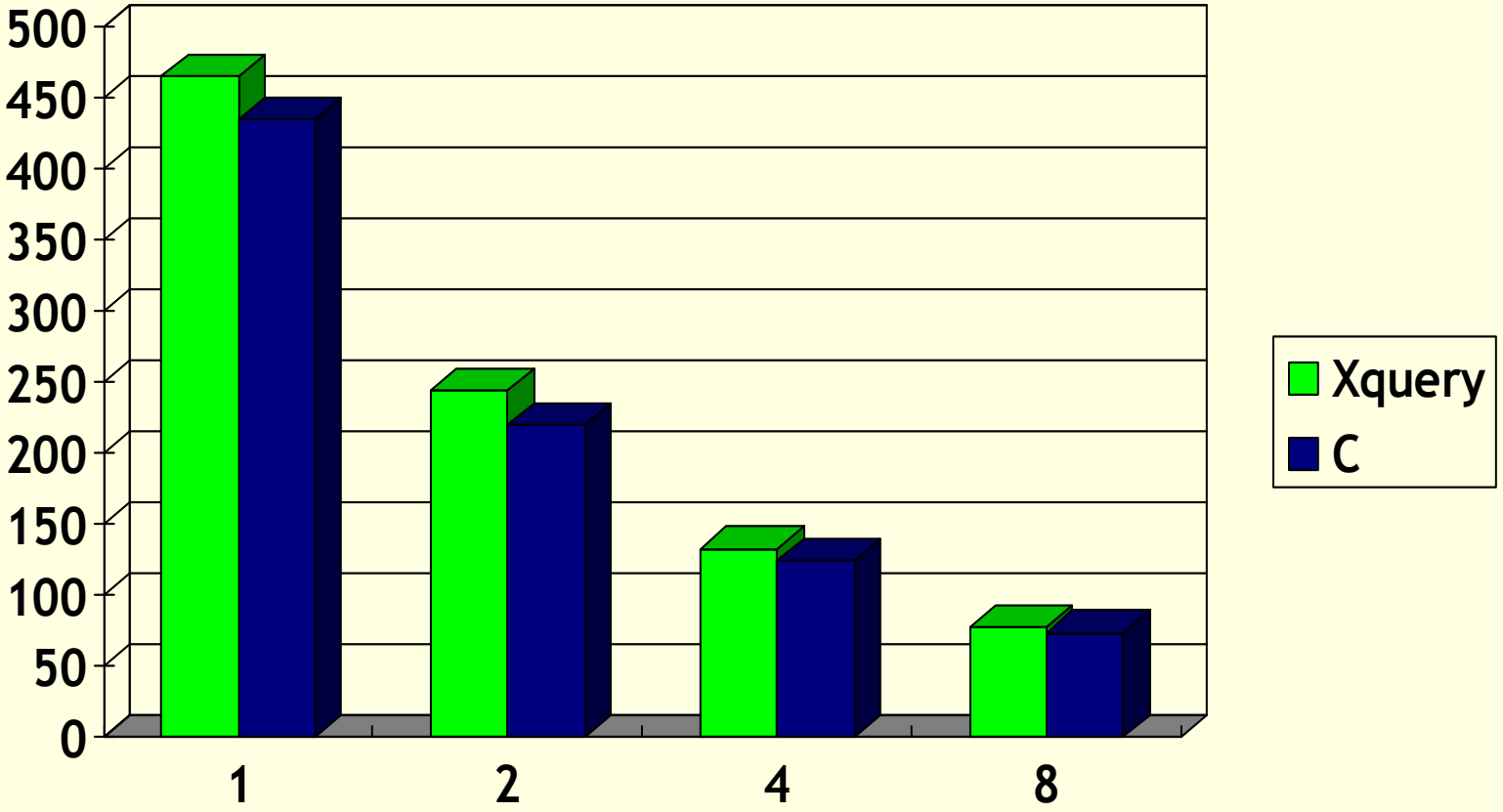


Satellite

Comparison with Manual - VMScope



Comparison with Manual - Satellite



Conclusions

- A case for the use of XML technologies in scientific data analysis
- XQuery - a data parallel language ?
- Identified and addressed compilation challenges
- A compilation system has been built
  - Very large performance gains from data-centric transformations
  - Preliminary evidence that high-level abstractions and query language do not degrade performance substantially