การศึกษาเปรียบเทียบวิธีบีบอัดข้อมูลที่เหมาะสมสำหรับแต่ละประเภทข้อมูล

A Comparative Study of Compression Algorithms for Each Data Type.

ชนาภา ศิลาวงษ์ และ ธนภัทร์ อนุศาสน์อมรกุล
ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ
คณะวิทยาศาสตร์ประยุกต์มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ
Email: chachachar.sila@gmail.com 1, Email: tanapata@kmutnb.ac.th2

บทคัดย่อ

ปัจจุบันอินเทอร์เน็ตถูกนำมาใช้ในการแลกเปลี่ยนข้อมูล เช่น ข้อความ รูปภาพ เสียงและวิดีโอซึ่งต้องการความรวดเร็วในการรับส่งข้อมูล อย่างไรก็ตามเมื่อขนาดของข้อมูลเพิ่มมากขึ้น เวลาที่ใช้ในการส่งข้อมูลก็มากขึ้น ดังนั้นจึงมีการนำเอาเทคนิคการบีบอัดข้อมูลที่ใช้ในการลดขนาดของข้อมูลก่อนที่ข้อมูลจะถูกส่ง ไม่เพียงแต่ประเภทข้อมูลที่ส่งผ่านอินเทอร์เน็ตที่มีความหลากหลาย ขั้นตอนวิธีการบีบอัดข้อมูลก็มีความหลากหลายด้วยเช่นกัน ในงานวิจัยฉบับนี้ได้วิเคราะห์ประเภทข้อมูลหลายรูปแบบเพื่อหาอัลกอริทึมที่เหมาะสมสำหรับข้อมูลโดยรวม โดยวัดจากอัตราการบีบอัดข้อมูล เวลาที่ใช้ในการบีบอัดและคลายข้อมูล เวลารวมที่ใช้ในการรับ-ส่งข้อมูลผ่านเครือข่าย และอัตราการส่งข้อมูล ในงานวิจัยนี้ได้ศึกษาเปรียบเทียบอัลกอริทึมแบบไม่สูญเสีย 4 ประเภท ได้แก่ Huffman LZ77 LZW และ Deflate ผลลัพธ์แสดงให้เห็นว่า อัลกอริทึม Deflate มีประสิทธิภาพโดยรวมดีที่สุด นอกจากนี้ ถ้าขนาดข้อมูลเล็กมาก และเครือข่ายมีแบนด์วิดท์สูง ไม่จำเป็นต้องใช้อัลกอริทึมการบีบอัดข้อมูล

คำสำคัญ

การบีบอัดข้อมูล อัลกอริทึมแบบไม่สูญเสีย อัลกอริทึม Deflate

Abstract

Recently, Internet is used to exchange information, such as text, picture, audio and video, which needs to be fast. However, the more data size, the more time to wait. Therefore, data compression techniques are used to reduce the data size before the data is transmitted. Not only the data type sent over the Internet is various, but also data compression algorithms. In this research, various data types are analyzed to find an appropriate compression algorithm in terms of a compression rate, a compression delay, decompression delay andoverall time. In this paper, four lossless compression algorithms, Huffman, LZ77, LZW and Deflate, are studied and compared. The results show that Deflate algorithm gives the best overall performance. In addition, if the data size is smalland the bandwidth is high,compression algorithms are not necessary.

Keywords

data compression, lossless compression algorithms, Deflate algorithms

1. บทนำ

เทคโนโลยีการสื่อสารในปัจจุบันมีการติดต่อสื่อสารแลกเปลี่ยนข้อมูลกันผ่านเครือข่ายอินเทอร์เน็ตเป็นจำนวนมากไม่ว่าจะเป็นข้อมูลข่าวสารความบันเทิงต่างๆ ข้อมูลตลาดหุ้น ที่จะเป็นข้อมูลที่ประกอบไปด้วยตัวเลขและตัวอักษรที่บอกชื่อและราคาหุ้น บทความภาษาต่างๆ เช่น ภาษาไทยและภาษาอังกฤษ เกมออนไลน์ เช่น ตำแหน่ง คุณสมบัติของตัวละคร ฉาก [1,2] หรือแม้กระทั่งการติดต่อสื่อสารผ่านทางโซเชียลเน็ตเวิร์คที่จะมีการอัพเดตข้อมูลรูปภาพ การแชร์ตำแหน่ง วิดีโอ หรือการอัพเดตสถานะผ่านอุปกรณ์สื่อสารทั้งอุปกรณ์แบบพกพา เช่น มือถือ หรือแทบเล็ตต่างๆ หรือแม้กระทั่งเว็บเบราว์เซอร์บนเครื่องคอมพิวเตอร์ทั่วไป เป็นต้น

ข้อมูลที่ส่งในเครือข่ายอินเทอร์เน็ตส่วนใหญ่นั้นต้องการเวลาในการส่งและรับข้อมูลที่รวดเร็ว แต่ในบางกรณีข้อมูลที่ต้องการส่งอาจมีขนาดใหญ่ซึ่งการส่งข้อมูลขนาดใหญ่ต้องใช้เวลาในการส่งข้อมูลนาน จึงมีการนำเทคนิคที่เกี่ยวกับการบีบอัดข้อมูลมาใช้ในการลดขนาดของข้อมูลก่อนที่จะส่ง ทำให้ข้อมูลที่ใช้ส่งในเครือข่ายมีขนาดเล็กลง จึงสามารถลดทรัพยากรที่ใช้ไม่ว่าจะเป็นในเรื่องของแบนด์วิดท์ และเวลาที่ใช้ในการส่งข้อมูลจากผู้ส่งไปยังผู้รับ แต่รูปแบบของข้อมูลที่ใช้ส่งในเครือข่ายอินเทอร์เน็ตค่อนข้างมีความหลากหลายไม่ว่าจะเป็นข้อความ รูปภาพ เสียง อีกทั้งวิธีการที่จะนำมาใช้ในการลดขนาดข้อมูลหรือที่เรียกว่าการบีบอัดข้อมูลก็มีอยู่ด้วยกันหลายวิธี แต่ละวิธีจะมีวิธีการทำงานและเวลาที่ใช้ในการประมวลผลแตกต่างกันออกไป ซึ่งวิธีการที่จะนำเข้ามาใช้สำหรับการบีบอัดข้อมูลแต่ละประเภทจึงต้องพิจารณาจากหลายปัจจัยด้วยกัน ไม่ว่าจะเป็นรูปแบบที่ใช้ในการเก็บข้อมูล ประเภทของข้อมูล เวลาที่ใช้ในการบีบอัดและคลายข้อมูล เป็นต้น

ในงานวิจัยนี้ได้วิเคราะห์รูปแบบของข้อมูลแบบต่างๆ เพื่อเลือกวิธีการบีบอัดข้อมูลที่เหมาะสม โดยจะพิจารณาในเรื่องของอัตราการบีบอัดข้อมูล เวลาที่ใช้ในการบีบอัดและคลายข้อมูล รวมไปถึงเวลาที่ใช้ในการส่งข้อมูลจากเครื่องต้นทางไปยังเครื่องปลายทาง เพื่อให้ได้วิธีการบีบอัดข้อมูลที่เหมาะสมสำหรับข้อมูลแต่ละประเภทที่ใช้กันอยู่ในปัจจุบัน

ส่วนที่ 2 อธิบายทฤษฎีที่สำคัญสำหรับงานวิจัยนี้พร้อมทั้งงานวิจัยที่เกี่ยวข้อง ถัดมาส่วนที่ 3 นำเสนอวิธีการทดลอง ส่วนที่ 4 เป็นการวิเคราะห์ผลการทดลอง และส่วนสุดท้ายเป็นสรุปผลการทดลอง

2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

การบีบอัดข้อมูล (Data Compression) เป็นการบีบอัด (Compress) เพื่อให้ใช้จำนวนบิตในการจัดเก็บข้อมูลน้อยลงกว่าเดิม [3-5] การบีบอัดข้อมูลมีประโยชน์ในการช่วยลดปริมาณการใช้ทรัพยากร เช่น ประหยัดพื้นที่ที่ใช้ในการจัดเก็บข้อมูล หรือใช้แบนด์วิดท์น้อยลงเพื่อส่งข้อมูลที่บีบอัดแล้ว ในทางตรงกันข้ามก่อนที่จะนำข้อมูลที่ถูกบีบอัดข้อมูลมาใช้ต้องคลายข้อมูล (Decompress) ก่อนการบีบอัดข้อมูล แบ่งออกเป็น 2 ประเภท คือ 1) การบีบอัดข้อมูลแบบไม่สูญเสีย (Lossless) คือไฟล์ที่นำมาคลายข้อมูลจะได้ไฟล์เหมือนต้นฉบับทุกประการ 2) การบีบ

อัดข้อมูลแบบสูญเสียข้อมูลบางส่วน (Lossy) คือเมื่อนำไฟล์มาคลายข้อมูลแล้วจะได้ไฟล์ที่ไม่เหมือนกับต้นฉบับเดิม

2.1 ทฤษฎีการบีบอัดข้อมูล

ในงานวิจัยฉบับนี้จึงพิจารณาเฉพาะอัลกอริทึมแบบไม่สูญเสีย 4 ชนิดได้แก่ Huffman, LZ77, LZW และ Deflate มีงานวิจัยหลายชิ้นชี้ให้เห็นว่าอัลกอริทึมที่เลือกมาพิจารณามีประสิทธิภาพในการบีบอัดข้อมูลและใช้เวลาในการบีบอัดข้อมูลและการคลายข้อมูลน้อย

- HuffmanAlgorithm [4-7] ข้อมูลจะถูกแปลงเป็น binary ซึ่ง characters ที่พบบ่อยจะมีรหัสสั้นกว่าตัวที่พบน้อย

- LZ77 Algorithm [5,8,9] ข้อมูลที่ซ้ำจะถูกกำหนดโดย offset และ length ของข้อมูลที่เกิดขึ้นก่อนหน้า

- LZW Algorithm [4,5,8,9,10] ข้อมูลที่ซ้ำจะถูกกำหนดเป็น index ไปยัง dictionary ที่สร้างขึ้นแบบ dynamic

- Deflate Algorithm [11-13] เป็นอัลกอริทึมที่เกิดจากการนำข้อดีของอัลกอริทึม Huffman และ LZ77 มารวมกันซึ่ง Deflate มีการจัดการการบีบอัดข้อมูลที่มีความยืดหยุ่นมากกว่า

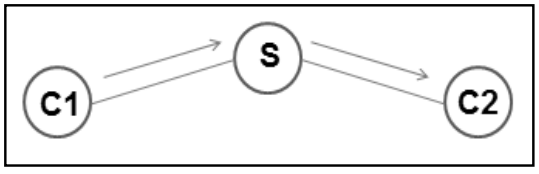
2.2 งานวิจัยที่เกี่ยวข้อง

การเปรียบเทียบประสิทธิภาพของอัลกอริทึมแต่ละตัว มีงานวิจัยมากมายที่ศึกษาวิเคราะห์ประสิทธิภาพของอัลกอริทึมว่ามีประสิทธิภาพการทำงานเป็นอย่างไร ดีมาก-น้อยเพียงใด เมื่อเทียบกับอัลกอริทึมตัวอื่น การเปรียบเทียบการทำงานของอัลกอริทึมแต่ละตัว เช่นในงานวิจัยของ Mohammed Al-laham และ Ibrahiem [5] ได้เปรียบเทียบอัลกอริทึมที่ใช้ในการบีบอัดข้อมูลระหว่าง Huffman และ LZW จากนั้นออกแบบทดลองการบีบอัดข้อมูลจากไฟล์ประเภทต่างๆ เช่น doc, txt, bmp, tif, gif, และ jpg โดยใช้วิธีการบีบอัดข้อมูลที่แตกต่างกัน โดยสิ่งที่พิจารณาคือ อัตราการบีบอัดข้อมูลและไฟล์ข้อมูลที่ถูกบีบอัดแล้ว ผลการทดลองแสดงให้เห็นว่าข้อมูลประเภทที่ไม่ผ่านการบีบอัด (.doc และ .bmp) หลังจากมีการบีบอัดข้อมูลแล้วจะมีขนาดไฟล์เล็กลง ซึ่งอัตราส่วนของการบีบอัดข้อมูลของทั้งสองวิธีค่อนข้างใกล้เคียงกัน แต่สำหรับไฟล์ข้อมูลประเภทที่ผ่านการบีบอัดมาแล้ว (.jpg และ .gif) พบว่าข้อมูลหลังจากมีการบีบอัดข้อมูลมีขนาดใหญ่ขึ้นกว่าไฟล์ต้นฉบับ ซึ่งไฟล์ที่มีการบีบอัดข้อมูลด้วยวิธี LZW มีขนาดไฟล์เพิ่มขึ้นมากกว่าไฟล์ที่มีการบีบอัดข้อมูลด้วยวิธี Huffman งานวิจัยนี้ชี้ให้เห็นว่า ไฟล์ข้อมูลที่ผ่านการบีบอัดข้อมูลมาแล้วไม่เหมาะที่จะนำมาบีบอัดข้อมูลอีกครั้ง เพราะอาจจะลดขนาดของไฟล์ได้เพียงเล็กน้อยหรือในบางกรณีอาจจะทำให้ขนาดของไฟล์เพิ่มขึ้น

3. วิธีการดำเนินการทดลอง

3.1 เครื่องมือที่ใช้ในการพัฒนาและทดสอบ

เครื่องคอมพิวเตอร์สำหรับใช้ในการทดลองมีจำนวน 3 เครื่องเชื่อมต่อกันในระบบ LAN ดังภาพที่ 1



ภาพที่ 1 รูปแบบการเชื่อมต่อของเครื่องที่ใช้ในการทดลอง

เครื่องเซิร์ฟเวอร์

- ฮาร์ดแวร์ เครื่องคอมพิวเตอร์ตั้งโต๊ะ 1 เครื่อง ประกอบด้วย ซีพียู Intel Core i5-25003.30 GHz หน่วยความจำหลัก 4 GB ฮาร์ดดิสก์ขนาด 1 TB

- ซอฟต์แวร์ระบบปฏิบัติการ Windows 7 64 bit, เครื่องมือพัฒนาโปรแกรม NetBeans IDE 7.2.1, Java JDK 7, Browser Chrome version 14 ขึ้นไป

เครื่องไคลเอนต์

- ฮาร์ดแวร์ เครื่องคอมพิวเตอร์ Notebook2 เครื่อง เครื่องที่ 1 ประกอบด้วย ซีพียู Intel CoreTM 2 Duo T81002.1 G หน่วยความจำหลัก 2 GB ฮาร์ดดิสก์ขนาด 200 GB เครื่องที่ 2 Mac Book 1 ชุด ประกอบด้วย ซีพียู Intel Core i7 2 GHz หน่วยความจำหลัก 4 GB ฮาร์ดดิสก์ขนาด 500 GB

- ซอฟต์แวร์ เครื่องที่ 1 ประกอบด้วย ระบบปฏิบัติการ Windows 7 32 bit เครื่องที่ 2 ประกอบด้วยระบบปฏิบัติการ Mac OS X Lion 10.7.5, เครื่องมือพัฒนาโปรแกรม NetBeans IDE 7.2, Java JDK 7, Browser Chrome version 14 ขึ้นไป

3.2 ข้อมูลที่ใช้ในการวิจัย

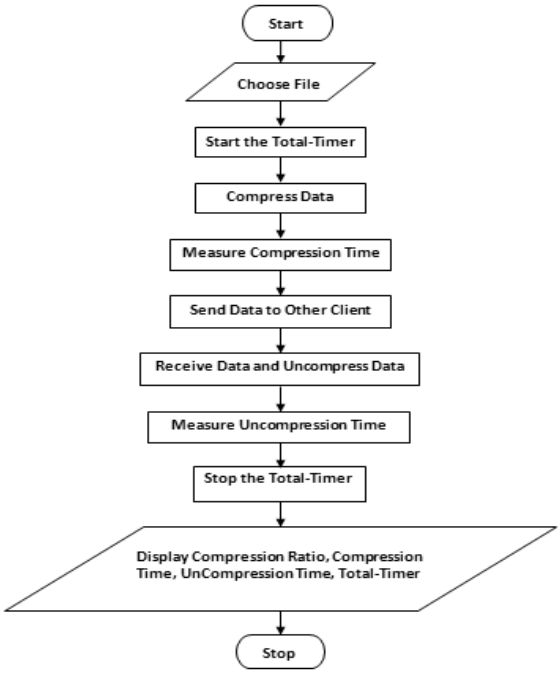
ข้อมูลที่นำมาใช้ในการทำวิจัย เป็นข้อมูลจากรูปแบบไฟล์ประเภทต่างๆ ที่นิยมใช้กันอยู่ในปัจจุบัน ผู้วิจัยได้เลือกตัวอย่างของไฟล์ที่มีความหลากหลาย เพื่อให้สามารถทดสอบประสิทธิภาพการทำงานของตัวอัลกอริทึมแต่ละแบบได้อย่างมีประสิทธิภาพ ซึ่งข้อมูลที่ใช้ในการทดสอบมี 10 ประเภท แต่ละประเภทมีขนาดที่แตกต่างกัน 15 ขนาด (ประมาณ 90 กิโลไบต์ถึง 5 เมกะไบต์) รวมทั้งสิ้น 150 ไฟล์ แบ่งออกเป็นหมวดใหญ่ๆ ได้ 3 หมวด คือ ไฟล์เอกสาร ไฟล์รูปภาพ และไฟล์เสียง แสดงในตารางที่ 1

ตารางที่ 1 ประเภทของไฟล์ที่นำมาใช้ในการทดลอง

Type	Compressed File	Non-compressed File
Document	DOCX	DOC, TXT, PDF
Graphic	JPG, GIF, PNG	BMP
Sound	MP3	WAV

3.3 การออกแบบระบบงาน

ระบบที่ออกแบบสามารถวัดประสิทธิภาพอัตราการบีบอัดข้อมูลรวมถึงอัตราเร็วที่ใช้ในการบีบอัดและคลายข้อมูลของแต่ละอัลกอริทึม รวมถึงเวลารวมในการส่งข้อมูลแบ่งตามอัลกอริทึมแต่ละตัว เริ่มจากการที่ผู้ใช้เลือกไฟล์ข้อมูล จากนั้นระบบจะบีบอัดและคลายข้อมูลโดยการจับเวลาที่ใช้บีบอัดและคลายข้อมูลแยกตามแต่ละอัลกอริทึม โดยจับเวลาตั้งแต่เริ่มบีบอัดข้อมูล จากนั้นส่งข้อมูลที่บีบอัดแล้วไปยังเซิร์ฟเวอร์ เพื่อส่งต่อไปยังเครื่องปลายทาง และคลายข้อมูลเรียบร้อย การพัฒนาจะแบ่งออกเป็น 2 ส่วน คือ ทางฝั่งไคลเอนต์พัฒนาด้วยภาษา JavaScript [14] ฝั่งเซิร์ฟเวอร์พัฒนาด้วยภาษาจาวา ภาพที่2 แสดงขั้นตอนการทำงานของระบบที่ใช้โพรโทคอลเว็บซ็อกเก็ตในการส่งข้อมูล



ภาพที่ 2 ขั้นตอนการทำงานของระบบที่มีการบีบอัดข้อมูล

3.4 การวัดประสิทธิภาพ

ประสิทธิภาพการบีบอัดข้อมูลจะวัดจากอัตราส่วนการบีบอัดข้อมูล ซึ่งแสดงถึงจำนวนข้อมูลที่เหลือหลังจากการบีบอัด ความสัมพันธ์ดังกล่าวจะอยู่ในรูปของร้อยละขนาดของข้อมูลดั้งเดิม (Filesize) และขนาดของข้อมูลที่ถูกบีบอัดแล้ว (compression size) ดังสมการที่ (1)

$$Compression\ ratio = \frac{compression\ \ size}{file\ \ size} \quad (1)$$

การวัดอัตราเร็วการบีบอัดข้อมูลและการคลายข้อมูลของแต่ละอัลกอริทึมจะนำปริมาณข้อมูลหารด้วยเวลาที่ได้ ดังสมการที่ (2) และ (3)

$$Compression\ Speed\ = \frac{File\ \ size\ 1}{compression\ \ \ time} \quad (2)$$

Compression Speed คืออัตราเร็วในการบีบอัดข้อมูลซึ่งเป็นอัตราส่วนระหว่างปริมาณข้อมูลและเวลาที่ใช้ในการบีบอัดข้อมูล มีหน่วยเป็นไบต์ต่อวินาที File size1 คือ ขนาดข้อมูลก่อนการบีบอัดมีหน่วยเป็นไบต์ compression Time คือ เวลาที่ใช้ในการบีบอัดข้อมูล มีหน่วยเป็นวินาที

$$\text{Decompression speed} = \frac{\text{File size 2}}{\text{Decompression time}} \quad (3)$$

Decompression Speed คืออัตราเร็วในการคลายข้อมูล ซึ่งเป็นอัตราส่วนระหว่างปริมาณข้อมูลและเวลาที่ใช้ในการคลายข้อมูล มีหน่วยเป็นไบต์ต่อวินาที File size2 คือขนาดข้อมูลหลังการบีบอัดมีหน่วยเป็นไบต์ Decompression Time คือ เวลาที่ใช้ในการคลายข้อมูลมีหน่วยเป็นวินาที

การวัดประสิทธิภาพการส่งข้อมูลจากเครื่องต้นทางไปยังเครื่องปลายทางของระบบที่มีการบีบอัดข้อมูลแบ่งตามอัลกอริทึมแต่ละตัว การทดลองจะวัดเวลารวมของระบบที่ใช้ในการส่งข้อมูล 1 ไฟล์จากเครื่องต้นทางไปยังปลายทาง เนื่องจากโมเดลนี้ทดสอบในระบบ LAN และต้องการพิจารณาค่าความหน่วงเวลาที่เกิดขึ้นในชั้น Application Layer อีกทั้งการส่งข้อมูลในการทดลองจะส่งทีละไฟล์จึงไม่นำ Queuing Delay และ Propagation Delay มาคิดเพราะอยู่ในระบบ LAN และระยะทางสั้น สมการที่ได้คือ

$$\begin{aligned} Overall\ Time_{compression} \\ &= Compression\ Time \\ &+ Processing\ Time \\ &+ Decompression\ \ Time + \\ &\ \ Transmission\ Time \end{aligned} \quad (4)$$

Overall Timecompression คือเวลารวมของระบบที่ใช้ในการส่งข้อมูลจากเครื่องต้นทางไปยังเครื่องปลายทางในระบบที่มีการใช้การบีบอัดข้อมูล Processing Time คือเวลาที่เซิร์ฟเวอร์ใช้ในการประมวลผลTransmission Time คือเวลาที่ใช้ในการส่งข้อมูลจากเครื่องต้นทางไปยังปลายทาง ขึ้นกับขนาดข้อมูลและอัตราเร็วในการส่งข้อมูลหรือแบนด์วิดท์ (Bandwidth)

เวลาที่ระบบสามารถวัดได้ มี 4 ตัวเท่านั้น คือ Overall Time, Compression Time, Decompression Time และ Processing Time ถ้าต้องการรู้เวลา Transmission Time สามารถคำนวณหาได้จากสมการที่ (5) และ (6) ดังนี้

$$\begin{aligned} Transmission\ Time = \\ Overall\ Time - (Compression\ Time\ + \\ Processing\ Time\ + \\ Decompression\ Time) \end{aligned} \quad (5)$$

ในการส่งข้อมูลเครือข่ายที่มีแบนด์วิดท์ต่ำควรนำการบีบอัดข้อมูลเข้ามาใช้ จึงจำเป็นต้องพิจารณาแบนด์วิดท์ที่เหมาะสมสำหรับการใช้การบีบอัดข้อมูลให้มีประสิทธิภาพ

$$Transmission\ Time = \frac{Filesize\ 1 \times 8}{Bandwidt\ h} \quad (6)$$

Bandwidth อัตราการส่งข้อมูลมีหน่วยเป็นบิตต่อวินาที

การหาแบนด์วิดท์ที่เหมาะสมกับการบีบอัดข้อมูลไปใช้นั้นจะพิจารณาจากค่าของเวลารวมทั้งแบบที่มีการบีบอัดข้อมูลและแบบที่ไม่ใช้การบีบอัดข้อมูล

$$Overall\ Time_{compression} = \frac{File\ size\ 1}{Compression\ \ Speed} + \frac{File\ size\ 2}{Decompression\ \ Speed} + Processing\ Time + Transmission\ Time \quad (7)$$

File size1 คือ ขนาดข้อมูลก่อนการบีบอัด File size2 คือ ขนาดของข้อมูลหลังถูกบีบอัดแล้ว (File size1 × Compression Ratio) ส่วนเวลารวมของระบบที่ใช้ส่งข้อมูลแบบที่ไม่มีการบีบอัดข้อมูลนั้นจะใช้ชื่อตัวแปรว่า Overall Time no compression มีสมการในการคิดเวลาดังสมการที่ (8)

$$\text{Overall Time}_{\text{no compression}} = \text{Processing Time} + \text{Transmission Time} \quad (8)$$

เมื่อ Overall Time no Compression คือเวลารวมของระบบที่ไม่มีการบีบอัดข้อมูลที่ใช้ในการส่งข้อมูลจากเครื่องต้นทางไปยังปลายทางมีหน่วยเป็นวินาที

แบนด์วิดท์ที่เหมาะสมที่จะนำการบีบอัดข้อมูลเข้ามาใช้ต้องหาแบนด์วิดท์ที่ทำให้เกิดจุดตัดระหว่าง Overall Time no compression และ Overall Time compression สามารถวิเคราะห์ได้จากสมการที่ (7) และ (8) กำหนดให้ Overall Time no compression เท่ากับ Overall Time Compression โดยสมมุติให้ Processing time มีค่าเท่ากันเพราะทดสอบในระบบเดียวกันได้สมการใหม่ดังนี้

$$Bandwidth = \frac{16(1-Compression\ \ Ratio) \times Compression\ \ Speed\ \times Decompression\ \ Speed}{Decompression\ \ Speed + (Compression\ \ Speed \times Compression\ \ Ratio)} \quad (9)$$

โดยสมการที่ (9) คำนวณมาจากรูปแบบการเชื่อมต่อดังแสดงในภาพที่ 1 จะมีการส่งข้อมูล 2 ครั้งก่อนถึงปลายทาง

4. ผลการดำเนินการวิจัย

ตารางที่ 2 ร้อยละการบีบอัดข้อมูลแยกตามอัลกอริทึมและประเภทไฟล์

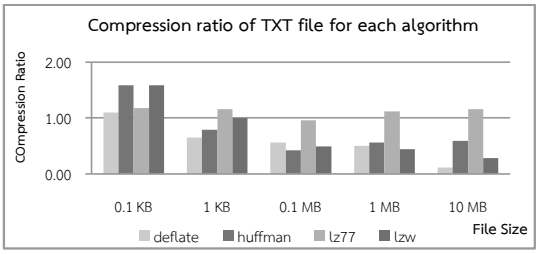
Type	Deflate	Huffman	LZ77	LZW
DOC	88.29%	96.23%	156.33%	125.66%
DOCX	96.40%	99.65%	170.11%	137.62%
PDF	86.64%	97.19%	162.14%	124.45%
TXT	39.93%	53.17%	96.41%	36.70%
PNG	101.17%	100.26%	176.35%	145.67%
JPG	100.10%	99.84%	175.28%	140.01%
GIF	100.99%	100.00%	172.21%	144.40%
BMP	23.85%	51.36%	42.19%	21.44%
MP3	98.30%	99.23%	172.41%	139.42%
WAV	89.71%	87.35%	131.02%	106.55%

ตารางที่ 2 ใช้สมการที่ (1) แสดงร้อยละของการบีบอัดข้อมูลแยกตามอัลกอริทึม และประเภทของไฟล์ข้อมูลแต่ละประเภท แสดงเป็นค่าเฉลี่ยซึ่งเหมาะสมกับอัลกอริทึมแตกต่างกัน ไฟล์ที่มีการซ้ำกันของข้อมูลและยังไม่ได้ผ่านการบีบอัดมาก่อน จะเหมาะกับอัลกอริทึมแบบ LZW เช่นไฟล์ TXT และ BMP เพราะตัวอัลกอริทึมมีการสร้างตารางข้อมูลซ้ำไว้ เมื่อพบข้อมูลที่ซ้ำมากทำให้ลดขนาดของข้อมูลได้มาก

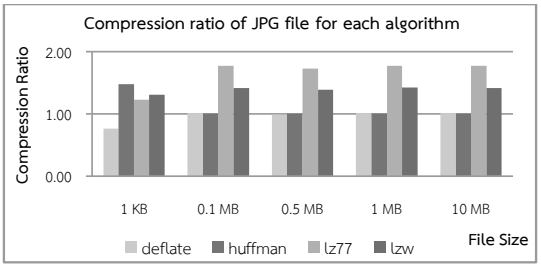
ไฟล์ข้อมูลประเภท PNG, JPG, GIF, WAV บางไฟล์เป็นข้อมูลไม่ได้ถูกบีบอัดมาก่อน (WAV) ข้อมูลมีลักษณะเป็นค่าสุ่ม ทำให้เหมาะกับอัลกอริทึมแบบ Huffman มากกว่า เพราะจะมองข้อมูลเป็นไฟล์ๆ ไป

ถ้ามีความแตกต่างน้อย ก็จะลดขนาดข้อมูลได้มาก ถึงผลลัพธ์ของอัลกอริทึมนี้ จะไม่สามารถลดขนาดของข้อมูลได้มาก แต่ให้ผลลัพธ์ดีที่สุดเมื่อเทียบกับอัลกอริทึมตัวอื่นๆ

ไฟล์ประเภท DOC, DOCX, PDF และ MP3 เหมาะกับอัลกอริทึมแบบ Deflate มากที่สุด เนื่องจากหลักการทำงานของอัลกอริทึมแบบ Deflate มีฟังก์ชั่นให้เลือกทำให้อัลกอริทึมนี้มีความยืดหยุ่น จึงค่อนข้างเหมาะกับไฟล์ที่มีรูปแบบของข้อมูลที่ไม่ซ้ำมาก แต่ยังไม่ผ่านการบีบอัด



ภาพที่ 3 อัตราการบีบอัดข้อมูลของไฟล์ TXT แยกตามอัลกอริทึม



ภาพที่ 4 อัตราการบีบอัดข้อมูลของไฟล์ JPG แยกตามอัลกอริทึม

ภาพที่ 3 และ 4 เป็นผลลัพธ์การบีบอัดข้อมูลของไฟล์ TXT และ JPG แสดงให้เห็นว่าถ้าข้อมูลที่ไม่ผ่านการบีบอัดมาก่อนอัลกอริทึมแต่ละแบบจะมีประสิทธิภาพแตกต่างกัน แต่สามารถบีบอัดข้อมูลได้ดังแสดงในภาพที่ 3 ถ้าข้อมูลที่ผ่านการบีบอัดมาแล้วและบีบอัดซ้ำ จะไม่สามารถบีบอัดได้ดีเท่าที่ควรดังแสดงในภาพที่ 4

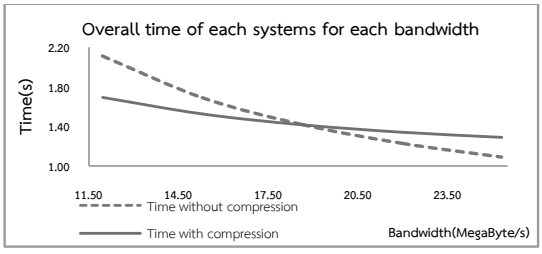
ข้อสังเกตอีกอย่างคือไฟล์ที่มีขนาดต่างกันมีอัตราการบีบอัดข้อมูลที่ใกล้เคียงกัน แต่ถ้าไฟล์ TXT ที่มีขนาดเล็กมาก ไม่ควรนำมาบีบอัดข้อมูล ดังแสดงในภาพที่ 3 เพราะจะทำให้ข้อมูลเพิ่มขึ้นหลังการบีบอัด เมื่อพิจารณาจากค่า Compression Ratio ที่น้อยที่สุดเบื้องต้นสามารถสรุปได้ว่า ไฟล์ประเภท DOC และ PDF เหมาะกับอัลกอริทึมแบบ Deflate ไฟล์ประเภท PNG, JPG และ GIF ไม่ควรนำมาบีบอัดอีกครั้ง ไฟล์ประเภท DOCX และ MP3 ถ้าจะนำการบีบอัดข้อมูลเข้ามาใช้ ต้องใช้ในระบบที่มีแบนด์วิดท์ที่ต่ำมากๆ และใช้อัลกอริทึม Deflate ส่วนไฟล์ประเภท TXT, BMP และ WAV ต้องนำมาพิจารณาต่อ

จากตารางที่ 3 เมื่อพิจารณาถึงเวลารวมที่ใช้ในการส่งข้อมูลจากเครื่องต้นทางไปยังเครื่องปลายทาง พบว่าอัลกอริทึมแบบ Deflate ใช้เวลาน้อยที่สุดเมื่อเทียบกับอัลกอริทึมตัวอื่นๆ ถึงแม้จะไม่สามารถลดขนาดของข้อมูลได้มากที่สุดสำหรับไฟล์ประเภท เช่น อัลกอริทึมแบบ LZW ที่ลดขนาดของไฟล์ประเภท TXT, BMP ได้มากที่สุด และ Huffman ที่ลดขนาดของไฟล์ประเภท WAV ได้มากที่สุด เพราะฉะนั้น ไฟล์ประเภท TXT, WAV และ BMP จึงควรใช้อัลกอริทึมแบบ Deflate

ตารางที่ 3 เวลาที่ใช้ในการส่งข้อมูลผ่านเครือข่าย แยกตามอัลกอริทึม

Data Type		TXT	BMP	WAV
Size (KB)		1,480	1,405	1,396
Deflate	Compress size(KB)	699	343	1,292
	Overall Time(s)	1.335	0.736	1.398
	Processing Time(s)	0.176	0.077	0.331
	Compression Time(s)	0.925	0.474	0.78
	Decompression Time(s)	0.167	0.139	0.185
Huffman	Compress size(KB)	773	830	1,246
	Overall Time(s)	3.084	2.343	5.493
	Processing Time(s)	0.189	0.205	0.324
	Compression Time(s)	1.384	1.147	2.983
	Decompression Time(s)	1.436	0.92	2.085
LZ77	Compress size(KB)	1,492	586	1,854
	Overall Time(s)	5.988	2.514	8.708
	Processing Time(s)	0.349	0.132	0.42
	Compression Time(s)	2.603	1.421	4.145
	Decompression Time(s)	2.932	0.893	4.023
LZW	Compress size(KB)	609	296	1,545
	Overall Time(s)	2.392	2.713	5.358
	Processing Time(s)	0.129	0.054	0.335
	Compression Time(s)	1.829	2.415	3.803
	Decompression Time(s)	0.373	0.194	1.117

ในการวัดประสิทธิภาพในการส่งข้อมูลนั้นขึ้นอยู่กับหลายปัจจัยด้วยกัน ถ้าระบบมีความเร็วของเครือข่ายที่สูงมาก การนำข้อมูลที่ผ่านการบีบอัดข้อมูลเข้ามาใช้จะไม่ได้ช่วยเพิ่มประสิทธิภาพการทำงานมากนัก แต่ถ้าระบบที่ทำงานมีความเร็วของเครือข่ายต่ำ การนำการบีบอัดข้อมูลเข้ามาใช้จะช่วยเพิ่มประสิทธิภาพในเรื่องของการลดปริมาณข้อมูล และเวลาที่ใช้ในการส่งข้อมูลลงได้ จากผลการทดลองโดยใช้สมการที่ (9) โดยบีบอัดข้อมูลไฟล์ TXT ด้วยอัลกอริทึมแบบ Deflate ถ้ามีแบนด์วิดท์สูงกว่า 19 เมกะไบต์ต่อวินาที ไม่จำเป็นต้องนำการบีบอัดข้อมูลเข้ามาใช้ ดังแสดงในภาพที่ 5 เมื่อเทียบกับความเร็วเครือข่ายอินเทอร์เน็ตในปัจจุบัน นอกจากนี้ยังต้องพิจารณาถึงขนาดของข้อมูล เพราะข้อมูลที่มีขนาดเล็กมาก ไม่ควรนำการบีบอัดข้อมูลเข้ามาใช้ เพราะนอกจากจะไม่ช่วยลดขนาดของข้อมูลแล้ว อาจจะส่งผลให้เสียเวลาในการประมวลผลที่ต้องใช้ในการบีบอัดข้อมูลและการคลายข้อมูล รวมถึงเวลาที่เพิ่มขึ้นในการส่งข้อมูล เพราะต้องส่งข้อมูลที่มีขนาดใหญ่เพิ่มขึ้น เวลาที่ต้องใช้ในการส่งข้อมูลก็จะเพิ่มมากขึ้นด้วยเช่นกัน



ภาพที่ 5 เวลารวมกับแบนด์วิดท์ของไฟล์ TXT โดยใช้ อัลกอริทึม Deflate

5. สรุปและข้อเสนอแนะ

จากการทดลองเมื่อพิจารณาจาก Compression Ratio, Compression Time, Decompression Time และ Overall Time พบว่าการใช้การบีบอัดข้อมูลโดยใช้อัลกอริทึมแบบ Deflate เหมาะกับข้อมูลเกือบทุกประเภทถึงแม้ว่าจะไม่ใช่อัลกอริทึมที่มีค่า Compression Ratio Compression Time และ Decompression Time น้อยที่สุด แต่อัลกอริทึมแบบ Deflate มีค่า Overall Time น้อยที่สุด ปริมาณข้อมูลที่ใช้ในการส่งแต่ละครั้งมีผลต่อประสิทธิภาพการทำงาน ถ้าข้อมูลมีขนาดเล็กมาก ไม่เหมาะที่จะนำการบีบอัดข้อมูลไปใช้ เพราะอาจจะทำให้ข้อมูลมีขนาดเพิ่มขึ้น ต้องใช้เวลาเพิ่มขึ้นในการส่งข้อมูล นอกจากนี้ยังต้องพิจารณาถึงแบนด์วิดท์ที่ใช้ เพราะถ้าระบบที่ใช้มีแบนด์วิดท์น้อย การนำการบีบอัดข้อมูลเข้ามาใช้จะช่วยลดเวลาในการส่งข้อมูลได้มาก

งานวิจัยฉบับนี้ศึกษาการใช้อัลกอริทึมในการบีบอัดข้อมูลแบบไม่สูญเสียได้แก่ Deflate, Huffman, LZ77 และ LZW มาใช้ในการบีบอัดข้อมูลหลายรูปแบบหลายขนาด การวัดประสิทธิภาพจะอยู่ในรูปของค่าเฉลี่ย นอกจากนี้การทดลองทำอยู่ในเครือข่ายระบบ LAN ผลการทดลองอาจแตกต่าง ถ้าทดลองผ่านเครือข่ายอินเทอร์เน็ต อีกทั้งตัวภาษาที่นำมาใช้ในการพัฒนาคือภาษา JavaScript ซึ่งมีลักษณะการทำงานแบบ Single thread จึงควรมีการพัฒนาตัวไลบรารีด้วยภาษาหรือใช้เทคโนโลยีที่เป็นแบบ Multi Thread เช่น ภาษา Java หรือ JSP Servlet หรือ เทคโนโลยีใช้ Web worker ร่วมกันในการพัฒนาระบบ

เอกสารอ้างอิง

[1] B. Chen and Z. Xu, “A Framework for Browser-Based Multiplayer Online Games Using Webgl and Websocket”, Proc. Int’l Conf. Multimedia Technology(ICMT 11), IEEE Press, pp. 471–474, 2011.

[2] Kuryanovich, Egor, “HTML5 Games Most Wanted Build the Best HTML5 Games”, Apress, pp.213-238, March 26, 2012.

[3] Sayood Khalid, “Introduction to data compression”, The Morgan Kaufmann series in multimedia information and systems, 2006.

[4] Werner Bergmans, “Data compression theory and algorithms”, http://www.maximum compression.com/algoritms.php

[5] Mohammed Al-laham, “Comparative Study between Various Algorithm of Data Compression Techniques”, IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.4, pp.281-291, April 2007.

[6] Mamta Sharma, “Compression Using Huffman Coding”, IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.5, pp.133-141, May 2010.

[7] ดอนสัน ปงผา, “การเข้ารหัสแลการถอดรหัสฮัฟฟ์แมนโดยใช้ค่าสถิติแบบแยกตัวอักษรภาษาไทยและภาษาอังกฤษ”, ปริญญาวิศวกรรมศาสตร์มหาบัณฑิต สาขาวิศวกรรมไฟฟ้า มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ, 2546.

[8] Werner Bergmans, “Large Text Compression Benchmark”, http://mattmahoney.net/dc/text.html

[9] David Salomon, “Data compression”,3nd Edition, Springer, 2004.

[10] นิเวศ จิระประวัติชัย, “การพัฒนาประสิทธิภาพในการบีบอัดข้อมูลตัวอักษรโดยใช้ BWT ร่วมกับ LZW”, National Conference on Computing and Information Technology (NCCIT), pp. 298-303, 2005.

[11] P. Deutsch, “Deflate Compressed Data Format Specification ver.1.3”,http://tools.ietf.org/html/rfc1951, May 1996.

[12] T. Yoshino, “Deflate, WebSocket Per-Frame Compression draft ver.06”, http://tools.ietf.org/html/draft-tyoshino-hybi-websocket-perframe-deflate-06, March 10, 2012.

[13 Yazdanpanah, “A simple lossless preprocessing algorithm for hardware implementation of Deflate data compression”, Electrical Engineering (ICEE), pp. 1-5, May 2011.

[14] Danny Goodman, “JavaScript & DHTML Cookbook”, 2nd Edition, O’RELLY, pp. 23-26, 2007.