รายงานฉบับสมบูรณ์

การบีบอัดข้อมูล XML โดยไม่ใช้โครงสร้างของเอกสาร

Schema Free on XML Data Compression

รหัสโครงการ

F-31-205-12-07

ผศ. ดร. กานดา รุณนะพงศา
นายประพันธ์ เลขาโสภณ

16 ตุลาคม 2550

รายงานฉบับสมบูรณ์

การบีบอัดข้อมูล XML โดยไม่ใช้โครงสร้างของเอกสาร

Schema Free on XML Data Compression

รหัสโครงการ

F-31-205-12-07

รายนามคณะผู้วิจัย

1. ผศ. ดร. กานดา รุณนะพงศา
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
มหาวิทยาลัยขอนแก่น อ. เมือง ขอนแก่น 40002
2. นายประพันธ์ เลขาโสภณ
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
มหาวิทยาลัยขอนแก่น อ. เมือง ขอนแก่น 40002

วันเริ่มต้นโครงการ 1 ตุลาคม 2548

วันสิ้นสุดโครงการ 16 ตุลาคม 2550

กิตติกรรมประกาศ

โครงการวิจัยนี้ได้รับการสนับสนุนจาก

ศูนย์ประสานงานนักเรียนทุนรัฐบาลทางด้านวิทยาศาสตร์และเทคโนโลยี

กระทรวงวิทยาศาสตร์และเทคโนโลยี

และสำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ

ปีงบประมาณที่ได้รับทุน ปีงบประมาณ 2548

บทคัดย่อ

ภาษาเอกซ์เอ็มแอล (Extensible Markup Language, XML) ในปัจจุบันได้วิวัฒนาการมาเป็นภาษามาตรฐานในการเสนอและแลกเปลี่ยนข้อมูลบนอินเทอร์เน็ต เนื่องจากภาษาเอกซ์เอ็มแอลเป็นภาษาที่ใช้งานง่ายมีความยืดหยุ่น และไม่ขึ้นอยู่กับระบบปฏิบัติการของเครื่องคอมพิวเตอร์ แต่ว่าข้อมูลที่เป็นภาษาเอกซ์เอ็มแอลนั้นมักจะมีขนาดใหญ่และมีข้อมูลที่ซ้ำซ้อนอันเนื่องมาจากการใช้แท็กที่ซื่อซ้ำๆ กันในการอธิบายข้อมูล เอกสารเอกซ์เอ็มแอลจึงมีขนาดใหญ่ซึ่งทำให้ต้องการพื้นที่ในการจัดเก็บในปริมาณมากและใช้เวลานานในการส่งข้อมูล ดังนั้นงานวิจัยที่ต้องการจะบีบอัดข้อมูลในภาษาเอกซ์เอ็มแอลจึงมีความจำเป็นอย่างยิ่ง ในงานวิจัยที่ได้ทำมาแล้วนั้น ข้อมูลที่ถูกบีบอัดแล้วอาจจะอยู่ในรูปแบบของไบนารีหรือรูปแบบที่เฉพาะผู้พัฒนาเครื่องมือเท่านั้นที่เข้าใจ ในงานวิจัยนี้ได้นำเสนอวิธีบีบอัดข้อมูลแบบใหม่ชื่อว่า “เอกซ์เบร็ววิตี้ (XBrevity)” ซึ่งเป็นวิธีการที่ใช้ในการบีบอัดข้อมูลหรือคลายการบีบอัดข้อมูลโดยที่ข้อมูลที่ถูกบีบอัดสามารถเป็นที่เข้าใจได้ง่ายกับคนทั่วไป โดยที่วิธีบีบอัดที่นำเสนอนี้สามารถจะใช้ได้กับเอกสารเอกซ์เอ็มแอลใดๆ รวมทั้ง สามารถใช้กับเอกสารเอกซ์เอ็มแอลที่ไม่มีเอกสารที่อธิบายโครงสร้างของเอกสารเอกซ์เอ็มแอล (เอกสาร XML Schema)

งานวิจัยนี้เสนอแนวทางการบีบอัดข้อมูลโดยที่เอกสารที่ถูกบีบอัดอยู่ในรูปแบบของภาษาเอกซ์เอ็มแอลแบบย่อซึ่งจะทำให้เอกสารนั้นยังคงข้อดีของภาษาเอกซ์เอ็มแอล ขอบเขตของงานวิจัยนี้ไม่ได้มีเป้าหมายในการบีบอัดเอกสารเอกซ์เอ็มแอลให้ขนาดของเอกสารเอกซ์เอ็มแอลมีขนาดเล็กที่สุดเมื่อเปรียบเทียบกับวิธีการบีบอัดข้อมูลอื่นๆ แต่งานวิจัยนี้มีจุดประสงค์ให้ข้อมูลของเอกสารเอกซ์เอ็มแอลที่ถูกบีบอัดแล้วเป็นที่เข้าใจง่ายและยังอยู่ในรูปแบบของเอกสารเอกซ์เอ็มแอล วิธีการวิจัยเริ่มจากการออกแบบโครงสร้างของเอกสารเอกซ์เอ็มแอลในรูปแบบย่อ ออกแบบอัลกอริธึมที่ใช้ในโปรแกรมที่ใช้ในการบีบอัดและการคลายการบีบอัด จากนั้นทำการพัฒนาโปรแกรม และทำการวัดประสิทธิภาพของการบีบอัดตลอดจนเวลาที่ใช้ในการบีบอัด โดยใช้ข้อมูล เอกซ์เอ็มแอล จากการดาวน์โหลดไฟล์ เอกซ์เอ็มแอล จากอินเทอร์เน็ตและใช้โปรแกรม XMark ในการสร้างไฟล์เอกซ์เอ็มแอลแบบสุ่ม จากผลการทดลองพบว่าโปรแกรมบีบอัดข้อมูลสามารถทำให้ข้อมูลเล็กลง และได้ข้อมูลที่อยู่ในรูปแบบเอกซ์เอ็มแอล

คำสำคัญ: เอกซ์เอ็มแอล การบีบอัดข้อมูล การคลายการบีบอัดข้อมูล

Abstract

eXtensible Markup Language (XML) becomes the standard language for data representation and exchange on the Internet because it is simple, flexible, and platform independent. However, XML data is often large and verbose since it consists of many repeated tags which are used to self-describe data. The large size of data results in an excessive amount of space required for storing data on the disk space and time required for transmitting data over the network. Thus, it is necessary to find an effective compression technique for XML data. In previous compression techniques, the compressed data is in the binary form or in the format that is understandable for only a few researchers who own those techniques. In this work, we propose XBREVITY, an XML compressor which supports compressing and uncompressing XML data. XBREVITY adopts a novel encoding method that has the compressed XML data in the format that is easy to understand for anyone. This method can apply for any XML document including one without its associated XML Schema file defined.

This research work proposes the method in compressing the original XML data into the compressed XML data using a brevity format. Thus, the compressed XML data still preserves the advantage features of XML but the XML document has a smaller size. The scope of this research work does not aim to minimize the size of the compressed document to be the smallest among all compressing methods. The goal of this research work is rather to make the compressed data easy to understand and still be in the XML format. The research method consists of designing the brevity form of compressed XML document, designing the algorithms in compressing and decompressing XML data, developing the programs, and measuring the effectiveness and time in compressing XML data. To test the compression and decompression techniques, we use input files which are downloaded from the Internet and the random generated files using XMark. Based on the experimental results, we found that the compressed data is in the XML file format and has the smaller size.

Keywords: XML, data compression, data decompression

สารบัญเรื่อง

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญตาราง

บทนำ

ปัจจุบัน XML (Extensible Markup Language) [20] ได้เข้ามามีบทบาทและเป็นมาตรฐานในการแลกเปลี่ยนข้อมูล เนื่องจาก XML มีความสามารถในการอธิบายความหมายของข้อมูลและมีความยืดหยุ่นในการใช้งาน การนำ XML มาใช้งานสามารถทำได้โดยการใช้แท็กเป็นตัวกำกับและการตั้งชื่อแท็กที่สื่อถึงความหมายของข้อมูล ทำให้เอกสารที่ถูกสร้างขึ้นเข้าใจได้ง่าย จึงเป็นส่วนสำคัญที่ทำให้การเข้าถึงข้อมูลได้ง่ายขึ้น แต่จากการที่มีการใช้แท็กเข้ามาช่วยในการสื่อถึงความหมายทำให้เกิดการบันทึกแท็ก ชนิดเดียวกันบ่อยครั้งในเอกสาร

```
<?xml version="1.0" encoding="UTF-8"?>
<weblog>
<entry>
<host>133.5.33.76</host>
<referer>-</referer>
<userAgent>-</userAgent>
<dateTime>9/Aug/2001:00:05:30</dateTime>
<reqID>-0500</reqID>
<reqType>GET</reqType>
<resource>/publish/programming/pippy1.gif
</resource>
<protocol>HTTP/1.1</protocol>
<statusCode>200</statusCode>
<byteCount>5440</byteCount>
</entry>
<entry>
<host>133.5.33.76</host>
<referer>-</referer>
```

```
<userAgent>-</userAgent>
<dateTime>9/Aug/2001:00:05:30</dateTime>
<reqID>-0500</reqID>
<reqType>GET</reqType>
<resource>/publish/programming/pippy_multi
eval.gif</resource>
<protocol>HTTP/1.1</protocol>
<statusCode>200</statusCode>
<byteCount>4236</byteCount>
</entry>
</weblog>
```

ภาพที่ 1 ตัวอย่างเอกสาร XML ที่ใช้งานจริง

จากภาพที่ 1 จะเห็นได้ว่ามีการบันทึกข้อมูลประเภทเดียวกันหลายครั้ง ซึ่งในแต่ละครั้งจะมีรายละเอียดแตกต่างกัน ด้วยเหตุนี้ ขนาดของเอกสารจึงมีขนาดใหญ่เมื่อเทียบกับขนาดข้อมูลจริงภายในเอกสารนั้น ส่งผลให้สิ้นเปลืองเนื้อที่หากต้องการจัดเก็บเอกสารและสิ้นเปลืองเวลาในการรับส่ง หากต้องการแลกเปลี่ยนข้อมูลระหว่างองค์กรผ่านระบบเครือข่าย โดยทั่วไปแล้วข้อมูล XML จะเก็บอยู่ในรูปแบบของไฟล์ ดังนั้นข้อมูล XML ที่จะมีการนำไปใช้กันได้จริงควรจะเป็นการบีบอัดไฟล์ XML ที่ทำให้มีขนาดเล็กและนำไปใช้งานได้ง่าย เนื่องจากข้อมูล XML จำนวนมากจะเก็บไว้ในไฟล์ ดังนั้นการบีบอัดไฟล์ XML จึงมีความจำเป็นอย่างยิ่ง

ปัจจุบัน XML ได้ถูกนำมาใช้งานในหลายสาขาวิชาชีพ ไม่ว่าจะเป็นทางธุรกิจซึ่งมีการนำ ebXML (Electronic Business XML) [18] และ BPEL (Business Process Execution Language) [2] ไปใช้ด้านกราฟิกซึ่งมีการนำ SVG (Scalable Vector Graphics) [4] ไปใช้ หรือแม้กระทั่งด้านวิทยาศาสตร์ สาขาเคมี ซึ่งมีการนำภาษา CML (Chemical Markup Language) [14] ไปใช้ ไม่ว่าจะเป็น BPEL, SVG, หรือ CML ต่างก็เป็นภาษา XML ประเภทหนึ่ง

นอกจากนี้แอปพลิเคชันอีกอันหนึ่งที่สำคัญของภาษา XML ซึ่งก็คือเว็บเซอร์วิส (Web Services) [26] เป็นซอฟต์แวร์ที่ได้นำเอามาใช้อย่างมากในปัจจุบัน และคาดว่าจะมี

การใช้อย่างแพร่หลายมากขึ้นในอนาคต ในปัจจุบันได้มีการนำเว็บเซอร์วิสเอามาใช้ในการให้บริการ ผ่านอินเทอร์เน็ตโดยบริษัทชั้นนำ อาทิเช่น Yahoo Search Web Services [28], Google Web APIs [6], Amazon Web Services [1], และ eBay API [3] จุดเด่นของเทคโนโลยีเว็บเซอร์วิสคือ การที่มันทำให้โปรแกรมที่พัฒนาโดยภาษาและใช้แพลตฟอร์มที่แตกต่างกันสามารถติดต่อ และทำงานร่วมกันได้โดยใช้ภาษา XML เป็นภาษาสื่อกลางในการแลกเปลี่ยนข้อมูล ฉะนั้นจะเห็นได้ว่าข้อมูล XML จะมีปริมาณการใช้งานเพิ่มมากขึ้น และขนาดของข้อมูลของ XML ที่มักจะมีขนาดใหญ่จะส่งผลกระทบต่อประสิทธิภาพการทำงานของแอปพลิเคชันที่ใช้ XML เป็นภาษาในการบันทึกข้อมูล

จากปัญหาในเรื่องขนาดของข้อมูลที่มักจะใหญ่นั้น แนวทางที่สามารถนำมาใช้ในการแก้ปัญหาได้คือ การบีบอัดข้อมูล XML (XML data compression) เพื่อลดขนาดของเอกสารซึ่งเป็นการบีบอัดข้อมูล XML โดยเฉพาะ ทำให้สามารถเพิ่มประสิทธิภาพในการบีบอัดได้ดีกว่าการใช้วิธีการบีบอัดข้อมูลทั่วไป ฉะนั้นในการทำวิจัยครั้งนี้ทำเพื่อการแก้ไขปัญหาในเรื่องขนาดที่ใหญ่ของข้อมูล XML โดยการบีบอัดข้อมูลแบบไม่ใช้ Schema ซึ่งการที่ไม่ใช้ Schema จะทำให้สามารถบีบอัดข้อมูล XML ได้กับทุกเอกสารและทำการเข้ารหัสข้อมูลเพื่อให้ได้ข้อมูลที่มีขนาดเล็กลง [9]

วัตถุประสงค์ของโครงการวิจัย

1. เพื่อศึกษาและพัฒนาอัลกอริทึม (Algorithm) ในการบีบอัดข้อมูล
2. เพื่อลดขนาดของข้อมูล XML
3. เพื่อให้เอกสารที่ถูกบีบอัดง่ายต่อการเข้าใจโดยไม่ต้องขยายการบีบอัด

การทบทวนวรรณกรรม

1. บทบาทและความสำคัญของการใช้ XML

เนื่องจากโลกทุกวันนี้มีการนำเทคโนโลยีสารสนเทศเข้ามาใช้ในชีวิตประจำวันมากขึ้น และได้เข้ามาเป็นส่วนหนึ่งของการทำงาน ที่เห็นได้ชัดคือ เทคโนโลยีบนอินเทอร์เน็ต ไม่ว่าจะเป็นการเข้าไปเยี่ยมชมเว็บไซต์ต่าง ๆ หรือการส่งจดหมายอิเล็กทรอนิกส์ที่เรียกว่า อีเมล (E-mail) ซึ่งกำลังเป็นมาตรฐานของการติดต่อสื่อสารสำหรับอนาคต ทำให้ต้องมีการคิดเพื่อพัฒนาให้มีความก้าวหน้ามากขึ้น การเขียนเว็บไซต์ในปัจจุบันนี้ โดยปกติแล้วสิ่งที่จะใช้สร้างเว็บได้คือ ภาษา HTML

[21] เท่านั้น แต่วันนี้โลกได้มีการพัฒนาเทคโนโลยีใหม่ที่ใช้สำหรับการแลกเปลี่ยนข้อมูลผ่านทางอินเทอร์เน็ต นั่นคือ XML ซึ่งเป็นสิ่งที่หลายผลิตภัณฑ์ให้การสนับสนุน

ปัจจุบันภาษา XML ได้เข้ามามีบทบาทและเป็นมาตรฐานในการแลกเปลี่ยนข้อมูล เนื่องจากมีความสามารถในการอธิบายความหมายของข้อมูลและมีความยืดหยุ่นในการใช้งาน โดยข้อมูลภายใน XML เป็นข้อความตัวอักษร (Text) ซึ่งเครื่องคอมพิวเตอร์ไม่ว่าจะเป็นแพลตฟอร์มใดหรือมีระบบในการใช้งานที่แตกต่างกัน ก็สามารถอ่านข้อความตัวอักษรได้ ภาษา XML จึงถือได้ว่าเป็นเทคโนโลยีที่ไม่ขึ้นกับแพลตฟอร์มใดๆ เช่น เครื่องคอมพิวเตอร์ที่ใช้ภาษาวิชวลเบสิก (Visual Basic) บนระบบปฏิบัติการของไมโครซอฟต์ (Windows System) สามารถติดต่อแลกเปลี่ยนข้อมูล XML กับเครื่องคอมพิวเตอร์ที่ใช้ภาษาจาวา (Java) บนระบบปฏิบัติการลีนุกซ์ (Linux System) ได้ จากจุดเด่นของภาษา XML นี้เองจึงเป็นโอกาสที่จะนำภาษา XML ไปใช้ประโยชน์ในเชิงพาณิชย์ เช่นในการพัฒนาเว็บเซอร์วิส (Web Services) ซึ่งเป็นเทคโนโลยีที่ทำให้โปรแกรมที่พัฒนาโดยภาษาและบนแพลตฟอร์มที่แตกต่างกันสามารถติดต่อและทำงานร่วมกันได้โดยอัตโนมัติ นอกจากนี้ยังทำให้โปรแกรมสามารถถูกเรียกใช้ได้จากอุปกรณ์อิเล็กทรอนิกส์ใด ๆ ที่ติดต่อกับระบบอินเทอร์เน็ต

XML เป็นภาษาที่ให้ความชัดเจนในการให้รายละเอียดเกี่ยวกับข้อมูล และการเปลี่ยนแปลงข้อมูลโดยแอปพลิเคชันบนเว็บและใช้ฟอร์มที่ยืดหยุ่นได้ตามมาตรฐาน HTML ได้เปิดโลกแห่งการแสดงข้อมูลต่างๆ มานำเสนอ ส่วน XML จะทำให้การทำงานกับข้อมูลโดยตรงที่ช่วยเสริมกับการทำงานของ HTML

การนำ XML มาใช้งานนั้นสามารถทำได้โดยการใช้แท็กเป็นตัวกำกับและการตั้งชื่อแท็กที่สื่อถึงความหมายของข้อมูล แต่จากการที่มีการใช้แท็กเข้ามาช่วยในการสื่อถึงความหมาย ทำให้เกิดการบันทึกแท็ก ชนิดเดียวกันบ่อยครั้งและมีอยู่จำนวนมากในเอกสาร ดังแสดงในภาพที่ 2

```
<?xml version="1.0" encoding="UTF-8"?>
<authors>
        <name>
                <firstname>Pissamai</firstname>
                <lastname>Tumpadcha</lastname>
        </name>
        <name>
                <firstname>Veerasak</firstname>
                <lastname>Lekhasophon</lastname>
        </name>
        <name>
                <firstname>Rukchon</firstname>
                <lastname>Sakpaisan</lastname>
        </name>
</authors>
```

ภาพที่ 2 ตัวอย่างเอกสาร XML

จากภาพที่ 2 จะเห็นได้ว่ามีการบันทึกข้อมูลประเภทเดียวกันหลายครั้ง ซึ่งในแต่ละครั้งจะมีรายละเอียดแตกต่างกัน ด้วยเหตุนี้ขนาดของเอกสารจึงมีขนาดใหญ่เมื่อเทียบกับขนาดข้อมูลจริงภายในเอกสารนั้น ส่งผลให้สิ้นเปลืองเนื้อที่หากต้องการจัดเก็บเอกสารและสิ้นเปลืองเวลาในการรับส่งหากต้องการแลกเปลี่ยนข้อมูลระหว่างองค์กรผ่านระบบเครือข่าย

จากปัญหาในเรื่องขนาดของข้อมูล แนวทางที่สามารถนำมาใช้ในการแก้ปัญหาได้คือ การบีบอัดข้อมูลเพื่อลดขนาดของเอกสาร เนื่องจากเอกสาร XML เป็นเอกสารแบบข้อความ วิธีการหนึ่งซึ่งสามารถนำมาใช้ได้คือ การบีบอัดข้อมูล XML ซึ่งใช้ในการบีบอัดข้อมูล XML โดยเฉพาะ ทำให้สามารถเพิ่มประสิทธิภาพในการบีบอัดได้ดีกว่าการใช้วิธีการบีบอัดด้วยการบีบอัดข้อมูลทั่วไป ในการให้รายละเอียดส่วนของโครงสร้างเอกสารส่วนใหญ่จะใช้ DTD (Document Type Definition) หรือ XML Schema [23,24,25]

ในการบีบอัดข้อมูล XML ที่มีอยู่ในปัจจุบันนั้นมีทั้งแบบที่ใช้และไม่ใช้ DTD หรือ XML Schema เป็นตัวกำหนดกฎเกณฑ์และข้อกำหนดโครงสร้างของเอกสาร การบีบอัดข้อมูลที่ใช้ DTD หรือ XML Schema นั้นมีข้อจำกัดคือต้องอาศัย DTD หรือ XML Schema ในการนำไปใช้งานด้วยทุกครั้งและเอกสารบางชนิดมีการเปลี่ยนแปลงอยู่ตลอดเวลาทำให้โครงสร้างเอกสารต้องมีการ

เปลี่ยนแปลงตามไปด้วย ซึ่งไม่สะดวกต่อการใช้งาน อีกทั้งเอกสาร XML ไม่จำเป็นต้องมี DTD หรือ XML Schema เสมอไป ซึ่งตัวอย่างของเอกสารที่ไม่จำเป็นต้องมี DTD หรือ XML Schema เช่น จดหมาย , ข้อความ (Message) หรือข้อมูลที่ถูกสร้างขึ้นมาโดยอัตโนมัติในเว็บเพจที่มีการให้การบริการเฉพาะบุคคล (Dynamically generated for personal customization) เป็นต้น

2. ประวัติของภาษามาร์คอัพ

จากแนวคิดเกี่ยวกับรูปแบบของข้อมูลที่ใช้ในการแลกเปลี่ยนกัน ระหว่างโปรแกรมที่แตกต่างกัน เพื่อให้การรวบรวมความสามารถในการแลกเปลี่ยนข้อมูลและความสามารถในการจัดเก็บข้อมูลเข้าด้วยกัน จึงได้มีการกำหนดมาตรฐานเพื่อใช้ในงานดังกล่าว SGML (Standard Generalized Markup Language) [19] จึงถูกสร้างขึ้นมาเพื่อเป็นมาตรฐานในการกำหนดรูปแบบของข้อมูลในวัตถุประสงค์ที่หลากหลาย และถูกนำไปใช้อย่างแพร่หลายในการจัดเก็บเอกสารขนาดใหญ่และมีความซับซ้อน

ข้อกำหนดในการพัฒนา SGML มักจะใช้ในการจัดการระบบเอกสารเป็นส่วนใหญ่ จึงเป็นการดีหากมีการกำหนดไวยากรณ์ขึ้นมาโดยเฉพาะ เพื่อใช้ในการแสดงข้อมูลและเชื่อมโยงส่วนต่างๆของข้อมูลเข้าด้วยกัน แอปพลิเคชันของ SGML ที่เป็นที่รู้จักกันมากที่สุดคือ HTML (HyperText Markup Language) ซึ่งถือได้ว่าเป็นการปฏิวัติและสร้างมิติใหม่ในการกระจายข้อมูลและสารสนเทศ การนำเสนอ ตลอดจนการค้นข้อมูลต่างๆ โดยผู้ใช้สามารถใช้สารสนเทศที่ต้องการได้ง่ายด้วยเบราว์เซอร์ (Browser) และมีเครื่องมือในการช่วยค้นหา (Search Engine) นอกจากนั้นยังมีการประยุกต์ไปใช้กับเครือข่ายในสำนักงานหรืออินเทอร์เน็ต และใช้สำหรับการบริการข้อมูลสำหรับลูกค้าและคู่ค้าให้สามารถตอบสนองทางด้านสารสนเทศที่ต้องการ ได้อย่างมีประสิทธิภาพมากขึ้น โดยสามารถแสดงผลได้ในทุกๆ แอปพลิเคชันที่มีความเข้าใจใน HTML ซึ่งก็คือ เว็บเบราว์เซอร์ (Web Browser)

เว็บเบราว์เซอร์ต่างๆ ไม่เพียงแต่แสดงผลเอกสาร HTML เท่านั้น หากมีเพจใดที่มีไฮเปอร์ลิงก์ (Hyper Link) เชื่อมโยงเอกสารอื่น เว็บเบราว์เซอร์นั้นจะต้องสามารถรองรับเอกสารที่ปลายทางด้วย จากการใช้งานของไฮเปอร์ลิงก์นี่เองที่ทำให้ “เวิล์ดไวด์เว็บ (Wolrd Wide Web : WWW)” มีลักษณะ “ใยแมงมุม” ซึ่งสามารถเชื่อมโยงเอกสารใดๆที่อยู่บนเว็บต่างๆได้ เพราะ HTML เป็นไฟล์ที่มีลักษณะเป็นเท็กซ์ (Text) ทำให้การสร้างเอกสาร HTML สามารถทำได้โดยใช้เท็กซ์เอดิเตอร์ (Text Editor) หรือเว็บเพจเอดิเตอร์ (Web Page Editor) ทั่วๆ ไปได้

3. XML คืออะไร

เนื่องจาก SGML เป็นภาษาที่มีความสลับซับซ้อนจึงไม่เหมาะกับการนำมาใช้ในกระบวนการแลกเปลี่ยนข้อมูลบนเว็บ และถึงแม้ว่า HTML จะมีผู้นำไปใช้กันอย่างแพร่หลาย แต่ก็ยังมีข้อจำกัดคือ HTML ถูกสร้างขึ้นมาให้ทำหน้าที่ในการนำเสนอข้อมูลบนเว็บเบราว์เซอร์เท่านั้น โดยที่ HTML ไม่สามารถแยกแยะหรืออธิบายข้อมูลจากเอกสารว่าเป็นข้อมูลที่เกี่ยวข้องกับส่วนนั้นๆ ได้เลย

XML เป็นส่วนหนึ่งของ SGML ที่เป็นข้อกำหนดในการสร้างเอกสารอิเล็กทรอนิกส์ที่กำหนดโดย W3C หรือ World Wide Web Consortium โดย XML ถูกสร้างขึ้นมาเพื่อให้การนำไปใช้ง่ายที่สุด และ XML ได้รับการออกแบบมาให้มีความเข้ากันได้กับ SGML ซึ่งเอกสารใดๆ ที่มีการสร้างตามหลักไวยากรณ์ของ XML แล้วจะเป็นไปตามหลักไวยากรณ์ของ SGML ด้วย อีกทั้งยังสามารถเปิดเอกสารนั้นได้ด้วยเครื่องมือที่ใช้กับ SGML ได้ทันที ทำให้ XML เป็นมาตรฐานของการสร้างภาษาที่เข้ากันได้ตามหลักเกณฑ์ของ XML ซึ่งมีคำอธิบายในตัวของมันเอง (Self-describing) จากภาพที่ 3 จะเห็นความแตกต่างระหว่างโครงสร้าง HTML กับโครงสร้าง XML ซึ่งโครงสร้าง HTML มีส่วนของข้อมูลใน <P> เท่านั้นไม่สามารถกำหนดให้เป็นอย่างอื่นได้ทำให้ไม่สามารถแยกแยะข้อมูลหรืออธิบายได้ว่าเป็นข้อมูลแบบใด ซึ่งต่างกับโครงสร้าง XML ที่มีส่วนของข้อมูลเกี่ยวกับ <name> , <first> และ <last> โดยผู้สร้างข้อมูลสามารถกำหนดชื่อใดๆ ของแท็กได้ตามความต้องการ ซึ่งเป็นข้อดีของการสื่อความหมายของข้อมูล ทำให้เกิดความยืดหยุ่นในการกำหนดโครงสร้างของข้อมูลและสามารถจัดโครงสร้างให้มีความเหมาะสมกับแอปพลิเคชันกับใช้งานร่วมกับ XML ข้อมูลใน XML เป็นข้อความตัวอักษร (Text) ซึ่งเครื่องคอมพิวเตอร์ไม่ว่าจะเป็นแพลตฟอร์มใดก็สามารถอ่านข้อความตัวอักษรได้ จึงกล่าวได้ว่า XML เป็นเทคโนโลยีที่ไม่ขึ้นกับแพลตฟอร์มใดๆ เช่น เครื่องคอมพิวเตอร์ที่ใช้ภาษาวิชวลเบสิก (Visual Basic) บนระบบปฏิบัติการของไมโครซอฟต์ (Windows System) สามารถติดต่อแลกเปลี่ยนข้อมูลกับเครื่องคอมพิวเตอร์ที่ใช้ภาษาจาวา (Java) บนระบบปฏิบัติการลีนุกซ์ (Linux System) ได้อย่างสะดวก [7]

```
<HTML>
        <HEAD><TITLE>name</TITLE>
</HEAD>
        <BODY>
                <P>Pissamai Tumpadcha
        </BODY>
</HTML>
```

(ก) เอกสาร HTML

```
<?xml version="1.0"?>
<name>
        <firstname>Pissamai</firstname>
        <lastname>Tumpadcha</lastname>
</name>
```

(ข) เอกสาร XML

ภาพที่ 3 ข้อแตกต่างระหว่าง เอกสาร HTML และเอกสาร XML

ถึงแม้ว่า XML จะมีข้อดีอยู่หลายประการแต่ข้อเสียของ XML ก็มีเช่นกัน โดยปกติแล้วเอกสารภาษา XML จะมีขนาดใหญ่กว่าเอกสารของเท็กซ์ไฟล์ (Text File) เนื่องจากเอกสารภาษา XML มักจะมีการใช้แท็กกำกับความหมายของข้อมูลและบ่อยครั้งที่มีชื่อแท็กที่ซ้ำๆ กัน การส่งข้อมูลโดยใช้ภาษา XML ทำให้เกิดความต้องการแบนด์วิดธ์ (Bandwidth) ขนาดใหญ่เนื่องจากขนาดของไฟล์ใหญ่กว่านั่นเอง วิธีหนึ่งที่จะช่วยลดแบนด์วิดธ์ในการส่งข้อมูล คือการใช้ XPath [22] ซึ่งเป็นภาษาที่ระบุถึงส่วนเฉพาะเจาะจงข้อมูลของ XML ทำให้เราสามารถดึงข้อมูลเฉพาะบางส่วนจาก XML ได้โดยไม่ต้องดึงข้อมูลทั้งหมดออกมา และอีกวิธีหนึ่งก็คือการบีบอัดข้อมูล

4. **เอกซ์เอ็มแอลพาสเซอร์** (XML Parser)

เมื่อสร้างเอกสารที่เป็นไปตามหลักเกณฑ์ของ XML จะมีโปรแกรมที่เรียกว่า XML Parsers ซึ่งเป็นตัวที่สามารถอ่านและเปลี่ยนแปลงข้อมูลในเอกสาร XML

```
<?xml version="1.0"?>
<name>
        <firstname>Pissamai</lastname>
        <nickname>Aom</nickname>
        <lastname>Tumpadcha</lastname>
</name>
```

ภาพที่ 4 แสดงส่วนเพิ่มเติมในเอกสาร XML

เมื่อมีการเปลี่ยนแปลงข้อมูลบางส่วนภายในเอกสารจากโครงสร้าง XML ตามภาพที่ 3 (ข) ซึ่งปรากฏผลดังภาพที่ 4 Parser จะบอกให้ทราบได้ว่ามีส่วนของข้อมูลที่ชื่อว่า <nickname> อยู่ โดยที่ผู้พัฒนาโปรแกรม Parser ไม่จำเป็นต้องทราบกฎเกณฑ์ต่างๆ ของไวยากรณ์ นอกจากนี้ Parser ตัวเดียวกันยังสามารถใช้กับแอปพลิเคชันอื่นๆ ที่มีความแตกต่างกันได้ และยังสามารถทำงานร่วมกับภาษาใดๆ เช่น ภาษาอังกฤษ ภาษาไทย หรือภาษาอื่นๆ กับ Parser ตัวเดิมได้ แม้ผู้พัฒนา Parser นั้นๆ จะไม่เข้าใจภาษาที่นำมาใช้ใน XML เลยก็ตาม

เมื่อนำโครงสร้าง XML จากภาพที่ 3 (ข) ซึ่งจะมีส่วนของข้อมูลเฉพาะชื่อ <firstname> และนามสกุล <lastname> แอปพลิเคชันนั้นๆ ยังสามารถใช้งานร่วมกับโครงสร้างใหม่ตามภาพที่ 4 ที่มีข้อมูล <nickname> เพิ่มเติมเข้าไป โดยไม่ต้องมีการแก้ไขโค้ดใดๆ ในโครงสร้างเดิม เนื่องจาก Parser จะจัดการในส่วนของการนำข้อมูลออกจากเอกสาร และแอปพลิเคชันยังสามารถใช้ประโยชน์จากข้อมูลที่เพิ่มเข้าไปได้

5. โครงสร้างของ XML

XML ใช้การซ้อนกันของอิลิเมนต์เพื่อบ่งบอกโครงสร้างของเอกสาร หากพิจารณาตัวอย่างรูปแบบโครงสร้างของผู้แต่งบทความวิชาการ เมื่อผู้แต่งประกอบด้วยจำนวนคนหลายคนในแต่ละคนประกอบด้วยข้อมูลที่เป็น ชื่อ <name> เป็นต้น

```
Begin authors
   Begin name 1
                Pissamai  Tumpadcha
   End name 1
   Begin name 2
                Pathumrat  Lekhasophon
   End name 2
End authors
```

(ก) โครงสร้างข้อมูล

```
<?xml version="1.0" encoding="tis-620"?>
        <authors>
                <name>Pissamai
Tumpadcha</name>
                <name>Pathumrat
Lekhasophon</name>
        </authors>
```

(ข) นำโครงสร้างข้อมูลมาเขียนเป็น XML

ภาพที่ 5 โครงสร้างของเอกสาร XML ตัวอย่าง

ข้อมูลส่วนบุคคล จะมีโครงสร้างที่มีรายละเอียดที่ซับซ้อนมากกว่านี้ เช่น รหัสประจำตัว, ที่อยู่, เบอร์โทรศัพท์ เป็นต้น กล่าวคือ ภายในส่วนของข้อมูลส่วนบุคคล ยังประกอบด้วยโครงสร้าง

ย่อย คือ รหัสประจำตัว (id) ลักษณะของเอกสาร XML เมื่อเขียนตามโครงสร้างนั้น สามารถอธิบายโดยใช้ภาพที่ 6 ได้ ดังนี้

```
<?xml version="1.0" encoding="tis-620"?>
<authors>
        <name id="a1">Pissamai Tumpadcha</name>
        <name id="a2">Pathumrat  Lekhasophon</name>
</authors>
```

ภาพที่ 6 เอกสาร XML ที่มีแอตทริบิวต์ (Attributes)

จากภาพที่ 6 บรรทัดที่ 1 หมายความว่าเราประกาศเอกสารนี้เป็นเอกสาร XML และมีการเข้ารหัสอักขระเป็น tis-620 เพื่อให้ใช้ภาษาไทยได้ ในภาษา XML จะแบ่ง โครงสร้างเป็น 2 ส่วนใหญ่ ๆ คือ แท็ก (Tag) และอิลิเมนต์ (Element) โดยภายในอิลิเมนต์อาจมีแอตทริบิวต์หรือไม่ก็ได้ ซึ่งสามารถอธิบายเพิ่มเติมได้ดังนี้

5.1 แท็ก (Tag)

สำหรับใน XML แล้ว แท็กมีความหมายในลักษณะเดียวกับที่ใช้ใน HTML tag คือข้อความที่เริ่มต้นด้วยสัญลักษณ์ “<” และลงท้ายด้วยสัญลักษณ์ “>”

5.1.1 แท็กเปิด (Start Tag) มีสัญลักษณ์คือ <...> จากภาพที่ 6 แท็กเปิดคือ <authors>

5.1.2 แท็กปิด (End Tag) จะเหมือนกับแท็กเปิดแต่จะมีเครื่องหมาย / อยู่หลังสัญลักษณ์ < ดังนั้นลักษณะของแท็กปิดจึง มีรูปแบบคือ </...> หากพิจารณาระหว่างแท็กเปิดกับแท็กปิดแล้ว ข้อแตกต่างอีกข้อหนึ่งคือแท็กเปิดเป็นแท็กที่สามารถใส่ข้อมูลแอตทริบิวต์ (Attribute) ลงไปภายในแท็กได้แต่จะไม่ทำในแท็กปิด จากภาพที่ 6 แท็กปิดคือ </authors>

5.2 อิลิเมนต์ (Element)

โครงสร้างหลักของ XML ซึ่งอยู่ในรูปของแท็กเช่นเดียวกัน ตามตัวอย่างข้างล่างอิลิเมนต์สามารถจะมีลักษณะซ้อนกันเป็นชั้นๆ (Nested)

```
<authors>
        <name>
            .....
        </name>
</authors>
```

ภาพที่ 7 โครงสร้างของอิลิเมนต์

ซึ่งเมื่อเทียบจากภาพที่ 7 ซึ่ง <authors> เป็นรูทอิลิเมนต์ (Root element) ซึ่งเป็นอิลิเมนต์ที่ไม่ได้อยู่ภายใต้อิลิเมนต์ใดๆ ในเอกสารและอิลิเมนต์อื่นๆ จะต้องอยู่ภายใต้อิลิเมนต์ เพื่อให้เข้าใจความหมายของศัพท์หลายๆ คำศัพท์ ก่อนที่จะนำไปใช้ในการสร้าง XML ซึ่งจำเป็นอย่างมาก เมื่อพิจารณาข้อมูลบางส่วนของเอกสารดังภาพที่ 8

```
<name id="a1"> Pissamai Tumpadcha </name>
```

ภาพที่ 8 ตัวอย่างของอิลิเมนต์

จากลักษณะโครงสร้างตั้งแต่ <name ..... ไปจนถึง </name> ถูกเรียกว่าอิลิเมนต์ หรืออธิบายได้อีกแบบคือ อิลิเมนต์ จะเริ่มต้นที่แท็กเปิด และสิ้นสุดที่แท็กปิด ในแท็กคำสั่งเดียวกัน เพื่อให้ง่ายแก่การแยกแยะ รายละเอียดของแต่ละส่วนที่ให้ไว้ในตัวอย่าง แสดงความหมายที่เกี่ยวข้องกับ XML ได้ดีขึ้น อิลิเมนต์จะมีส่วนประกอบของเนื้อหา (Content) และ แอตทริบิวต์ (Attribute) ซึ่งอธิบายได้ดังนี้

5.2.1 เนื้อหา (Content Element) หรือ “เนื้อความ” เป็นข้อมูลเพื่อใช้ในการแสดงให้ผู้อ่านเอกสารได้เห็น หรือกล่าวอีกนัยหนึ่งคือเนื้อหา จะอยู่หลังแท็กเปิด และจบที่ก่อนถึงแท็กปิดนั่นเอง

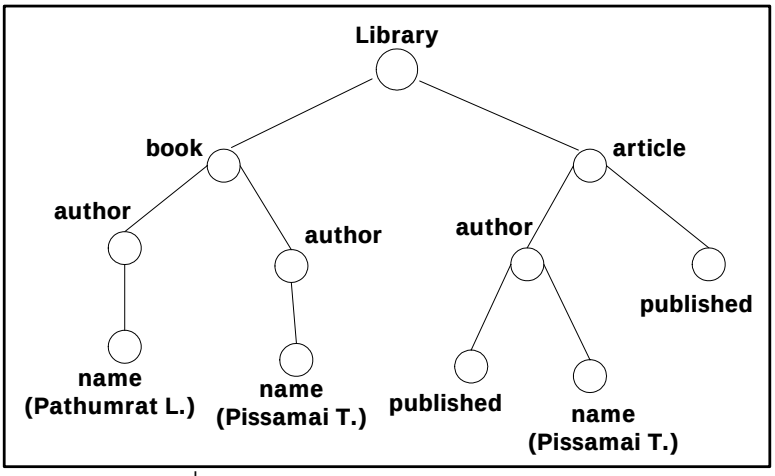
5.2.2 แอตทริบิวต์ (Attribute) คือข้อมูลที่มีความหมายเพิ่มเติม เพื่อแยกแยะสิ่งที่เหมือนกันออกจากกันได้ โดยจะถูกบรรจุอยู่ภายในแท็กเปิด ซึ่งแอตทริบิวต์ อาจจะมีหรือไม่มีก็ได้ ในบางครั้งภายในแท็กเปิดอาจจะมีได้มากกว่าหนึ่งแอตทริบิวต์ จากภาพที่ 9 ส่วนที่เป็นแอตทริบิวต์คือ id="a1" และ gender="Female"

```
<name id="a1" gender="Female"> Pissamai Tumpadcha </name>
```

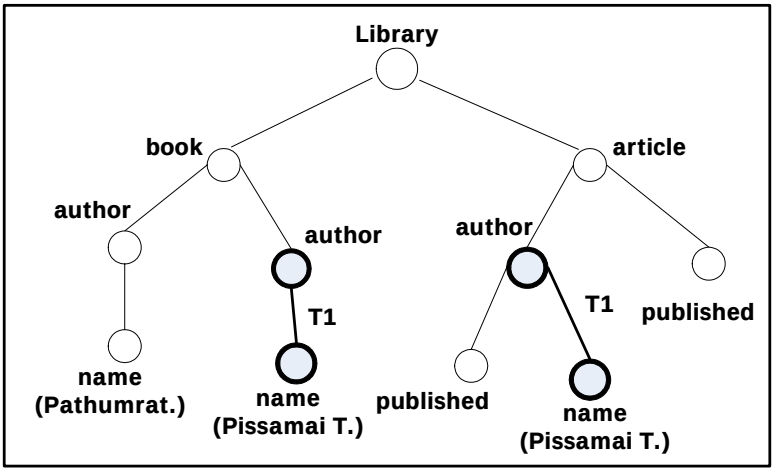
ภาพที่ 9 อิลิเมนต์ที่มีหลายแอตทริบิวต์

6. งานวิจัยที่เกี่ยวข้อง

Multi-Colored Trees (MCT) [8] เป็นแนวคิดที่ได้นำเอาโครงสร้างของ XML ที่มีอิลิเมนต์เหมือนกันมาสร้างเป็นทรี (tree) ใหม่โดยการแยกแยะอิลิเมนต์นั้นด้วยสีเป็นตัวแทนในการเก็บ tree ดังกล่าว เพื่อใช้กับเอกสาร XML ที่มีรายละเอียดของอิลิเมนต์เหมือนกันหรือมีการใช้ซ้ำภายในเอกสารนั้น ทำให้โครงสร้างของข้อมูลมีขนาดที่เล็กลง



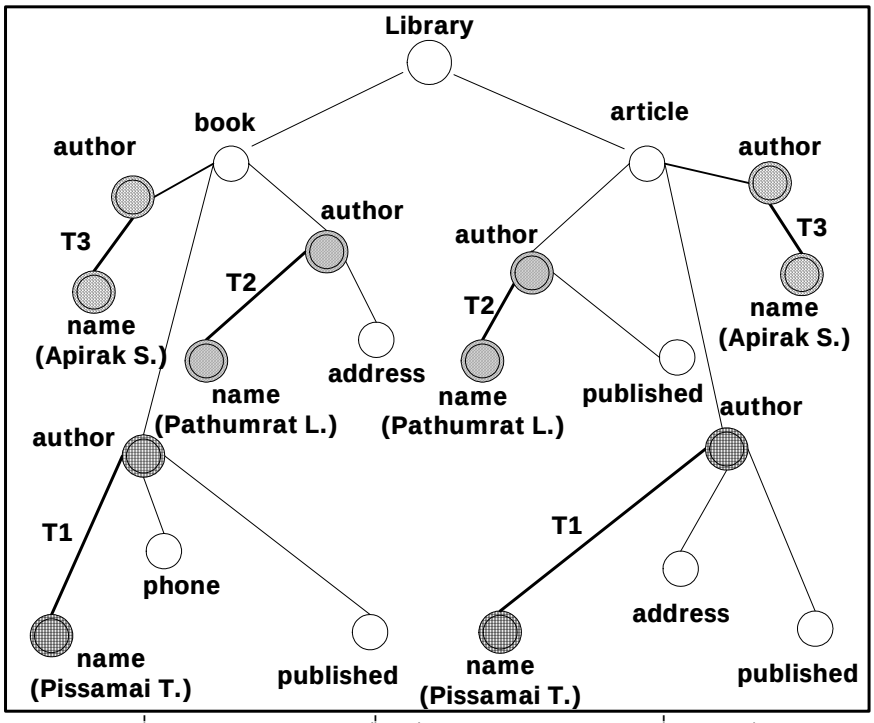
ภาพที่ 10 โครงสร้าง Tree ของอิลิเมนต์



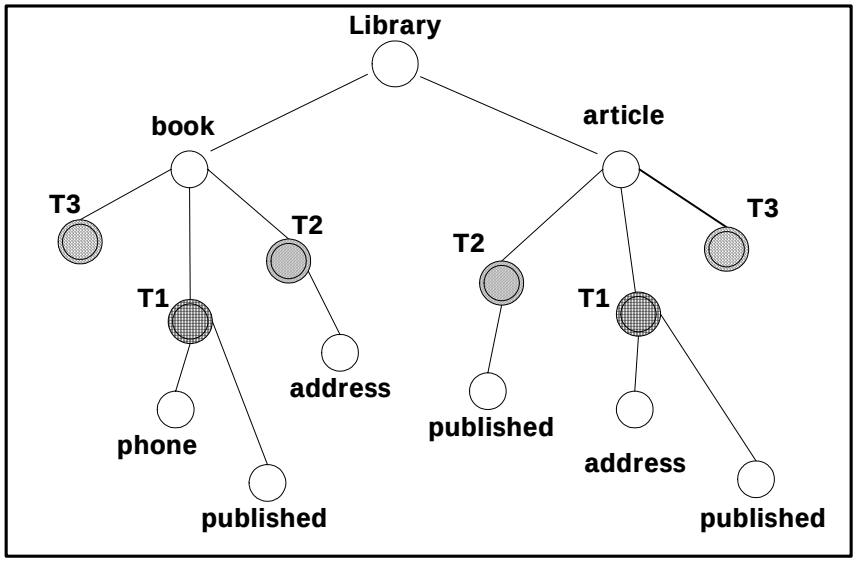
ภาพที่ 11 การจัดกลุ่มของอิลิเมนต์ที่เหมือนกัน

จากภาพที่ 10 แสดงให้เห็นโครงสร้าง Tree ของ XML ที่มีอิลิเมนต์ต่างๆ และจะเห็นว่ามีอิลิเมนต์ที่เหมือนกันใช้งานซ้ำกันอยู่ จึงทำการจัดกลุ่มของอิลิเมนต์ที่เหมือนกันจะจัดให้อยู่ในรูปแบบของโหนด (Node) ซึ่งในที่นี้คือ Node T1 ซึ่งจะใช้สีบอกถึงความแตกต่างระหว่างอิลิเมนต์ แต่ในรายงานนี้จะใช้รูปแบบ (pattern) แทนสีดังแสดงในภาพที่ 12 โดยที่ Node T1 มีความสัมพันธ์ระหว่างอิลิเมนต์ author และอิลิเมนต์ name (Pissamai T.) ซึ่งใช้สัญลักษณ์ T1 เป็นตัวแทนของกลุ่มที่มีความสัมพันธ์แบบนี้

เมื่อข้อมูลในเอกสารมีความซับซ้อนมากขึ้น จะทำการแทนความหมายของอิลิเมนต์ที่ถูกจัดเป็นโหนดต่างๆ จากภาพที่ 12 มีอิลิเมนต์ที่มีการใช้งานซ้ำเกิดขึ้นอีก ซึ่งก็คือ Node T2 และ Node T3 โดยที่ Node T2 มีความสัมพันธ์ระหว่างอิลิเมนต์ author และอิลิเมนต์ name (Pathumrat L.) ส่วน Node T3 มีความสัมพันธ์ระหว่างอิลิเมนต์ author และอิลิเมนต์ name (Apirak S.) และในทำนองเดียวกันเมื่อมีอิลิเมนต์ที่ซ้ำกันเกิดขึ้นในส่วนอื่นๆ จะกำหนด Node ให้กับอิลิเมนต์ที่มีความสัมพันธ์กัน



ภาพที่ 12 โครงสร้างเดิมเมื่อเทียบกับ Node ต่างๆ ที่กำหนดไว้



ภาพที่ 13 โครงสร้างที่แทนด้วย Node

จากภาพที่ 13 เมื่อโครงสร้างถูกแทนด้วย Node แล้วจะเห็นได้ว่าขนาดโครงสร้างจะเล็กลงเมื่อเทียบกับโครงสร้างที่ยังไม่มีการแทนด้วย Node

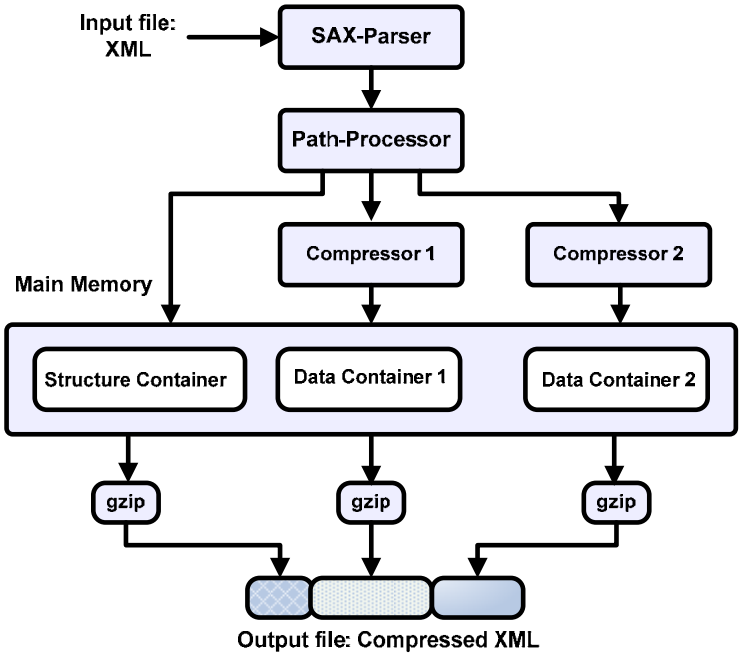
ในขั้นตอนการบีบอัดข้อมูล XML เพื่อให้มีขนาดที่เล็กลงนั้น โดยเทคนิคที่มีการนำเสนอซึ่งได้แก่ XMill [11], XGRIND [17], XPRESS [13], และ XPACK [12] ซึ่งทั้ง 4 วิธีเป็นการนำเสนอโดยใช้เทคนิคในการเข้ารหัสและบีบอัดข้อมูลด้วย zlib และ gzip [5] ซึ่งเป็นเทคนิคที่ใช้ในการบีบอัดข้อมูล

XMill เป็นวิธีการแรกที่ได้มีการนำเสนอวิธีการและเทคนิคในการบีบอัดเอกสาร XML ในการบีบอัดข้อมูลโดยวิธีนี้จะใช้ StAX Parser เป็นตัวจัดการกับข้อมูล XML เพื่อนำข้อมูลที่อ่านได้เข้าสู่กระบวนการบีบอัดของ XMill โดยแผนภาพการทำงานได้แสดงไว้ในภาพที่ 14 ซึ่งมีกระบวนการ 3 กระบวนการ ได้แก่

1) แยกโครงสร้างออกจากข้อมูล เป็นการแยกโครงสร้างที่ประกอบไปด้วยแท็กและแอตทริบิวต์ออกจากส่วนที่เป็นข้อมูลโดยเริ่มจากการแยกส่วนแท็ก ซึ่งภายในประกอบไปด้วยอิลิเมนต์และแอตทริบิวต์ ออกจากข้อมูลที่เป็นตัวอักขระ

` 2) จัดกลุ่มให้กับข้อมูลต่างๆ จากนั้นจะทำการจัดความสัมพันธ์ของกลุ่มข้อมูลเป็นเหมือนภาชนะบรรจุ หรือเรียกว่า Containers โดยจัดให้ข้อมูลที่เป็นแบบเดียวกันอยู่ในกลุ่มเดียวกัน ส่วนข้อมูลจะเรียงกันไปต่อจากแท็กที่ได้จัดกลุ่มไว้แล้ว

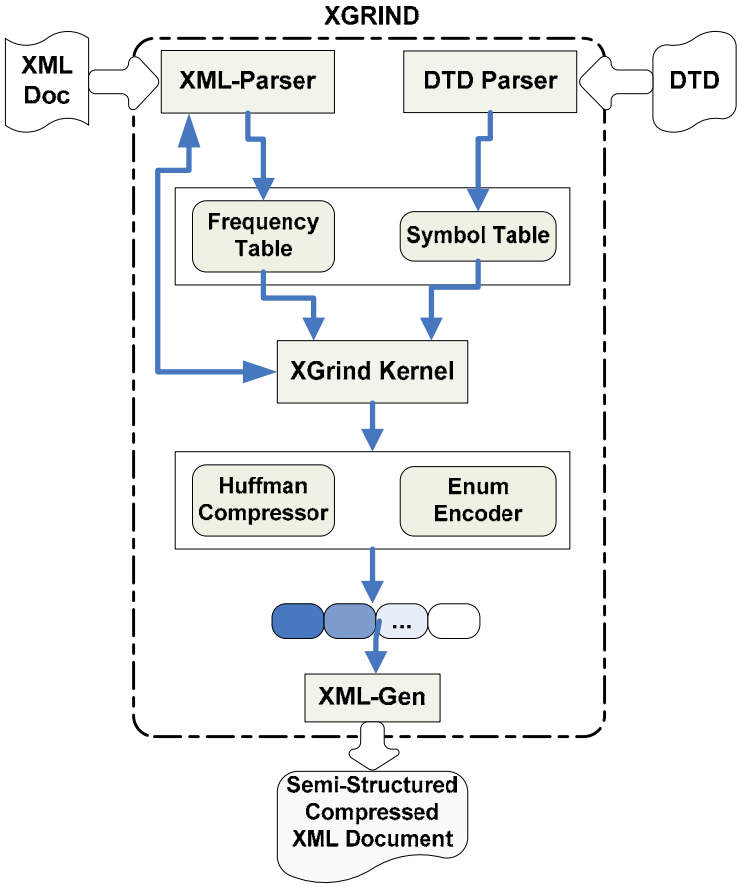
3) นำข้อมูลที่ได้เข้าสู่กระบวนการสร้างความสัมพันธ์ทางกลุ่มคำกับสัญลักษณ์ ในขั้นตอนนี้จะนำ Containers แต่ละกลุ่มมาทำการคัดแยกออกเป็นชุด จากนั้นให้นำ Containers ทั้งหมดโดยอาศัย gzip ทำการบีบอัดอีกครั้งเพื่อให้ได้ข้อมูลที่ออกมาเป็นไฟล์เดียว



ภาพที่ 14 การทำงานของ XMill [11]

เนื่องจากการจัดกลุ่มข้อความต้องอาศัยความเข้าใจความหมายของข้อมูล ซึ่งขึ้นอยู่กับชนิดของแอปพลิเคชัน ฉะนั้น XMill จึงมักต้องให้ผู้ใช้โปรแกรมพิจารณาความหมายของข้อมูล อีกทั้งผู้ใช้โปรแกรมไม่สามารถค้นหาข้อมูลที่ต้องการจากเอกสารที่ถูกบีบอัดแล้ว ถึงกระนั้นก็ตาม XMill ถือว่าเป็นบทความวิจัยแรกๆ ที่ทำให้ผู้วิจัยทั้งหลายตระหนักถึงความสำคัญของปัญหาและวิธีการแก้ปัญหาในการบีบอัดข้อมูล XML ซึ่งได้มีผู้นำเทคนิคบางเทคนิคของ XMill ไปใช้ในการเก็บและค้นหาข้อมูล XML [15]

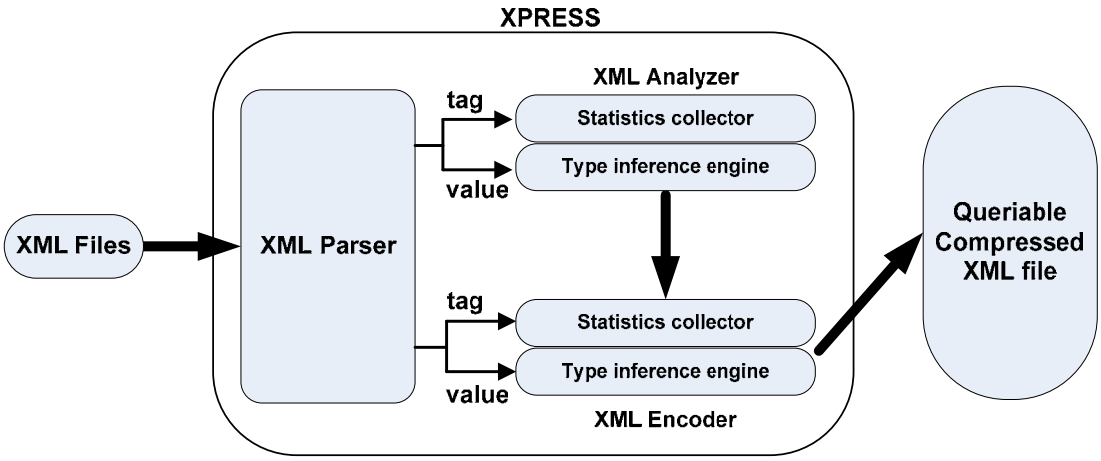
XGRIND เป็นวิธีการบีบอัดอีกแบบหนึ่งที่ได้รับการพัฒนาขึ้นมา โดยมีข้อดีที่ XMill ยังไม่สามารถทำได้ซึ่งก็คือ ผู้ใช้ XGRIND สามารถค้นหาข้อมูลได้จากเอกสารที่ถูกบีบอัด คุณสมบัตินี้เป็นผลมาจากการที่ข้อมูลที่ถูกบีบอัดแล้วยังคงมีโครงสร้างของเอกสารเดิม แต่อย่างไรก็ตามผู้ใช้โปรแกรม XGRIND ไม่สามารถตอบคำถามบางประเภทโดยไม่ได้คลายการบีบอัดของข้อมูล ตัวอย่างของคำถามที่ต้องมีการคลายการบีบอัดข้อมูลคือ คำถามที่ถามช่วงของค่า (Range Query) และคำถามที่ถามหาข้อมูลค้นหาบางส่วนที่ตรงกับข้อมูลที่มีอยู่ (Partial Match) โดยแผนภาพการทำงานของ XGRIND ได้แสดงไว้ในภาพที่ 15 ดังนี้



ภาพที่ 15 การทำงานของ XGRIND [17]

นอกจากนี้ XGRIND จะบีบอัดเฉพาะเอกสาร XML ที่มีเอกสาร DTD ซึ่งเป็นเอกสารที่ระบุโครงสร้างของเอกสาร XML ซึ่งเอกสาร XML บางเอกสารอาจจะไม่มี DTD ก็ได้ ฉะนั้นผู้ใช้โปรแกรม XGRIND จะต้องสร้าง DTD สำหรับเอกสาร XML ที่ยังไม่มี DTD

XPRESS ได้นำเสนอแนวคิดใหม่ที่ใช้การเข้ารหัสทางคณิตศาสตร์แบบผันกลับ (Reverse Arithmetic Encoding) เป็นวิธีการในการจัดเรียงข้อมูลเพื่อให้การหาคำตอบสำหรับนิพจน์ของ XPath เป็นไปได้อย่างมีประสิทธิภาพ นอกจากนี้ XPRESS ได้พัฒนาการหาชนิดของข้อมูลโดยไม่ต้องอาศัยข้อมูลอินพุตจากผู้ใช้โปรแกรม การแยกอิลิเมนต์ที่เป็นชนิดของข้อมูลและข้อมูลที่มีการใช้บ่อยครั้ง ออกเป็น 2 ส่วนคือส่วนที่เป็นแท็กกับส่วนที่เป็นค่าของข้อมูล (Value) โดยมี XML Parser เป็นตัวเปลี่ยนแปลงข้อมูลในเอกสาร เพื่อทำการวิเคราะห์เอกสารและเข้ารหัสเอกสาร เข้าสู่กระบวนการบีบอัดข้อมูล จากนั้นจึงนำข้อมูลที่ได้จากการบีบอัดแล้วไปทำการ query ต่อไป ซึ่งสามารถแสดงแผนภาพการทำงานของ XPRESS ได้ในภาพที่ 16



ภาพที่ 16 การทำงานของ XPRESS [13]

จากภาพที่ 16 วิธีการนี้ขึ้นอยู่กับชนิดของข้อมูลโดยจะมีหลักการทำงานคือ

1) ตัววิเคราะห์ข้อมูล XML (XML Analyzer) อัลกอริทึมของตัววิเคราะห์ข้อมูล XML จะสร้าง Hash table ที่เรียกว่า Elemhash ซึ่งจะเก็บข้อมูลต่างๆ อันได้แก่ ชนิดของข้อมูล, ความถี่ในการใช้งาน เป็นต้น ค่าดังกล่าวจะถูกจัดเก็บให้อยู่ในช่วง Interval เรียกค่านี้ว่า Statistics collector ซึ่งจะคอยนับจำนวนของอิลิเมนต์ที่เกิดขึ้น อย่างไรก็ตามถ้าแท็กนั้นเป็นอิลิเมนต์ระดับที่สูงกว่า เช่น Root element จะเป็นการยากที่จะกำหนดช่วงเส้นทางของข้อมูล จึงจำเป็นต้องใช้หลักการความถูกต้องระดับสูงทางคณิตศาสตร์ (High Precision Floating Arithmetic)

2) เข้ารหัสข้อมูล XML (XML Encoder) ในการเข้ารหัสข้อมูลจะทำการเปลี่ยนรูปให้เป็นไบนารี ซึ่งในแต่ละช่วงของการเข้ารหัสจะแทนด้วย u8 ใช้ 7 บิต จำนวน 1 ไบต์, u16 ใช้ 15 บิต จำนวน 2 ไบต์และ u32 ใช้ 31 บิต จำนวน 4 ไบต์ เมื่อเข้ารหัสแล้วจึงทำการบีบอัดข้อมูลต่อไป ดังตารางที่ 1

ตารางที่ 1 Data Encoders ที่ XPRESS ใช้

Encoder	Description
u8	encoder for integers where max-min < 27
u16	encoder for integers where< 27+1 < max-min < 215
u32	encoder for integers where < 215+1 <max-min< 231
f32	encoder for floating values
dict8	dictionary encoder of textual data

กระบวนการค้นหาข้อมูลที่ถูกบีบอัดแล้ว (Query Processing) เมื่อข้อมูลถูกบีบอัดโดย XPRESS แล้ว กระบวนการค้นหา (Query) จะทำให้ชื่อที่มีความยาวสั้นลงแล้วทำการแปลงรูปไปเป็นลำดับช่วง ซึ่งโดยทั่วไปแล้วหนึ่งลำดับช่วงจะเท่ากับเส้นทางของชื่อที่ทำให้สั้นลงแล้ว หรือเป็นการลดความยาวของชื่อให้สั้นลงไปนั่นเอง เมื่อต้องการค้นหาข้อมูลจะทำการค้นหาจาก Hash table ซึ่งจะเก็บช่วงของข้อมูลนั้นๆ ที่ได้ทำการบีบอัดแล้ว เพื่อให้ได้ช่วงของข้อมูลที่ตรงกับการค้นหานั้น

XPRESS มีข้อดีเช่นเดียวกับ XGRIND ตรงที่สามารถจะค้นหาข้อมูลจากเอกสารที่ถูกบีบอัดแล้ว แต่ว่า XPRESS ไม่ได้ใช้ DTD ถึงอย่างไรก็ตาม XPRESS มีข้อจำกัดตรงที่ XPRESS จะไม่สามารถเข้าใจเอกสาร XML ที่มีการใช้ ID และ IDREF และไม่ได้กล่าวถึงวิธีการขยายข้อมูลที่ถูกบีบอัดให้เป็นข้อมูล XML ปกติ

XPACK เป็นการบีบอัดเอกสาร XML ด้วยวิธีการเชิงไวยากรณ์ ซึ่งใช้วิธีการเชิงไวยากรณ์ในการบีบอัดและการขยายเอกสาร XML ซึ่งแสดงแผนภาพการทำงานในภาพที่ 17 และมีส่วนประกอบหลักของ XPACK คือ

1) Grammar Generator ทำหน้าที่ในการสร้างกฎไวยากรณ์

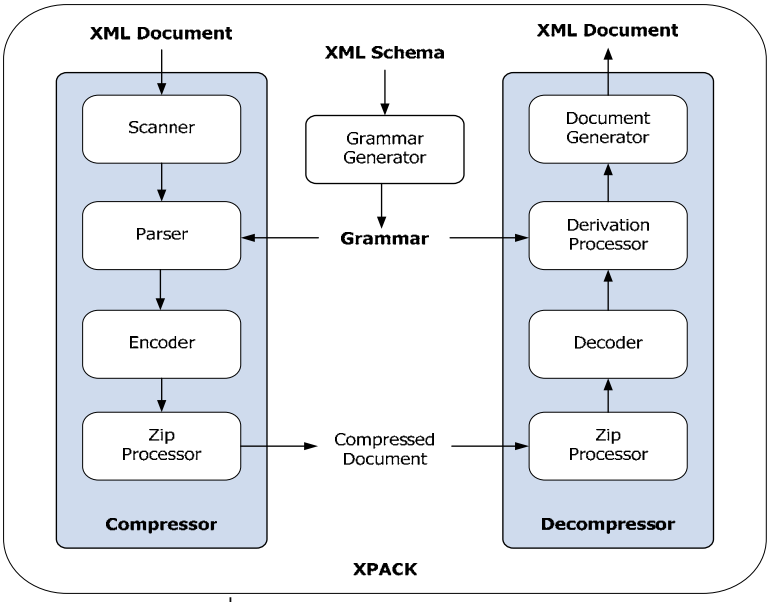
2) Compressor ทำหน้าที่บีบอัดเอกสาร

3) Decompressor ทำหน้าที่ขยายเอกสารที่ผ่านการบีบอัด

1) ตัวสร้างกฎไวยากรณ์ (Grammar Generator) มีหน้าที่ในการสร้างกฎไวยากรณ์เพื่อใช้ในการบีบอัดเอกสาร โดยการวิเคราะห์คำอธิบายโครงสร้างเอกสาร XML ภาษาที่ใช้ในการอธิบายโครงสร้างดังกล่าวได้แก่ XML Schema

2) ตัวบีบอัดข้อมูล (Compressor) ทำหน้าที่ในการบีบอัดเอกสารโดยประกอบไปด้วยส่วนประกอบย่อย 4 ส่วนด้วยกันคือ Scanner, Parser, Encoder และ Zip Processor ดังภาพที่ 17 โดยการทำงานภายใน Compressor เริ่มต้นที่ Scanner รับเอกสาร XML เข้ามาเพื่อเปลี่ยนให้เป็นเครื่องหมาย (Token) ที่ใช้ในกฎไวยากรณ์ จากนั้น Parser ทำการวิเคราะห์ด้วยวิธีการเชิงไวยา กรณ์ เพื่อให้ได้ผลลัพธ์ออกมาเป็น Parser Tree ต่อมาจึงนำ Parser Tree มาเข้ารหัสให้อยู่ในรูปของแถวลำดับด้วย Encoder ซึ่งผลลัพธ์ที่ได้จะนำไปทำการบีบอัดในขั้นตอนสุดท้ายที่ Zip Processor โดยทำการบีบอัดข้อมูลด้วย อัลกอริทึมการบีบอัดข้อมูลทั่วไป ผลลัพธ์จากการบีบอัดคือเอกสารที่ได้รับการบีบอัด (Compressed Document) ซึ่งสามารถนำไปจัดเก็บ หรือแลกเปลี่ยนระหว่างองค์กรผ่านเครือข่ายต่อไป

3) ตัวขยายข้อมูลที่ถูกบีบอัด (Decompressor) เมื่อต้องการใช้เอกสารที่ได้รับการบีบอัดแล้ว จำเป็นต้องทำการขยายเอกสารก่อนโดยใช้ Decompressor ซึ่งประกอบไปด้วย 4 ส่วนย่อย คือ Unzip Processor, Decoder, Derivation Processor และ Document Generator ดังภาพที่ 17 Decompressor มีการทำงานโดยเริ่มที่ Unzip Processor ทำการขยายเอกสารที่ผ่านการบีบอัด ให้อยู่ในรูปของข้อมูลที่เข้ารหัส จากนั้น Decode ทำการถอดรหัสข้อมูลให้อยู่ในรูปของ Parser Tree เพื่อให้ Derivation Processor ใช้วิธีการเชิงไวยากรณ์ให้ได้มา ซึ่งสัญลักษณ์ที่จะใช้ในการสร้างเอกสาร สุดท้าย Document Generator ทำการสร้างเอกสาร XML ที่มีความหมายของข้อมูลภายในเหมือนกับเอกสารต้นฉบับ



ภาพที่ 17 การทำงานของ XPACK [12]

การจัดโครงสร้างไวยากรณ์ของ XPACK เป็นดังนี้

<t> ::= t # / d (a, v)

โดยที่

<t>	แทนรูปแบบการจัดเรียงข้อมูลในเอกสาร XML (ไม่ใช่แท็ก!!)
t	แทนแท็กเปิด (Open Tag)
#	แทนข้อมูลภายในแท็ก
/	แทนแท็กปิด (Close Tag)
d	แทนประเภทข้อมูลภายในแท็ก (Element Data Type)
a	แทนชื่อแอตทริบิวส์ (Attribute Name)
v	แทนประเภทข้อมูลของแอตทริบิวต์ (Attribute Data Type)

ข้อจำกัดของ XPACK คือโปรแกรมที่บีบอัดข้อมูลไม่สามารถจัดการกับเอกสาร XML ที่มีอิลิเมนต์เป็นแบบผสมผสาน (Mixed-content element) ซึ่งคืออิลิเมนต์ที่มีทั้งอิลิเมนต์และตัวอักขระอยู่ข้างใน และผู้ใช้โปรแกรมไม่สามารถค้นหาข้อมูลจากเอกสาร XML ที่ถูกบีบอัดแล้ว

การออกแบบการวิจัย

1) ศึกษาและค้นคว้าภาษา XML
2) ศึกษาและค้นคว้าการทำงานของการบีบอัดข้อมูล XML
3) วางขอบเขตและกำหนดเป้าหมายของงานวิจัย
4) ออกแบบและพัฒนาอัลกอริทึม
5) ออกแบบการทดลอง
6) เขียนโปรแกรม ทำการทดลอง และปรับปรุงแก้ไข
7) สรุปผลการทดลองและเขียนรายงานสรุปผล

ขอบเขตของการวิจัย

ในการศึกษาครั้งนี้จะทำการพัฒนาอัลกอริทึม เพื่อใช้ในการบีบอัดข้อมูล XML โดยที่ไม่ใช้โครงสร้างของเอกสาร ใช้เทคนิคจัดกลุ่มของข้อมูลที่เป็นแท็กชนิดเดียวกันและมีการใช้งาน รวมเข้าไปแล้วจัดรูปแบบของกลุ่มดังกล่าวขึ้นมาเป็นจุดศูนย์รวมของข้อมูลแต่ละกลุ่ม เพื่อใช้เป็นตัวแทนในการอ้างอิงถึงของข้อมูลนั้น ขั้นตอนต่อไปทำการบีบอัดโดยการเข้ารหัสข้อมูลด้วยวิธีการจัดเรียงแบบกระชับเพื่อให้ได้ข้อมูลที่มีขนาดเล็กลง เมื่อได้ข้อมูล XML ที่บีบอัดแล้ว ทำการเปรียบเทียบขนาดกับข้อมูลเดิมที่ยังไม่ได้ทำการบีบอัด ขั้นตอนต่อไปคือการขยายข้อมูลที่บีบอัดไว้ให้เป็นข้อมูล XML ในรูปแบบปกติ

ระเบียบวิธีวิจัย

1. ข้อกำหนดแบบจำลอง

1.1 ข้อตกลงเบื้องต้น

1) ข้อมูลที่ใช้ในการทดสอบต้องเป็น XML

2) ข้อมูล XML จะต้องมีรูปแบบที่ถูกต้องตามแบบฟอร์ม (Well-Formed)

3) เครื่องคอมพิวเตอร์ที่ใช้ทดสอบต้องสามารถใช้งานภาษา JAVA ได้

4) การคลายการบีบอัดต้องใช้กับข้อมูลที่ถูกบีบอัดด้วย XBrevity เท่านั้น

1.2 ข้อกำหนดของข้อมูล

1) การทดสอบบีบอัดใช้โปรแกรม 3 โปรแกรมคือ

1.1) การทดสอบบีบอัดด้วย gzip

1.2) ทำการทดสอบบีบอัดด้วย XMill

1.3) ทำการทดสอบบีบอัดด้วย XBrevity

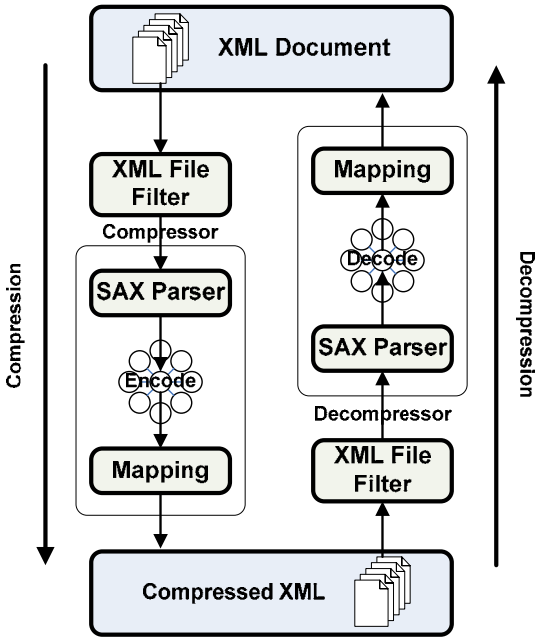
2) ข้อมูล XML ที่ใช้ในการทดสอบ

2.1) ดาวน์โหลดตัวอย่างจากเว็บไซต์

2.2) สร้างจากโปรแกรม XMark โดยกำหนดแฟกเตอร์ในการสร้าง

2. องค์ประกอบและสถาปัตยกรรมของ XBrevity

ในเทคนิคการบีบอัดด้วย XBrevity [10] นั้นจะประกอบไปด้วย ส่วนของการบีบอัดเอกสาร XML (Compressor) และการขยายเอกสาร (Decompressor) เพื่อทำให้เอกสารที่ถูกบีบอัดด้วย XBrevity กลับมาเป็นเอกสาร XML ต้นฉบับเดิมได้ ในโครงสร้างของ XBrevity จะใช้ StAX Parser เป็นตัวอ่านเอกสารทั้งในกระบวนการ การบีบอัดเอกสารและขยายการบีบอัดโดยในแต่ละกระบวนการจะมีการเข้ารหัสและถอดรหัสเอกสาร XML ที่ได้อ่าน ซึ่งโครงสร้างการทำงานของ XBrevity ได้แสดงไว้ในภาพที่ 18



ภาพที่ 18 โครงสร้างการทำงานของ XBrevity

2.1 โปรแกรมบีบอัดข้อมูล (Compressor)

มีหน้าที่ในการบีบอัดเอกสาร XML ซึ่งจะใช้ StAX Parser เป็นตัวอ่านวิเคราะห์เอกสารที่จะทำการบีบอัดเข้ามา ในระหว่างที่ทำการอ่านวิเคราะห์เอกสารอยู่นั้นจะทำการเข้ารหัสโดยการแทนที่ (Mapping) เพื่อเปลี่ยนรูปแบบให้เป็นตัวอักษรแล้วบันทึกค่าต่างๆ ลงไปบนไฟล์ โดยในการสร้างไฟล์ที่มีข้อมูลที่ถูกบีบอัดแล้วนั้น จะยังคงอยู่ในรูปแบบของภาษา XML และถูกต้องตามแบบฟอร์ม (Well-formed)

โดยในเอกสารใหม่ที่ได้ประกอบไปด้วย <d>…</d> โดยที่ d ย่อมาจาก Data หมายถึงส่วนของข้อมูล ซึ่งมีส่วนของข้อมูลที่ถูกบีบอัดแล้วอยู่ภายใน และ <m …/> โดยที่ m ย่อมาจาก Metadata หมายถึงส่วนของการอธิบายความหมายของข้อมูลภายในเอกสาร ซึ่งเอกสารที่ถูกบีบอัดนี้สามารถจัดเก็บ หรือแลกเปลี่ยนข้อมูลระหว่างเครือข่ายได้ และสามารถเข้าใจได้ง่ายโดยที่ไม่ต้องขยายข้อมูลกลับไปเป็นต้นฉบับเดิม เนื่องจากอยู่ในรูปแบบของเอกสาร XML แล้วและมีตัวอธิบายข้อมูลอยู่ภายในให้ หรือสามารถทำการขยายเอกสารให้เป็นเอกสารเดิมได้ด้วยตัวขยายเอกสาร (Decompressor) ที่ออกแบบไว้

2.2 โปรแกรมคลายการบีบอัดข้อมูล (Decompressor)

เมื่อเอกสารที่ถูกบีบอัดด้วย XBrevity ต้องการขยายให้กลับมาเป็นเอกสาร XML ต้นฉบับสามารถทำได้โดยใช้ StAX Parser ที่ออกแบบมาเพื่อวิเคราะห์เอกสาร โดยทำการถอดรหัสเอกสาร

ที่ถูกบีบอัดนั้นออกมา จากนั้นทำการ Mapping ค่าที่ได้ ในแท็ก <m …/> แล้วนำค่าที่อ่านได้ไปแทนที่ตัวแปรเดิมที่อยู่ภายใน <d>…</d> เพื่อให้กลับออกมาเป็นเอกสารต้นฉบับ

2.3 ตัวอย่างการทำงานของ XBrevity

ในรายงานนี้ได้นำเสนอการบีบอัดข้อมูล โดยไม่ใช้โครงสร้างของเอกสาร XML (XML Schema) ซึ่งแสดงตามภาพที่ 19 ซึ่งเป็นตัวอย่างของเอกสารที่ประกอบไปด้วยอิลิเมนต์ และแอตทริบิวต์ต่างๆ จะเห็นได้ว่ามีการใช้งานของแท็กที่ซ้ำๆ กัน แต่ข้อมูลภายในแตกต่างกัน ทำให้โครงสร้างดูซับซ้อนและเอกสารมีขนาดใหญ่

```
<?xml version="1.0"?>
<bib>
   <article>
      <authors>
         <author aid="a1" eid="e1">
            <name>Pissamai<nickname>Aom</nickname></name>
         </author>
         <author aid="a2"><name>Veerasuk</name></author>
      </authors>
      <year>2005</year>
   </article>
   <article>
      <authors>
         <author aid="a3"><name>Patumrut</name></author>
      </authors>
   </article>
</bib>
```

ภาพที่ 19 เอกสาร XML ตัวอย่าง A

ในการบีบอัดข้อมูลนั้นจะมีข้อมูลเมต้าดาต้า เพื่อใช้อ้างอิงอิลิเมนต์และแอตทริบิวต์ ซึ่งทำให้โครงสร้างที่มีอยู่เดิมสั้นลง ทำให้ขนาดของเอกสารเล็กลงตามไปด้วย

```
<c>
  <d>
    <e1e2e3 a4="a1" a5="e1" e6v="Pissamai" e7v="Aom" />
    <xe1e2 e3a4="a2" e6v="Veerasuk" />
    <xe1 e8v="2005" />
    <e1e2e3 a4="a3" e6v="Patumrut" />
  </d>
  <m f="bib article authors author aid eid name nickname year "
    b="0 1 2 3 a4 a5 6 7 8 "/>
</c>
```

ภาพที่ 20 เอกสาร XML ตัวอย่าง A ที่ถูกบีบอัดแล้ว

โดยที่ c = การบีบอัด (Compressed)

d = ข้อมูล (Data)

m = ข้อมูลที่อธิบายข้อมูล (Metadata)

f = ชื่อเต็ม (Full name)

b = ชื่อย่อ (Brevity name)

v = ค่าของข้อมูล (Value)

x = การอ้างอิงอิลิเมนต์ที่เรียงซ้อนกันก่อนหน้านี้

จากภาพที่ 20 ในการบีบอัดข้อมูลซึ่งเริ่มต้นด้วยแท็ก c คือส่วนที่ได้ทำการ Mapping ข้อมูลต่างๆ ให้เป็นตัวแปร และทำให้ข้อมูลเมื่อบีบอัดแล้วจะได้ขนาดที่เล็กลง ซึ่งข้อมูลจริงจะถูกบีบอัดอยู่ในแท็ก d โดยแท็กที่ขึ้นต้นด้วย e เป็นแท็กที่เป็นชื่อย่อของอิลิเมนต์ ส่วนแท็กที่ขึ้นต้นด้วย a เป็นแท็กที่เป็นชื่อย่อของแอตทริบิวต์ v เป็นตัวอักษรที่ชี้ให้เห็นว่าจะมีการเก็บค่าที่อยู่ในอิลิเมนต์ ตัวอักษร x เป็นตัวอักษรที่บ่งบอกว่าเป็นอิลิเมนต์อ้างอิงเพื่อใช้การเรียง ลำดับของการซ้อนกันของอิลิเมนต์ที่เป็นบรรพบุรุษ (Ancestor elements) ในอิลิเมนต์ที่เพิ่งพบก่อนหน้านี้ อย่างเช่น <e1e2e3 a5="a1" …/> ตามด้วย <xe1e2 e3a4="a2" .../> นั้นบ่งบอกว่า <xe1e2 e3a4="a2" …/> มีการเรียงลำดับอิลิเมนต์ที่อยู่ภายใต้ e1e2 ในขณะที่บรรทัดถัดลงมานั้นมีการจัดเรียงลำดับอิลิเมนต์เป็น <xe1 e8v …/> ตามลำดับไปจนจบ ส่วน e0 ไม่จำเป็นต้องอยู่ในข้อมูล

ของการจัดเรียง เนื่องจากในแต่ละเอกสารมีรูทอิลิเมนต์ (Root element) ได้เพียงตัวเดียวเท่านั้น

ดังนั้นในเอกสารที่ถูกบีบอัดจึงมี <e1e2e3 …/> แทนที่จะเป็น <e0e1e2e3 …/> และในการอ่านทำความเข้าใจเอกสารนั้นมีวิธีการคือ ในหนึ่งแท็กของเอกสารที่ถูกบีบอัด เช่น <e1e2e3 a4="a1" a5="e1" e6v="Pissamai" e7v="Aom" /> จะอ่านโดยไล่จาก e1 ซึ่งไม่มีเนื้อหา ภายใต้ e1 มี e2 ซึ่งไม่มีเนื้อหาเช่นกันจะเรียงชิดติดกันกับ e1 ภายใต้ e2 มี e3 ซึ่งมีแอตทริบิวต์อยู่ 2 ค่าด้วยกัน เมื่อเริ่มมีค่าของข้อมูลจะมีการแยกด้วยช่องว่างเพื่อให้อยู่ในรูปของแอตทริบิวต์ แล้วจึงแสดงค่าของข้อมูลออกมา จากนั้นในทุกๆ ค่าที่ถัดมาจะแยกด้วยช่องว่าเสมอ ภายใต้ e3 ยังมี e6 ซึ่งมีเนื้อหาประกอบอยู่ด้วย และภายใต้ e6 ยังมี e7 ที่มีเนื้อหาซ้อนอยู่อีกชั้น ซึ่งเป็นชั้นสุดท้ายของแท็กนี้ โดยการไล่ลำดับชั้นของข้อมูลจะเริ่มจาก e ตัวแรกเป็นต้นไป ตัวอย่าง ถ้าต้องการทราบว่า "Aom" อยู่ภายใต้อิลิเมนต์ใดบ้าง จะทำการอ่านเรียงค่า e ภายในแท็กไปเรื่อยๆจนถึงค่าของข้อมูลที่ต้องการทราบ ซึ่งจะได้ว่า "Aom" อยู่ภายใต้ e1, e2, e3 และ e6 เรียงลำดับกันไปนั่นเอง

ในกรณีที่ข้อมูลมีความยาว หรือมีการปิดแท็กเกิดขึ้นในเอกสารจะเกิดแท็กใหม่และมี x เป็นตัวอ้างอิงลำดับชั้นของอิลิเมนต์ก่อนหน้าที่จะแสดงข้อมูลต่อท้าย เช่น <xe1e2 e3a4="a2" e6v="Veerasuk" /> จะเห็นได้ว่า e3 ซึ่งมีค่าของข้อมูล จะอยู่ภายใต้ e1 และ e2 ตามลำดับ และ e6 ซึ่งมีค่าของข้อมูลจะอยู่ภายใต้ e1, e2 และ e6 ตามลำดับ และมีข้อสังเกตว่า <e1e2e3 a4="a1" .../> ระหว่าง e3 กับ a4 จะแยกกัน แต่ถ้ามีการอ้างอิงถึงอิลิเมนต์ก่อนหน้า อย่างเช่น <xe1e2 e3a4="a2" .../> ระหว่าง e3 กับ a4 จะอยู่ติดกัน

จะเห็นได้ว่าโครงสร้างยังคงเป็นเอกสาร XML จากการบีบอัดด้วยวิธีการนี้จะเห็นได้ว่าเอกสารใหม่มีความกระชับและขนาดเล็กลงกว่าต้นฉบับ อีกทั้งยังเป็นเอกสารที่สามารถอ่านเข้าใจได้ง่าย และยังอยู่ในรูปแบบของเอกสาร XML ที่ถูกต้องตามแบบฟอร์มด้วย

เอกสาร XML อีกตัวอย่างหนึ่งเป็นเอกสารที่เก็บข้อมูลเกี่ยวกับบทความวิจัยต่างๆ ตัวอย่างในลักษณะนี้แสดงไว้ในภาพที่ 21

```
<?xml version="1.0" encoding="UTF-8"?>
<bib>
  <inproceedings>
    <authors>
      <author>N. Zhang</author>
```

```
        <author>V. Kacholia</author>
        <author>M. T. Ozsu</author>
    </authors>
    <title>A Succinct Physical Storage Scheme for Efficient Evaluation for Path Queries in XML</title>
    <booktitle>Proceedings of the IEEE International Conference on Data Engineering
    </booktitle>
    <month>March</month> <year>2004</year>
    <pages>55-65</pages>
    </inproceedings>
<inproceedings>
        <authors>
            <author>B. B. Yao</author>
            <author>M. T. Ozsu</author>
            <author>N. Khandelwal</author>
        </authors>
        <title>XBench Benchmark and Performance Testing of XML DBMSs</title>
        <booktitle>Proceedings of the IEEE International Conference on Data Engineering
</booktitle>
        <month>March</month>
        <year>2004</year>
        <pages>621-632</pages>
    </inproceedings>
</bib>
```

ภาพที่ 21 เอกสาร XML ตัวอย่าง B

หลังจากใช้โปรแกรม XBrevity ทำการบีบอัดข้อมูลแล้วจะได้เอกสารที่ถูกบีบอัดดังแสดงในภาพที่ 22

```
<c>
  <d>
    <e1e2e3 v="N. Zhang" />
    <xe1e2 e3v="V. Kacholia" />
    <xe1e2 e3v="M. T. Ozsu" />
    <xe1 e4v="A Succinct Physical Storage Scheme for Efficient Evaluation for Path Queries in
XML" />
    <xe1 e5v="Proceedings of the IEEE International Conference on Data Engineering" />
    <xe1 e6v="March" />
    <xe1 e7v="2004" />
    <xe1 e8v="55-65" />
    <e1e2e3 v="B. B. Yao" />
    <xe1e8 e3v="M. T. Ozsu" />
    <xe1e8 e3v="N. Khandelwal" />
    <xe1 e4v="XBench Benchmark and Performance Testing of XML DBMSs" />
    <xe1 e5v="Proceedings of the IEEE International Conference on Data Engineering" />
    <xe1 e6v="March" />
    <xe1 e7v="2004" />
    <xe1 e8v="621-632" />
  </d>
  <m f="bib inproceedings authors author title booktitle month year pages "
        b="0 1 2 3 4 5 6 7 8 "/>
</c>
```

ภาพที่ 22 เอกสาร XML ตัวอย่าง B ที่ถูกบีบอัดแล้ว

ในส่วนของเอกสารที่มีอิลิเมนต์แบบผสมผสานนั้นสามารถใช้ XBrevity ในการบีบอัดได้เช่นกัน โดยมีตัวอย่างของเอกสารดังตัวอย่างที่อยู่ในภาพที่ 23 ซึ่งเป็นตัวอย่างของเอกสารที่เป็น Mixed-Content Element โดยในเอกสารประกอบไปด้วย <movie> ที่เป็น Root element และมี <mtitle>, <mname> และ <actor> เป็นลูก (Child) ซึ่งใน <mname> จะมี “shawshank” เป็นเนื้อหา (Content) และอยู่ระหว่างเนื้อหาของ <mtitle> ซึ่งมี “the prisoner of” และมี

“redemption” เป็นเนื้อหาโดยมี “year” เป็นแอตทริบิวต์ซึ่งมีค่าเป็น “1994” ส่วนแท็ก actor มีชื่อแท็กเหมือนกันซ้อนกันอยู่สูงสุด 3 ชั้น ซึ่งมีเนื้อหาที่แตกต่างกันไป

```
<?xml version="1.0"?>
<movie>
   <mtitle year="1994">the prisonner of
<mname>shawshank</mname>redemption</mtitle>
  <actor>Tim Robbins
     <actor>Morgan Freeman <actor>Bob Gunton
     </actor>William Sadler<actor>Clancy Brown
     </actor>Gil Bellows</actor>
   </actor>
</movie>
```

ภาพที่ 23 เอกสาร XML ตัวอย่าง C

```
<c>
 <d>
  <e1 a2="1994" v="the prisonner of"    e3v="shawshank"/>
  <xe1 v="redemption"/>
  <e4 v="Tim Robbins" e4v="Morgan Freeman" e4v2=" Bob Gunton" />
  <xe4e4 v="William Sadler" e4v="Clancy Brown"/>
  <xe4e4 v="Gil Bellows" />
 </d>
<m f="movie mtitle year mname actor "
    b="0 1 a2 3 4 "/>
</c>
```

ภาพที่ 24 เอกสาร XML ตัวอย่าง C ที่ถูกบีบอัดแล้ว

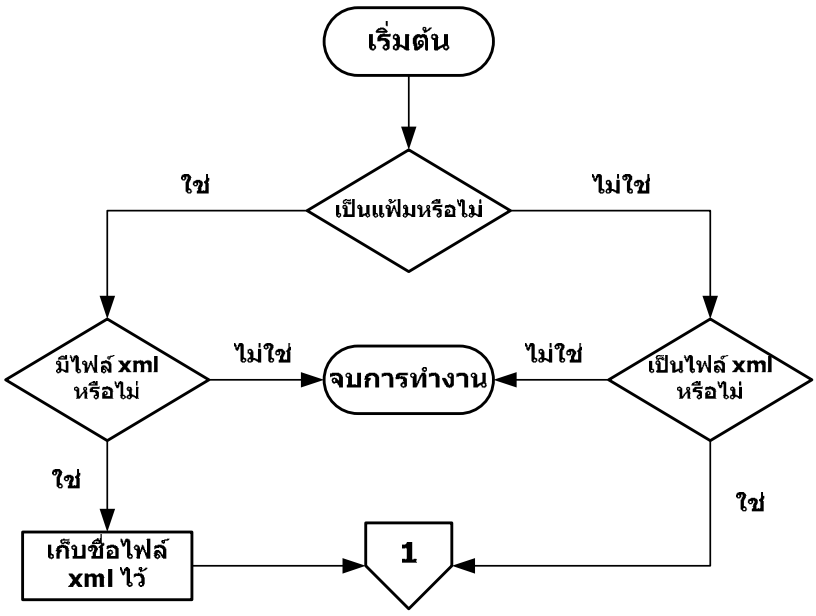
เมื่อมีเอกสารที่มีอิลิเมนต์แบบผสมผสานเกิดขึ้น XBrevity จะพิจารณาบีบอัดเอกสารดังกล่าว โดยถ้าพบแท็กปิดเกิดขึ้นในตัวอย่างนี้คือ </mname> ซึ่งจะได้ <e1 a2="1994" v="the prisoner

of" e3v="shawshank"/> จะจัดให้อยู่ในอิลิเมนต์เดียวกัน ส่วน “redemption” จะแยกเป็นอิลิเมนต์ใหม่โดยภายในจะแทนด้วย x เพื่อบ่งบอกว่าเป็นเนื้อหาที่อยู่ในอิลิเมนต์เดียวกันกับอิลิเมนต์อ้างอิงก่อนหน้านี้ซึ่งก็คือ <mtitle> นั่นเอง ในส่วนแท็ก <actor> ที่มีชื่ออิลิเมนต์เดียวกันและมีจำนวนชั้นซ้อนกันตั้งแต่ 3 ชั้นขึ้นไป จะแยกเป็น 1 อิลิเมนต์ คือ e4 และมี 3 แอตทริบิวต์ คือ v ซึ่งเป็นค่าของ actor ตัวแรก ตัวถัดมาคือ e4v เป็นค่าของ actor ตัวที่สอง และ e4v2 เป็นค่าของ actor ตัวสุดท้าย จะสังเกตได้ว่าเมื่อมีอิลิเมนต์ซ้อนกันหลายตัวและมีชื่อเดียวกัน XBrevity จะทำการกำหนดค่าตัวเลขให้ตั้งแต่การซ้อนกัน 3 ชั้นขึ้นไป โดยเริ่มใส่ตัวเลข 2 เป็นต้นไป ผลที่ได้เป็นไปตามภาพที่ 24 ซึ่งจะเห็นได้ว่าแม้เอกสารจะมีอิลิเมนต์แบบเนื้อหาผสมระหว่างอักขระและแท็ก (Mixed-Content Element) เอกสารนั้นก็สามารถถูกบีบอัดด้วย XBrevity ได้เช่นกัน

3. ส่วนประกอบต่าง ๆ ของโปรแกรม

3.1 ตัวจำแนกเอกสาร XML (XML File Filter)

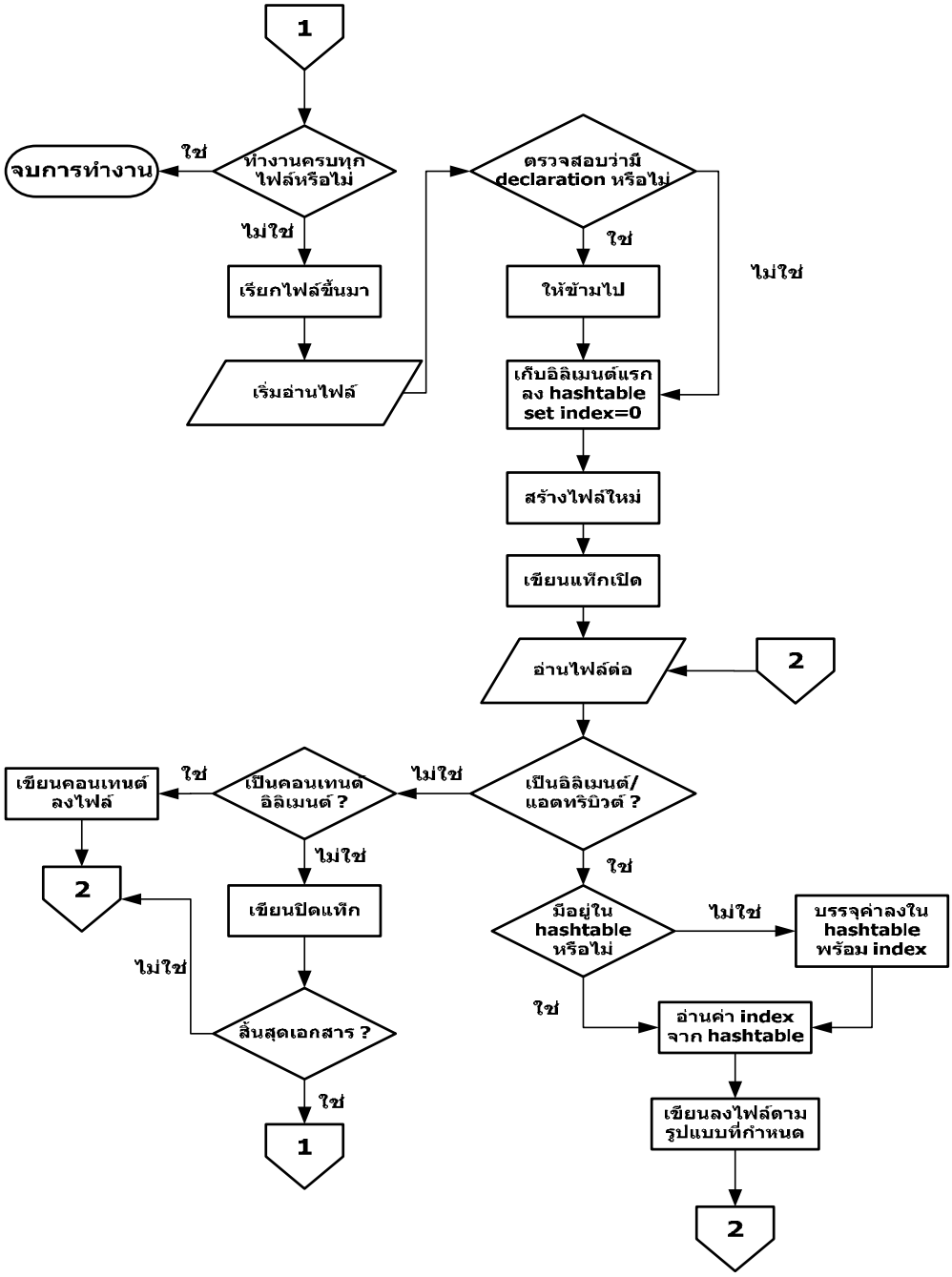
ในการทำงานของ XBrevity นั้นสามารถจำแนกเอกสารที่เข้ามาได้โดยรับคำร้องขอเป็นแบบ ไฟล์ หรือเลือกทำได้ทั้งแฟ้มที่มีเอกสาร XML อยู่ภายใน ซึ่งแสดงในภาพที่ 25



ภาพที่ 25 ผังการจำแนกเอกสาร XML ว่าเรียกจากไฟล์หรือแฟ้ม

3.2 **ตัวบีบอัด** (Compressor)

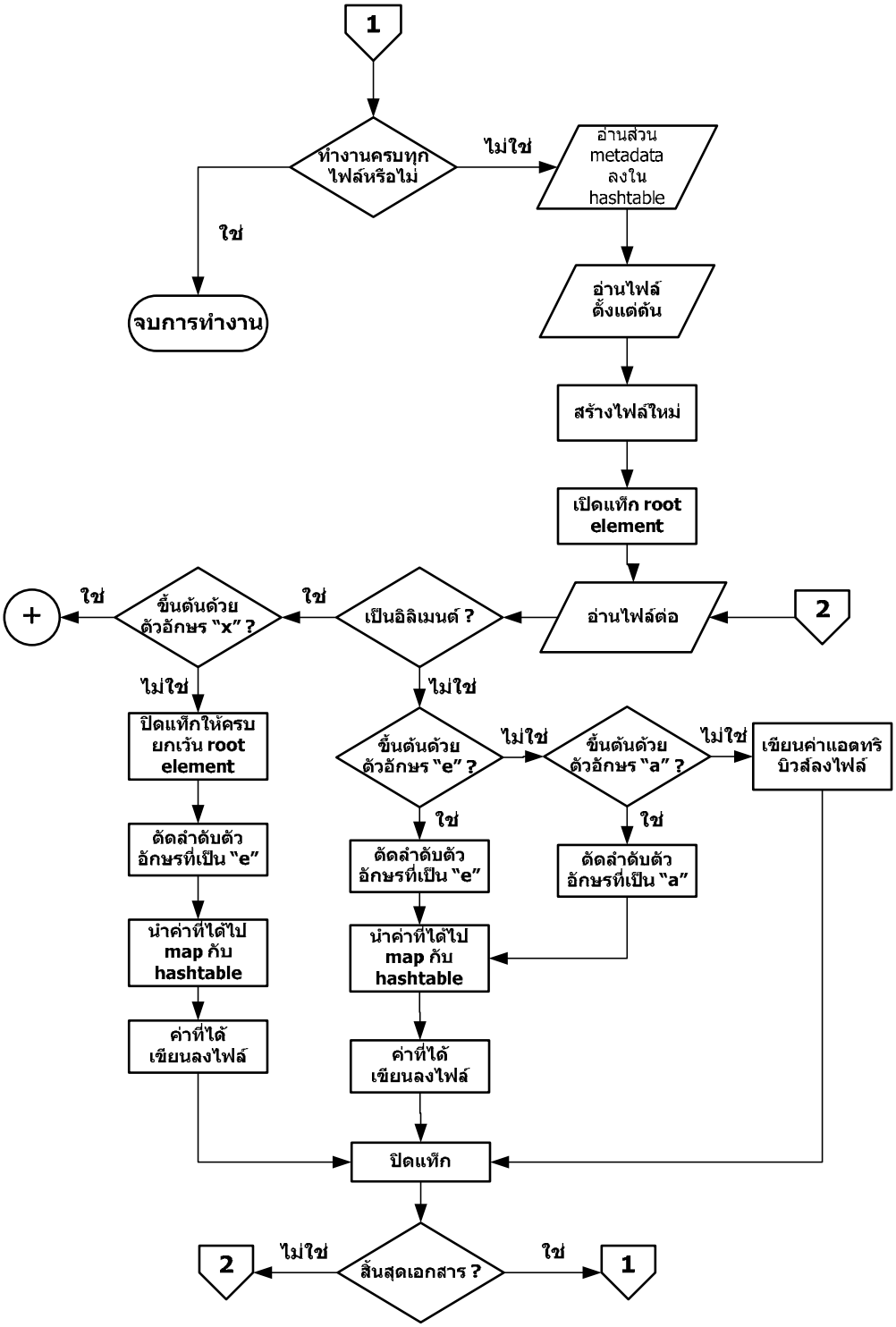
ในกระบวนการบีบอัดนี้สามารถเลือกบีบอัดทั้งแบบไฟล์เดียวหรือทั้งแฟ้มได้ เมื่อจำแนกไฟล์ที่จะทำการบีบอัดจะเข้าสู่กระบวนการอ่านข้อมูลภายในไฟล์นั้น โดยอาศัยการเก็บข้อมูลลงใน Hash table แล้วทำการเขียนข้อมูลดังกล่าวลงบนไฟล์ซึ่งจะมีการจัดรูปแบบต่างๆให้เหมาะสมและทำการจัดเรียงแบบกระชับ ซึ่งแสดงผังการทำงานในภาพที่ 26



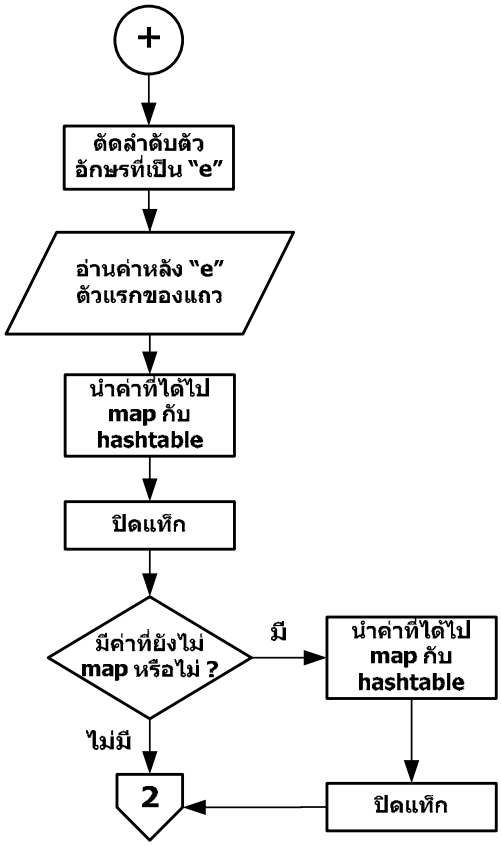
ภาพที่ 26 ผังกระบวนการบีบอัดด้วยการจัดเรียงแบบกระชับ

3.3 **ตัวขยายการบีบอัด** (Decompressor)

ก่อนการขยายตัวขยายจะทำการจำแนกเอกสารเหมือนกับการจำแนกของตัวบีบอัด เมื่อผ่านส่วนของการจำแนกเอกสารแล้วจะเข้าสู่กระบวนการขยายการบีบอัด



ภาพที่ 27 ผังกระบวนการคลายการบีบอัด



ภาพที่ 28 ผังกระบวนการคลายการบีบอัด (ต่อ)

การประมวลผลข้อมูลที่ได้จากการจำลอง

4.1 วัดอัตราส่วนของการบีบอัด (Compression Ratio) ซึ่งเป็นอัตราส่วนร้อยละในการบีบอัด โดยหาได้จากสมการที่ *(1)*

$$\text{อัตราส่วนของการบีบอัด} = \left(1 - \frac{\text{ขนาดของเอกสารที่บีบอัดแล้ว}}{\text{ขนาดของเอกสารต้นฉบับ}}\right) \times 100\% \qquad (1)$$

4.2 เวลาในการบีบอัด (Compression Time) คือเวลาที่ใช้ในการบีบอัดเอกสาร XML โดยมีหน่วยในการวัดเป็นวินาที

ผลการวิจัย

1. คุณลักษณะของเอกสารที่ใช้ทดลอง

ในการทดลองนี้ได้ใช้ตัวอย่างในการทดลองเป็น 2 ส่วนคือส่วนที่ได้จากการสร้างเอกสารด้วย XMark [16] ซึ่งทำการสร้างเอกสาร XML ขึ้นมาทั้งหมด 5 เอกสาร โดยมีขนาดของเอกสารที่แตกต่างกันขึ้นอยู่กับค่าแฟกเตอร์ (XMark factor) ที่ใช้ในการกำหนดขนาดของเอกสารที่จะสร้างขึ้นมา ซึ่งแสดงคุณลักษณะของเอกสารดังตารางที่ 2

ตารางที่ 2 คุณลักษณะของเอกสารที่ใช้ในการทดสอบโดยสร้างจาก XMark

XML Files	Size (bytes)	Depth	Element Types	Attribute Types	XMark factor
a.xml	27,233	8	58	3	0.0001
b.xml	118,274	9	72	4	0.001
c.xml	1,182,547	10	72	5	0.01
d.xml	11,875,066	10	72	5	0.1
e.xml	118,552,713	10	72	5	1

จากข้อมูลในตารางที่ 2 ข้อมูลซึ่งบ่งบอกชนิดของข้อมูลซึ่งได้แก่

Size คือขนาดของเอกสาร XML มีหน่วยเป็นไบต์ (bytes)

Depth คือความลึกหรือจำนวนชั้นที่ซ้อนกันสูงสุดที่เกิดขึ้นในเอกสาร

Element Types คือจำนวนชนิดของอิลิเมนต์ในเอกสาร

Attribute Types คือจำนวนชนิดของแอตทริบิวต์ในเอกสาร

Xmark factor คือค่าแฟกเตอร์ที่ใช้กำหนดขนาดในการสร้างเอกสารตัวอย่าง

ในส่วนที่สองเป็นเอกสาร XML ที่มีการใช้งานจริง ซึ่งได้ทำการดาวน์โหลดมาจากเครือข่ายอินเทอร์เน็ตซึ่งมีที่มาจากแหล่งต่างๆ และมีคุณลักษณะทั่วไปที่นำมาใช้เหมือนกับเอกสารที่สร้างจาก XMark ดังตารางที่ 3

ตารางที่ 3 คุณลักษณะของเอกสารที่ใช้ในการทดสอบจากอินเทอร์เน็ต

XML Files	Size (bytes)	Depth	Element Types	Attribute Types
1998statistics.xml[1]	669,309	7	46	0
factbook.xml[2]	4,222,646	7	199	0
format.xml[3]	1,166,916	6	15	1
shakespeare.xml	251,865	6	17	0
weblog.xml[4]	3,021,921	3	12	0

2. ผลการทดลอง

เมื่อทำการบีบอัดในแต่ละวิธีแล้ว ผลที่ได้ปรากฏอยู่ในตารางที่ 4 โดยวัดจากขนาดที่บีบอัดได้ซึ่งมีหน่วยเป็นกิโลไบต์ (KByte)

ตารางที่ 4 การเปรียบเทียบขนาดของเอกสารที่บีบอัดแล้ว

XML Files	Compressed File Size (KByte)			
	Original	gzip	XMill	XBrevity
a.xml	27.0	10.4	10.4	25.6
b.xml	115.5	39.8	37.9	112.1
c.xml	1154.8	371.5	338.1	1,119.1
d.xml	11596.7	3,731.6	3,320	11,238.9
e.xml	115774.1	37,357.6	32,996	112,223.2
1998statistics.xml	653.6	65.2	34.70	648.0
factbook.xml	4,123.7	978.7	683.7	3,259.6
format.xml	1,139.6	111.7	81.66	1,012.1
shakespeare.xml	246.0	66.4	62.68	252.5
weblog.xml	2,951.1	124.5	72.69	2,055.4

[1] http://www.ibiblio.org/xml/examples/1998statistics.xml

[2] http://www.w3.org/XML/Binary/2005/03/test-data/Over100K/factbook.xml

[3] http://www.eda.org/pub/ibis/xml/sample1/format.xml

[4] http://gnosis.cx/download/weblog.xml

โดยการทดลองนี้ได้ทำบน Intel Pentium 4 CPU 1.70 GHz หน่วยความจำ 512 MB ระบบปฏิบัติการ Windows XP Professional with Service Pack 2 และใช้ภาษาจาวา (JDK 5.0 Update 1.6.0) ในการสร้างและพัฒนาระบบการบีบอัด

จากตารางที่ 4 จะเห็นได้ว่าขนาดที่บีบอัดด้วย gzip และ XMill มีขนาดเล็กมากเมื่อเทียบกับ XBrevity เนื่องจากว่า gzip และ XMill เป็นการบีบอัดในรูปแบบของไบนารี แต่ XBrevity ทำการบีบอัดเฉพาะส่วนของแท็กเท่านั้น และยังคงเป็นเอกสารในรูปแบบ XML อยู่ แต่อย่างไรก็ตาม เมื่อเรานำเอา XBrevity ที่ได้ทำการจัดเรียงข้อมูลแบบกระชับแล้ว มาใช้ร่วมกับ gzip จะได้ผลการทดลองที่แตกต่างออกไป ดังแสดงในตารางที่ 5

ตารางที่ 5 การเปรียบเทียบขนาดเมื่อ XBrevity+gzip

XML Files	Compressed File Size (KByte)			
	Original	gzip	XMill	XBrevity+gzip
a.xml	27.0	10.40	10.4	10.5
b.xml	115.5	39.8	37.9	40.4
c.xml	1154.8	371.5	338.1	373.4
d.xml	11596.7	3,731.6	3,320	3744.9
e.xml	115774.1	37,357.6	32,996	37482.4
1998statistics.xml	653.6	65.2	34.70	66.1
factbook.xml	4,123.7	978.7	683.7	945.6
format.xml	1,139.6	111.7	81.66	109.4
shakespeare.xml	246.0	66.4	62.68	66.3
weblog.xml	2,951.1	124.5	72.69	114.6

จากการนำผลการบีบอัดที่ได้จาก XBrevity มารวมกับ gzip ทำให้ประสิทธิภาพในการบีบอัดเอกสาร ทำได้ดีขึ้นกว่าการใช้ XBrevity บีบอัดเพียงอย่างเดียว เนื่องจากว่าเอกสารที่ผ่านการจัดเรียงแบบกระชับด้วย XBrevity แล้วนั้นขนาดของเอกสารจะเล็กกว่าต้นฉบับเดิมและเมื่อรวมกับ gzip จึงทำให้ขนาดที่บีบอัดได้ใกล้เคียงกับการบีบอัดด้วย gzip เพียงอย่างเดียว แต่การรวมกันของ XBrevity+gzip นั้นจะมีคุณสมบัติเหมือน gzip ซึ่งค้นหาข้อมูลที่ถูกบีบอัดไม่ได้และไม่ Well-formed โดยผลของการบีบอัดที่ได้แสดงไว้ในตารางที่ 6 และ 7

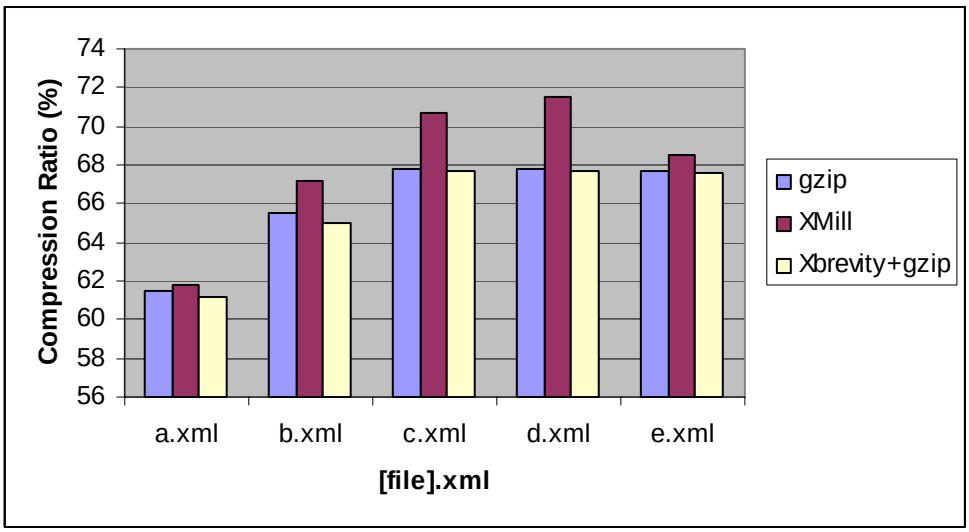
ตารางที่ 6 การเปรียบเทียบอัตราส่วนในการบีบอัดไฟล์จาก XMark

XML Files	Compression Ratio (%)		
	gzip	XMill	XBrevity+gzip
a.xml	61.47	61.83	61.2
b.xml	65.55	67.18	65.0
c.xml	67.83	70.72	67.7
d.xml	67.82	71.50	67.7
e.xml	67.73	68.52	67.6
Average	**66.08**	**68.52**	**65.8**

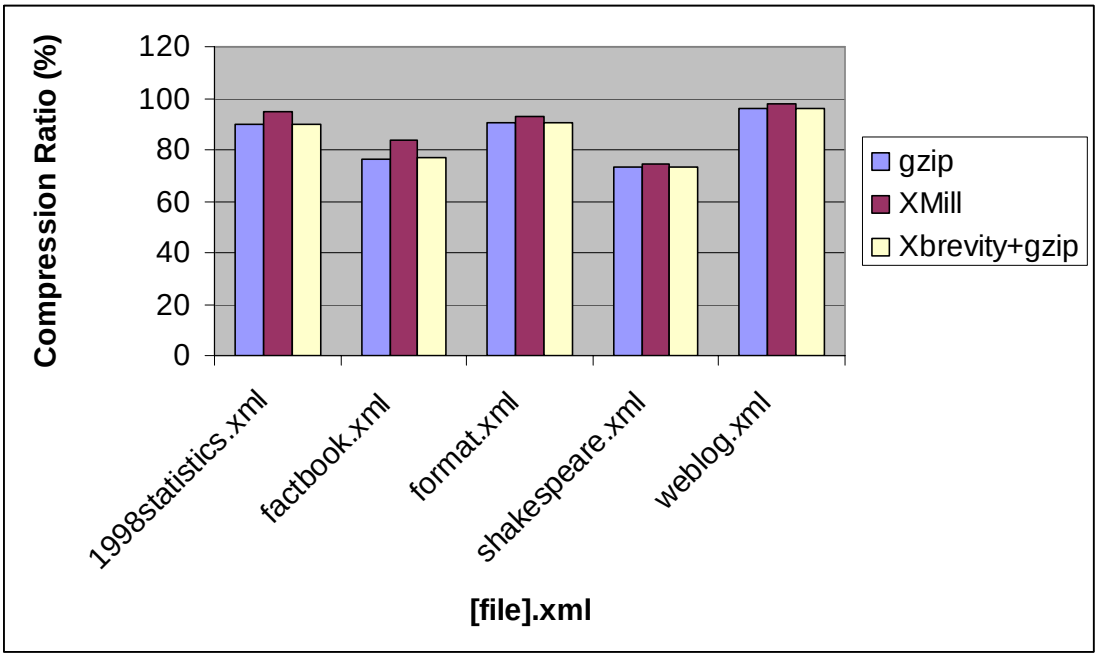
ตารางที่ 7 การเปรียบเทียบอัตราส่วนในการบีบอัดไฟล์จากอินเทอร์เน็ต

XML Files	Compression Ratio (%)		
	Gzip	XMill	XBrevity+gzip
1998statistics.xml	90.03	94.69	89.9
factbook.xml	76.27	83.42	77.1
format.xml	90.20	92.83	90.4
shakespeare.xml	73.00	74.52	73.0
weblog.xml	95.78	97.54	96.1
Average	**85.06**	**88.60**	**85.3**

แต่เมื่อเทียบกับ XMill แล้วเอกสารยังมีขนาดใหญ่กว่าเล็กน้อย จะเห็นได้ว่าเมื่อขนาดของ XBrevity ที่รวมกับ gzip มีขนาดเล็กลงทำให้อัตราส่วนในการบีบอัดสูงขึ้นตามไปด้วย และขนาดมีความใกล้เคียงกันมากเมื่อเทียบกับการบีบอัดด้วย gzip เพียงอย่างเดียว ในบางเอกสารที่มีอัตราการบีบอัดน้อยกว่าเล็กน้อยนั้น เนื่องมาจากอิลิเมนต์และเนื้อหาจำนวนไม่น้อยอยู่ในชั้นที่ลึกลงไปภายในเอกสาร ทำให้เกิดการอ้างอิงอิลิเมนต์ก่อนหน้ามีความยาวและมีจำนวนมากตามไปด้วย โดยประสิทธิภาพการบีบอัดเมื่อนำ XBrevity ที่รวมกับ gzip ได้แสดงไว้ดังกราฟจากภาพที่ 29 และ 30



ภาพที่ 29 กราฟแสดงประสิทธิภาพการบีบอัดไฟล์จาก XMark



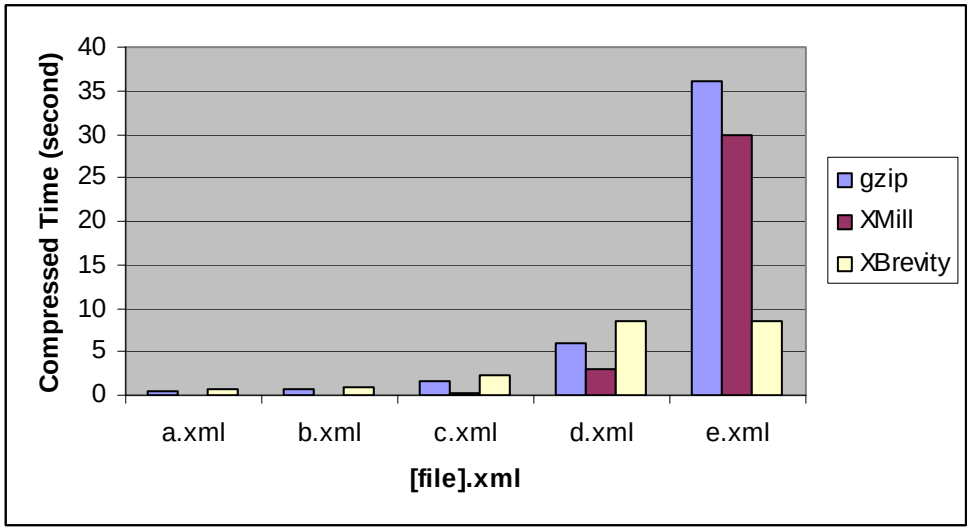
ภาพที่ 30 กราฟแสดงประสิทธิภาพการบีบอัดไฟล์จากอินเทอร์เน็ต

จากประสิทธิภาพการบีบอัดที่ได้แสดงไว้แล้ว เมื่อทำการวัดเวลาที่ใช้ในการบีบอัดแล้วซึ่งได้ผลตามตารางที่ 8 ซึ่งแสดงเวลาที่ใช้ในการบีบอัดด้วยวิธีต่างๆ

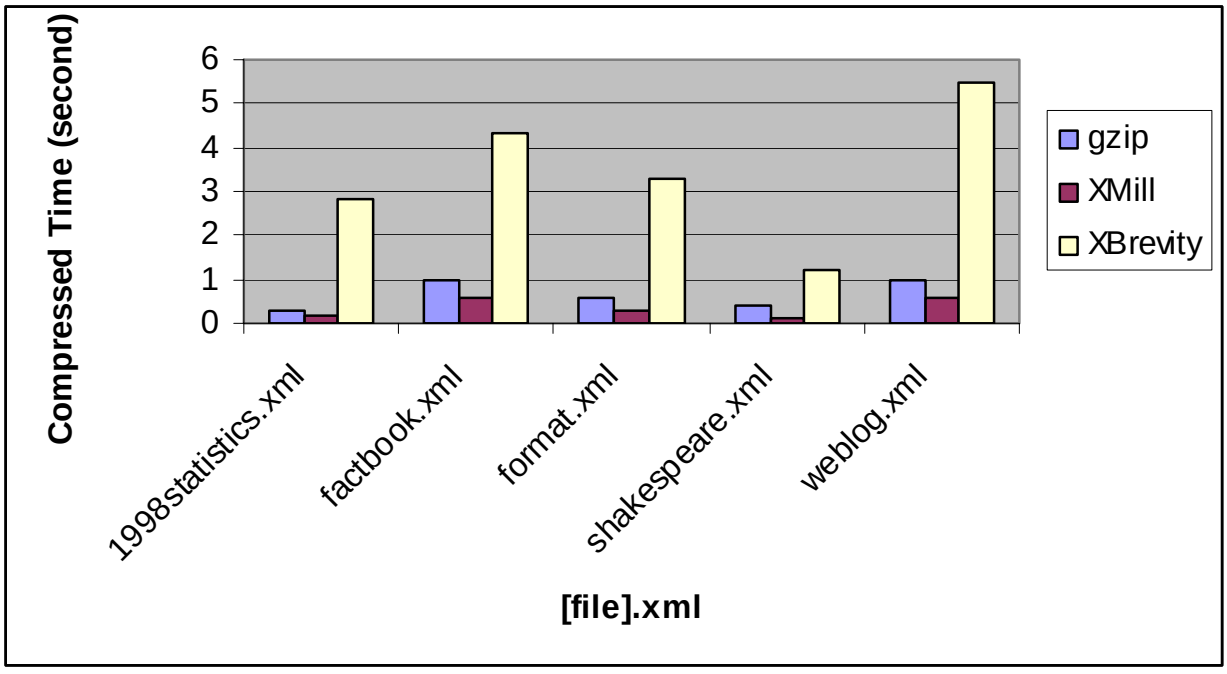
ตารางที่ 8 การเปรียบเทียบเวลาที่ใช้ในการบีบอัด

XML	Compressed Time (second)		
Files	gzip	XMill	XBrevity
a.xml	0.4	< 0.1	0.7
b.xml	0.8	0.1	1.0
c.xml	1.5	0.3	2.3
d.xml	6	3	8.5
e.xml	36	30	8.5
1998statistics.xml	0.3	0.2	2.8
factbook.xml	1	0.6	4.3
format.xml	0.6	0.3	3.3
shakespeare.xml	0.4	0.1	1.2
weblog.xml	1	0.6	5.5

เมื่อนำเวลามาประเมินผลแล้วผลการบีบอัดที่ได้จาก XMill ใช้เวลาน้อยที่สุดและ XBrevity ใช้เวลามากกว่าวิธีการอื่นๆ เมื่อเอกสารมีขนาดเล็กความแตกต่างเรื่องเวลาถือได้ว่าน้อย แต่ถ้าเอกสารมีขนาดใหญ่ขึ้นทำให้เห็นความแตกต่างได้ชัดเจนมากขึ้นตามไปด้วย อันเนื่องมาจาก XBrevity ใช้ภาษาจาวา (Java language) เป็นเครื่องมือในการสร้างโปรแกรมเพื่อทดสอบเทคนิคนี้ โดยมีขั้นตอนของการอ่านไฟล์และแยกส่วนของข้อมูลออกเป็นอิลิเมนต์ แอตทริบิวต์ และเนื้อหาของเอกสาร เมื่ออ่านเอกสารในแต่ละส่วนแล้วจะต้องเก็บค่าลงใน Hashtable ก่อนแล้วจึงเรียกข้อมูลมาทำการเขียนลงไปบนไฟล์ โดยในการเขียนลงไฟล์นั้นต้องทำการตรวจสอบให้เป็นไปตามแบบฟอร์มของภาษา XML อีกด้วย ซึ่งกระบวนการต่างๆ ดังกล่าวมีหลายขั้นตอนทำให้เวลาที่ใช้บีบอัดมากตามไปด้วย แต่ XMill ใช้ภาษาซีพลัสพลัส (C++ language) ในการพัฒนาโดยเมื่ออ่านเอกสารและทำการคัดแยกแล้วจะทำการเข้ารหัสซึ่งเป็นแบบไบนารีไปจนเสร็จกระบวนการ และ gzip ใช้ภาษาซี (C language) ในการพัฒนาโดยอ่านไฟล์แล้วเข้ารหัสแบบไบนารีไปจนเสร็จกระบวนการโดยไม่ได้ทำการแยกแยะส่วนประกอบต่างๆ ของภาษา XML แต่อย่างใด ซึ่งไฟล์ที่ได้จากการบีบอัดของ XBrevity เป็นรูปแบบที่แตกต่างกันกับ XMill และ gzip กล่าวคือ XBrevity ยังเป็นไฟล์ XML อยู่สามารถนำไปใช้ได้โดยไม่ต้องขยาย ส่วน XMill และ gzip ต้องขยายการบีบอัดก่อนจึงจะสามารถใช้งานเอกสารนั้นได้ ซึ่งผลที่ได้แสดงเป็นกราฟไว้ในภาพที่ 31 และ 32



ภาพที่ 31 กราฟแสดงเวลาที่ใช้ในการบีบอัดไฟล์จาก XMark



ภาพที่ 32 กราฟแสดงเวลาที่ใช้ในการบีบอัดไฟล์จากอินเทอร์เน็ต

จากผลการทดลองที่ได้นั้น จะเห็นได้ถ้าเปรียบเทียบอัตราส่วนของการบีบอัดแล้ว XMill และ gzip มีค่าค่อนข้างสูง และระยะเวลาในการบีบอัดก็ใช้เวลาที่น้อย เมื่อเทียบกับการใช้ XBrevity เพียงอย่างเดียวเพราะ XBrevity บีบอัดเฉพาะส่วนของแท็กเท่านั้น แต่อย่างไรก็ตามถ้า

นำ XBrevity ที่ได้ทำการบีบอัดแล้วมาใช้ร่วมกับ gzip จะเห็นว่ามีขนาดของข้อมูลที่ลดลงมาค่อนข้างมาก รวมถึงเวลาที่ใช้ในการบีบอัดก็เร็วกว่าเดิมแต่ไม่สามารถทำการค้นหาข้อมูลที่ผ่านการรวมกันได้ ซึ่งผลที่ได้ใกล้เคียงกับการใช้ gzip และ XMill โดยเวลาที่ใช้ในการขยายไฟล์ออกเป็นเอกสารต้นฉบับเดิมน้อยกว่าการบีบอัดเอกสาร ไม่ว่าจะใช้วิธีการใด แต่เนื่องจาก XBrevity ต้องเขียนข้อมูลที่อ่านได้ลงไปบนไฟล์ที่เป็น .xml ด้วยจึงทำให้ใช้เวลานานแต่มีความกระชับ อีกทั้งข้อมูลที่เขียนลงไปนั้นเป็นรูปย่อของเอกสาร ซึ่งยังคงข้อดีของภาษา XML ไว้ ทำให้เป็นผลดีในแง่ของการอ่านเอกสารเพื่อทำความเข้าใจเอกสารที่อยู่ในรูปแบบที่บีบอัดแล้วได้ง่าย เพราะว่าในการนำไปใช้งานจริงนั้น การที่ผู้ใช้ปลายทางสามารถถอดรหัสที่ถูกบีบอัดออกมาให้เป็นข้อมูลเดิมได้นั้น จำเป็นจะต้องใช้ตัวขยายข้อมูลจากข้อมูลที่ถูกบีบอัดแล้ว ซึ่งอาจทำให้ไม่สะดวกกับการใช้งาน และเป็นข้อจำกัดในการนำไปใช้งานกับกลุ่มคนทั่วไป รวมถึงการที่ต้องใช้เวลาในการขยายข้อมูลเพิ่มขึ้นตามไปด้วย

ในกระบวนการคลายการบีบอัดนั้น จะนำไฟล์ที่ถูกบีบอัดมาทำการขยายออกเป็นเอกสารต้นฉบับ ผลที่ได้เป็นเอกสารต้นฉบับที่มีความสมบูรณ์ และถูกต้องตามหลักของภาษา XML และเป็นไปในทางเดียวกันทุกเอกสารที่มีการบีบอัด แต่การขยายเอกสารจะทำได้เร็วกว่าการบีบอัด และในส่วน XBrevity ที่ใช้การบีบอัดรวมกับ gzip มีคุณสมบัติเดียวกันกับ gzip และจะต้องทำการ gunzip ก่อนจึงจะได้เอกสารในรูปแบบที่ถูกบีบอัดด้วย XBrevity

สรุปผล

การบีบอัดข้อมูล XML มีความสำคัญเพราะ XML เป็นภาษาที่มีข้อดีอยู่หลายประการ ซึ่งปัจจุบันได้มีการใช้กันอย่างแพร่หลาย แต่ข้อจำกัดของ XML ก็มีเช่นกัน โดยปกติแล้วเอกสารภาษา XML จะมีขนาดใหญ่กว่าเอกสารของเท็กซ์ไฟล์ (Text File) เนื่องจากเอกสารภาษา XML มักจะมีการใช้แท็กกำกับความหมายของข้อมูล การส่งข้อมูลโดยใช้ภาษา XML ทำให้เกิดความต้องการเน็ตเวิร์กแบนด์วิดธ์ (Network Bandwidth) ขนาดใหญ่เนื่องจากขนาดของไฟล์ วิธีหนึ่งที่จะช่วยลดเน็ตเวิร์กแบนด์วิดธ์ในการส่งข้อมูล คือการบีบอัดข้อมูลให้มีขนาดเล็กลง อีกทั้งยังช่วยให้ใช้พื้นที่ในการเก็บข้อมูลลดลงตามไปด้วย จากงานวิจัยที่ผ่านมาเกี่ยวกับการบีบอัดข้อมูล XML นั้นได้ทำการบีบอัดข้อมูลให้มีขนาดเล็กลง ซึ่งในบางวิธีการจะต้องใช้โครงสร้างของภาษา เป็น

ตัวกำหนดไวยากรณ์เป็นของตัวเอง และบางวิธีการต้องขยายเอกสารให้เป็นต้นฉบับก่อนจึงจะสามารถใช้งานได้ ซึ่งยุ่งยากต่อการนำไปใช้งานทั่วๆ ไป

เทคนิคที่ได้นำเสนอคือการบีบอัดข้อมูล XML โดยใช้วิธีจัดเรียงข้อมูลแบบกระชับ (XBrevity) ซึ่งเป็นเทคนิคการจัดเรียงแบบกระชับ เพื่อให้เอกสารมีขนาดเล็กลงจากเดิม และยังสามารถเข้าใจความหมายเอกสารที่ถูกบีบอัดได้ง่าย โดยไม่ต้องมีการขยายเอกสารให้เป็นต้นฉบับเดิม เพราะอยู่ในรูปแบบของภาษา XML อยู่แล้ว ทำให้ยังคงข้อดีที่มีอยู่ในภาษา XML อีกทั้งยังง่ายต่อการใช้งานทั่วๆ ไป ซึ่งไม่ต้องมีตัวกำหนดไวยากรณ์แบบใดแบบหนึ่งโดยเฉพาะ และสามารถที่จะค้นหาข้อมูลที่ถูกบีบอัดได้ด้วย XQuery [27] เพื่อที่จะได้ไม่เสียเวลาในการขยายข้อมูลอีก หรือหากจะขยายข้อมูลก็สามารถทำได้เช่นกัน ในการพัฒนานี้ได้มีการเพิ่มเติมให้โปรแกรมสามารถเลือกบีบอัดและคลายการบีบอัดแฟ้มที่มีเอกสาร XML ได้ทั้งแฟ้มอีกด้วย นอกเหนือจากการบีบอัดทีละไฟล์

งานวิจัยนี้มีผลงานตีพิมพ์ดังต่อไปนี้

1. Lakhasophon P, Runapongsa K. “A Survey of XML Data Compression”. In: NECSEC2005. Proceeding of the 1st Northeastern Computer Science and Engineering Conference; 2005 March 31–April 1; Khon Kaen, THAILAND; 2005. p. 349-360. [in Thai]

2. Lakhasophon P, Runapongsa K. “XML Data Compression using Brevity Encoding”. In: NCSEC2005. Proceeding of the 9th Nation Computer Science and Engineering Conference; 2005 October 27-28; Bangkok, THAILAND; 2005. p. 349-360. [in Thai]

3. Lakhasophon P, Runapongsa K. “XBrevity: XML Data Compression using Brevity Encoding”. In: ITC-CSCC 2006. Proceeding of the 21st International Technical Conference on Circuits/Systems, Computers and Communications; 2006 July 10-13; Chiang Mai, Thailand; 2006.

ปัญหาและอุปสรรค

เนื่องจากเวลาที่ใช้บีบอัดค่อนข้างนาน จึงจำเป็นต้องเข้าไปแก้ไขในส่วนของการทำงานภายในใหม่บางส่วนเพื่อให้ได้รูปแบบที่ถูกต้องและสามารถทำงานได้เร็วขึ้นกว่าเดิม โดยที่สาเหตุที่โปรแกรมในเวอร์ชันก่อนหน้านี้ทำงานช้าเนื่องจากมีการเปลี่ยนไลบรารีที่ใช้ในการอ่านเอกสารเอกซ์เอ็มแอลจาก Simple API for XML (SAX) เป็น Streaming API for XML (StAX) และได้ปรับปรุงโค้ดในโปรแกรม โดยให้มีการสร้างออปเจกต์ใหม่ให้น้อยที่สุดเท่าที่จะทำได้ เพื่อประหยัดพื้นที่หน่วยความจำ

การทดลองนี้ได้เปลี่ยนไลบรารีที่ใช้ในการอ่านเอกสารเอกซ์เอ็มแอลจาก Simple API for XML (SAX) เป็น Streaming API for XML (StAX) เนื่องจากการทำงานของ StAX นั้นจะเร็วกว่าการทำการของ SAX ดังตารางที่ 9 ซึ่งเป็นการเปรียบเทียบเวลาในการประมวลผลไฟล์โดยใช้ SAX และ StAX โดยเวลาเฉลี่ยในการประมวลผลไฟล์ของ SAX จะประมาณ 7.28 วินาที ส่วนเวลาเวลาเฉลี่ยในการประมวลผลไฟล์ของ StAX จะประมาณ 6.60 วินาที ซึ่งน้อยกว่า SAX ถึง 0.68 วินาที

ตารางที่ 9 การเปรียบเทียบเวลาในการประมวลผลไฟล์โดยใช้ SAX และ StAX

XML	Processing Time (second)	
Files	SAX	StAX
1998statistics.xml	8.07	5.84
a.xml	0.70	0.65
b.xml	1.14	1.13
format.xml	10.59	9.48
shakespeare.xml	2.13	2.02
weblog.xml	21.04	20.48
Average	7.28	6.60

และนอกจากนั้น SAX ยังมีข้อเสียตรงที่ SAX แต่ละเหตุการณ์ของ SAX นั้น อาจจะมีการประมวลผลมากกว่า 1 ครั้งก็ได้ ซึ่งมีผลทำให้ผลลัพธ์ที่ได้ไม่ถูกต้อง แต่การทำงานของ StAX แต่ละเงื่อนไขของตัวอ่าน จะประมวลผลเพียงแค่ครั้งเดียวซึ่งจะได้ผลลัพธ์ที่ถูกต้อง

ตารางที่ 10 เปรียบเทียบเวลาที่ใช้ในการบีบอัดไฟล์ของโปรแกรมเวอร์ชันเก่า (ซึ่งใช้ในการรายงานผลการทดลองในรายงานความก้าวหน้าของงานวิจัย) กับโปรแกรมเวอร์ชันใหม่ และตารางที่ 11 เปรียบเทียบเวลาที่ใช้ในการขยายไฟล์ของโปรแกรมเวอร์ชันเก่ากับโปรแกรมเวอร์ชันใหม่

ตารางที่ 10 การเปรียบเทียบเวลาที่ใช้ในการบีบอัดไฟล์ของโปรแกรมเวอร์ชันกับโปรแกรมเวอร์ชันใหม่

XML	Compressed Time (second)	
Files	XBrevity (old)	XBrevity (new)
a.xml	2.4	0.7
b.xml	8.6	0.9
c.xml	80.4	2.3
d.xml	776.6	2.8
e.xml	7735.6	8.5
1998statistics.xml	157.2	2.9
factbook.xml	212.3	4.3
format.xml	172.2	3.4
shakespeare.xml	26.9	1.2
weblog.xml	296.3	5.5

ตารางที่ 11 การเปรียบเทียบเวลาที่ใช้ในการขยายไฟล์ของโปรแกรมเวอร์ชันเก่ากับโปรแกรมเวอร์ชันใหม่

XML	Decompressed Time (second)	
Files	XBrevity (old)	XBrevity (new)
a.xml	2.3	0.7
b.xml	6.5	1.2
c.xml	N/A	2.8
d.xml	N/A	9.7
e.xml	N/A	6.8
1998statistics.xml	83.3	3.8
factbook.xml	N/A	7.1
format.xml	8.3	4.3
shakespeare.xml	17.3	1.6
weblog.xml	85.5	9.2

* หมายเหตุ ค่า N/A หมายถึง โปรแกรมไม่สามารถประมวลผลไฟล์นั้นๆได้ เนื่องจากเกิดข้อผิดพลาด

และในโปรแกรมเก่านั้นยังไม่สามารถทำงานร่วมกับเอกสารเอกซ์เอ็มแอลบางเอกสารที่มีเครื่องหมายพิเศษ เช่น <, >, &, " ดังนั้นจึงได้มีการแก้ไขปัญหานี้กับโปรแกรมใหม่

ในเทคนิค XBrevity ที่ได้นำเสนอไป ซึ่งเน้นถึงความสำคัญในการบีบอัดข้อมูลในส่วนของโครงสร้างแท็ก แต่ตัวอักขระซึ่งเป็นข้อมูลภายในแท็ก (Content element) ยังไม่ได้รับการบีบอัด ดังนั้นจึงยังสามารถพัฒนาประสิทธิภาพของเทคนิคการบีบอัดได้มากขึ้นไปอีกโดยการบีบอัดอักขระภายในแท็ก ซึ่งจะมีผลให้ขนาดของเอกสารลดลงได้อีก รวมไปถึงการพัฒนาให้การบีบอัดใช้เวลาน้อยลงจากเดิมที่เป็นอยู่ นอกจากนี้แนวทางในการนำเอาเอกสารที่ถูกบีบอัดแล้วซึ่งอยู่ในรูปแบบ XML อยู่แล้วไปใช้เพื่อการค้นหาข้อมูลก็เป็นงานวิจัยที่น่าสนใจ และผลการทดลองจากการบีบอัดข้อมูล XML หากสามารถนำไปประยุกต์ใช้งานที่ต้องใช้ภาษา XML รวมถึงการใช้งานบนเว็บเซอร์วิส (Web Services) ซึ่งเป็นอีกงานวิจัยหนึ่งที่น่าสนใจ และเกิดประโยชน์ในอนาคต ใน

ส่วนของการ Query นั้นสามารถทำได้แต่ต้องปรับปรุงวิธีการเดิมที่มีอยู่แล้วซึ่งคือ XQuery มาประยุกต์วิธีการเพื่อให้ใช้กับเอกสารที่บีบอัดด้วย XBrevity ซึ่งสามารถทำได้ด้วยการ Query สองชั้น โดยชั้นแรกให้ทำการค้นหาชื่ออิลิเมนต์ที่ต้องการแล้วทำการ Mapping กับ Index ที่เป็นหมายเลขซึ่งอยู่ภายในแท็ก <m …/> เมื่อได้ผลจากชั้นนี้แล้ว ในชั้นที่สองจึงทำการ Query ในเอกสารซึ่งอยู่ภายในแท็ก <d …/> ต่อไป ทำให้การค้นหาข้อมูลที่ถูกบีบอัดด้วย XBrevity สามารถทำได้เช่นกัน

บรรณานุกรม

1. Amazon. **Amazon Web Services.** Retrieved 14 July, 2004, from http://www.amazon.com/gp/aws/ landing.html
2. BEA Systems, IBM, Microsoft, SAP AG, Siebel Systems. **Business Process Execution Language for Web Services.** Retrieved 19 July, 2004, from http://www-128.ibm.com/developerworks/library/specification/ws-bpel/
3. eBay. **eBay API.** Retrieved 19 July, 2004, from http://developer.ebay.com/common/api
4. Ferraiolo J, Fujisawa J, Jackson D. **Scalable Vector Graphics (SVG) 1.1 Specification.** Retrieved 6 August, 2004, from http://www.w3.org/TR/SVG
5. Gailly J.L, Adler M. **gzip : The Compressor Data.** Retrieved 19 June, 2004, from http://www.gzip.org
6. Google. Google Web APIs (beta). Retrieved 25 July, 2004, from http://www.google.com/apis/
7. Hunter D, Cagle C, Dix C, Kovack R, Pinnock J, Rafter J. Beginning XML. 2nd ed. Wiley Publishing, Inc., 10475 Crosspoint Blvd., Indianapolis, IN 46256, (317) 572-3447, fax (317) 572-4355: Wrox Press; 2002.
8. Jagadish H.V, Lakshmanan L.V.S, Scannapieco M, Srivastava D, Wiwatwattana N. Colorful XML : One Hierarchy Isn't Enough. In: **the 2004 ACM SIGMOD.** Proceeding of the 2004 ACM SIGMOD Internaional Conference on Management of Data; 2004 June 13-18; 2004. p. 251-262.
9. Lakhasophon P, Runapongsa K. A Survey of XML Data Compression. In: **NECSEC2005.** Proceeding of the 1st Northeastern Computer Science and Engineering Conference; 2005 March 31–April 1; Khon Kaen; THAILAND; 2005. p. 80-87. [in Thai]
10. Lakhasophon P, Runapongsa K. XML Data Compression using Brevity Encoding. In: **NCSEC2005.** Proceeding of 9th Nation Computer Science and Engineering Conference; 2005 October 27-28; Bangkok; THAILAND; 2005. p. 349-360. [in Thai]

11. Liefke H, Suciu D. XMill : an Efficient Compressor for XML Data. In: **the 2000 ACM SIGMOD**. Proceeding of the 2000 ACM SIGMOD Internaional Conference on Management of Data; 2000 May; 2000. p. 153-164.
12. Mairiang K, Pluempitiwiriyawej C. XPACK : A Grammar-based XML Document Compression. In: **NCSEC2003**. Proceeding of 7th Nation Computer Science and Engineering Conference; 2003 October 28-30; Conburi; THAILAND; 2003. [in Thai]
13. Min J.K, Park M.J, Chung C.W. XPRESS : A Queriable Compression for XML Data. **the 2003 ACM SIGMOD**. Proceeding of the 2003 ACM SIGMOD Internaional Conference on Management of Data; 2003 June 9-12; 2003. p. 122-133.
14. Murray P, Rzepa H. **Chemical Markup Language (CML)**. Retrieved 30 July, 2004, from http://www.xml-cml.org
15. Runapongsa K, Patel J.M. Storing and Querying XML Data in Object-Relational DBMSs. In: **EDBT Workshops 2002**. Conference on Extending Database Technology. 2002 March 24-28; Prague, Czech Republic; 2002. p. 266-285.
16. Schmidt A.R, Waas F, Kersten M.L, Carey M.J, Manolescu I, Busse R. XMark: A Benchmark for XML Data Management. In: **VLDB2002**. Proceedings of the International Conference on Very Large Data Bases. 2002 August; Hong Kong, China; 2002. p. 974-985.
17. Tolani P.M, Haritsa J.R. XGRIND : A Query-friendly XML Compressor. In: **ICDE'02**. Proceeding on the 18th International Conference on Data Engineering. 2002. p. 225-234.
18. UN/CEFACT, OASIS. **ebXML: Enabling a Global Electronic Market**. Retrieved 25 July, 2004, from http://www.ebxml.org
19. W3C. **On SGML and HTML**. Retrieved 15 June, 2004, from http://www.w3.org/TR/REC-html40/intro/sgmltut.html
20. W3C. **Extensible Markup Language (XML) 1.0 (Third Edition)**. (2004 Feb 4) Retrieved 20 July, 2004, from http://www.w3.org/TR/2004/REC-xml-20040204
21. W3C. **HTML 4.01 Specification**. (1999 Dec 24). Retrieved 9 June, 2004, from http://www.w3.org/TR/html401

22. W3C. **XML Path Language (XPath) Version 1.0.** (1999 Nov 16). Retrieved 6 August, 2004, from http://www.w3.org/TR/1999/REC-xpath-19991116
23. W3C. **XML Schema Part 0 : Primer.** (2001 May 2). Retrieved 1 July, 2004, from http://www.w3.org/TR/xmlschema-0/
24. W3C. **XML Schema Part 1 : Structures.** (2001 May 2). Retrieved 1 July, 2004, from http://www.w3.org/TR/xmlschema-1/
25. W3C. **XML Schema Part 2 : Datatypes.** (2001 May 2). Retrieved 1 July, 2004, from http://www.w3.org/TR/xmlschema-2/
26. W3C. **Web Services.** Retrieved 8 September, 2004, from http://www.w3.org/2002/ws/
27. W3C. **XQuery: An XML Query Language.** Retrieved 14 January, 2005, from http://www.w3.org/XML/Query
28. Yahoo. **Yahoo! Search Web Services.** Retrieved 4 July, 2004, from http://developer.yahoo.net/

ภาคผนวก ก
ประวัติย่อและที่อยู่ของคณะผู้วิจัย

ประวัติของหัวหน้าโครงการ

ผศ. ดร. กานดา รุณนะพงศา ได้รับทุนรัฐบาลในการศึกษาในระดับปริญญาตรี โท เอก ณ ประเทศสหรัฐอเมริกาโดยที่จบการศึกษาปริญญาตรีด้วยเกียรตินิยมในสาขาวิศวกรรมไฟฟ้าและวิศวกรรมคอมพิวเตอร์ จากมหาวิทยาลัยคาร์เนกีเมลลอน ในปี พ.ศ. 2540 จบการศึกษาปริญญาโทและปริญญาเอกในสาขาวิทยาการคอมพิวเตอร์และวิศวกรรมคอมพิวเตอร์ จากมหาวิทยาลัยมิชิแกน แอนอาร์เบอร์ ในระหว่างที่เรียนปริญญาเอกในช่วงปิดเทอมภาคฤดูร้อน ได้ฝึกงานที่สถาบันวิจัยของไอบีเอ็มที่โตรอนโต ประเทศแคนาดา และที่นิวยอร์ก และแคลิฟอร์เนีย ประเทศสหรัฐอเมริกา

ผศ. ดร. กานดา รุณนะพงศา ปัจจุบันเป็นอาจารย์ประจำที่ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น และเป็นผู้ช่วยคณบดีฝ่ายวิชาการ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น มีความสนใจทางด้านเอกซ์เอ็มแอล เว็บเซอร์วิส และเว็บ 2.0 ได้รับเชิญให้เป็นวิทยากรในการอบรมเอกซ์เอ็มแอลและเว็บเซอร์วิสทั้งในหน่วยงานภาครัฐและบริษัทเอกชน และเป็นที่ปรึกษาของบริษัทที่นำเทคโนโลยีเอกซ์เอ็มแอลและเว็บเซอร์วิสมาใช้

ประวัติของผู้ร่วมวิจัย

นายประพันธ์ เลขาโสภณ เกิดเมื่อวันที่ 6 สิงหาคม พ.ศ. 2519 สำเร็จการศึกษาระดับปริญญาตรีวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมไฟฟ้า สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี เมื่อปี พ.ศ. 2542 เคยทำงานในตำแหน่งอาจารย์ประจำคณะวิชาไฟฟ้า สถาบันเทคโนโลยีราชมงคล วิทยาเขตสกลนคร ในระหว่างปี พ.ศ. 2542-2547 เคยทำงานในตำแหน่งหัวหน้าแผนกวิจัย สถาบันเทคโนโลยีราชมงคล วิทยาเขตสกลนคร ในระหว่างปี พ.ศ. 2546-2547 และศึกษาต่อในระดับปริญญาโทสาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น ในปี พ.ศ. 2547 และจบการศึกษาระดับปริญญาโท ในปี พ.ศ. 2549

ภาคผนวก ข

บทความของผู้วิจัย

บทความของผู้วิจัย

1. Lakhasophon P, Runapongsa K. “A Survey of XML Data Compression”. In: **NECSEC2005**. Proceeding of the 1st Northeastern Computer Science and Engineering Conference; 2005 March 31–April 1; Khon Kaen, THAILAND; 2005. p. 349-360. [in Thai]

2. Lakhasophon P, Runapongsa K. “XML Data Compression using Brevity Encoding”. In: **NCSEC2005**. Proceeding of the 9th Nation Computer Science and Engineering Conference; 2005 October 27-28; Bangkok, THAILAND; 2005. p. 349-360. [in Thai]

3. Lakhasophon P, Runapongsa K. “XBrevity: XML Data Compression using Brevity Encoding”. In: **ITC-CSCC 2006**. Proceeding of the 21st International Technical Conference on Circuits/Systems, Computers and Communications; 2006 July 10-13; Chiang Mai, Thailand; 2006.

การสำรวจการบีบอัดข้อมูลเอกซ์เอ็มแอล

A Survey of XML Data Compression

ประพันธ์ เลขาโสภณ
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
Email: jingxth@yahoo.com

กานดา รุณนะพงศา
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น
Email: krunapon@kku.ac.th

บทคัดย่อ

ข้อมูลบนเว็บที่แลกเปลี่ยนกันในปัจจุบันส่วนใหญ่นิยมแสดงในรูปแบบของเอกสาร XML (Extensible Markup Language) ซึ่งโดยทั่วไปเอกสาร XML มีขนาดใหญ่เมื่อเทียบกับขนาดของข้อมูลจริงในเอกสาร เนื่องจากมีการใช้แท็กที่ซ้ำกันในการอธิบายข้อมูลประเภทเดียวกันที่มีรายละเอียดต่างกัน ในบทความนี้ได้สำรวจถึงเทคนิคที่ใช้ในการบีบอัดเอกสาร XML ถึงแม้ว่าแต่ละเทคนิคที่ใช้ในการบีบอัดเอกสาร XML มีวิธีการที่แตกต่างกันซึ่งพบว่าทุกเทคนิคมีความจำเป็นต้องใช้ข้อมูลเกี่ยวข้องกับโครงสร้างของเอกสารทั้งสิ้น โดยการกำหนดกฎไวยากรณ์จากการวิเคราะห์โครงสร้างเอกสาร จากผลการสำรวจนั้นแต่ละเทคนิคมีจุดเด่นและจุดด้อยที่แตกต่างกันและทำให้ความสามารถในการใช้งานแตกต่างกันอีกด้วย

คำสำคัญ การบีบอัดข้อมูล, XML, กฎไวยากรณ์

1. บทนำ

ปัจจุบัน XML (Extensible Markup Language) [6] ได้เข้ามามีบทบาทและเป็นมาตรฐานในการแลกเปลี่ยนข้อมูล เนื่องจากการสร้างเอกสารอาจมีโครงสร้างที่หลากหลาย และเมื่อมีการเปลี่ยนแปลงข้อมูลเกิดขึ้นย่อมต้องมีการเปลี่ยนแปลงโครงสร้างตามไปด้วย ซึ่งจะมีความยุ่งยากในการแก้ไขสำหรับแอปพลิเคชันขนาดใหญ่ ด้วยเหตุนี้ XML จึงมีบทบาทมากขึ้น ซึ่ง XML มีความสามารถในการอธิบายความหมายของข้อมูลและมีความยืดหยุ่นในการใช้งาน การนำ XML มาใช้งานสามารถทำได้โดยการใช้แท็กเป็นตัวกำกับและการตั้งชื่อแท็กที่สื่อถึงความหมายของข้อมูล ทำให้เอกสารที่ถูกสร้างขึ้นเข้าใจได้ง่าย จึงเป็นส่วนสำคัญที่ทำให้การเข้าถึงข้อมูลได้ง่ายขึ้น แต่จากการที่มีการใช้แท็กเข้ามาช่วยในการสื่อถึงความหมายทำให้เกิดการบันทึกแท็ก ชนิดเดียวกันบ่อยครั้งในเอกสาร

```
<Book>
   <Author>
      <Name>        Pissamai
</Name>
   </Author>
   <Author>
      <Name>        Porntip
</Name>
   </Author>
</Book>
```

รูปที่ 1 ตัวอย่างเอกสาร XML

จากรูปที่ 1 จะเห็นได้ว่ามีการบันทึกข้อมูลประเภทเดียวกันหลายครั้ง ซึ่งในแต่ละครั้งจะมีรายละเอียดแตกต่างกัน ด้วยเหตุนี้ ขนาดของเอกสารจึงมีขนาดใหญ่เมื่อเทียบกับขนาดข้อมูลจริงภายในเอกสารนั้น ส่งผลให้สิ้นเปลืองเนื้อที่หากต้องการจัดเก็บเอกสารและสิ้นเปลืองเวลาในการรับส่ง หากต้องการแลกเปลี่ยนข้อมูลระหว่างองค์กร ผ่านระบบเครือข่าย

จากปัญหาในเรื่องขนาดของข้อมูลแนวทางที่สามารถนำมาใช้ในแก้ปัญหาได้คือการบีบอัดข้อมูล XML (XML data compression) เพื่อลดขนาดของเอกสารซึ่งเป็นการบีบอัดข้อมูล XML โดยเฉพาะทำให้สามารถเพิ่มประสิทธิภาพในการบีบอัดได้ดีกว่าการใช้วิธีการบีบอัดข้อมูลทั่วไป

ในการให้รายละเอียดของโครงสร้างเอกสารส่วนใหญ่จะใช้ DTD หรือ XML Schema ซึ่งรายละเอียดของโครงสร้างเอกสาร XML มักจะถูกอธิบายด้วย DTD (Document Type Definition) [1] ถ้าหากมีการส่งข้อมูล XML โดยมี DTD เป็นส่วนเพิ่มเติม XML จะทำให้การแลกเปลี่ยนข้อมูลง่ายขึ้นเนื่องจากมีการกำหนดว่า แท็กอะไรที่ควรมีและความสัมพันธ์ระหว่างข้อมูลประเภทต่างๆ ควรจะเป็นอย่างไร แต่ในการสร้าง DTD มีรายละเอียดและขั้นตอนที่ยุ่งยากพอสมควร ทำให้การสร้างเอกสารมีความยุ่งยาก บ่อยครั้งที่มีเอกสารซึ่งประกอบด้วยเนื้อหาที่มีชนิดต่างกัน และต้องการตรวจสอบชนิดของข้อมูล (Datatypes) [10] เหล่านี้ได้ แต่ DTD ไม่ได้ถูกออกแบบมาเพื่อตรวจสอบชนิดของข้อมูลเหล่า นี้หรือการตรวจสอบขอบเขตของค่า (Value) ดังนั้น XML Schema [8,9] จึงถูกสร้างขึ้นมาเพื่อการแก้ปัญหาเหล่านี้ XML Schema ต่างจาก DTD ตรงที่ DTD มีรูปประโยค (Syntax) เป็นของตัวเอง ส่วน XML Schema นั้นถูกเขียนขึ้นโดยใช้ไวยากรณ์ของภาษา XML นอกจากการจัดสร้างข้อมูลที่ DTD นำเสนอแล้ว XML Schema ยังช่วยกำหนดชนิดของข้อมูลโดยใช้เนมสเปซ (Namespace) [7] และกำหนดช่วงค่าของแอตทริบิวต์ (Attribute) และอิลิเมนต์ (Element)

ตัวอย่าง DTD ของเอกสาร XML จากรูปที่ 1

<!ELEMENT Book (Author+)>

<!ELEMENT Author (Name)>

<!ELEMENT Name (#PCDATA)>

จากตัวอย่างข้างต้นในลิสต์ลำดับ จะประกอบไปด้วยการประกาศให้ <Book> เป็นอิลิเมนต์ที่มี Author อย่างน้อยหนึ่ง Author โดยอิลิเมนต์ Author นั้นมีข้อมูลของอิลิเมนต์ Name และอิลิเมนต์ Name เก็บข้อมูลตัวอักษรซึ่งถูกประกาศเนื้อหาเป็น PCDATA (Parsed Character Data)

เทคนิคในการบีบอัดข้อมูล XML ที่มีอยู่ในปัจจุบันนั้น จะใช้ XML Schema เป็นกฎเกณฑ์และข้อกำหนดในการทำการบีบอัดข้อมูล XML และในบทความนี้ในหัวข้อที่ 2 จะได้นำเสนอกระบวนการบีบอัดแบบต่างๆ ที่มีอยู่ในปัจจุบันซึ่งได้แก่ XMILL, XPRESS และ XPACK โดยที่ทั้ง 3 วิธีดังกล่าวจะได้นำเสนอภาพรวมถึงข้อเด่นข้อด้อยและแนว ทางในการพัฒนาเทคนิคใหม่ๆในการบีบอัดข้อมูล XML จะนำเสนอในหัวข้อที่ 3 และบทสรุปในหัวข้อที่ 4

2. งานวิจัยที่เกี่ยวข้อง

งานวิจัยที่กล่าวถึงการบีบอัดข้อมูล XML ในปัจจุบันนั้นมีอยู่ 3 วิธีได้แก่ XMILL, XPRESS และ XPACK ซึ่งทั้ง 3 เทคนิคมุ่งเน้นในการลดขนาดของข้อมูลที่เป็น XML ให้เล็กลง โดยแต่ละวิธีจะใช้เทคนิคที่แตกต่างกันไป ดังที่จะได้เสนอต่อไปนี้

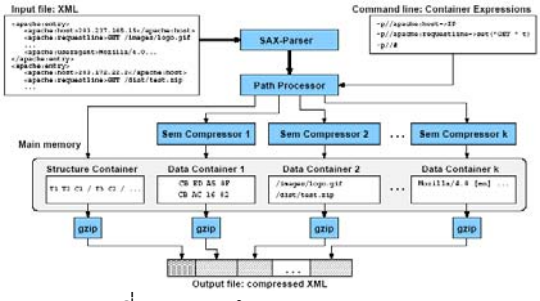
2.1 XMILL [2]

เป็นเทคนิคแรกที่ได้ทำการบีบอัดเอกสาร XML เพื่อให้มีขนาดเล็กลงโดยได้ใช้ zlib เป็นไลเบอรี่ในตัว XMILL ซึ่งเป็นตัวเดียวกันกับที่ใช้ใน GZip โดยใช้หลักการ 3 ข้อ คือ 1) แยกโครงสร้างออกจากข้อมูล 2) จัดกลุ่มให้กับข้อมูลต่างๆ 3) นำข้อมูลที่ได้เข้าสู่กระบวนการสร้างความสัมพันธ์ทางกลุ่มคำกับสัญลักษณ์ ซึ่งในการบีบอัดข้อมูลโดยวิธีนี้จะใช้ SAX Parser เป็นตัวจัดการกับข้อมูล XML เพื่อนำข้อมูลที่ได้เข้าสู่กระบวนการบีบอัดของ XMILL โดยที่หลักการดังกล่าวมีรายละเอียดดังนี้

2.1.1 **แยกโครงสร้างออกจากข้อมูล** เป็นการแยกโครงสร้างที่ประกอบไปด้วยแท็ก และแอตทริบิวต์ออกจากส่วนที่เป็นข้อมูล

2.1.2 **จัดกลุ่มให้กับข้อมูลต่าง ๆ** ที่ได้จากการแยกโครงสร้างออกจากข้อมูล โดยให้ข้อมูลที่เป็นแบบเดียวกันจัดอยู่ในกลุ่มเดียวกัน ส่วนข้อมูลจะเรียงกันไปต่อจากแท็กที่ได้จัดกลุ่มแล้ว

2.1.3 **ข้อมูลที่ได้จะเข้าสู่กระบวนการสร้างความสัมพันธ์ทางกลุ่มคำกับสัญลักษณ์** เพื่อให้สามารถอ้างถึงกลุ่มคำที่ได้จัดแยกแล้ว และสามารถบอกความหมายของกลุ่มคำนั้นๆได้



รูปที่ 2 ผังการทำงานของ XMILL

ตัวอย่างในการทำงานของ XMILL

เอกสาร XML ต้นฉบับ

```
<Book>
        <Author>
                <Name>          Pissamai
        </Name>
        </Author>
        <Author>
                <Name> Porntip </Name>
        </Author>
</Book>
```

ทำการแยกแท็กและข้อมูลออกจากกันได้

Book = T1, Author = T2, Name = T3

C1=Pissamai, C2=Porntip

โครงสร้างที่สมบูรณ์

Structure = T1 T2 T3 C1 / T2 T3 C2 / /

จากตัวอย่าง XMILL จะทำการจัดการโครงสร้างของเอกสารใหม่ก่อนการบีบอัดและบีบอัดเสร็จแล้ว จะเห็นได้ว่าโครงสร้างเล็กลงจากต้นฉบับ แต่เรายังไม่สามารถเข้าใจโครงสร้างที่ทำขึ้นมาใหม่ได้ ต้องอาศัยการขยายข้อมูลที่ถูกบีบอัดนี้ให้เป็นข้อมูลเดิมก่อนจึงจะสามารถเข้าใจข้อมูลภาย ในได้

2.2 XPRESS [4]

เป็นเทคนิคการบีบอัดข้อมูล XML ที่สนับสนุนการค้นหา (Query) ข้อมูลที่ถูกบีบอัดแล้วได้โดยใช้การเข้ารหัสที่เรียกว่า reverse arithmetic encoding ซึ่งวิธีการนี้ขึ้นอยู่กับชนิดของข้อมูลโดยจะมีหลักการทำงานคือ 1) วิเคราะห์ข้อมูล XML 2) เข้ารหัสข้อมูล XML 3) กระบวนการ ค้นหา ข้อมูลที่ถูกบีบอัดแล้ว (Query Processing)

2.2.1 **ตัววิเคราะห์ข้อมูล** XML (XML Analyzer) อัลกอริทึมของตัววิเคราะห์ข้อมูล XML จะสร้าง Hash table ที่เรียกว่า Elemhash ซึ่งจะ

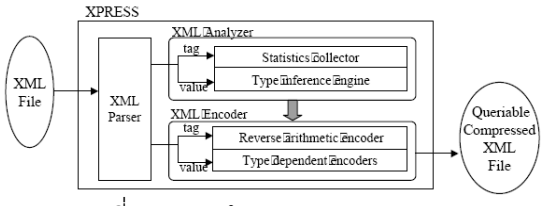
เก็บข้อมูลต่างๆอันได้แก่ ชนิดของข้อมูล, ความถี่ในการใช้งานเป็นต้น ค่าดังกล่าวจะถูกจัดเก็บให้อยู่ในช่วง Interval เรียกค่านี้ว่า Statistics collector ซึ่งจะคอยนับจำนวนของอิลิเมนต์ที่เกิดขึ้น อย่างไรก็ตามถ้าแท็กนั้นเป็นอิลิเมนต์ระดับที่สูงกว่า เช่น root element จะเป็นการยากที่จะกำหนดช่วงเส้นทางของข้อมูล จึงจำเป็นต้องใช้หลักทางคณิตศาสตร์ชั้นสูง (high precision floating arithmetic)

2.2.2 **เข้ารหัสข้อมูล XML (XML Encoder)** ในการเข้ารหัสข้อมูลจะทำการเปลี่ยนรูปให้เป็นไบนารี ซึ่งในแต่ละช่วงของการเข้ารหัสจะแทนด้วย u8 ใช้ 7 บิต จำนวน 1 ไบต์, u16 ใช้ 15 บิต จำนวน 2 ไบต์และ u32 ใช้ 31 บิต จำนวน 4 ไบต์ เมื่อเข้ารหัสแล้วจึงทำการบีบอัดข้อมูลต่อไป ดังตารางที่ 1

ตารางที่ 1 Data Encoders

Encoder	Description
u8	encoder for integers where max-min < 2^7
u16	encoder for integers where< 2^7+1 < max-min < 2^{15}
u32	encoder for integers where < 2^{15}+1 <max-min< 2^{31}
f32	encoder for floating values
dict8	dictionary encoder of textual data

2.2.3 **กระบวนการค้นหาข้อมูลที่ถูกบีบอัดแล้ว (Query Processing)** เมื่อข้อมูลถูกบีบอัดโดย XPRESS แล้วกระบวนการ ค้นหา (Query) จะทำให้ชื่อที่มีความยาวสั้นลงแล้ว ทำการแปลงรูปไปเป็นลำดับช่วง ซึ่งโดยทั่วไปแล้ว หนึ่งลำดับช่วงจะเท่ากับเส้นทางของชื่อที่ทำให้สั้นลงแล้ว หรือเป็นการลดความยาวของชื่อให้สั้นลงไปนั่นเอง เมื่อต้องการค้นหา (Query) ข้อมูลจะทำการค้นหาจาก Hash table ซึ่งจะเก็บช่วงของข้อมูลนั้นๆ ที่ได้ทำการบีบอัดแล้ว เพื่อให้ได้ช่วงของข้อมูลที่ตรงกับการค้นหา (Query) นั้น



รูปที่ 3 ผังการทำงานของ XPRESS

ตัวอย่างการทำงานของ XPRESS

เอกสาร XML ต้นฉบับ

<Book>

<Author>

<Name> Pissamai

</Name>

</Author>

<Author>

<Name> Porntip </Name>

</Author>

</Book>

ทำการแยกโครงสร้างแท็กออกจากข้อมูล

Book=T1, Author=T2, Name=T3, C1=Pissamai, C2=Porntip

โครงสร้างใหม่ที่ได้

Non-homomorphic

X0

T1

T2 T3 C1 / /

T2 T3 C2 / /

/

X1

encode(Pissamai)

encode(Porntip)

Homomorphic

T1

T2

T3 encode(Pissamai) /

/

T2

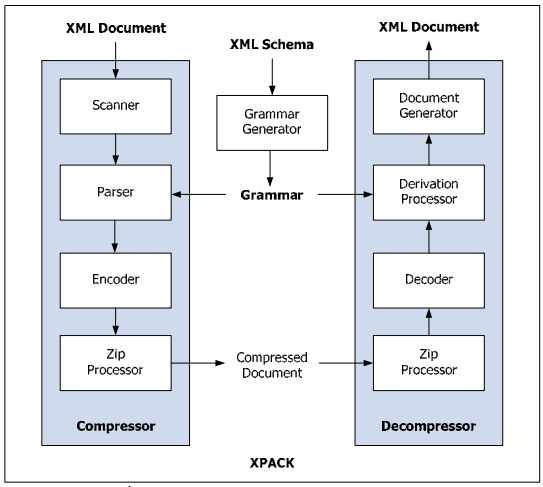
T3 encode(Porntip) /

/

จากตัวอย่างจะสามารถแบ่งการบีบอัดเป็น 2 แบบคือ แบบแยกการบีบอัดแท็กและข้อมูลออกจากกัน (Non-homomorphic) และแบบรวมส่วนที่เป็นแท็กและข้อมูลไว้ด้วยกัน (Homomorphic) แล้วทำการบีบอัด ซึ่งทั้ง 2 แบบให้ผลที่ไม่แตกต่างกันมากนัก จะเห็นได้ว่าขนาดของโครงสร้างใหม่ที่ได้เล็กลง และข้อมูลยังดูเข้าใจยากอยู่ และไม่มีตัวขยายข้อมูลที่บีบอัดแล้ว

2.3 XPACK [3]

เป็นการบีบอัดเอกสาร XML ด้วยวิธีการเชิงไวยากรณ์ ในหัวข้อนี้นำเสนอภาพรวมส่วนประกอบของเทคนิคที่เรียก ว่า XPACK ซึ่งใช้วิธีการเชิงไวยากรณ์ในการบีบอัดและการขยายเอกสาร XML ส่วนประกอบหลักของ XPACK คือ 1) Grammar Generator ทำหน้าที่ในการสร้างกฎไวยากรณ์ 2) Compressor ทำหน้าที่บีบอัดเอกสาร และ 3) Decompressor ทำหน้าที่ขยายเอกสารที่ผ่านการบีบอัด ซึ่งส่วนประกอบต่างๆมีรายละเอียดดังปรากฏในรูปที่ 4



รูปที่ 4 ผังการทำงานของ XPACK

2.3.1 **ตัวสร้างกฎไวยากรณ์** (Grammar Generator) มีหน้าที่ในการสร้างกฎไวยากรณ์เพื่อใช้ในการบีบอัดเอกสาร โดยการวิเคราะห์คำอธิบายโครงสร้างเอกสาร XML ภาษาที่ใช้ในการอธิบายโครงสร้างดังกล่าวได้แก่ XML Schema

2.3.2 **ตัวบีบอัดข้อมูล** (Compressor) ทำหน้าที่ในการบีบอัดเอกสารโดยประกอบไปด้วยส่วนประกอบย่อย 4 ส่วนด้วยกันคือ Scanner, Parser, Encoder และ Zip Processor ดังรูปที่ 4 โดยการทำงานภายใน Compressor เริ่มต้นที่ Scanner รับเอกสาร XML เข้ามาเพื่อเปลี่ยนให้เป็นของเครื่องหมาย (Token) ที่ใช้ในกฎไวยากรณ์ จากนั้น Parser ทำการวิเคราะห์ด้วยวิธีการเชิงไวยากรณ์ เพื่อให้ได้ผลลัพธ์ออกมาเป็น Parser Tree ต่อมาจึงนำ Parser Tree มาเข้ารหัสให้อยู่ในรูปของแถวลำดับด้วย Encoder ซึ่งผลลัพธ์ที่ได้จะนำไปทำการบีบอัดในขั้นตอนสุดท้ายที่ Zip Processor โดยทำการบีบอัดข้อมูลด้วย อัลกอริทึมการบีบอัดข้อมูลทั่วไป ผลลัพธ์จากการบีบอัดคือเอกสารที่ได้รับการบีบอัด (Compressed

Document) ซึ่งสามารถนำไปจัดเก็บ หรือแลกเปลี่ยนระหว่างองค์กรผ่านเครือข่ายต่อไป

2.3.3 ตัวขยายข้อมูลที่ถูกบีบอัด (Decompressor) เมื่อต้องการใช้เอกสารที่ได้รับการบีบอัดแล้ว จำเป็นต้องทำการขยายเอกสารก่อนโดยใช้ Decompressor ซึ่งประกอบไปด้วย 4 ส่วนย่อย คือ Unzip Processor, Decoder, Derivation Processor และ Document Generator ดังรูปที่ 4 Decompressor มีการทำงานโดยเริ่มที่ Unzip Processor ทำการขยายเอกสารที่ผ่านการบีบอัด ให้อยู่ในรูปของข้อมูลที่เข้ารหัส จากนั้น Decode ทำการถอดรหัสข้อมูลให้อยู่ในรูปของ Parse Tree เพื่อให้ Derivation Processor ใช้วิธีการเชิงไวยากรณ์ให้ได้มา ซึ่งสัญลักษณ์ที่จะใช้ในการสร้างเอกสาร สุดท้าย Document Generator ทำการสร้างเอกสาร XML ที่มีความหมายของข้อมูลภายในเหมือนกับเอกสารต้นฉบับ

การจัดโครงสร้างไวยากรณ์ของ XPACK เป็นดังนี้

<t> ::= t # / d (a, v)

โดยที่

<t> แทนรูปแบบการจัดเรียงข้อมูลในเอกสาร XML (*ไม่ใช่แท็ก!!*)

t แทนแท็กเปิด (Open Tag)

แทนข้อมูลภายในแท็ก

/ แทนแท็กปิด (Close Tag)

d แทนประเภทข้อมูลภายในแท็ก (Element Data Type)

a แทนชื่อแอตทริบิวส์ (Attribute Name)

v แทนประเภทข้อมูลของแอตทริบิวต์ (Attribute Data Type)

จากหลักการจัดโครงสร้างไวยากรณ์ข้างต้นนี้ เมื่อมีเอกสารที่จะทำการบีบอัดข้อมูล XPACK จะทำการจัดโครงสร้างใหม่ได้จากตัวอย่างต่อไปนี้

ตัวอย่างการทำงานของ XPACK

เอกสาร XML ต้นฉบับ

```
<Book>
        <Author>
                <Name>          Pissamai
        </Name>
        </Author>
        <Author>
                <Name> Porntip </Name>
        </Author>
</Book>
```

สร้างกฎไวยากรณ์

1 <Book> ::= Book <Author><Name> /

2 <Author> ::= Author <Name> /

3 <Name> ::= Name # / string

โครงสร้างใหม่ที่ได้

Book Author Name Pissamai / Author Name Porntip/ /

จากตัวอย่างจะเห็นได้ว่ามีการนำเอกสาร XML ต้นฉบับมาทำการสร้างกฎไวยากรณ์ขึ้นมาแล้วจากนั้นจึงทำการจัดรูปแบบโครงสร้าง ที่สอดคล้องกับกฎไวยากรณ์ที่ได้กำหนดขึ้นเมื่อเปรียบเทียบกับโครงสร้างเดิมจะเห็นได้ว่ามีขนาดที่เล็กลง แต่อย่างไรก็ตามยังต้องมีการขยายข้อมูลที่ถูกบีบอัดออกมาโดยอ้างอิงกับกฎไวยากรณ์นั้นๆ ให้เป็น

เอกสารต้นฉบับก่อน จึงจะสามารถเข้าใจข้อมูลภายในได้

3. ภาพรวมของการบีบอัดข้อมูล XML และแนวทางในการพัฒนาการบีบอัด

จากการทำงานของการบีบอัดข้อมูล XML ทั้ง 3 เทคนิคนั้นในแต่ละเทคนิคจะมีทั้งจุดเด่นและจุดด้อยอยู่ ซึ่งสามารถจำแนกส่วนที่สำคัญๆได้ดังนี้

XMILL ใช้เทคนิคในการแยกโครงสร้างของเอกสารโดยการแยกส่วนที่เป็นแท็กออกจากส่วนที่เป็นข้อมูล ซึ่งทำให้ง่ายในการจำแนกชนิดของข้อมูล จากนั้นในกระบวนการบีบอัดได้มีการใช้ zlib เป็นไลบารีซึ่งไลบารีดังกล่าวเป็นส่วนหนึ่งที่ใช้ในโปรแกรม GZip ที่มีความสามารถในการบีบอัดและสามารถขยายข้อมูลที่ถูกบีบอัดได้ แต่อย่างไรก็ตาม XMILL ยังมีข้อที่ควรปรับปรุงบางประการคือในการบีบอัดด้วยเทคนิคนี้ยังต้องอาศัยการใช้ Schema เป็นตัวสร้างไวยากรณ์ก่อนการบีบอัดข้อมูล อีกทั้งข้อมูลที่ถูกบีบอัดด้วยเทคนิคนี้ยังไม่สามารถค้นหา (Query) ข้อมูลที่ถูกบีบอัดอยู่ได้

XPRESS ใช้เทคนิคในการแยกโครงสร้างก่อนการบีบอัด โดยแยกส่วนที่เป็นแท็กและส่วนที่เป็นข้อมูลออกจากกันโดยใช้ hash table เป็นที่เก็บตัวบ่งชี้ข้อมูลต่างๆ จากนั้นจะใช้ XML Parser ร่วมในกระบวนการบีบอัด และสามารถค้นหาข้อมูลที่ถูกบีบอัดแล้วได้ ซึ่งเป็นส่วนที่สำคัญของเทคนิคนี้ แต่อย่างไรก็ตาม XPRESS ยังคงมีข้อปรับปรุงบางประการคือ การใช้การบีบอัดด้วยเทคนิคนี้ยังคงใช้ Schema เป็นตัวสร้างไวยากรณ์ก่อนการบีบอัดและข้อมูลที่ถูกบีบอัดด้วยเทคนิคนี้ยังไม่มีตัวขยายข้อมูลที่ถูกบีบอัดให้เป็นข้อมูลเดิมได้

XPACK การบีบอัดด้วยเทคนิคนี้จะมีตัวสร้างกฎไวยากรณ์ขึ้นมา ทำให้ง่ายในการจัดการกับเอกสารก่อนการบีบอัดข้อมูล อีกทั้งยังสามารถขยายเอกสาร XML ที่ได้ทำการบีบอัดแล้วให้สามารถเป็นเอกสาร XML ปกติได้ แต่ถึงอย่างไรก็ตาม XPACK ยังคงมีข้อที่ควรปรับปรุงคือจะต้องใช้สกีม่า (Schema) เพื่อสร้างกฎไวยากรณ์เป็นหลักซึ่งเป็นส่วนสำคัญของเทคนิคนี้ อีกทั้งยังไม่สามารถค้นหา (Query) ข้อมูลที่ถูกบีบอัดอยู่ได้

จากภาพรวมของเทคนิคการบีบอัดข้อมูล XML จะสังเกตเห็นได้ว่าทุกเทคนิคจะทำการแยกส่วนที่เป็น แท็กและส่วนที่เป็นข้อมูลออกจากกัน เพื่อง่ายต่อการจำแนกชนิดของข้อมูล ส่วนการบีบอัดข้อมูลจะมีความแตกต่างกันไป โดยที่วิธี XMILL จะใช้ zlib ที่เป็นไลบารีของ GZip และเมื่อพัฒนาแล้วจะบีบอัดข้อมูลได้ดีกว่า GZip โดยตรง ส่วน XPRESS จะใช้ XML Parser ซึ่งให้ผลในการบีบอัดดีกว่า XMILL [4] และวิธี XPACK จะใช้ Zip Processor ซึ่งไม่ได้บอกว่าเป็นแบบใดในการบีบอัดข้อมูลและบทความที่อ้างถึงได้กล่าวว่า สามารถบีบอัดข้อมูลได้ดีกว่า XMILL [3] จากการเปรียบเทียบข้อมูลดังกล่าวจะเห็นได้ว่า XMILL จะเป็นตัวเปรียบเทียบประสิทธิภาพในการบีบอัดข้อมูล ที่ทำให้ข้อมูลลดลง แต่ยังไม่มีบทความที่รายงานผลเปรียบเทียบประสิทธิภาพของการบีบอัดข้อมูลระหว่างเทคนิค XPRESS และเทคนิค XPACK

โดยสรุปแล้วทั้งสามเทคนิคนี้ จะให้ความสำคัญกับการลดขนาดของข้อมูลเป็นหลักและจะใช้ Schema เป็นตัวกำหนดไวยากรณ์ให้กับข้อมูลที่จะทำการบีบอัด ซึ่งในการนำไปใช้งานจริงนั้น การที่ผู้ใช้ปลายทางสามารถถอดรหัสที่ถูกบีบอัดออกมาให้เป็นข้อมูลเดิมได้นั้น จำ-เป็นจะต้องใช้ตัวขยายข้อมูลจาก วิธีการที่ถูกบีบอัดจากต้นทางนั้นๆ ทำให้ไม่สะดวกกับการใช้งานซึ่งอาจเป็นข้อจำกัดในการนำไปใช้งานกับกลุ่มคนทั่วไป และนอก จากนั้นเอกสารบางชนิดมีการเปลี่ยนแปลงอยู่ตลอดเวลา ทำให้โครงสร้างเอกสารต้องมีการ

เปลี่ยนแปลงและสกีม่า เปลี่ยนแปลงตามไปด้วย ดังนั้นควรจะมีเทคนิคในการบีบอัดข้อมูล XML โดยไม่ใช้ Schema เพื่อลดข้อจำกัดดังกล่าวได้

4. บทสรุป

การบีบอัดข้อมูล XML มีความสำคัญเพราะ XML เป็นภาษาที่มีข้อดีอยู่หลายประการ ซึ่งปัจจุบันได้มีการใช้กันอย่างแพร่หลาย แต่ข้อเสียของ XML ก็มีเช่นกัน โดยปกติแล้วเอกสารภาษา XML จะมีขนาดใหญ่กว่าเอกสารของเท็กซ์ไฟล์ (Text File) เนื่องจากเอกสารภาษา XML มักจะมีการใช้แท็กกำกับความหมายของข้อมูล การส่งข้อมูลโดยใช้ภาษา XML ทำให้เกิดความต้องการเน็ตเวิร์กแบนด์วิดธ์ (Network Bandwidth) ขนาดใหญ่เนื่องจากขนาดของไฟล์ วิธีหนึ่งที่จะช่วยลดเน็ตเวิร์กแบนด์วิดธ์ในการส่งข้อมูล คือการบีบอัดข้อมูลให้มีขนาดเล็กลง อีกทั้งยังช่วยให้ใช้พื้นที่ในการเก็บข้อมูลลดลงตามไปด้วย จากงานวิจัยที่ผ่านมาเกี่ยวกับการบีบอัดข้อมูล XML นั้นได้ทำการบีบอัดข้อมูลให้มีขนาดเล็กลง ซึ่งในแต่ละวิธีจะใช้ สกีม่า (Schema) เป็นตัวกำหนดไวยากรณ์เป็นของตัวเอง ซึ่งยากต่อการนำไปใช้งานทั่วๆไป โดยเทคนิคที่ควรมีคือการบีบอัดข้อมูลไม่จำเป็นต้องใช้สกีม่า(Schema) เพื่อให้ง่ายต่อการใช้งานทั่วๆไปซึ่งไม่ต้องมีตัวกำหนดไวยากรณ์แบบใดแบบหนึ่งโดยเฉพาะ และสามารถที่จะค้นหา (Query) ข้อมูลที่ถูกบีบอัดได้ เพื่อที่จะได้ไม่เสียเวลาในการขยายข้อมูล

เอกสารอ้างอิง

[1] D. Hunter, C. Cagle, D. Gibbons, N. Ozu, J. Pinnock and P. Spencer. “Beginning XML” , Wrox Press, 2002.

[2] H. Liefke and D. Suciu. “XMill: an Efficient Compressor for XML Data.” , In Proceeding of the 2000 ACM SIGMOD International Conference on Management of Data, pages 153-164, May 2000.

[3] Mairiang K, Pluempitiwiriyawej C. XPACK : A Grammar-based XML Document Compression. In: NCSEC2003. Proceeding of 7th Nation Computer Science and Engineering Conference; 2003 October 28-30; Conburi; THAILAND; 2003.

[4] J.-K. Min, M.-J. Park, and C.-W. Chung. “XPRESS: A Queriable Compression for XML Data.” In Proceeding of the 2003 ACM SIGMOD International Conference on Management of Data, pages 122-133, June 9-12, 2003.

[5] W3C. “Extensible Markup Language (XML) 1.0 (Third Edition)”, Feb 4, 2004, Available at http://www.w3.org/TR/2004/REC-xml-20040204/.

[6] W3C. “Extensible Markup Language (XML) 1.1”, Feb 4, 2004, Available at http://www.w3.org/TR/2004/REC-xml11-20040204/.

[7] W3C. “Namespaces in XML 1.1”, Feb 4, 2004, Available at http://www.w3.org/TR/xml-names11/.

[8] W3C. “XML Schema Part 0 : Primer ” , May 2, 2001, Available at http://www.w3.org/TR/xmlschema-0/.

[9] W3C. “XML Schema Part 1 : Structures”, May 2., 2001, Available at http://www.w3.org/TR/xmlschema-1/.

[10] W3C. “XML Schema Part 2 : Datatypes ”, May 2, 2001, Available at http://www.w3.org/TR/xmlschema-2/

การบีบอัดข้อมูล XML โดยใช้วิธีจัดเรียงข้อมูลแบบกระชับ
XML Data Compression using Brevity Encoding

ประพันธ์ เลขาโสภณ กานดา รุณนะพงศา
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น 40002
Email: superjing@gmail.com krunapon@kku.ac.th

บทคัดย่อ

ภาษา XMLได้วิวัฒนาการมาเป็นภาษามาตรฐานในการเสนอและแลกเปลี่ยนข้อมูลบนอินเทอร์เน็ตเนื่องจากภาษา XML เป็นภาษาที่ใช้ง่ายมีความยืดหยุ่น และไม่ขึ้นอยู่กับระบบปฏิบัติการของเครื่องคอมพิวเตอร์ แต่ว่าข้อมูลที่เป็นภาษา XML นั้นมักจะมีขนาดใหญ่และมีข้อมูลที่ซ้ำซ้อนอันเนื่องมาจากการใช้แท็กที่ซื่อซ้ำ ๆ ในการอธิบายข้อมูล เอกสาร XML จึงมีขนาดใหญ่ซึ่งทำให้ต้องการพื้นที่ในการจัดเก็บในปริมาณมากและใช้เวลานานในการส่งข้อมูล ดังนั้นงานวิจัยที่ต้องการจะบีบอัดข้อมูลในภาษา XML จึงมีความจำเป็นอย่างยิ่ง งานวิจัยที่ได้ทำมาแล้วนั้น ข้อมูลที่ถูกบีบอัดแล้วอาจจะอยู่ในรูปแบบของไบนารีหรือรูปแบบที่เป็นที่เข้าใจกับผู้พัฒนาเครื่องมือเท่านั้น

งานวิจัยนี้ได้นำเสนอวิธีบีบอัดข้อมูลแบบใหม่ชื่อว่า เอ็กซ์เบร็ววิตี้ (XBrevity*) ซึ่งเป็นวิธีการที่ใช้ในการบีบอัดข้อมูลหรือคลายการบีบอัดข้อมูลโดยที่ข้อมูลที่ถูกบีบอัดสามารถเป็นที่เข้าใจได้กับคนทั่วไป การบีบอัดข้อมูลโดยที่เอกสารที่ถูกบีบอัดอยู่ในรูปแบบของภาษา XML แบบย่อนั้นจะทำให้เอกสารนั้นยังคงข้อดีของภาษา XML ในขณะที่ขนาดของเอกสารเล็กลงข้อมูลที่ถูกบีบอัดอยู่ในรูปแบบของภาษา XML การบีบอัด

*http://gear.kku.ac.th/~krunapon/research/xbrevity

Abstract

XML becomes the standard language for data representation and exchange on the Internet because it is simple, flexible, and platform neutral. However, XML data is often large and verbose since it consists of many repeated tags which are used to self-describe data. The large size of data results in an excessive amount of space required for storing data on the disk space and that of time required for transmitting data over the network. Thus, it is necessary to find an effective compression technique for XML data. Previous compression techniques generate the compressed XML data that is in the binary format or in the proprietary format to only that XML compressor tool. Such format is often incomprehensible to others or difficult to automatically parse.

In this work, we propose XBrevity, an XML compressor which supports compressing and uncompressing XML data. XBrevity adopts a novel encoding method that has the compressed XML data in the XML format. Thus, the compressed XML data still preserve the advantage features of XML but the XML document has the smaller size.

Keywords: XML, Compression, Data exchange

1. บทนำ

ปัจจุบัน XML (Extensible Markup Language) [16] ได้เข้ามามีบทบาทและเป็นมาตรฐานในการแลกเปลี่ยนข้อมูล เนื่องจาก XML มีความสามารถ

ในการอธิบายความหมายของข้อมูลและมีความยืดหยุ่นในการใช้งาน การนำ XML มาใช้งานสามารถทำได้โดยการใช้แท็กเป็นตัวกำกับและการตั้งชื่อแท็กที่สื่อถึงความหมายของข้อมูล ทำให้เอกสารที่ถูกสร้างขึ้นเข้าใจได้ง่าย จึงเป็นส่วนสำคัญที่ทำให้การเข้าถึงข้อมูลได้ง่ายขึ้น แต่จากการที่มีการใช้แท็กเข้ามาช่วยในการสื่อถึงความหมายทำให้เกิดการบันทึกแท็ก ชนิดเดียวกันบ่อยครั้งในเอกสาร

```
<Book>
  <Author>
    <Name> Pissamai </Name>
  </Author>
  <Author>
    <Name> Porntip </Name>
  </Author>
</Book>
```

รูปที่ 1 ตัวอย่างเอกสาร XML

จากรูปที่ 1 จะเห็นได้ว่ามีการบันทึกข้อมูลประเภทเดียวกันหลายครั้ง ซึ่งในแต่ละครั้งจะมีรายละเอียดแตกต่างกัน ด้วยเหตุนี้ ขนาดของเอกสารจึงมีขนาดใหญ่เมื่อเทียบกับขนาดข้อมูลจริงภายในเอกสารนั้น ส่งผลให้สิ้นเปลืองเนื้อที่หากต้องการจัดเก็บเอกสารและสิ้นเปลืองเวลาในการรับส่ง หากต้องการแลกเปลี่ยนข้อมูลระหว่างองค์กรผ่านระบบเครือข่าย

โดยทั่วไปแล้วข้อมูล XML จะเก็บอยู่ในรูปแบบของไฟล์ ดังนั้นการบีบอัดข้อมูล XML ที่จะมีการนำไปใช้กันได้จริงจะเป็นการบีบอัดไฟล์ XML เนื่องจากข้อมูล XML จำนวนมากจะเก็บไว้ในไฟล์ ดังนั้นการบีบอัดไฟล์ XML จึงมีความจำเป็นอย่างยิ่ง

ปัจจุบัน XML ได้ถูกนำมาใช้งานในหลายสาขาวิชาชีพ ไม่ว่าจะเป็นทางธุรกิจซึ่งมีการนำ ebXML (Electronic Business XML) [15] และ BPEL (Business Process Execution Language) [2] ไปใช้ ด้านกราฟิกซึ่งมีการนำ SVG (Scalable Vector Graphics) [4] ไปใช้ หรือแม้กระทั่งด้านวิทยาศาสตร์ซึ่งมีการนำภาษา CML (Chemical Markup Language) [11] ไปใช้ ไม่ว่าจะเป็น BPEL, SVG, หรือ CML ต่างก็เป็นภาษา XML ประเภทหนึ่ง

นอกจากนี้แอปพลิเคชันอีกอันหนึ่งที่สำคัญของภาษา XML ซึ่งก็คือเว็บเซอร์วิส (Web Service) [21] เป็นซอฟต์แวร์ที่ได้นำเอามาใช้อย่างมากในปัจจุบันและคาดว่าจะมีการใช้อย่างแพร่หลายมากขึ้นในอนาคต ในปัจจุบันได้มีการนำเว็บเซอร์วิสเอามาใช้ในการให้บริการ ผ่านอินเทอร์เน็ตโดยบริษัทชั้นนำอาทิเช่น Yahoo Search Web Services [23], Google Web APIs [6], Amazon Web Services [1], และ eBay API [3] จุดเด่นของเทคโนโลยีเว็บเซอร์วิสคือการที่มันทำให้โปรแกรมที่พัฒนาโดยภาษาและใช้แพลตฟอร์มที่แตกต่างกันสามารถติดต่อและทำงานร่วมกันได้โดยใช้ภาษา XML เป็นภาษาสื่อกลางในการแลกเปลี่ยนข้อมูล ฉะนั้นจะเห็นได้ว่าข้อมูล XML จะมีปริมาณเพิ่มมากขึ้นและขนาดของข้อมูลของ XML ที่มักจะมีขนาดใหญ่จะส่งผลกระทบต่อประสิทธิภาพการทำงานของแอปพลิเคชันที่ใช้ XML เป็นภาษาในการบันทึกข้อมูล

จากปัญหาในเรื่องขนาดของข้อมูลแนวทางที่สามารถนำมาใช้ในแก้ปัญหาได้คือการบีบอัดข้อมูล XML (XML data compression) เพื่อลดขนาดของเอกสารซึ่งเป็นการบีบอัดข้อมูล XML โดยเฉพาะทำให้สามารถเพิ่มประสิทธิภาพในการบีบอัดได้ดีกว่าการใช้วิธีการบีบอัดข้อมูลทั่วไป

ในบทความนี้ในหัวข้อที่ 2 จะได้นำเสนอกระบวนการบีบอัดแบบต่างๆ ที่มีอยู่ในปัจจุบันซึ่งได้แก่ XMill, XGrind, XPRESS และ XPACK จากนั้นหัวข้อที่ 3 จะมีเนื้อหาในส่วนของแนวทางในการพัฒนาเทคนิคใหม่ๆในการบีบอัดข้อมูล ผลการทดลองจะนำเสนอในหัวข้อที่ 4 บทสรุปอยู่ในหัวข้อที่ 5 และในหัวข้อ 6 กล่าวถึงแนวทางในการพัฒนาในอนาคต

2. งานวิจัยที่เกี่ยวข้อง

โดยปกติแล้วในการบีบอัดไฟล์ข้อมูลทั่วไปนั้นจะนิยมใช้ gzip [5] เนื่องจากเป็นซอฟต์แวร์ที่นำไปใช้ได้โดยไม่ต้องเสียค่าใช้จ่าย อีกทั้งสามารถจะบีบอัดข้อมูลได้ดี และไม่จำเป็นจะต้องมีข้อมูลเกี่ยวกับโครงสร้างของเอกสาร แต่ว่าการบีบอัดไฟล์ XML โดยใช้ gzip นั้นมีข้อจำกัดที่ gzip ไม่สามารถตรวจสอบพบอิลิเมนต์ที่ซ้ำ ๆ ที่อาจจะไม่ได้อยู่ติดกัน

งานวิจัยที่กล่าวถึงการบีบอัดข้อมูล XML ในปัจจุบันนั้นมีอยู่หลายวิธีการได้แก่ XMill [8], XGrind [14], XPRESS [10] และ XPACK [9] ซึ่งมุ่งเน้นในการลดขนาดของข้อมูลที่เป็น XML ให้เล็กลง โดยแต่ละวิธีจะใช้เทคนิคที่แตกต่างกันไป

2.1 XMill

XMill เป็นวิธีการแรกที่ได้มีการนำเสนอวิธีการและเทคนิคในการบีบอัดเอกสาร XML โดยเริ่มจากการแยกส่วนแท็ก ซึ่งภายในประกอบไปด้วยอิลิเมนต์และแอตทริบิวต์ ออกจากข้อมูลที่เป็นตัวอักขระ จากนั้นจะทำการจัดความสัมพันธ์ของกลุ่มข้อมูลเป็น containers โดยจัดให้ข้อมูลที่เป็นแบบเดียวกันอยู่ในกลุ่มเดียวกัน ในขั้นตอนต่อมาจะนำ containers แต่ละตัวมาทำการบีบอัดเข้าไป จากกันแล้วทำการบีบอัดทั้งหมดโดยอาศัย gzip เพื่อให้ได้ข้อมูลที่ออกมาเป็นไฟล์เดียว

เนื่องจากการจัดกลุ่มข้อความต้องอาศัยความเข้าใจของความหมายของข้อมูล ซึ่งขึ้นอยู่กับชนิดของแอปพลิเคชัน ฉะนั้น XMill จึงมักต้องให้ผู้ใช้โปรแกรมพิจารณาความหมายของข้อมูล อีกทั้งผู้ใช้โปรแกรมไม่สามารถค้นหาข้อมูลที่ต้องการจากเอกสารที่ถูกบีบอัดแล้ว ถึงกระนั้นก็ตาม XMill ถือว่าเป็นบทความวิจัยแรก ๆ ที่ทำให้ผู้วิจัยทั้งหลายตระหนักถึงความสำคัญของปัญหาและวิธีการแก้ปัญหาในการบีบอัดข้อมูล XML ซึ่งได้มีผู้นำเทคนิคบางเทคนิคของ XMill ไปใช้ในการเก็บและค้นหาข้อมูล XML [12]

2.2 XGrind

XGrind มีข้อดีที่ XMill ยังไม่สามารถทำได้ซึ่งก็คือ ผู้ใช้ XGrind สามารถค้นหาข้อมูลได้จากเอกสารที่ถูกบีบอัด คุณสมบัตินี้เป็นผลมาจากการที่ข้อมูลที่ถูกบีบอัดแล้วยังคงมีโครงสร้างของเอกสารเดิม แต่อย่างไรก็ตามผู้ใช้โปรแกรม XGrind ไม่สามารถตอบคำถามบางประเภทโดยไม่ได้คลายการบีบอัดของข้อมูล ตัวอย่างของคำถามที่ต้องมีการคลายการบีบอัดข้อมูลคือ คำถามที่ถามช่วงของค่า (range query) หรือคำถามที่ถามหาข้อมูลค้นหาบางส่วนที่ตรงกับข้อมูลที่มีอยู่ (partial match)

นอกจากนี้ XGrind จะบีบอัดเฉพาะเอกสาร XML ที่มีเอกสาร DTD (Document Type Definition) ซึ่งเป็นเอกสารที่ระบุโครงสร้างของเอกสาร XML ซึ่งเอกสาร XML บางเอกสารอาจจะไม่มี DTD ก็ได้ ฉะนั้นผู้ใช้โปรแกรม XGrind จะต้องสร้าง DTD สำหรับเอกสาร XML ที่ยังไม่มี DTD

2.3 XPRESS

XPRESS มีข้อดีเช่นเดียวกับ XGrind ตรงที่สามารถจะค้นหาข้อมูลจากเอกสารที่ถูกบีบอัดแล้ว แต่ว่า XPRESS ไม่ได้ใช้ DTD

XPRESS ได้นำเสนอแนวคิดใหม่ที่ใช้ reverse arithmetic encoding เป็นวิธีการในการจัดเรียงข้อมูลเพื่อให้การหาคำตอบสำหรับ XPath expressions เป็นไปได้อย่างมีประสิทธิภาพ นอกจากนี้ XPRESS ได้พัฒนาการหาชนิดของข้อมูลโดยไม่ต้องอาศัยข้อมูลอินพุตจากผู้ใช้โปรแกรม แต่อย่างไรก็ตาม XPRESS มีข้อจำกัดที่ XPRESS จะไม่สามารถเข้าใจเอกสาร XML ที่มีการใช้ ID และ IDREF และไม่มีวิธีการขยายข้อมูลที่ถูกบีบอัดให้เป็นข้อมูล XML ปกติ

2.4 XPACK

XPACK เป็นการบีบอัดเอกสาร XML โดยใช้วิธีการเชิงไวยากรณ์ในการบีบอัดและการขยายเอกสาร XML ส่วนประกอบหลักของ XPACK คือ Grammar Generator ทำหน้าที่ในการสร้างกฎไวยากรณ์ ส่วนที่สองคือCompressor ทำหน้าที่บีบอัดเอกสาร และในส่วนสุดท้ายจะเป็น Decompressor ซึ่งทำหน้าที่ขยายเอกสารที่ผ่านการบีบอัดโดยอาศัยโครงสร้างเดิมของเอกสาร

ข้อจำกัดของ XPACK ก็คือโปรแกรมไม่สามารถบีบอัดจัดการกับเอกสาร XML ที่มีอิลิเมนต์เป็นแบบ mixed-content ซึ่งคืออิลิเมนต์ที่มีทั้งอิลิเมนต์และตัวอักขระอยู่ข้างใน และผู้ใช้โปรแกรมไม่สามารถค้นหาข้อมูลจากเอกสาร XML ที่ถูกบีบอัดแล้ว

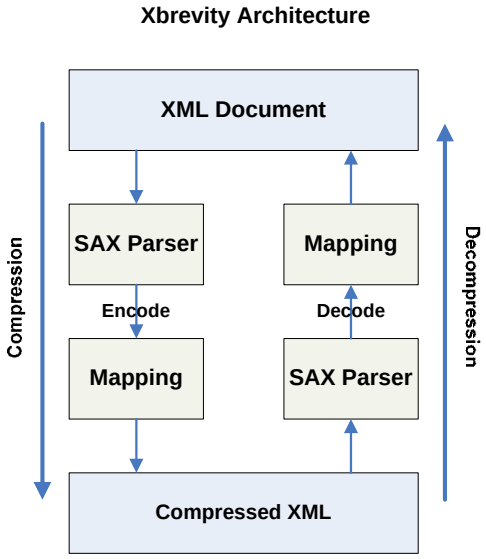
3. เทคนิคการบีบอัดแบบ XBrevity

ในเทคนิคการบีบอัดด้วย XBrevity นั้นจะประกอบไปด้วย ส่วนของการบีบอัดเอกสาร XML (Compressor) และการขยายเอกสาร (Decompressor) เพื่อทำให้เอกสารที่ถูกบีบอัดด้วย XBrevity กลับมาเป็นเอกสาร XML ต้นฉบับเดิมได้

ในโครงสร้างของ XBrevity จะใช้ SAX Parser เป็นตัวอ่านเอกสารทั้งในกระบวนการการบีบอัดเอกสารและขยายการบีบอัด ซึ่งโครงสร้างการทำงานของ XBrevity ได้แสดงไว้ในรูปที่ 2

3.1 โปรแกรมบีบอัดข้อมูล (Compressor)

มีหน้าที่ในการบีบอัดเอกสาร XML ซึ่งจะใช้ SAX Parser เป็นตัวอ่านวิเคราะห์เอกสารที่จะทำการบีบอัดเข้ามา ในระหว่างที่ทำการอ่านวิเคราะห์เอกสารอยู่นั้นจะทำการเข้ารหัสโดยการ Mapping เพื่อเปลี่ยนรูปแบบให้เป็นตัวอักษรแล้วเขียนค่าลงไปบนไฟล์ โดยในการสร้างไฟล์ที่มีข้อมูลที่ถูกบีบอัดแล้วนั้น จะยังคงอยู่ในรูปแบบของภาษา XML อยู่ และถูกต้องตามหลัก (Well-formed)



รูปที่ 2 แสดงโครงสร้างการทำงานของ XBrevity

โดยในเอกสารใหม่ที่ได้ประกอบไปด้วย <d>…</d> ซึ่งเป็นส่วนของข้อมูลที่ถูกบีบอัดแล้ว กับ <m …/> ซึ่งเป็นส่วนของการอธิบายความหมายของข้อมูลภายในเอกสาร (Metadata) ซึ่งเอกสารที่

ถูกบีบอัดนี้สามารถไปจัดเก็บ หรือแลกเปลี่ยนข้อมูลระหว่างเครือ ข่ายได้ และสามารถเข้าใจได้ง่ายโดยที่ไม่ต้องขยายข้อมูลกลับไปเป็นต้นฉบับเดิม เนื่องจากอยู่ในรูปแบบของเอกสาร XML แล้ว และมีตัวอธิบายข้อมูลอยู่ภายในให้ หรือจะทำการขยายเอกสารให้เป็นเอกสารเดิมได้ด้วยตัวขยายเอกสาร (Decompressor) ที่ออกแบบไว้

3.2 โปรแกรมคลายการบีบอัดข้อมูล (Decompressor)

เมื่อเอกสารที่ถูกบีบอัดด้วย XBrevity ต้องการขยายให้กลับมาเป็นเอกสาร XML ต้นฉบับสามารถทำได้โดยใช้ SAX Parser ที่ออกแบบมาเพื่อวิเคราะห์เอกสาร โดยทำการถอดรหัสเอกสารที่ถูกบีบอัดนั้นออกมา จากนั้นทำการ Mapping ค่าที่ได้ในแท็ก <m …/> แล้วนำค่าที่อ่านได้ไปแทนที่ตัวแปรเดิมที่อยู่ภายใน <d>…</d> เพื่อให้กลับออกมาเป็นเอกสารต้นฉบับ

3.3 ตัวอย่างการทำงานของ XBrevity

ในงานวิจัยนี้ได้นำเสนอการบีบอัดข้อมูล โดยไม่ใช้ Schema ซึ่งแสดงตามรูปที่ 3 ซึ่งเป็นตัวอย่างของเอกสารที่ประกอบไปด้วย อิลิเมนต์ และแอตทริบิวต์ต่างๆ จะเห็นได้ว่ามีการใช้งานของแท็กที่ซ้ำๆ กันแต่ข้อมูลภายในแตก-ต่างกัน ทำให้โครงสร้างดูซับซ้อนและเอกสารมีขนาดใหญ่

```
<?xml version="1.0"?>
  <bib>
    <article>
      <authors>
        <author aid="a1" eid="e1">
        <name>Pissamai<nickname>Aom
      </nickname></name>
    </author>
    <author aid="a2">
      <name>Porntip</name>
    </author>
      </authors>
      <year>2005</year>
    </article>
    <article>
      <authors>
    <author aid="a3">
      <name>Thongchai</name>
    </author>
      </authors>
    </article>
  </bib>
```

รูปที่ 3 เอกสาร XML ตัวอย่าง A

ในการบีบอัดข้อมูลนั้นจะมีข้อมูลเมต้าดาต้า (Meta data) เพื่อใช้อ้างอิงอิลิเมนต์และแอตทริบิวต์ ซึ่งทำให้โครงสร้างที่มีอยู่เดิมสั้นลง ทำให้ขนาดของเอกสารเล็กลงตามไปด้วย

```
<?xml version="1.0" encoding="UTF-8"?>
<c>
  <d>
    <e1e2e3 a4="a1" a5="e1" e6v="Pissamai"
 e7v="Aom"/>
    <xxe3 a4="a2" e6v="Porntip"/>
    <xe8 v="2005"/>
    <e1e2e3 a4="a3" e6v="Thongchai"/>
  </d>
  <m f="bib article authors author aid eid name
nickname year " b="0 1 2 3 a4 a5 6 7 8 "/>
</c>
```

รูปที่ 4 เอกสาร XML ตัวอย่าง A ที่ถูกบีบอัดแล้ว

โดยที่ **c** = การบีบอัด (Compressed)

d = ข้อมูล (Data)

m = ข้อมูลที่อธิบายข้อมูล (Metadata)

f = ชื่อเต็ม (Full name)

b = ชื่อย่อ (Brieviation name)

v = ค่าของข้อมูล (Value)

x = ใช้การเรียงซ้อนของอิลิเมนต์ที่พบก่อนหน้านี้

จากรูปที่ 4 ก่อนการบีบอัดข้อมูล (แท็ก **c**) จะทำการ mapping ข้อมูลต่างๆให้เป็นตัวแปร และทำให้ข้อมูลเมื่อบีบอัดแล้วจะเห็นได้ว่ามีขนาดเล็กลง ซึ่งข้อมูลจริงจะถูกบีบอัดอยู่ในแท็ก **d** แท็กที่ขึ้นต้นด้วย **e** เป็นแท็กที่เป็นชื่อย่อของอิลิเมนต์ ส่วนแท็กที่ขึ้นต้นด้วย **a** เป็นแท็กที่เป็นชื่อย่อของแอตทริบิวต์ **v** เป็นตัวอักษรที่ชี้ให้เห็นว่าจะมีการเก็บค่าที่อยู่ในอิลิเมนต์ ตัวอักษร x เป็นตัวอักษรที่บ่งบอกว่า อิลิเมนต์ปัจจุบันจะใช้การเรียง ลำดับของการซ้อนกันของอิลิเมนต์ที่เป็นบรรพบุรุษ (ancestor elements) ของอิลิเมนต์ที่เพิ่งพบก่อนหน้านี้ อย่างเช่น <e1e2e3 a5="a1" …/> ตามด้วย <xxe3 …/> นั้นบ่งบอกว่า <xxe3 …/> มีการเรียงลำดับอิลิเมนต์เป็น <e1e2e3 …/> ในขณะที่ <xe8 …/> นั้นมีการจัดเรียงลำดับอิลิเมนต์เป็น <e1e8 …/> ส่วน e0 ไม่จำเป็นต้องอยู่ในข้อมูลของการจัดเรียงเนื่องจากในแต่ละเอกสารมี root element มีเพียงตัวเดียว ดังนั้นในเอกสารที่ถูกบีบอัดจึงมี <e1e2e3 …/> แทนที่จะเป็น <e0e1e2e3 …/> จะเห็นได้ว่าโครงสร้างยังคงเป็นเอกสาร XML จากการบีบอัดด้วยวิธีการนี้จะเห็นได้ว่า เอกสารใหม่มีขนาดเล็กลงกว่าต้นฉบับ อีกทั้งยังเป็นเอกสารที่สามารถอ่านเข้าใจได้ง่าย และยังอยู่ในรูปแบบของเอกสาร XML ที่ถูกต้องตามหลักด้วย (Well-formed)

เอกสาร XML อีกตัวอย่างหนึ่งเป็นเอกสารที่เก็บข้อมูลของบทความวิจัยต่างๆ เอกสารตัวอย่างในลักษณะนี้แสดงในรูปที่ 5

```
<?xml version="1.0" encoding="UTF-8"?>
<bib>
  <inproceedings>
    <authors>
      <author>N. Zhang</author>
      <author>V. Kacholia</author>
      <author>M. T. Ozsu</author>
    </authors>
    <title>A Succinct Physical Storage Scheme for Efficient Evaluation for Path Queries in XML</title>
    <booktitle>Proceedings of the IEEE International Conference on Data Engineering</booktitle>
    <month>March</month>
    <year>2004</year>
    <pages>55-65</pages>
  </inproceedings>
  <inproceedings>
    <authors>
      <author>B. B. Yao</author>
      <author>M. T. Ozsu</author>
      <author>N. Khandelwal</author>
    </authors>
    <title>XBench Benchmark and Performance Testing of XML DBMSs</title>
    <booktitle>Proceedings of the IEEE International Conference on Data Engineering</booktitle>
    <month>March</month>
```

```
    <year>2004</year>
    <pages>621-632</pages>
  </inproceedings>
</bib>
```

รูปที่ 5 เอกสาร XML ตัวอย่าง B

หลังจากใช้โปรแกรม XBrevity ทำการบีบอัดข้อมูลแล้วจะได้เอกสารที่ถูกบีบอัดดังแสดงในรูปที่ 6

```
<c>
  <d>
    <e1e2e3 v="N. Zhang"/>
    <xxe3 v="V. Kacholia"/>
    <xxe3 v="M. T. Ozsu"/>
    <xe4 v="A Succinct Physical Storage Scheme for Efficient Evaluation for Path Queries in XML"/>
    <xe5 v="Proceedings of the IEEE International Conference on Data Engineering"/>
    <xe6 v="March"/>
    <xe7 v="2004"/>
    <xe8 v="55-65"/>
    <e1e2e3 v="B. B. Yao"/>
    <xxe3 v="M. T. Ozsu"/>
    <xxe3 v="N. Khandelwal"/>
    <xe4 v="XBench Benchmark and Performance Testing of XML DBMSs"/>
    <xe5 v="Proceedings of the IEEE International Conference on Data Engineering"/>
    <xe6 v="March"/>
    <xe7 v="2004"/>
    <xe8 v="621-632"/>
  </d>
  <m f="bib inproceedings authors author title booktitle month year pages " b="0 1 2 3 4 5 6 7 8 "/>
</c>
```

รูปที่ 6 เอกสาร XML ตัวอย่าง B ที่ถูกบีบอัดแล้ว

ในส่วนของเอกสารที่เป็น Mixed-Content Element นั้นสามารถใช้ XBrevity ในการบีบอัดได้เช่นกันดังนี้

```
<?xml version="1.0"?>
<Conversation>
  <word1>hello<word2>thai  people</word2>i love
    Thailand</word1>
</Conversation>
```

รูปที่ 7 เอกสาร XML ตัวอย่าง C

```
<c>
  <d>
    <e1 v="hello" e2v="thai people"/>
   <x v="i love Thailand"/>
  </d>
  <m f="Conversation word1 word2 "
    b="0 1 2 "/>
</c>
```

รูปที่ 8 เอกสาร XML ตัวอย่าง C ที่ถูกบีบอัดแล้ว

จากตัวอย่างที่อยู่ในรูปที่ 7 เป็นตัวอย่างของเอกสารที่เป็น Mixed-Content Element โดยในเอกสารประกอบไปด้วย <Conversation> ที่เป็น root element และมี <word1> และ <word2> เป็น child ซึ่ง <word2> จะมี “thai people” เป็น content และอยู่ระหว่าง content ของ <word1> ซึ่งมี “hello”

และ “I love Thailand” เป็น content เมื่อมี mixed-content เกิดขึ้น XBrevity จะพิจารณาบีบอัดเอกสารดังกล่าวถ้าพบแท็กปิดเกิดขึ้นในตัวอย่างนี้คือ </word2> ซึ่งจะได้ *<e1 v="hello"e2 v="thai people"/>* จะจัดให้อยู่ในอิลิเมนต์เดียวกัน ส่วน “I love Thailand” จะแยกเป็นอิลิเมนต์ใหม่โดยภายในจะแทนด้วย x เพื่อบ่งบอกว่าเป็น content ที่อยู่ในอิลิเมนต์เดียวกันกับอิลิเมนต์ก่อนหน้านี้คือ <word1> นั่นเอง ผลที่ได้เป็นไปตามรูปที่ 8 ซึ่งจะเห็นได้ว่าแม้จะเป็น mixed-content ก็สามารถบีบอัดด้วย XBrevity ได้เช่นกัน

4. การทดลอง

ในการทดลองการบีบอัดข้อมูลนี้ ได้นำเอกสาร XML ตัวอย่างโดยเป็นเอกสารที่ถูกสร้างขึ้นมาจากที่ต่างๆ จากแหล่งดาวน์โหลดตามข้อมูลที่อ้างอิงไว้ หรืออาจจะใช้ XMark [13] ในการสร้างไฟล์ขึ้นมาก็ได้ ซึ่งขนาดเอกสารที่นำมาทดสอบจะมีค่าต่างๆกัน และนำเอกสารที่ได้นี้ไปทดสอบกับ gzip และ XMill เพื่อเปรียบเทียบประสิทธิภาพการบีบอัดข้อมูล และพัฒนาระบบการบีบอัด โดยการทดลองนี้ได้ทำบน Intel Pentium4 1.80GHz หน่วยความจำ 256 MB ระบบปฏิบัติการ Windows XP Professional with Service Pack 2 และใช้ภาษาจาวา (JDK 5.0 Update 4) ในการสร้างและพัฒนาระบบการบีบอัด แต่ในการทดลองนี้ไม่ได้ทดสอบร่วมกับ XGrind, XPRESS และ XPACK เนื่องจาก XGrind เป็นโปรแกรมที่ใช้ทดสอบบนระบบปฏิบัติการ Linux หรือบนระบบ Unix เท่านั้น ส่วน XPRESS และ XPACK ไม่มีโปรแกรมให้ดาวน์โหลดเพื่อทดสอบ จึงไม่สามารถนำมาใช้ทดสอบเพื่อเปรียบเทียบประสิทธิภาพกับ Compressor ตัวอื่นๆ ได้ ดังนั้นในการทดสอบนี้ จึงทำการเปรียบการบีบอัดด้วย gzip, XMill และ XBrevity

4.1 การวัดประสิทธิภาพของการบีบอัด

ประสิทธิภาพของการบีบอัดวัดจากอัตราส่วนในการบีบอัด ซึ่งหาได้จากสมการที่ *(1)*

$$\text{อัตราส่วน} = 1 - \frac{\text{ขนาดของเอกสารที่บีบอัดแล้ว}}{\text{ขนาดของเอกสารต้นฉบับ}} \quad (1)$$

ตารางที่ 1 คุณลักษณะของเอกสารที่ใช้ในการทดสอบ

XML Files	Size (bytes)	Depth	Elem	Attr
f1.xml[1]	669,309	7	46	0
f2.xml[2]	4,222,646	7	199	0
f3.xml[3]	1,166,916	6	15	1
f4.xml[4]	251,865	6	17	0
f5.xml[5]	3,021,921	3	12	0

จากข้อมูลในตารางที่ 1 f1.xml, f2.xml, …, f5.xml เป็นชื่ออ้างอิงถึงเอกสาร XML ที่ใช้ในการทดสอบ ซึ่งเอกสารเหล่านี้ผู้อ่านบทความสามารถดาวน์โหลดได้ ตามเว็บไซต์ที่ได้ระบุไว้ นอกจากนี้ในตารางที่ 1 ยังมีข้อมูลซึ่งบ่งบอกชนิดของข้อมูลดังนี้ *Size* ขนาดของเอกสาร XML เป็นไบต์ (bytes) *Depth* ความลึกหรือจำนวนชั้นสูงสุดที่เกิดขึ้นในเอกสาร *Element* คือจำนวนอิลิเมนต์ในเอกสารที่เป็นชื่อเดียวกัน *Attributes* คือจำนวนแอตทริบิวต์ในเอกสารที่เป็นชื่อเดียวกัน เมื่อทำการบีบอัดในแต่ละวิธีแล้ว ผลที่ได้ปรากฏอยู่ในตารางที่ 2

[1] http://www.ibiblio.org/xml/examples/1998statistics.xml

[2] http://www.w3.org/XML/Binary/2005/03/test-data/Over100K/factbook.xml

[3] http://www.eda.org/pub/ibis/xml/sample1/format.xml

[4] ไฟล์ใน /examples/shakespeare.xml ของ XMill ดาวน์โหลดได้จาก http://www.research.att.com/sw/tools/xmill/

[5] http://gnosis.cx/download/weblog.xml

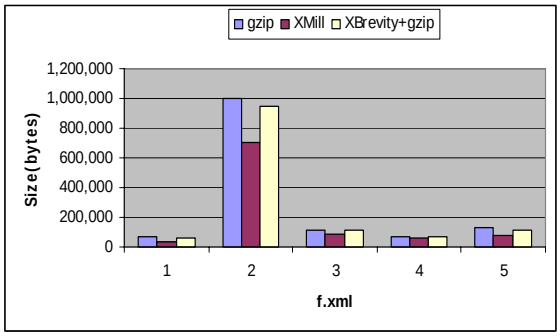
ตารางที่ 2 การเปรียบเทียบขนาดของเอกสารที่บีบอัดแล้ว

XML Files	Compressed File Size (bytes)		
	gzip	XMill	XBrevity
f1.xml	66,752	35,528	503,806
f2.xml	1,002,187	700,137	3,095,133
f3.xml	114,361	83,618	862,410
f4.xml	68,000	64,184	225,339
f5.xml	127,509	74,439	1,868,629

จากตารางที่ 2 จะเห็นได้ว่าขนาดที่บีบอัดด้วย gzip และ XMill มีขนาดเล็กมากเมื่อเทียบกับ XBrevity เนื่องจากว่า XBrevity ทำการบีบอัดเฉพาะส่วนของแท็กเท่านั้น ซึ่งผู้วิจัยกำลังพัฒนาเพิ่มเติมเพื่อให้บีบอัดส่วนที่เป็นข้อความได้ด้วยแต่อย่างไรก็ตามเมื่อเรานำเอา XBrevity ที่ได้ทำการจัดเรียงข้อมูลแบบกระชับแล้ว มาใช้ร่วมกับ gzip จะได้ผลการทดลองดังตารางที่ 3 และกราฟแสดงขนาดที่ได้จากการบีบอัดจากรูปที่ 7

ตารางที่ 3 การเปรียบเทียบขนาดเมื่อ XBrevity+gzip

XML Files	Compressed File Size (bytes)		
	gzip	XMill	XBrevity+gzip
f1.xml	66,752	35,528	63,452
f2.xml	1,002,187	700,137	950,807
f3.xml	114,361	83,618	108,338
f4.xml	68,000	64,184	66,727
f5.xml	127,509	74,439	115,990



รูปที่ 9 กราฟแสดงขนาดการบีบอัดเมื่อ XBrevity+gzip

จากการนำ XBrevity มารวมกับ gzip ทำให้ประสิทธิภาพในการบีบอัดเอกสาร ทำได้ดีขึ้นกว่าการใช้ XBrevity เพียงอย่างเดียว เนื่องจากว่าเมื่อเอกสารที่ผ่านการจัดเรียงแบบกระชับด้วย XBrevity แล้วนั้นขนาดของเอกสารจะเล็กกว่าต้นฉบับเดิม เมื่อรวมกับ gzip จึงทำให้ขนาดที่บีบอัดได้เล็กกว่า gzip ซึ่งผลที่ได้แสดงไว้ในตารางที่ 3 และรูปที่ 9 แล้ว แต่เมื่อเทียบกับ XMill แล้วยังมีขนาดใหญ่กว่าเล็กน้อย

ตารางที่ 4 การเปรียบเทียบอัตราส่วนในการบีบอัด

XML Files	Compression Ratio (%)		
	gzip	XMill	XBrevity+gzip
f1.xml	90.0	94.7	90.5
f2.xml	76.3	83.4	77.5
f3.xml	90.2	92.8	90.7
f4.xml	73.0	74.5	73.5
f5.xml	95.8	97.5	96.2
Average	85.0	88.6	85.7

จากตารางที่ 4 จะเห็นได้ว่าเมื่อขนาดของ XBrevity ที่รวมกับ gzip มีนาดเล็กลงทำให้อัตราส่วนในการบีบอัดสูงขึ้นตามไปด้วย ดังนั้น

เวลาที่ใช้ในการบีบอัดก็เร็วขึ้นด้วย ซึ่งผลที่ได้แสดงในตารางที่ 5

ตารางที่ 5 การเปรียบเทียบเวลาในการบีบอัด

XML Files	**Compression Time (seconds)**		
	gzip	**XMill**	**XBrevity+gzip**
f1.xml	1.5	1	1.5
f2.xml	3	2.5	2.9
f3.xml	1	0.8	1
f4.xml	0.6	0.3	0.5
f5.xml	2.5	2	2.4

และเมื่อนำไฟล์ที่ถูกบีบอัดในตารางที่ 5 มาทำการขยายออกเป็นเอกสารต้นฉบับจะได้เวลาที่ใช้ในการขยาย ผลที่ได้คล้ายคลึงไปในทางเดียวกันกับการบีบอัดเอกสาร แต่การขยายจะเร็วกว่าเล็กน้อย และในส่วน XBrevity ที่ใช้การบีบอัดรวมกับ gzip มีคุณสมบัติเดียวกันกับ gzip แต่ขนาดของไฟล์เล็กกว่า การขยายไฟล์จึงทำได้เร็วกว่า gzip

จากผลการทดลองที่ได้นั้น จะเห็นได้ถ้าเปรียบเทียบอัตราส่วนของการบีบอัดแล้ว XMill และ gzip มีค่าค่อนข้างสูง และระยะเวลาในการบีบอัดก็ใช้เวลาที่น้อย เมื่อเทียบกับ XBrevity เพราะ XBrevity บีบอัดเฉพาะส่วนของแท็กเท่านั้น แต่ถ้านำ XBrevity ที่ได้ทำการบีบอัดแล้วมาใช้ร่วมกับ gzip จะเห็นว่ามีขนาดของข้อมูลที่ลดลงมาค่อนข้างมาก รวมถึงเวลาที่ใช้ในการบีบอัดก็เร็วกว่าเดิม ซึ่งผลที่ได้ใกล้เคียงกับการใช้ gzip และ XMill และจะเห็นได้ว่าเวลาที่ใช้ในการขยายไฟล์ออกเป็นเอกสารต้นฉบับจะน้อยกว่าการบีบอัด ไม่ว่าจะใช้วิธีการใด แต่เนื่องจาก XBrevity ต้องเขียนข้อมูลที่อ่านได้ลงไปบนไฟล์ที่เป็น .xml ด้วยจึงทำให้ใช้เวลานานอีก ทั้งข้อมูลที่เขียนลงไปนั้นเป็นรูปย่อของเอกสาร ซึ่งยังคงข้อดีของภาษา XML ไว้ อีกทั้งยังเป็นผลดีในแง่ของการอ่านเพื่อทำความเข้าใจเอกสารที่บีบอัดแล้วได้ง่าย เพราะว่าในการนำไปใช้งานจริงนั้น การที่ผู้ใช้ปลายทางสามารถถอดรหัสที่ถูกบีบอัดออกมาให้เป็นข้อมูลเดิมได้นั้น จำเป็นจะต้องใช้ตัวขยายข้อมูลจากข้อมูลที่ถูกบีบอัดแล้ว ซึ่งอาจทำให้ไม่สะดวกกับการใช้งาน และเป็นข้อจำกัดในการนำไปใช้งานกับกลุ่มคนทั่วไป

5. สรุป

การบีบอัดข้อมูล XML มีความสำคัญเพราะ XML เป็นภาษาที่มีข้อดีอยู่หลายประการ ซึ่งปัจจุบันได้มีการใช้กันอย่างแพร่หลาย แต่ข้อจำกัดของ XML ก็มีเช่นกัน โดยปกติแล้วเอกสารภาษา XML จะมีขนาดใหญ่กว่าเอกสารของเท็กซ์ไฟล์ (Text File) เนื่องจากเอกสารภาษา XML มักจะมีการใช้แท็กกำกับความหมายของข้อมูล การส่งข้อมูลโดยใช้ภาษา XML ทำให้เกิดความต้องการเน็ตเวิร์กแบนด์วิดธ์ (Network Bandwidth) ขนาดใหญ่เนื่องจากขนาดของไฟล์ วิธีหนึ่งที่จะช่วยลดเน็ตเวิร์กแบนด์วิดธ์ในการส่งข้อมูล คือการบีบอัดข้อมูลให้มีขนาดเล็กลง อีกทั้งยังช่วยให้ใช้พื้นที่ในการเก็บข้อมูลลดลงตามไปด้วย จากงานวิจัยที่ผ่านมาเกี่ยวกับการบีบอัดข้อมูล XML นั้นได้ทำการบีบอัดข้อมูลให้มีขนาดเล็กลง ซึ่งในบางวิธีการจะต้องใช้สกีม่า (Schema) เป็นตัวกำหนดไวยากรณ์เป็นของตัวเอง และบางวิธีการต้องขยายเอกสารให้เป็นต้นฉบับก่อนจึงจะสามารถใช้งานได้ ซึ่งยากต่อการนำไปใช้งานทั่วๆไป

เทคนิคที่ได้นำเสนอคือการบีบอัดข้อมูล XML โดยใช้วิธีจัดเรียงข้อมูลแบบกระชับ (XBrevity) ซึ่งเป็นเทคนิคการจัดเรียงแบบกระชับ เพื่อให้เอกสารมีขนาดเล็กลงจากเดิม และยังสามารถเข้าใจความหมายเอกสารที่ถูกบีบอัดได้ง่าย โดยไม่ต้องมีการขยายเอกสารให้เป็นต้นฉบับเดิมเพราะอยู่ใน

รูปแบบของภาษา XML อยู่แล้ว ทำให้ยังคงข้อดีที่มีอยู่ในภาษา XML อีกทั้งยังง่ายต่อการใช้งานทั่วๆไป ซึ่งไม่ต้องมีตัวกำหนดไวยากรณ์แบบใดแบบหนึ่งโดยเฉพาะ และสามารถที่จะค้นหา (Query) ข้อมูลที่ถูกบีบอัดได้ เพื่อที่จะได้ไม่เสียเวลาในการขยายข้อมูลอีก

6. แนวทางการพัฒนาในอนาคต

นอกจากงานวิจัย XBrevity ที่ได้นำเสนอไป ซึ่งเน้นถึงความสำคัญในการบีบอัดข้อมูลในส่วนของโครงสร้างแท็ก แต่ตัวอักขระซึ่งเป็นข้อมูลภายในแท็กยังไม่ได้รับการบีบอัด ดังนั้นจึงยังสามารถพัฒนาประสิทธิภาพของเทคนิคการบีบอัดได้มากขึ้นไปอีกโดยการบีบอัดอักขระภายในแท็ก ซึ่งจะมีผลให้ขนาดของเอกสารลดลงได้อีก นอกจากนี้แนวทางและวิธีการนำเอาเอกสารที่ถูกบีบอัดแล้วซึ่งอยู่ในรูปแบบ XML ไปใช้เพื่อการค้นหาข้อมูลก็เป็นงานวิจัยที่น่าสนใจ รวมถึงการพัฒนาให้โปรแกรมสามารถบีบอัดแฟ้มที่มีเอกสาร XML ได้ทั้งแฟ้มแทนที่จะบีบอัดทีละไฟล์ ผลการทดลองจากการเปรียบเทียบเวลารวมในการบีบอัดข้อมูลระหว่างเวลาในการส่งเอกสาร XML ที่ถูกบีบอัดแล้วกับเวลาที่ใช้ในการส่งเอกสาร XML เดิมเป็นอีกผลการทดลองต่อเนื่องที่ควรจะทำ

เอกสารอ้างอิง

[1] Amazon. “Amazon Web Services”, Available at http://www.amazon.com/gp/aws/landing.html

[2] BEA Systems, IBM, Microsoft, SAP AG and Siebel Systems. “Business Process Execution Language for Web Services”, Available at http://www-128.ibm.com/developerworks/library/specification/ws-bpel/

[3] eBay. “eBay API”, Available at http://developer.ebay.com/common/api

[4] J. Ferraiolo, J. Fujisawa, D. Jackson. “Scalable Vector Graphics (SVG) 1.1 Specification”, Available at http://www.w3.org/TR/SVG

[5] J.L. Gailly and M. Adler, “gzip : The compressor data”, Available at http://www.gzip.org/

[6] Google. Google Web APIs (beta), Available at http://www.google.com/apis/

[7] D. Hunter, C. Cagle, D. Gibbons, N. Ozu, J. Pinnock and P. Spencer. “Beginning XML” , *Wrox Press*, 2002.

[8] H. Liefke and D. Suciu. “XMill: an Efficient Compressor for XML Data.” , In *Proceeding of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 153-164, May 2000.

[9] K. Mairiang, and C. Pluempitiwiriyawej. “XPACK: A Grammar-based XML Document Compression”, In *Proceeding of NCSEC2003 the7th National Computer Science and Engineering Conference*, October 28-30, 2003.

[10] J.-K. Min, M.-J. Park, and C.-W. Chung. “XPRESS: A Queriable Compression for XML Data.” In *Proceeding of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 122-133, June 9-12, 2003.

[11] P. Murray-Rust and H. Rzepa. “Chemical Markup Language (CML)”, Available at http://www.xml-cml.org

[12] K. Runapongsa and J. M. Patel, “Storing and Querying XML Data in Object-Relational DBMSs”, EDBT Workshops 2002: 266-285

[13] A. R. Schmidt, F. Waas, M. L. Kersten, M. J. Carey, I. Manolescu, and R. Busse. “XMark: A Benchmark for XML Data Management”, Proceedings of the International Conference on Very Large Data Bases (VLDB), pp. 974-985, Hong Kong, China, August 2002, Available at http://monetdb.cwi.nl/xml/index.html

[14] P. M. Tolani and J. R. Haritsa. “XGRIND: A Query-friendly XML Compressor.” In *Proceedings of 18th International Conference on Databases Engineering*, February 2002.

[15] UN/CEFACT and OASIS. “ebXML: Enabling a Global Electronic Market”, Available at http://www.ebxml.org/

[16] W3C. “Extensible Markup Language (XML) 1.0 (Third Edition)”, Feb 4, 2004, Available at http://www.w3.org/TR/2004/REC-xml-20040204/.

[17] W3C. “Namespaces in XML 1.1”, Feb 4, 2004, Available at http://www.w3.org/TR/xml-names11/.

[18] W3C. “XML Schema Part 0 : Primer ” , May 2, 2001, Available at http://www.w3.org/TR/xmlschema-0/.

[19] W3C. “XML Schema Part 1 : Structures”, May 2., 2001, Available at http://www.w3.org/TR/xmlschema-1/.

[20] W3C. “XML Schema Part 2 : Datatypes ”, May 2, 2001, Available at http://www.w3.org/TR/xmlschema-2/

[21] W3C. “Web Services”, Available at http://www.w3.org/2002/ws/

[22] W3C. “XQuery : An XML Query Language”, Available at http://www.w3.org/XML/Query

[23] Yahoo. “Yahoo! Search Web Services”, Available at http://developer.yahoo.net/

XBrevity: XML Data Compression using Brevity Encoding

Praphan Lakhasophon and Kanda Runapongsa

Department of Computer Engineering, Faculty of Engineering, Khon Kaen University
Khon Kaen 40002, Thailand.
{superjing@gmail.com , krunapon@kku.ac.th}

ABSTRACT

XML becomes the standard language for data representation and exchange on the Internet because it is simple, flexible, and platform neutral. However, XML data is often large and verbose since it consists of many repeated tags which are used to self-describe data. The large size of data results in an excessive amount of space required for storing data on the disk space and that of time required for transmitting data over the network. Thus, it is necessary to find an effective compression technique for XML data. In this work, we propose XBREVITY, an XML compressor which supports compressing and uncompressing XML data. XBREVITY adopts a novel encoding method that has the compressed XML data in the XML format. Thus, the compressed XML data still preserves the advantage features of XML but the XML document has the smaller size.

Keywords: XML, Compression, Brevity

1. INTRODUCTION

Currently, XML [10] becomes the standard language for data exchange because it is self-described and flexible. XML uses tag names to describe data so that a created document is easy to understand. When XML tags are used to describe data, an XML document usually has many repeated tags. Therefore, the large size of data results in an excessive amount of space required for storing data on the disk space and that of time required for transmitting data over the network.

The remainder of the paper is organized as follows: In section 2 we present related work. Section 3 presents features of XBrevity. Section 4 presents compression techniques in XBrevity. The experiments of these techniques are studied in section 5. Finally, conclusions are given in section 6.

2. RELATED WORK

File compression often uses gzip [2] because it is freeware and does not need the information of document structure. The disadvantage of using gzip in XML is that it cannot check continued repeated elements since it is not designed for XML compression.

Important or recent related work of XML compression includes XMill, XGrind, XPRESS and XPACK [3]. These methods reduce the size of an XML document using different techniques as described in the following subsections.

2.1 XMILL

XMill [4] achieves better compression rate compared to gzip (by a factor of 2, for data-like XML documents) without sacrificing speed. This owes to

the fact that it separates structure from content. This makes it a clear winner for applications like data archiving since these applications require lesser disk space. At the same time, it reduces network bandwidth. XMill is moderately faster than gzip in XML publishing. However, relative advantage of XMill depends on the application it is used. However, compressed output of XMill is not queryable except the document has to be decompressed.

If the size of the input document is less than 20KB, XMill will not exhibit any significant advantage over gzip. Here the compression is targeted for applications, such as data exchanging, data archiving, but not for deriving a meaningful view of the input document as is the case of compressing images or video sequences. To apply specialized compressors to containers, human intervention is required to specify the required container. Path processor is configured by user commands to map values. XMill precludes incremental processing of compressed documents; it actually hinders compressors other than gzip, and requires user assistance to achieve the best compression [7].

2.2 XGRIND

XGrind [9] has a considerable improvement in query response time and disk bandwidth. Its effective compression is processed through increased information density so that memory hit buffer ratio increases. In addition, XGrind compresses at the granularity of element/attribute value using context-free compression scheme. Furthermore, for range and partial match queries, on the fly decompression is required for only those elements that feature in the query predicates.

However, XGrind does not support several operations such as non-equality selections. In addition, it cannot perform any join, aggregation, and nested queries or construct operations. XGrind depend on one-time statistics of an XML document before the compression, but the statistics can change due to updates made to the compressed XML document. Lastly, XGrind uses a fixed root-to-leaf navigation strategy, which is insufficient to provide alternative evaluation strategies.

2.3 XPRESS

Like XGrind, XPRESS [6] also allows queries on compressed data. XPRESS works only on XML trees. It cannot handle ID/IDREF tags. It creates bisimilar partitions of elements in the XML document. Then, it encodes partitions by assigning disjoint intervals and allows query evaluation by operations on these intervals. It uses an encoding method known as the reverse arithmetic encoding. It is a combination of differential and binary encoding methods. This is an efficient path encoding method, which yields fewer overheads of partial decompression and quicker path evaluation. XPRESS provides high compression ratio.

2.4 XPACK

XPACK [5] is an XML compressor that uses grammar for compressing and decompressing XML documents. XPACK is composed of the following three parts: the Grammar Generator, the Compressor, and the Decompressor. The Grammar Generator creates grammar. The Compressor compresses XML document and the Decompressor decompresses compressed XML document by using the old structure of the document.

The disadvantage of XPACK is that it cannot compress XML documents that have mixed-content elements (elements that have both elements and character data). Moreover, users cannot query data from compressed XML documents.

3. FEATURES OF XBREVITY

The important features of XBrevity are as follows:

- Being able to compress and decompress an XML document
- Being able to compress an XML document with mixed-content elements
- Being able to query compressed XML documents
- Being able to compress a folder of XML documents

Following paragraphs describe one of features of XBrevity which is that it can compress an XML document that has mixed-content elements. The example of the XML document with mixed-content elements is shown in Fig.1. This document consists of "movie" element which is the root element, then the "mtitle" element which has "mname" element. Here, it can be seen that "mtitle" element is a mixed-content element since it contains both sub-elements ("mtitle") and the character data ("the prisoner of"). "actor" element is also a mixed-content element since it contains both character data ("Tim Robbins") and other elements ("actor") which are nested. The "mtitle" element has one attribute which its name is "year" and its value is "1994", and two sections of character data that has value "the prisoner of" and "redemption".

```
<?xml version="1.0"?>
<movie>
   <mtitle year="1994">the prisonner of

<mname>shawshank</mname>redemption</mtitle>
  <actor>Tim Robbins
     <actor>Morgan Freeman <actor>Bob Gunton
     </actor>William Sadler<actor>Clancy Brown
     </actor>Gil Bellows</actor>
   </actor>
</movie>
```

Fig.1: An Original Sample XML File

```
<?xml version=="1.0"?>
<c>
<d>
   <e1    a2="1994"    v="the    prisonner    of"
          e3v="shawshank"/>
   <xe1 v="redemption"/>
   <e4   v="Tim   Robbins"   e4v="Morgan   Freeman"
          e4v2=" Bob Gunton" />
   <xe4e4 v="William Sadler" e4v="Clancy Brown"/>
   <xe4e4 v="Gil Bellows" />
</d>
<m f="movie mtitle year mname actor"
          b="0 1 a2 3 4 "/>
</c>
```

Fig.2: The Compressed Sample XML File

When an XML document has mixed-content elements, XBrevity uses the following rules in compressing the document. If a close tag occurs in a mixed-content element, XBrevity separates the results into multiple parts. For example, in the "mtitle" element, there is a close tag which is </mname>. The first part of the result is <e1 a2="1994" v="the prisoner of" e3v="shawshank"/>. Then, "redemption" is included in the second part of the result which is a new element that has "x" as a prefix to indicate that this element is the part of the previous element which is the "mtitle" element.

The actor tags are nested in three levels and are mapped into "e4" with three attributes which are "v" (which is corresponding to the content of the first actor), "e4v" (which is corresponding to the content of the second actor) , and "e4v2" (which is corresponding to the content of the third actor). When the same element is nested more than three levels, XBrevity adds level number starting from the number "2". Fig. 2 shows the compressed XML version of the sample XML document with mixed-content elements, which the document is shown in Fig. 1.

4. COMPRESSION TECHNIQUES IN XBREVITY

The architecture of XBrevity consists of Compressor which is used to encode the original XML document to the compressed XML document and Decompressor which is used to decode the compressed XML document back to the original XML document. XBrevity has the XML File Filter to distinguish the program input that is a single XML document and the program input that is a folder containing several XML documents. SAX Parser is used to parse and analyze documents for encoding and decoding the documents. The architecture of XBrevity is shown in Fig.3.

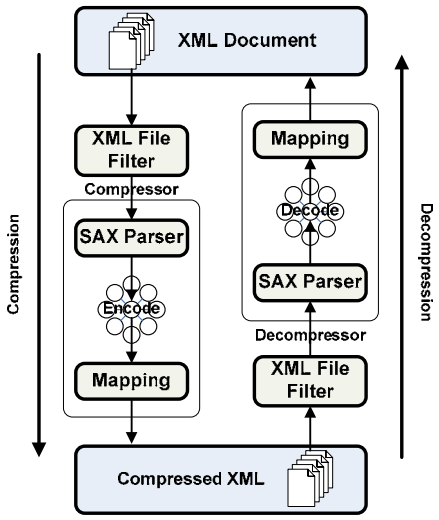


Fig.3: The Architecture of XBrevity

4.1 COMPRESSOR

The function of Compressor is to compress an XML document which uses SAX Parser for reading and analyzing the input XML document. While SAX Parser analyses the document, it encodes the data by using the mapping method that produces the result which is then saved in a new well-formed XML file.

The new XML document consists of the "c" element which is the root element. "c" stands for "compressed" data. The "c" element has two child elements: the "d" element and the "m" element. "d" stands for data and "m" stands for "metadata" Thus, the information between <d> and </d> is the compressed XML data and the information inside <m…/> is the metadata of the compressed XML data. The new compressed XML document can be stored, exchanged over the network, and is easy to understand since it is in an XML format and has metadata.

The next subsection will describe how the Decompressor component of XBrevity decodes the compressed XML document back to the original document.

4.2 DECOMPRESSOR

The Decompressor of XBrevity also uses SAX Parser to read and analyze the compressed XML document by mapping information inside <m…/> tag then using the mapped information to construct the data inside that is in between <d> and </d> in a new XML file. As a result, it will generate an XML document that is the same as the original XML document.

5. EXPERIMENTAL EVALUATION

In this section, we present the results from an experimental evaluation of XBrevity, and compare it with current compression techniques.

5.1 EXPERIMENTAL SETUP

XBrevity that we implemented is a stand-alone Java application. It uses the Apache Xerces Java version 2.8 [1] as SAX Parser APIs. The source code can be downloaded at

http://gear.kku.ac.th/~krunapon/research/xbrevity/

All our experiments operate on Intel Pentium 4 1.8 GHz, RAM 256 MB, Windows XP Professional with Service Pack 2 OS.

We use XMark [8] to randomly generate XML documents with different sizes. The properties of XML files are shown in Table 1. After generating XML documents, we compare the performance of XBrevity and other current techniques which include gzip and XMill. We cannot test XGrind, XPRESS, and XPACK because the available XGrind program is available only on the platform that is different from the platform that we use. XPRESS and XPACK do not have the programs available for downloading.

Table 1: XML Data Set

XML Files	Size (bytes)	Depth	Elem	Attr	XMark factor
a.xml	27,233	8	58	3	0.0001
b.xml	118,274	9	72	4	0.001
c.xml	1,182,547	10	72	5	0.01
d.xml	11,875,066	10	72	5	0.1
e.xml	118,552,713	10	72	5	1

5.2 EXPERIMENTAL RESULTS

In this section, we first present the compression ratio of each compressor. The compression ratio is defined as follows:

$$Compression\ ratio = \frac{Size\ of\ Compressed\ XML}{Size\ of\ Original\ XML}$$

Table 2 shows that compressed XML documents have smaller sizes than that of original documents.

Table 2: Compression Size after Performing gzip

XML Files	Compressed Size (bytes)		
	Original	XBrevity	XBrevity +gzip
a.xml	27,233	26,587	10,813
b.xml	118,274	116,051	41,535
c.xml	1,182,547	1,158,109	383,921
d.xml	11,875,066	11,631,650	3,850,626
e.xml	118,552,713	115,701,249	38,860,417

XMILL and gzip are binary compression but XBrevity compresses only tag. When we combine using XBrevity with gzip, the result from applying different techniques are shown in Table 3.

Table 3: Compression Ratio after Performing gzip

XML Files	Compression Ratio (%)		
	gzip	XMill	XBrevity+gzip
a.xml	61.47	61.83	60.30
b.xml	65.55	67.18	64.88
c.xml	67.83	70.72	67.53
d.xml	67.82	71.37	67.57
e.xml	67.73	71.50	67.22
Average	**66.08**	**68.52**	**65.50**

We also plot the graph to compare the performance of gzip, XMill, and XBrevity+gzip as shown in Fig. 4

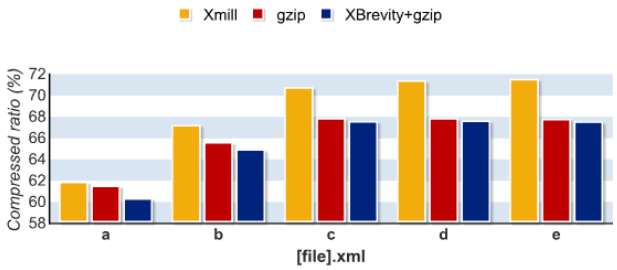


Fig.4: Compression Ratios of Different Techniques

6. CONCLUSIONS

XML compression is important because XML language has many advantages but its disadvantage is its large size since an XML document usually has repeated tags. Large size of data results in an excessive amount of time in transmitting XML data since it requires a large network bandwidth and also results in a large disk space. Several researchers have developed many compression techniques that can reduce XML data size but some methods need XML schema [11,12,13] in order to be able to compress data and some methods have to decode document to original document before applying queries [14].

In this paper, we propose the technique for compressing and decompressing XML data, which is called XBrevity. It can reduce the document size of a single XML file as well as a folder of several XML files. XBrevity has several advantages, such as it does not require the schema information of XML document and the user can query compressed document which saves decoded time.

In the future, we plan to enhance XBrevity to be able to compress the character data section in an XML document which will result in greater reduced size of the document.

7. REFERENCES

[1] Apache Software Foundation. "Xerces2 Java Parser 2.8.0 Release". Available at http://xerces.apache.org/xerces2-j/

[2] J.L. Gailly and M. Adler, "gzip : The compressor data", Available at http://www.gzip.org/

[3] P. Lakhasophon and K. Runapongsa. "A Survey of XML Data Compression". *Proceeding of the 1st Northeastern Computer Science and Engineering Conference (NECSEC2005)*, 2005 March 31–April 1, Khon Kaen, Thailand.

[4] H. Liefke and D. Suciu. "XMill: an Efficient Compressor for XML Data.", *Proceeding of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 153-164, May 2000.

[5] K. Mairiang, and C. Pluempitiwiriyawej. "XPACK: A Grammar-based XML Document Compression", In *Proceeding of NCSEC2003 the7th National Computer Science and Engineering Conference*, Oct 28-30, 2003.

[6] J.-K. Min, M.-J. Park, and C.-W. Chung. "XPRESS: A Queriable Compression for XML Data." *Proceeding of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 122-133, June 9-12, 2003.

[7] K. Runapongsa, J.M. Patel. "Storing and Querying XML Data in Object-Relational DBMSs". *Proceeding of the EDBT Workshops 2002. Conference on Extending Database Technology*. 2002 March 24-28, Prague, Czech Republic, 2002. p. 266-285.

[8] A. R. Schmidt, F. Waas, M. L. Kersten, M. J. Carey, I. Manolescu, and R. Busse. "XMark: A Benchmark for XML Data Management", *Proceedings of the International Conference on Very Large Data Bases (VLDB'02)*, pp. 974-985, Hong Kong, China, August 2002, Available at http://monetdb.cwi.nl/xml /index.html

[9] P. M. Tolani and J. R. Haritsa. "XGRIND: A Query-friendly XML Compressor." *Proceedings of the 18th International Conference on Databases Engineering (ICDE'02)*, Feb 2002.

[10] W3C. "Extensible Markup Language (XML) 1.0 (Third Edition)", Feb 4, 2004, Available at http://www.w3.org/TR/2004/REC-xml-20040204/.

[11] W3C. "XML Schema Part 0 : Primer " , May 2, 2001, Available at http://www.w3.org/TR /xmlschema-0/.

[12] W3C. “XML Schema Part 1 : Structures”, May 2., 2001, Available at http://www.w3.org/TR/xmlschema-1/.

[13] W3C. “XML Schema Part 2 : Datatypes ”, May 2, 2001, Available at http://www.w3.org/TR/xmlschema-2/

[14] W3C. “XQuery : An XML Query Language”, Available at http://www.w3.org/XML/Query

ภาคผนวก ค

คำสั่งเทียม (Pseudo code) ของ XBrevity

คำสั่งเทียมของตัวบีบอัด (Compressor)

```
START
        INTEGER index = 0
        STRING describe = “”
        STRING file_name = “”
        STRING compress_file = “”
        READ file_name
        OPEN file_name
        READ file_name
        IF not end file THEN
                IF declaration THEN
                        READ new describe
                END-IF
                PUSH describe AND index TO hashtable
                CREATE compress_file
                OPEN  compress_file
                WRITE open tag              \\(open tag is “<” )
                index = index + 1
        END-IF
        WHILE not end file
                READ new describe
                IF element OR attribute THEN
                        IF not in hashtable THEN
                                PUSH describe AND index TO hashtable
                                index = index + 1
                        END-IF
                        GET index value is describe FROM hashtable
                        WRITE index AND describe refer TO defined format
                ELSE-IF
                        content element THEN
                        WRITE describe refers TO defined format
                ELSE
                        WRITE close_tag             \\(close tag is “>”)
                END-IF
        END-WHILE
STOP
```

คำสั่งเทียมของตัวขยายการบีบอัด (Decompressor)

```
START
        STRING describe = “”
        STRING index = “”
        STRING file_name = “”
        STRING decompress_name = “”
        READ file_name
        OPEN file_name
        READ metadata_tag                    \\(data in “<m …>” tag )
        PUSH metadata AND index TO hashtable
        CREATE decompress_name
        OPEN decompress_name
        WRITE open_root_tag
        WRITE not_end_file
                READ describe
                IF describe is element THEN
                        IF starts with “e” THEN
                                SPRIT “e” FROM describe
                                GET index which it’s value is describe FROM hashtable
                                WRITE index AND describe refer TO defined format
                        ELSE-IF starts with “a” THEN
                                SPLIT “e” FROM describe
                                GET index value is describe FROM hashtable
                                WRITE index and describe refer TO defined format
                                ELSE
                                WRITE describe refers TO defined format
                        END-IF
                ELSE-IF starts with “x” THEN
                        SPRIT “e” FROM describe
                        GET index which it’s value is describe FROM hashtable
                        WRITE index AND describe refer TO defined format
                        ELSE
                                WRITE close tag_refers TO defined format
                END-IF
        End-WHILE
STOP
```

ภาคผนวก ง

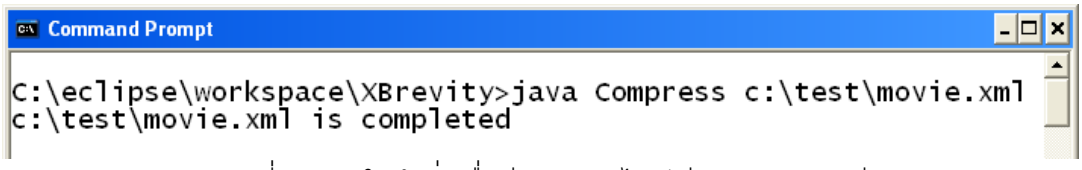
การใช้งาน XBrevity

การบีบอัด ซึ่งแบ่งออกเป็น 2 แบบคือ การบีบอัดไฟล์เดียว หรือบีบอัดทั้งแฟ้ม โดยทำการพิมพ์คำสั่งที่ command line ได้ดังนี้

สำหรับการบีบอัดไฟล์เดียวใช้คำสั่ง

java Compress [drive]:\folder\file.xml

ในตัวอย่างภาพที่ 33 ใช้ไฟล์ที่ชื่อ movie.xml



ภาพที่ 33 การใส่คำสั่งเพื่อบีบอัดแบบไฟล์เดียวและผลการบีบอัด

เมื่อบีบอัดเสร็จแล้วจะได้ไฟล์ใหม่ ซึ่งจะใช้ “_c” กำกับหลังชื่อไฟล์เดิมเพื่อบ่งบอกว่าเป็นไฟล์ที่ถูกบีบอัดของไฟล์ใด ซึ่งในตัวอย่างนี้คือ movie_c.xml

ภาพที่ 34 เมื่อเทียบไฟล์เดิมกับไฟล์ที่บีบอัดเสร็จแล้ว

```
<?xml version="1.0"?>
<movie>
    <mtitle year="1994">the prisonner
    of<mname>shawshank</mname>redemption</mtitle
    >
    <actor>Tim Robbins
    <actor>Morgan Freeman<actor>Bob
    Gunton</actor>William Sadler
    <actor>Clancy Brown</actor>Gil
    Bellows</actor>
    </actor>
</movie>
```

ภาพที่ 35 เอกสารภายในก่อนทำการบีบอัด

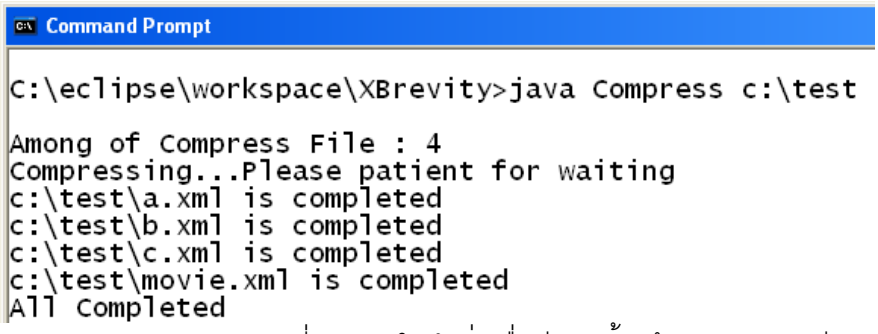
```
<c>
<d>
<e1 a2="1994" v="the prisonner of"
e3v="shawshank" />
<xe1 v="redemption" />
<e4 v="Tim Robbins" e4v="Morgan Freeman"
e4v2="Bob Gunton" />
<xe4e4 v="William Sadler" e4v="Clancy Brown" />
<xe4e4 v="Gil Bellows" />
</d>
<m f="movie mtitle year mname actor "
b="0 1 a2 3 4 "/>
</c>
```

ภาพที่ 36 เอกสารภายในเมื่อผ่านการบีบอัดแล้ว

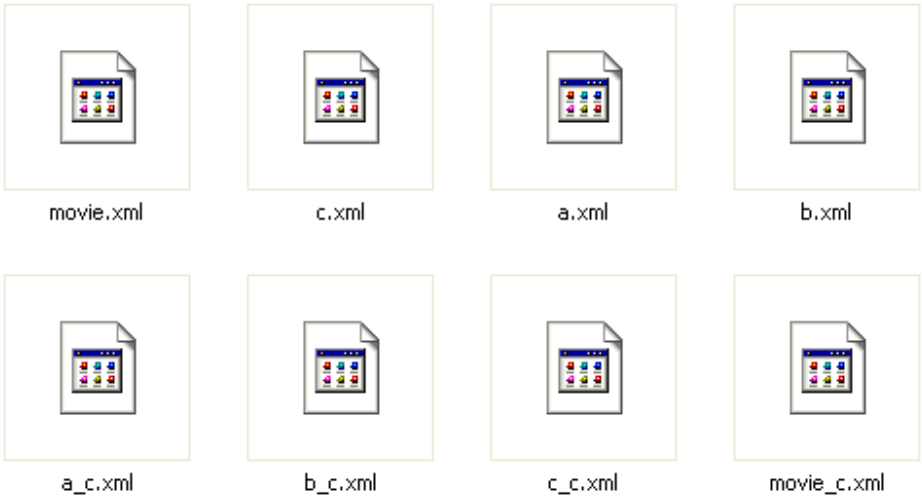
สำหรับการบีบอัดทั้งแฟ้มใช้คำสั่ง

java Compress [drive]:\folder

ในตัวอย่างภาพที่ 37 ใช้แฟ้มที่ชื่อ test ซึ่งมีไฟล์ทั้งสิ้น 4 ไฟล์



ภาพที่ 37 การใส่คำสั่งเพื่อบีบอัดทั้งแฟ้มและผลการบีบอัด



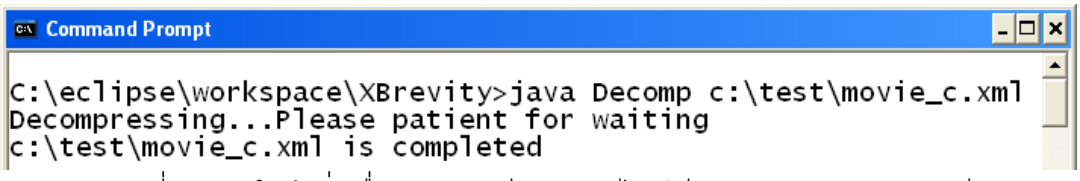
ภาพที่ 38 เมื่อเทียบไฟล์เดิมกับไฟล์ที่บีบอัดเสร็จแล้วภายในแฟ้ม

การขยายการบีบอัด ซึ่งแบ่งออกเป็น 2 แบบคือ ขยายการบีบอัดไฟล์เดียว หรือขยายการบีบอัดทั้งแฟ้ม โดยทำการพิมพ์คำสั่งที่ command line ได้ดังนี้

สำหรับขยายการบีบอัดไฟล์เดียวใช้คำสั่ง

java Decomp [drive]:\folder\file_c.xml

ในตัวอย่างภาพที่ 39 ใช้ไฟล์ที่ชื่อ movie_c.xml



ภาพที่ 39 การใส่คำสั่งเพื่อขยายการบีบอัดแบบไฟล์เดียวและผลการขยายการบีบอัด

และเมื่อเปิดเอกสารที่ผ่านการขยายการบีบอัดแล้วจะได้เอกสารที่มีความถูกต้องเช่นเดียวกับเอกสารก่อนการบีบอัดทุกประการ

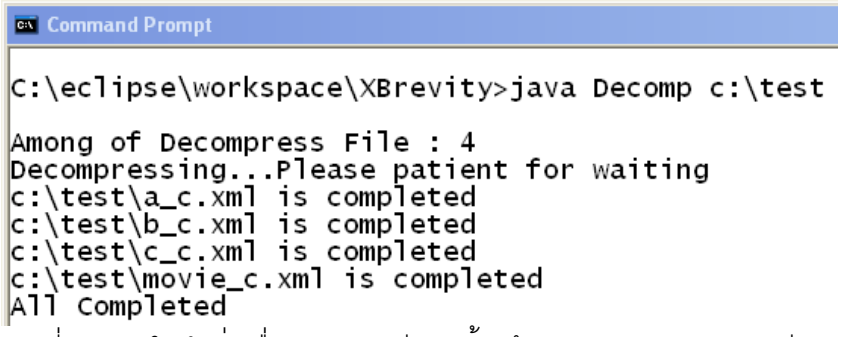
```
<?xml version='1.0' encoding='UTF-8'?>
<movie>
<mtitle year = "1994">the prisonner
of<mname>shawshank
</mname>
redemption</mtitle>
<actor>Tim Robbins<actor>Morgan Freeman
<actor>Bob Gunton
</actor>
William Sadler<actor>Clancy Brown
</actor>
Gil Bellows</actor>
</actor>
</movie>
```

ภาพที่ 40 เอกสารภายในเมื่อผ่านการขยายการบีบอัดแล้ว

สำหรับขยายการบีบอัดทั้งแฟ้มใช้คำสั่ง

java Decomp [drive]:\folder

ในตัวอย่างภาพที่ 41 ใช้แฟ้มที่ชื่อ test ซึ่งมีไฟล์ทั้งสิ้น 4 ไฟล์



ภาพที่ 41 การใส่คำสั่งเพื่อขยายการบีบอัดทั้งแฟ้มและผลการขยายการบีบอัด

หมายเหตุ

หลังจากบีบอัดหรือขยายการบีบอัดแล้วไฟล์เหล่านั้นจะอยู่ในโฟลเดอร์ปัจจุบันที่ได้ทำการบีบอัดหรือขยายการบีบอัด