zapthink
zapnote

Doc. ID: ZTZN-1191
Released: Apr. 21, 2006

ZAPTHINK ZAPNOTE™

RAX-J

RADICALLY IMPROVING THE PROCESSING OF XML APPLICATIONS

Analyst: Ronald Schmelzer

Abstract

XML is rapidly becoming the protocol and format of choice for interactions among disparate systems and organizations connected via networks. More than a text-based, metadata format for data interoperability, XML is now the answer to solving many of the long-standing issues with application and data integration as well as providing a *lingua franca* for developers to create application programming interfaces (APIs) that arbitrary systems can interact with.

Despite all the positive momentum that XML continues to garner in the enterprise, XML remains a highly inefficient and burdensome protocol to process. XML processing requires a dozen steps or more, including parsing, decryption, validation, and message transformation activities. This burdensome collection of tasks is increasingly bogging down systems with menial chores before they can even begin processing business logic. This paper aims to take the XML processing challenge one step further by suggesting that developers are the cause of many of the problems in their use of the increasingly obsolete DOM and SAX methods of XML parsing. Instead, this paper suggests a new approach to XML processing that improves upon the increasingly obsolete DOM and SAX methods of XML parsing: Random Access XML for Java, (RAX-J).

The Challenge of Efficient XML Processing

The evolution towards text-based metadata standards that the Extensible Markup Language (XML) spearheads continues unabated as companies seek the benefits of interoperability, reuse, and business agility. Much of this new kind of traffic on the network shifts control over the behavior of distributed systems toward metadata, which are content-oriented, rather than binary protocol-oriented, and as such, solutions responsible for routing, managing, securing, and processing XML traffic must make decisions based upon the content of the messages, rather than the protocols that underlie those messages. As a result, XML consumes significant bandwidth, processing, and storage capacity. XML processing tasks such as document transformation, parsing, message validation, classification, security, and intelligent routing are inherently processing-intensive, placing a significant burden on server infrastructure.

At the same time, enterprises are looking to add more layers of functionality, and thus complexity, to the XML traffic on the network. Simple point-to-point XML-based exchanges in free-text format are no longer sufficient to realize the benefits that XML promises. Companies increasingly desire a set of requirements for XML-aware network processing such as routing, transformation, compression, caching, security, and management tasks. As a result, as network traffic based on XML increases, IT data center administrators and developers are quickly realizing that the operational inefficiencies of XML are bogging down their general-purpose hardware and software. The addition of more advanced security, reliability, and process capabilities puts an overwhelming burden on existing network infrastructure that is already stretched to the limit handling basic XML processing tasks.

Why is XML inefficient? Text-based, metadata-laden XML is suitable for both machine processing and human readability. The combination of these two purposes results in message sizes that are easily 10 to 50 times larger than corresponding messages written in purpose-built, binary protocols. XML conversant endpoints must perform the steps to receive, decrypt, validate, parse, transform, perform business logic, serialize, canonicalize, sign, encrypt, and transmit on a per-message basis, imposing a significant load on processing machines. ZapThink research has shown that basic XML tasks such as canonicalization represent over 93% of the total processing time for processing of simple documents like an XML-Signature document (a typical XML document).

Add to these processing requirements the need for additional parsing and validation steps for XML schema, the growing number and complexity of security, reliability, and process headers, the need for partial-message security, XPath and XQuery processing of documents, message compression and decompression, business logic validity checking, and message integrity validation. The result is a bandwidth, processing, and storage nightmare results that grows in cost and complexity over time. Clearly, general-purpose processors, off-the-shelf software parsers and validation engines, application servers, and broad-based security solutions are not sufficient to meet the burgeoning XML processing challenge.

To date, two models have dominated XML processing: the *Document Object Model* (DOM) and the *Simple API for XML* (SAX). DOM is an in-memory approach to processing XML which loads an XML document into memory represented by a tree of nodes, which developers can then interact with through standard programming operations. The DOM is therefore simple for developers to implement and provides direct access to XML documents, but the construction of DOM nodes in memory makes it incredibly inefficient and resource-expensive for large documents.

An alternative to the DOM API is the SAX, which enables the processing of large XML documents as simply as if they were streaming files. Through the SAX API, developers register callbacks that are invoked as various parts of an XML document are parsed, requiring less memory than DOM approaches, but requiring much more complicated and sophisticated interaction with XML documents, especially if developers require multiple callbacks. Furthermore, users themselves must track certain context information, such as parent-child relationships.

Despite the inefficiency of these two approaches, the use of DOM and SAX for XML processing still represents the vast majority of all implementations, even though that neither approach is well-suited for hardware acceleration or software optimization. Rather than simply applying brute force XML acceleration technologies to DOM and SAX processing, companies must replace these methods of XML processing with a method that is both efficient to use as an end-point XML processing approach and well-suited for hardware acceleration and optimization.

RAX-J: Revolutionizing XML Processing

This new model of XML processing relies upon the concept of an *XML Cursor*. A cursor is like a sliding window over an XML document, exposing individual nodes of arbitrarily large and complex XML documents simultaneously. The Cursor API borrows some of the best traits of traditional SAX and DOM models: like SAX, it is lightweight and does not overly burden the processing system from a memory or processing requirement, and like DOM, it provides a simplified development approach that allows users to navigate anywhere in an XML document without undue constraints. Users simply navigate through XML documents using a cursor that provides a sequence of logical positions, called nodes, that developers can use to navigate to any other arbitrary node, get the node name, value, namespace context and other information, serialize the sub-tree rooted at the current node to various outputs, and/or apply XPath-defined queries. This cursor-based approach is part of the overall movement to token-oriented processing for documents that require the production of tokens for everything in the XML file and navigating on them instead of objects that represent nodes.

The primary challenges with the XML Cursor model for XML processing is that it's relatively new, and little software is available on the market for widespread adoption of the above techniques. Fortunately, a new approach for XML Cursor based processing known as *Random Access XML for Java* (RAX-J) is now available. RAX-J in essence recreates the current DOM and SAX-based XML processing stack and replaces those inefficient versions with an XML Cursor-based optimized stack. This RAX-J stack includes optimized versions of XPath, XSLT, XML Security, and other XML operations essential for XML and Web Services-based activities. As a result, users can switch to RAX-J approaches for XML processing without having to give up any existing, easy-to-use approaches for XML operations.

Tarari: Optimizing RAX-J in Hardware

Simply moving to RAX-J, without even adding any hardware or software acceleration, will give users a noticeable improvement in the operation of their systems. Not only optimized XML

processing, but also better memory handling, improved speed, and lowered cost of XML document navigation should provide users anywhere from 60% to 200% improvement in overall XML processing based on the cursor-based approach. All RAX-J features offer a software-only mode that delivers the functional equivalent of hardware with improved performance. RAX-J is thus very flexible in its development and deployment options. However, the cursor model is optimized for hardware acceleration. RAX-J takes full advantage of hardware-assisted processing that Tarari's Random Access XML Content Processor (RAX-CP) product supports, for example With this new XML document model, RAX-J enables developers to build end-to-end, standards-based XML applications with unprecedented performance.

RAX-J complies with all key XML standards, and offers complete implementations of the following W3C standards: XML 1.0, Namespaces in XML, XML Schema 1.0, XPath 1.0, XSLT 1.1, Canonical XML 1.0 and Exclusive XML Canonicalization 1.0. In addition, the package offers RAX-XSLT, which is a new approach to the old problem of improving XSLT performance. RAX-XSLT is optimized for the top performance use case for XSLT processing – transactional XML, network-based XML services, and Web Services. Tight integration between XSLT transformation processing and the underlying data model further optimizes performance, and the data model and transformation implementation is compatible with dynamic and adaptive optimization strategies.

Finally, Tarari offers a tool called *RAX Profiler for XSLT* (RPX), which identifies transformation processing bottlenecks and optimizes XSLT scripts. The profiler is not specific to the token-cursor data model, but when developers combine it with the RAX-J token and XML Cursor model for processing, they can realize substantial performance gains. After all, XML has proven to be a significant value for both IT and the business, it is now time to focus the emphasis on removing any remaining technological barriers to XML's long-term growth in the enterprise.

The ZapThink Take

Since XML is still an emerging area for many businesses, some companies are applying poor development practices, resulting in a variety of inefficient implementations that worsen the XML processing problem. A first step that companies should take to improve XML performance is to simply eliminate these poor practices and replace them with more effective methods. After all, the inefficiencies of XML rapidly stack up when developers build XML documents that contain frequently repeated element and tag sets. Before sending out XML documents on the wire, developers should seek to maximize performance by minimizing this unnecessary redundancy. It is clear that developers are part of the challenge to widespread adoption of XML, not because they are misapplying XML's use, but rather because they aren't optimizing their infrastructure in such a way that reinforces XML's use in the enterprise. Instead, developers are introducing key performance bottlenecks and stumbling blocks that prohibit the organization's long-term adoption of XML. Therefore, to make widespread adoption of XML a reality, not only do companies require changes to their architecture and overall approach, but also to the low-level infrastructure for the message-by-message processing of critical XML documents. ZapThink hopes that approaches like RAX-J contribute to the ever-improving performance, and thus ROI, of XML on the network.

Related Research

- *Service Orientation Market Trends* Report (ZTR-WS110)
- *SOA Tools and Best Practices* Report (ZTR-WS107)
- *High Performance and Appliance Approaches for XML* Report (ZTR-DI102)

- *Xenos* ZapNote (ZTZN-1190)
- *Reactivity* ZapNote (ZTZN-1180)
- *Conformative Systems* ZapNote (ZTZN-1171)
- *DataPower* ZapNote (ZTZN-1159)
- *Forum Systems* ZapNote (ZTZN-1170)
- *Sarvega* ZapNote (ZTZN-1165)

About ZapThink, LLC

ZapThink is an IT advisory and analysis firm that provides trusted advice and critical insight into the architectural and organizational changes brought about by the movement to XML, Web Services, and Service Orientation. We provide our three target audiences of IT vendors, service providers and end-users a clear roadmap for standards-based, loosely coupled distributed computing – a vision of IT meeting the needs of the agile business.

ZapThink helps its customers in three ways: by helping companies understand IT products and services in the context of Service-Oriented Architecture (SOA) and the vision of Service Orientation, by providing guidance into emerging best practices for Web Services and SOA adoption, and by bringing together all our audiences into a network that provides business value and expertise to each member of the network.

ZapThink provides market intelligence to IT vendors and professional services firms that offer XML and Web Services-based products and services in order to help them understand their competitive landscape, plan their product roadmaps, and communicate their value proposition to their customers within the context of Service Orientation.

ZapThink provides guidance and expertise to professional services firms to help them grow and innovate their services as well as promote their capabilities to end-users and vendors looking to grow their businesses.

ZapThink also provides implementation intelligence to IT users who are seeking guidance and clarity into the best practices for planning and implementing SOA, including how to assemble the available products and services into a coherent plan.

ZapThink's senior analysts are widely regarded as the "go to analysts" for XML, Web Services, and SOA by vendors, end-users, and the press. Respected for their candid, insightful opinions, they are in great demand as speakers, and have presented at conferences and industry events around the world. They are among the most quoted industry analysts in the IT industry.

ZapThink was founded in October 2000 and is headquartered in Baltimore, Maryland. Its customers include Global 1000 firms and government organizations, as well as many emerging businesses. Its analysts have worked at such firms as IDC, marchFIRST, and ChannelWave, and have sat on the working group committees for standards bodies such as RosettaNet, UDDI, and ebXML.

Call, email, or visit the ZapThink Web site to learn more about how ZapThink can help you to better understand how SOA will impact your business or organization.

ZAPTHINK CONTACT:
ZapThink, LLC
108 Woodlawn Rd
Baltimore, MD 21210
Phone: +1 (781) 207 0203
Fax: +1 (786) 524 3186
info@zapthink.com
www.zapthink.com