IBM Research

# Stream Processing of XML Documents

Vivek Sarkar

Programming Technologies Department

IBM T.J. Watson Research Center

vsarkar@us.ibm.com

# Acknowledgments


- Team members of DOMO (Data-Object Modeling and Optimization) project at IBM:
  - Rajesh Bordawekar
  - Michael Burke
  - Brendon Cahoon
  - Steve Fink
  - Mukund Raghavachari
  - Vivek Sarkar
  - Oded Shmueli


- References:


1. “Streaming XPath Processing with Forward and Backward Axes”, C.Barton et al. Proceedings of ICDE 2003, March 2003.
2. “Integrating Database and Programming Language Constraints”, M.Raghavachari et al. To appear in Proceedings of the 9th International Workshop on Data Base Programming Languages (DBPL), September 2003.



- Supported in part by DARPA under contract No. NBCHC020056

# Why is XML processing important?

- XML is a universal data interchange format for Web Services, Databases, Messaging, Grid Computing
- XML is based on open standards: XML Schema, XPath, XQuery, XSLT, …
- Relevance to database systems:
  - New XML data type and new XML-related operators
  - Queries over locally stored XML data and XML data streamed from external sources
- Relevance to Messaging and Publish-Subscribe systems:
  - XPath used to specify content-based subscriptions
  - System needs to scan a message quickly to determine if it satisifes a given predicate
  - Similar issues arise in routing/processing of SOAP messages in Web Services

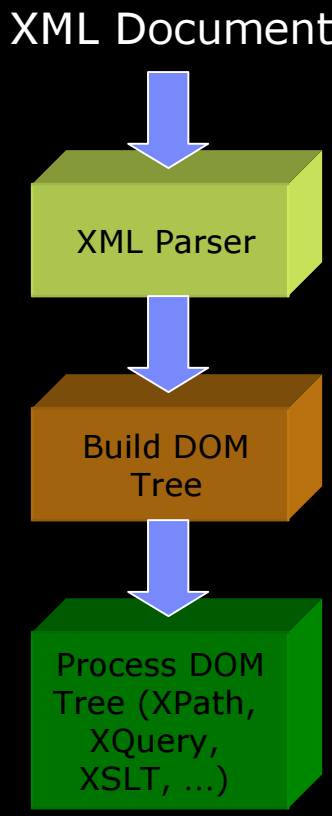
# Goals for XML processing

1. **Data Scalability**
   Ability to process large documents (larger than in-core memory)
2. **Efficiency**
   a. Locality --- improved exploitation of memory hierarchies
   b. Algorithmic --- use of linear-time algorithms
3. **Programmability**
   Use of high-level primitives instead of low level APIs

➔XML processing is an ideal candidate for streaming …

… but is it an ideal candidate for today's stream processors?

# XML Processing: Current Practice using DOM and SAX




**DOM approach:** build in-core representation of XML document, and process as needed using standard APIs

- Disadvantages:
  - Scalability --- cannot process large documents
  - Locality ---- even if document fits in memory, multiple DOM traversals exhibit poor locality
  - Algorithmic inefficiencies --- APIs perform multiple DOM traversals, leading to super-linear execution times

**SAX approach:** use a streaming event-based API for on-the-fly parsing of XML

- Disadvantages:

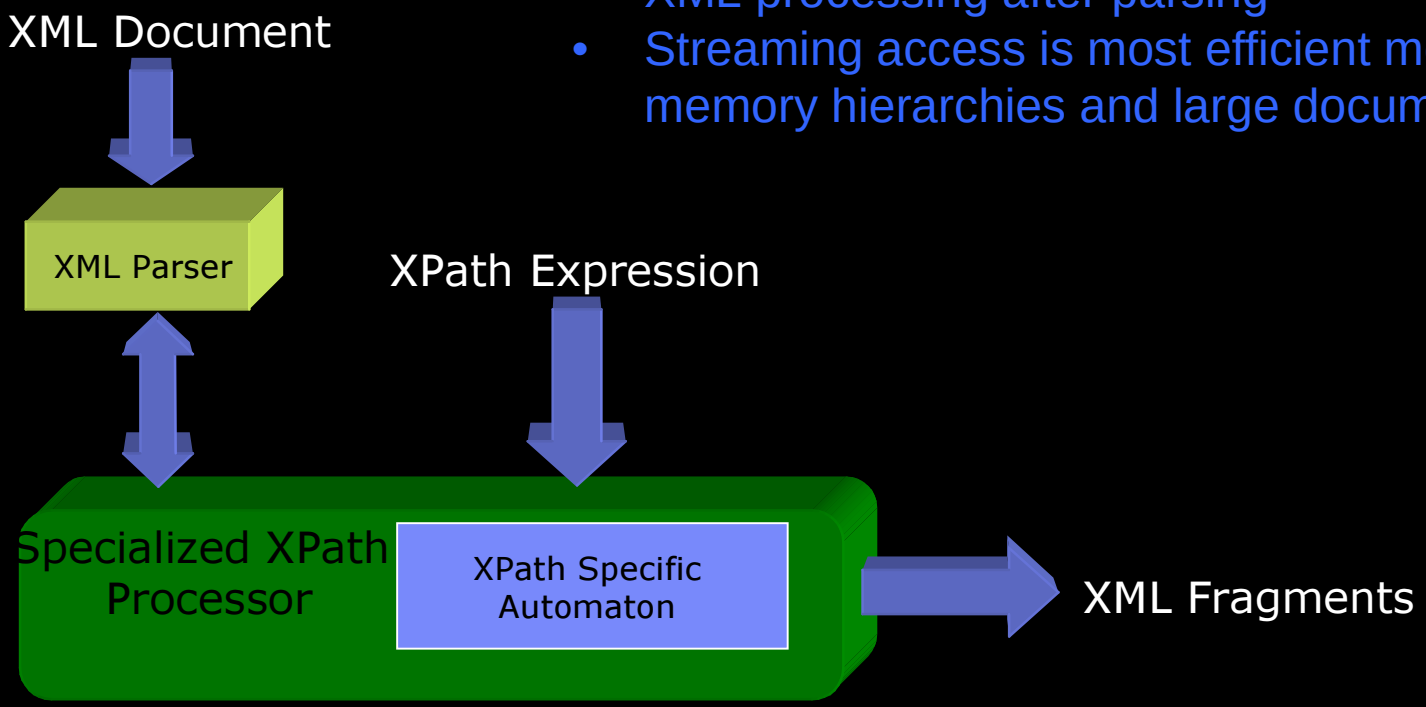
  Programmability: low-level event handling interface

  Limited functionality: no support for XPath search (especially with parent/ancestor axes)

# XML Processing: Our Approach

Motivation:

- XPath lookup is next major bottleneck in XML processing after parsing
- Streaming access is most efficient mode for memory hierarchies and large documents

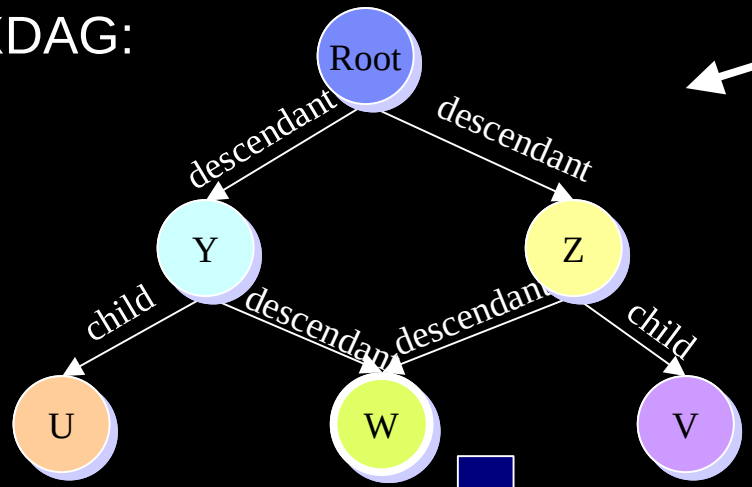



XAOS (Xpath Analysis and Optimization for Streaming)

# XAOS Example

//y[u]//w[ancestor::z/v]

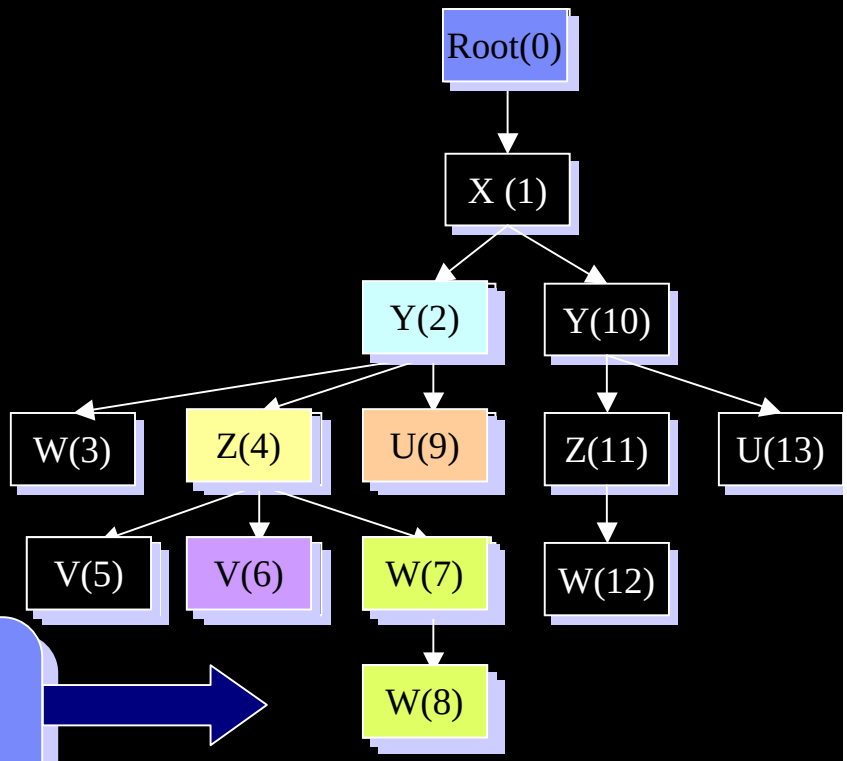
XDAG:




**XML Document**

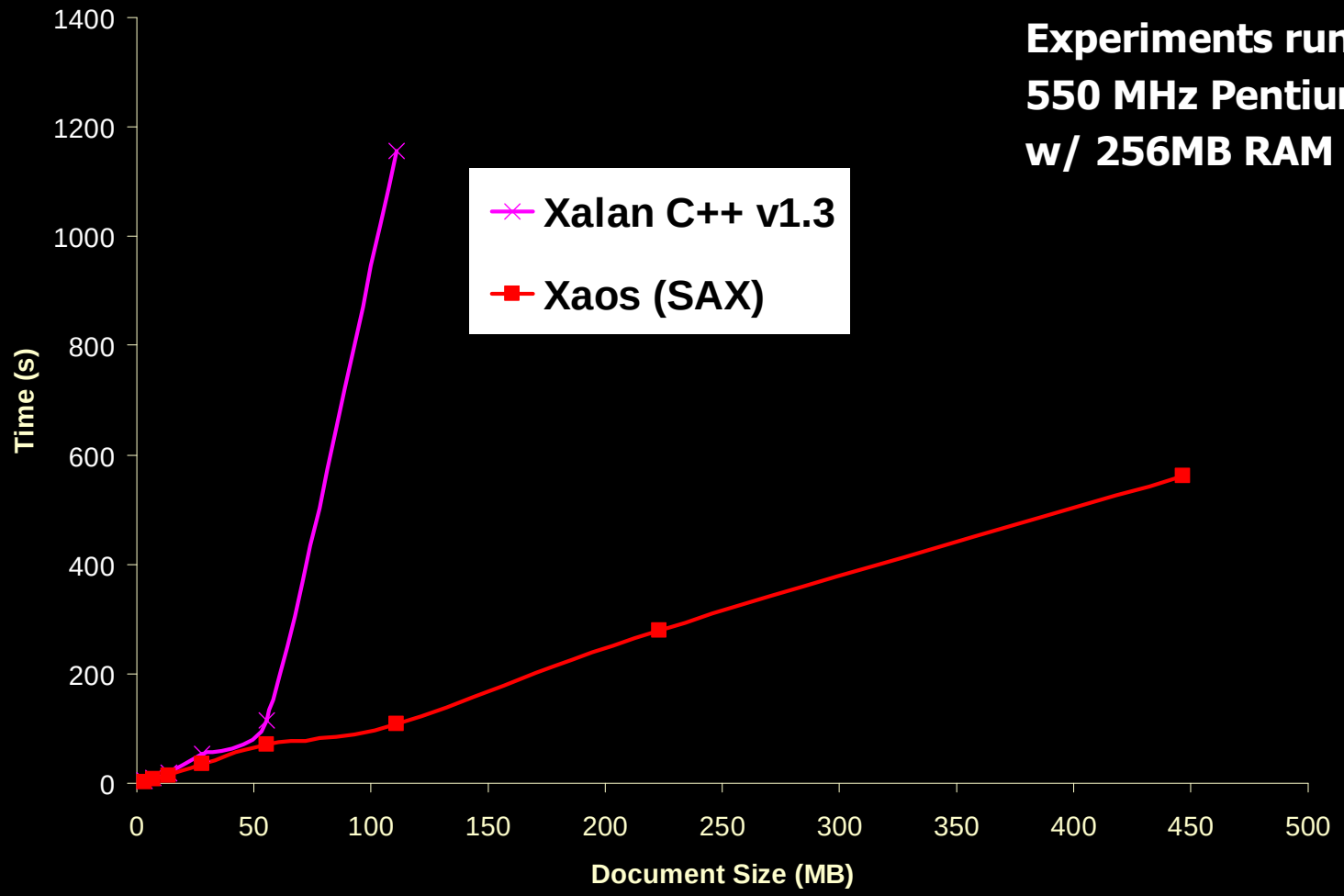
Parser + Specialized XPath Processor

Xpath-specific Automaton



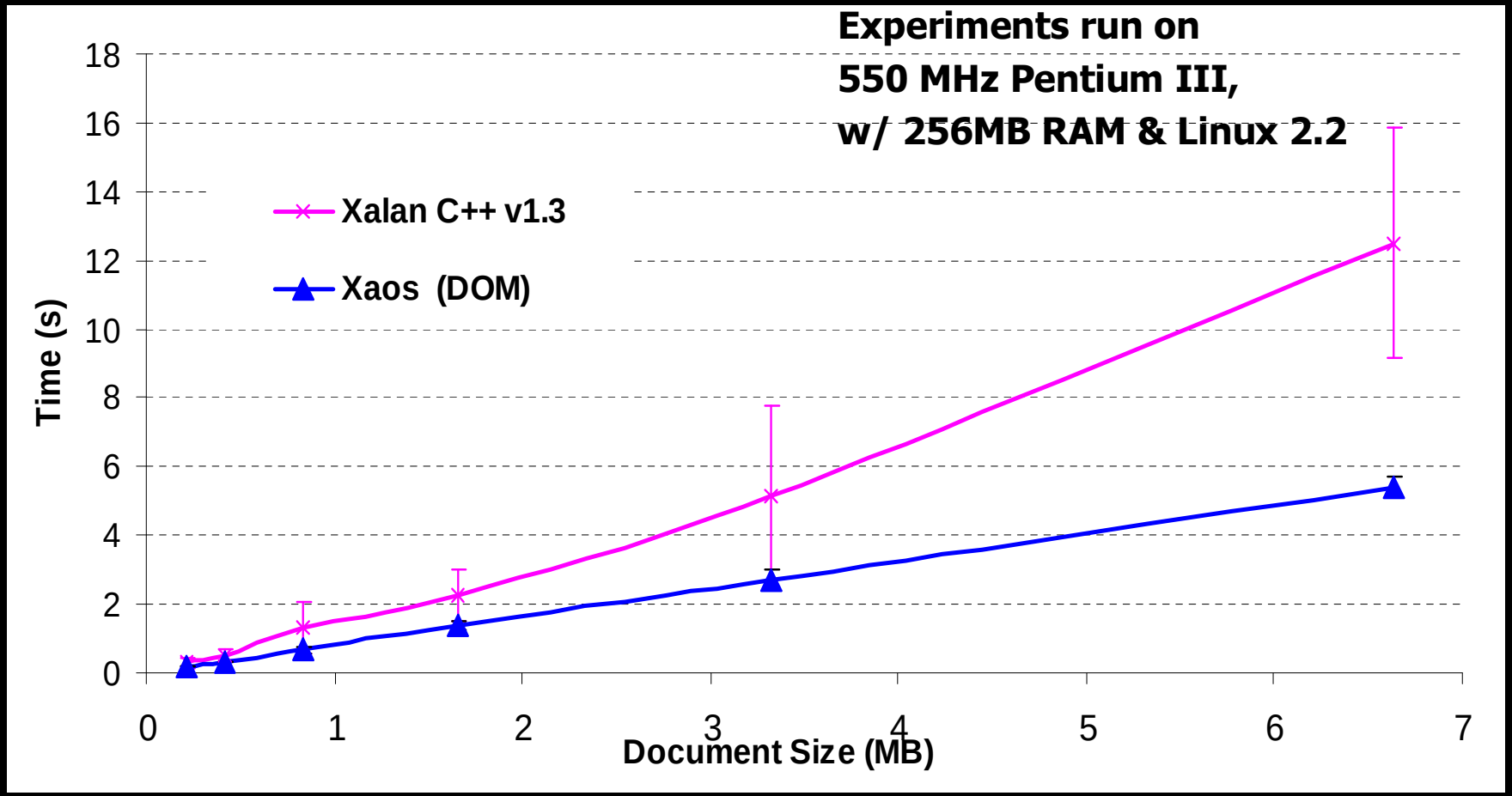

XPath Matchings in XML Document

# XAOS Scalability: Parsing + XPath Execution times




**Experiments run on**
**550 MHz Pentium III,**
**w/ 256MB RAM & Linux 2.2**

# XAOS In-Core Efficiency: XPath Search Time



# XAOS : Summary

- Streaming Algorithm for XPath evaluation that can handle expressions with both forward and backward (parent and ancestor) axes.
  - Handles recursive documents.
  - Handles multiple output nodes.
- Experiments demonstrate significant performance improvements:
  - Increased scalability : Can process documents over 1GB in size.
  - Greater than 2x performance improvement in searching time for in-core documents
- Future Work:
  - Develop programming model for stream processing of XML
  - Extend streaming paradigm to XML processing beyond XPath
  - Assemble set of benchmarks for representative XML processing scenarios to demonstrate benefits of streaming