การเปรียบเทียบอัลกอริทึมการบีบอัดแบบไม่สูญเสียข้อมูลบนเว็บแอปพลิเคชัน
A Comparison of Lossless Data Compression Algorithms on Web Applications

เศกสิทธิ์ พจมาร[1*] และจารี ทองคำ[1]
Seksit Podchaman[1*] and Jaree Thongkam[1]

Received: January 15, 2020; Revised: April 6, 2020; Accepted: April 22, 2020

บทคัดย่อ

งานวิจัยนี้มีวัตถุประสงค์เพื่อเปรียบเทียบอัลกอริทึมการบีบอัดแบบไม่สูญเสียข้อมูลเพื่อลดขนาดไฟล์ข้อมูลให้เล็กลงก่อนทำการส่งไฟล์ผ่านเครือข่ายอินเทอร์เน็ต และใช้เทคนิคการคลายข้อมูลที่เครื่องผู้ใช้ ซึ่งจะทำให้การส่งข้อมูลผ่านเว็บแอปพลิเคชันมีประสิทธิภาพและมีความรวดเร็วเพิ่มขึ้น กลุ่มไฟล์ข้อมูล 3 กลุ่มได้ถูกนำมาใช้ในการทดลองคือ กลุ่มไฟล์เอกสาร กลุ่มไฟล์รูปภาพ และกลุ่มไฟล์มัลติมีเดีย จำนวน 150 ไฟล์ โดยใช้อัลกอริทึมการบีบอัดแบบไม่สูญเสียข้อมูลจำนวน 5 อัลกอริทึม ได้แก่ Huffman Coding, Deflate, BZip2, LZMA, และ LZ4 ในการวัดประสิทธิภาพการบีบอัดข้อมูล ผู้วิจัยได้ใช้ความเร็วในการบีบอัด ความเร็วในการคลายข้อมูล อัตราส่วนการบีบอัด และเวลารวมทุกกระบวนการ จากการทดลองพบว่าอัลกอริทึม LZ4 มีประสิทธิภาพในการบีบอัดข้อมูล ด้วยความเร็วรวมทุกกระบวนการทำงานดีที่สุดที่อัตราเฉลี่ย 7.6865 วินาที

คำสำคัญ : การบีบอัดข้อมูล; อัลกอริทึมการบีบอัดแบบไม่สูญเสียข้อมูล; อัลกอริทึม LZ4

[1] *คณะวิทยาการสารสนเทศ มหาวิทยาลัยมหาสารคาม*
[1] *Faculty of Informatics, Mahasarakham University*
[*] *Corresponding Author E - mail Address: runtotree2000@gmail.com*

Abstract

The paper aims to compare different lossless data compression algorithms. These are used to reduce the size of data before transmitting it over the Internet. Because the data is preserved, it can be decompressed and restored to its original state. Facilitates fast and efficient data transfer in web applications. In this research, a total of 150 files of 3 different file types are used including text files, image files and multimedia files. Five lossless data compression algorithms including Huffman Coding, Deflate, BZip2, LZMA, and LZ4 are studied and compared. The compression speed, decompression speed, compression rate and total processing time are employed to evaluate the algorithms. The results show that LZ4 algorithm produces the best overall performance, with the average of 7.6865 second.

Keywords: Data Compression; Lossless Data Compression Algorithms; LZ4 Algorithms

บทนำ

ปัจจุบันเทคโนโลยีเว็บแอปพลิเคชันได้เข้ามามีบทบาทต่อชีวิตประจำวันมากขึ้น ซึ่งเป็นผลมาจากที่เทคโนโลยีเว็บแอปพลิเคชันได้มีการพัฒนาและเติบโตขึ้นมาอย่างรวดเร็ว ทำให้การรับ-ส่งข้อมูลมีปริมาณเพิ่มขึ้น [1] ส่งผลให้เครื่องบริการเว็บ (Web Server) มีความเร็วลดลง ไม่สามารถให้บริการได้อย่างมีประสิทธิภาพ หรือบางครั้งก็ไม่สามารถเข้าใช้งานระบบงานได้ เนื่องมาจากเครื่องบริการเว็บต้องใช้หน่วยความจำ (Memory) ซีพียู (CPU) พื้นที่เก็บข้อมูล (Disk) และแบนด์วิดธ์ (Bandwidth) มากเกินไปทำให้ประสิทธิภาพลดลง ปัจจัยหลักอีกประการหนึ่งคือ ปริมาณหรือขนาดของไฟล์ข้อมูลที่ถูกส่งผ่านเว็บแอปพลิเคชันก็มีผลโดยตรงต่อความเร็วในการส่งข้อมูล เมื่อไฟล์ข้อมูลมีขนาดที่ใหญ่ขึ้นเวลาในการรับ-ส่งก็ใช้เวลามากขึ้นไปด้วย [2] ซึ่งในปัจจุบันมีวิธีการต่าง ๆ ในการเพิ่มประสิทธิภาพให้กับการรับ-ส่งข้อมูลเพื่อเพิ่มความเร็วในการส่ง วิธีหนึ่งที่ถูกนำมาใช้ในการแก้ปัญหาเรื่องขนาดไฟล์ คือการบีบอัดไฟล์ให้มีขนาดเล็กลงก่อนส่งไฟล์ เพื่อให้มีความเหมาะสมต่อการจัดเก็บไฟล์ข้อมูล และการส่งข้อมูลผ่านเว็บแอปพลิเคชัน

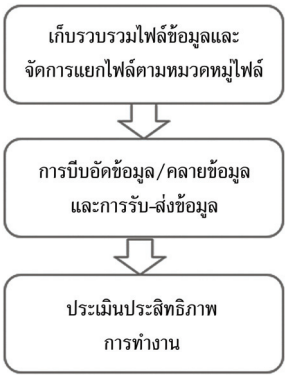
การบีบอัดข้อมูลเป็นที่นิยมในปัจจุบันซึ่งมีนักวิจัยหลายท่านได้พัฒนาอัลกอริทึม เพื่อมาใช้ในการบีบอัดข้อมูล [3] ได้นำวิธีการบีบอัดข้อมูลมาใช้ในเครือข่ายเซนเซอร์ไร้สาย (Wireless Sensor Networks: WSN) เนื่องจากโหนดเซนเซอร์ใช้พลังงานจากแบตเตอรี่ที่มีความจุจำกัด การส่งข้อมูลเป็นกระบวนการที่ต้องใช้พลังงานหลักใน WSN จึงมีการนำเทคนิคการประหยัดพลังงานหลาย ๆ รูปแบบมาใช้ และรูปแบบการบีบอัดข้อมูลเป็นเทคนิคที่ใช้พลังงานอย่างมีประสิทธิภาพ ซึ่งช่วยลดปริมาณข้อมูลที่จะส่งในเครือข่าย ทำให้ประหยัดพลังงานได้มาก จากการทดลองการนำวิธีการบีบอัดข้อมูลมาใช้สามารถประหยัดพลังงานได้มากถึง 87.57% [4] ได้วิจัยเรื่องการบีบอัดแบบไม่สูญเสียข้อมูล โดยใช้การบีบอัดข้อมูลในรูปแบบของไฟล์ JSON สาเหตุมาจากการแลกเปลี่ยนข้อมูลเดิมในระบบอินเทอร์เน็ตในสรรพสิ่ง (Internet of Things) ซึ่งมีการแลกเปลี่ยนข้อมูลจากฝั่งไคลเอ็นต์และฝั่งเซิร์ฟเวอร์เดิมเป็นแบบ XML ซึ่งมีขนาดที่ใหญ่ มาเป็นแบบ JSON พร้อมกับการบีบอัดข้อมูลแบบไม่สูญเสียข้อมูล จำนวน 4 อัลกอริทึมดังนี้ LZMA, GZIP,

BZIP, และ A-LZMA จากผลการวิจัยอัลกอริทึม BZIP มีประสิทธิภาพที่ดีกว่าวิธีอื่น ๆ [5] ได้ศึกษาอัลกอริทึมเพื่อเปรียบเทียบการบีบอัดรูปภาพจำนวน 71 รูปภาพ และแต่ละภาพมีความบิดเบือนจากความเป็นจริงสูง และทำการเลือกอัลกอริทึมมาทดสอบจำนวน 6 อัลกอริทึม ดังนี้ LZMA, PPM, BWT, LZW, Deflate, LZ77, และ Deflate64 จากผลการวิจัยอัลกอริทึม PPM, LZW และ Deflate64 ได้ผลการบีบอัดที่ดีกว่าวิธีอื่น ๆ

ดังนั้นผู้วิจัยจึงได้มีแนวคิดในการแก้ปัญหาเหล่านี้ ด้วยการนำวิธีการบีบอัดข้อมูลมาใช้ในการลดขนาดของข้อมูลให้เล็กลงก่อนที่จะส่ง และในการวิจัยในครั้งนี้ได้มุ่งเน้นการเปรียบเทียบประสิทธิภาพการบีบอัดแบบไม่สูญเสียข้อมูลจำนวน 5 อัลกอริทึม คือ Huffman Coding, LZMA, LZ4, Deflate, และ BZip2 โดยการเปรียบเทียบความเร็วในการบีบอัด ความเร็วในการคลายข้อมูล อัตราส่วนการบีบอัด และความเร็วในการรับ-ส่งไฟล์ข้อมูลผ่านเว็บแอปพลิเคชัน เพื่อนำผลการเปรียบเทียบที่ดีและมีประสิทธิภาพไปพัฒนาปรับปรุงระบบเว็บแอปพลิเคชันในหน่วยงานหรือองค์กรต่อไป

วิธีดำเนินการวิจัย

งานวิจัยนี้ดำเนินการวิจัยโดยมีวัตถุประสงค์ เพื่อเปรียบเทียบประสิทธิภาพอัลกอริทึมบีบอัดแบบไม่สูญเสียข้อมูลที่เหมาะสมกับหมวดไฟล์เอกสาร หมวดไฟล์รูปภาพ และหมวดไฟล์มัลติมีเดีย โดยวิธีการดำเนินงานวิจัยในครั้งนี้ ได้แบ่งขั้นตอนการทำงานออกเป็น 3 ขั้นตอน คือ 1) การเก็บรวบรวมไฟล์ข้อมูลและจัดแยกไฟล์ตามหมวดหมู่ไฟล์ 2) การบีบอัดข้อมูล/คลายข้อมูล และการรับ-ส่งข้อมูล 3) การประเมินประสิทธิภาพการทำงาน ซึ่งสามารถอธิบายรายละเอียดในแต่ละขั้นตอน ได้ดังนี้



รูปที่ 1 ขั้นตอนการทำงาน

1. **การเก็บรวบรวมไฟล์ข้อมูลและจัดแยกไฟล์ข้อมูลตามหมวดหมู่ไฟล์** ในการวิจัยครั้งนี้ผู้วิจัยได้ทำการรวบรวมไฟล์ข้อมูลจำนวน 150 ไฟล์ข้อมูล และจัดแยกไฟล์ตามหมวดหมู่ไฟล์ เพื่อต้องการทดสอบประสิทธิภาพแต่ละอัลกอริทึมให้มีความเหมาะสมกับหมวดหมู่ไฟล์ เนื่องจากไฟล์ข้อมูลในปัจจุบันมีมากมายหลายประเภท ผู้วิจัยจึงได้ทำการจัดแยกกลุ่มไฟล์ข้อมูลเพื่อรวบรวมไฟล์ที่มีการใช้งานอย่างแพร่หลายในปัจจุบันให้เป็นหมวดหมู่ และเพื่อนำผลการทดสอบในแต่ละหมวดหมู่ ไปปรับปรุงพัฒนาระบบการบีบอัดข้อมูลได้อย่างมีประสิทธิภาพ ด้วยการแบ่งหมวดหมู่ไฟล์ จำนวน 3 หมวดหมู่ คือ 1) หมวดไฟล์เอกสาร

จำนวน 60 ไฟล์ข้อมูล ประกอบไปด้วยนามสกุลไฟล์ (PDF, BAK, XLS, XLSX, LOG, TXT, DOC, DOCX) 2) หมวดไฟล์รูปภาพจำนวน 48 ไฟล์ข้อมูล ประกอบไปด้วยนามสกุลไฟล์ (JPG, GIF, TIF, TIFF, PNG, BMP, PSD) 3) หมวดไฟล์มัลติมีเดียจำนวน 42 ไฟล์ข้อมูล ประกอบไปด้วยนามสกุลไฟล์ (MP3, WAV, AVI, MP4, 3GP, WMA) และทำการแบ่งช่วงขนาดไฟล์ 15 ช่วงขนาดไฟล์ ซึ่งไฟล์ที่จะใช้ในการทดลองการบีบอัดข้อมูล จะมีขนาดไฟล์ตั้งแต่ 1 กิโลไบต์ – 100 เมกะไบต์ขึ้นไป

2. การบีบอัดข้อมูล/คลายข้อมูล และการรับ-ส่งข้อมูล การบีบอัดข้อมูล (Data Compression) เป็นการบีบอัด (Compress) เพื่อให้ใช้จำนวนบิตในการจัดเก็บข้อมูลน้อยลงกว่าเดิม การบีบอัดข้อมูลมีประโยชน์ในการลดปริมาณการใช้ทรัพยากร เช่น ประหยัดพื้นที่ของฮาร์ดดิสก์เมื่อจัดเก็บ เป็นต้น ในทางตรงข้ามข้อมูลที่ถูกบีบอัด (Compress) มาแล้วก็ต้องนำมาคลาย (Decompress) หรือถอดรหัส เพื่อให้ได้ข้อมูลเดิมกลับมาก่อนที่จะสามารถนำไปใช้งานได้ วิธีการบีบอัดข้อมูล แบ่งออกเป็น 2 ประเภทได้แก่

- การบีบอัดแบบไม่สูญเสีย (Lossless Data Compression) เป็นการบีบอัดข้อมูลแบบไม่สูญเสีย อาศัยหลักการที่ว่าปกติข้อมูลที่ใช้อยู่มักจะมีข้อมูลที่ซ้ำกันอาจใช้วิธีเก็บตำแหน่งที่ปรากฏคำนั้น ๆ แทน ก็จะสามารถลดความยาวข้อมูลที่จะเก็บได้ จะเห็นว่าการบีบอัดข้อมูลแบบนี้ข้อมูลต้นฉบับกับข้อมูลที่บีบอัดแล้วคลายออกมาจะเหมือนกันไม่ผิดเพี้ยน

- การบีบอัดแบบสูญเสียบางส่วน (Lossy Data Compression) จะมีแนวคิดต่างกันไป โดยใช้หลักการว่าความผิดเพี้ยนของข้อมูลเล็กน้อยเป็นสิ่งที่ยอมรับได้ เช่น ตาของมนุษย์ไม่สามารถแยกความแตกต่างของบางสีได้หมด ก็ไม่จำเป็นต้องเก็บข้อมูลทุกสี จะเห็นตัวอย่างจากไฟล์ประเภท JPG ใช้การบีบอัดข้อมูลแบบเสียบางส่วน จะทำให้ได้ขนาดไฟล์ภาพที่เล็กลงมาก แต่ก็สูญเสียรายละเอียดบางอย่างไป

ในขั้นตอนการบีบอัดและคลายข้อมูลในครั้งนี้ จะใช้อัลกอริทึมการบีบอัดไม่สูญเสียข้อมูล จำนวน 5 อัลกอริทึม ดังนี้

1) อัลกอริทึม Huffman Codes [6] คือการเข้ารหัสแบบ Huffman เป็นการใช้รหัสที่สั้นกว่าแทนสัญลักษณ์ที่เกิดขึ้นบ่อย โดยจะใช้ต้นไม้สองทางในการสร้างรหัสของสัญลักษณ์แต่ละตัวกระจายไปกับข้อมูลของโหนดใบที่เชื่อมอยู่กับต้นไม้สองทาง (Binary Tree) แต่ละโหนดจะมีน้ำหนักกำหนดอยู่ซึ่งก็คือความถี่หรือความน่าจะเป็นของการปรากฏของสัญลักษณ์นั้น ๆ โดยมีวิธีการสร้างต้นไม้ดังนี้ 1) กำหนดให้ทุก ๆ สัญลักษณ์เป็นโหนดใด ๆ 2) หา 2 โหนดใด ๆ ที่มีน้ำหนักน้อยที่สุด 3) สร้างโหนดแม่สำหรับโหนดสองโหนดนี้โดยมีน้ำหนักเท่ากับผลรวมของน้ำหนักของโหนดลูก 4) กำหนดให้โหนดแม่นี้เป็นโหนดใด ๆ และนำโหนดลูกออกจากโหนดใด ๆ 5) โหนดลูกโหนดหนึ่งจะถูกกำหนดให้มีทางผ่านจากโหนดแม่เมื่อทำการถอดรหัสด้วยบิต 0 และอีกโหนดจะถูกกำหนดด้วยบิต 1 จนกว่าจะเหลือโหนดใด ๆ หนึ่งโหนดซึ่งโหนดนี้จะถูกกำหนดให้เป็นรากของต้นไม้นั้นหมายถึง การสร้างต้นไม้เสร็จสิ้น

อัลกอริทึม Huffman Codes เป็นอัลกอริทึมที่สามารถบีบอัดข้อมูลไฟล์ที่เป็น ASCII ได้รวดเร็วมาก และประสิทธิภาพในการบีบอัดนั้นได้ผลดีเนื่องจากเราสามารถทราบ Symbol ทั้งหมดที่เป็นไปได้ (A-Z, a-z, 0-9) ทำให้ไม่ต้องมีการค้นหา Symbol เหมือนกับอัลกอริทึมอื่น ๆ แต่อัลกอริทึมนี้ไม่เหมาะกับการบีบอัดไฟล์ขนาดเล็ก เนื่องจากอาจทำให้ไฟล์ที่บีบอัดมีขนาดไฟล์หลังจากการบีบอัดมีขนาดพื้นที่เพิ่มขึ้น

2) อัลกอริทึม BZip2 [7] คืออัลกอริทึมการบีบอัดข้อมูลโดยใช้วิธีการเบอร์โรวส์-วีเลอร์ (Burrows-Wheeler) และการเข้ารหัสแบบ Huffman โดยปกติการบีบอัดข้อมูลแบบนี้จะดีกว่าการบีบอัดข้อมูลแบบ LZ77/LZ78 และมีประสิทธิภาพใกล้เคียงแบบ PPM ของการบีบอัดข้อมูลโดยใช้สถิติ

แต่เร็วกว่าทางด้านเวลา อัลกอริทึม BZip2 จะประมวลผลข้อมูลโดยแบ่งข้อมูลออกเป็นช่วงบล็อกขนาด 100,000 - 900,000 ไบต์ ขึ้นอยู่กับคำสั่งปกติขนาดของบล็อกจะเท่ากับ 900,000 ไบต์ จากนั้นจะอ่านข้อมูลครั้งละ 5,000 ไบต์จนกว่าจะครบเท่ากับขนาดของบล็อกที่กำหนด และประมวลผลบีบอัดข้อมูลและเขียนข้อมูลที่บีบอัดแล้วลงบนหน่วยเก็บข้อมูล ทำเช่นนี้ต่อไปเรื่อย ๆ จนกว่าบล็อกข้อมูลจะถูกบีบอัดข้อมูลแล้วทั้งหมด

อัลกอริทึมบี BZip2 เป็นอัลกอริทึมบีบอัดข้อมูลโดยใช้หลักสถิติเข้ามาใช้ในการจัดเก็บข้อมูล จึงส่งผลให้การบีบอัดมีอัตราส่วนการบีบอัดข้อมูลที่ดี แต่มีปัญหาในการใช้เวลาในการทำงานมากขึ้นทั้งในการบีบอัดและการคลายข้อมูล

3) อัลกอริทึม Deflate [8] คืออัลกอริทึมที่ทำการรวมกันของอัลกอริทึม LZ77 และการเข้ารหัส Huffman อัลกอริทึม Deflate จะตัดสตริงออกเป็นบล็อกย่อย ๆ บล็อกละ 32 กิโลไบต์ โดยแต่ละบล็อกนั้นจะใช้พื้นที่ 4 ไบต์ในการเก็บระยะทาง (Distance Code) โดยเป็นระยะทางที่จะถูกแทนที่ และสามารถเก็บ Symbol ที่จะถูกแทนที่ได้ 288 Symbols โดยมีหลักการทำงานคือ จะตัดข้อมูลออกเป็นบล็อกย่อย ๆ บล็อกละ 32 กิโลไบต์ก่อน จากนั้นแต่ละบล็อกจะใช้อัลกอริทึม LZ77 ในการหาบล็อกย่อย ๆ ที่มีบิตซ้ำกันในลักษณะที่เรียกว่า Sliding Window แล้วทำการสร้างตารางขึ้นมาเพื่อเก็บ Symbol และหลังจากนั้นจะสร้างตาราง Distance Code เพื่อเก็บระยะทางที่จะเข้าไปแทนที่ Symbol ที่ระบุไว้ในตัวข้อมูล

อัลกอริทึม Deflate นี้ได้ผลดีมากในการบีบอัดข้อมูลที่เป็นลักษณะไบนารี เช่น ซอฟต์แวร์ต่าง ๆ แต่อัลกอริทึมนี้ไม่เหมาะกับการบีบอัดไฟล์ขนาดเล็กและไฟล์เอกสาร เนื่องจากทำงานได้ช้าและต้องเสียพื้นที่ในการสร้าง Header ต่าง ๆ ใน แต่ละบล็อกมากกว่าอัลกอริทึมอื่น ๆ

4) อัลกอริทึม LZ4 [8] คืออัลกอริทึมที่อยู่ในตระกูล LZ77 ใช้ในการบีบอัดข้อมูลซึ่งมุ่งเน้นที่ความเร็วของการบีบอัดข้อมูลและถอดรหัสข้อมูล ค้นหาค่าที่ซ้ำซ้อนกัน โดยอาศัย Hash Table และไม่ค้นหาความเป็นไปได้ทั้งหมดของค่าที่ซ้ำซ้อนกัน และทำการบีบอัดกับข้อมูลในระดับไบต์

ข้อมูลที่ถูกบีบอัด (Compression Block) ของ LZ4 นั้นประกอบด้วย Sequence หลาย ๆ Sequence เรียงต่อกัน ในแต่ละ Sequence นั้น คือชุดของ Literals ซึ่งเป็นส่วนที่ไม่ถูกบีบอัดหรือข้อมูลที่ไม่ซ้ำ ตามด้วยส่วนที่ใช้ในการคัดลอก Match (ส่วนที่ระบุถึงการซ้ำของข้อมูล) ในแต่ละ Sequence จะเริ่มต้นด้วย Token ซึ่งมีขนาด 1 ไบต์ ซึ่งใน 1 ไบต์นี้จะแยกออกมาเป็น 2 ฟิลด์ ซึ่งแต่ละฟิลด์มีขนาด 4 บิต และมีค่าอยู่ระหว่าง 0 - 15 ค่าใน Token นั้นจะใช้สำหรับระบุความยาวและส่วนประกอบต่าง ๆ ใน Sequence นั้น โดย 4 บิตแรก (High-Bits) นั้นจะบอกถึงความยาวของ Literal และ 4 บิตหลัง (Low-Bits) นั้นจะบอกถึงความยาวของ Match ซึ่งความยาวของ Literal และ Match นั้น ถ้าหากว่าเกินกว่า 15 จะมีส่วน Literal Length เพิ่มขึ้นมา เพื่อเก็บค่าความยาวของ Literal โดยแต่ละไบต์จะเก็บค่าได้ 255 เมื่อ Literal Length มีความยาวมากขึ้น ก็สามารถเพิ่มขนาดของ Literal Length ได้โดยไม่ได้จำกัดขนาดไว้ (No Size Limit) จาก Literals คือ Operation ที่ใช้ในการคัดลอก Match ซึ่งเริ่มจาก Offset ซึ่งมีขนาด 2 ไบต์ แบบ Little Endian (ไบต์แรกคือ Low ไบต์ที่ 2 คือ High) ระบุถึงตำแหน่งของ Match ที่ถูกก๊อบปี้มาจาก Literals ซึ่ง 1 หมายถึงตำแหน่งปัจจุบัน ลบ 1 Byte และค่าสูงสุดของ Offset คือ 65535 ในการคำนวณความยาวของ Match จะใช้ฟิลด์ที่สองของ Token ซึ่งเป็น 4 บิตหลัง (Low-Bits) โดยความยาวที่น้อยที่สุดของ Match เท่ากับ 4 ไบต์ เรียกว่า Min Match ดังนั้นค่า 0 จึงหมายถึง 4 ไบต์ และ 15 หมายถึง 19+ ไบต์ ซึ่งสามารถขยายให้มากขึ้นได้เช่นเดียวกันกับ Literal Decoder สามารถสร้างข้อมูลต้นฉบับด้วยข้อมูลจาก Offset และความยาวของ Match

อัลกอริทึม LZ4 มีความเร็วในการบีบอัด 500 เมกะไบต์ต่อวินาทีต่อคอร์ และสามารถปรับความเร็วเพิ่มได้ถึงกิกะบิตต่อวินาที ในโหมดการทำงานของซีพียูที่มีหลายคอร์ และมีความเร็วในการแปลงข้อมูลกลับนั้นสามารถเร็วได้ถึงความเร็วของแรมที่มีในเครื่อง

5) อัลกอริทึม LZMA [7] คืออัลกอริทึมการบีบอัดแบบพจนานุกรมอยู่ในตระกูล LZ77 ซึ่งผลลัพธ์จะถูกเข้ารหัสด้วยตัวเข้ารหัสแบบเป็นช่วง โดยใช้แบบจำลองที่ซับซ้อนเพื่อทำนายความน่าจะเป็นของแต่ละบิต ตัวบีบอัดพจนานุกรมพบการจับคู่โดยใช้โครงสร้างข้อมูลพจนานุกรมที่ซับซ้อน และสร้างสัญลักษณ์การอ้างอิงวลีซึ่งถูกเข้ารหัสครั้งละหนึ่งบิต ในการบีบอัด LZMA สตรีมที่ถูกบีบอัดเป็นสตรีมของบิตเข้ารหัสโดยใช้ Coder Range แบบปรับตัวได้ สตรีมจะถูกแบ่งออกเป็นแพ็กเก็ต แต่ละแพ็กเก็ตจะมีขนาดหนึ่งไบต์ แต่ละส่วนของแต่ละแพ็กเก็ตถูกจำลองด้วยบริบทที่ต่างกัน ดังนั้นการทำนายความน่าจะเป็นสำหรับแต่ละบิตนั้นก็จะมีความสัมพันธ์กับค่าของบิตนั้น ๆ

อัลกอริทึม LZMA มีอัตราการบีบอัดที่ดีกว่าอัลกอริทึม Bzip2 โดยเฉพาะการบีบอัดไฟล์ที่เป็นประเภทข้อความธรรมดา (Plain Text) แต่ข้อเสียของอัลกอริทึมนี้คือต้องการทรัพยากรเครื่องมากทั้งหน่วยประมวลผลกลาง (CPU) และหน่วยความจำหลัก (RAM)

สำหรับขั้นตอนในการทำงานสามารถอธิบายรายละเอียด ได้ดังนี้

2.1 **การบีบอัดข้อมูล** เป็นขั้นตอนเริ่มต้นจากผู้ใช้งานทำการเลือกไฟล์ที่ต้องการอัปโหลด เมื่อไฟล์ถูกอัปโหลดเข้ามาจัดเก็บที่เครื่องบริการเว็บเสร็จเรียบร้อย จากนั้นจะทำการประมวลผลการบีบอัดข้อมูลตามอัลกอริทึมที่ผู้ใช้งานต้องการ เมื่อการประมวลผลการบีบอัดเสร็จสิ้น ไฟล์ข้อมูลจะถูกบันทึกจัดเก็บโดยนามสกุลไฟล์ข้อมูลจะต่อท้ายด้วยอัลกอริทึมที่ทำการบีบอัดเข้ามาด้วยทุกไฟล์ข้อมูล ในขั้นตอนการบีบอัดข้อมูล (Compression) จะประมวลผลการทำงานที่เครื่องบริการเว็บ โดยเครื่องมือที่นำมาใช้ในการทดสอบประสิทธิภาพการทำงานในขั้นตอนนี้ มีคุณสมบัติดังนี้

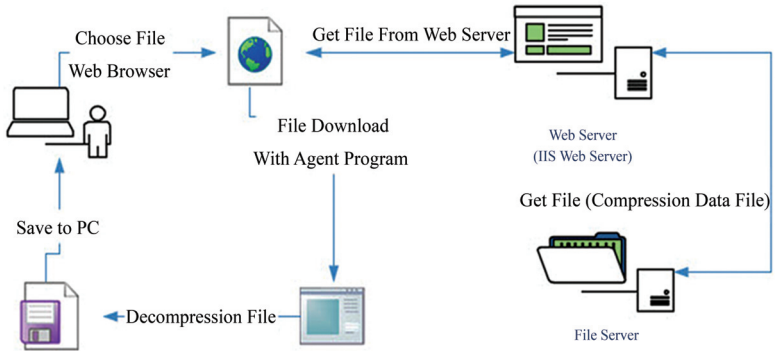
- ฮาร์ดแวร์ เครื่องคอมพิวเตอร์ Notebook ประกอบไปด้วย ซีพียู Intel Core i5-2410M @ 2.30 GHz หน่วยความจำหลัก 8 GB ฮาร์ดดิสก์ขนาด 500 GB
- ซอฟต์แวร์ ระบบปฏิบัติการ Windows 10 Professional 64 Bit, Visual Studio.Net 2019, พัฒนาด้วยภาษา ASP.NET Core MVC, Internet Information Services (IIS) 7.5, Browser Chrome Version 14

2.2 **การคลายข้อมูล** เป็นขั้นตอนในการคลายข้อมูลที่มีการบีบอัดมาแล้ว โดยเริ่มต้นการทำงานหลังจากโปรแกรมดาวน์โหลดไฟล์ จากเครื่องบริการเว็บลงมาจัดเก็บที่เครื่องผู้ใช้งานก่อน จากนั้นโปรแกรมทำการคลายการบีบอัดข้อมูล โดยการตรวจสอบนามสกุลไฟล์ก่อน เนื่องจากนามสกุลไฟล์เอกสารที่ทำการบีบอัดข้อมูลมาแล้วจะบ่งบอกชนิดของอัลกอริทึมที่ใช้ในการบีบอัด เมื่อบีบอัดด้วยอัลกอริทึมใด ๆ ต้องทำการคลายการบีบอัดด้วยอัลกอริทึมนั้น ๆ เสมอ ในขั้นตอนการคลายข้อมูลจะประมวลผลการทำงานที่เครื่องผู้ใช้งาน (Client) การประมวลผลการทำงานที่เครื่องผู้ใช้งานจะทำให้มีประสิทธิภาพมากกว่าการประมวลผลการทำงานที่เครื่องบริการเว็บ เนื่องจากในกรณีที่มีผู้ใช้งานจำนวนมากต้องการดาวน์โหลดไฟล์เอกสารไปใช้งานพร้อม ๆ กัน ส่งผลให้เครื่องบริการเว็บต้องประมวลผลข้อมูลในหลาย ๆ ขั้นตอน ซึ่งอาจจะส่งผลให้ประสิทธิภาพการทำงานลดลง ดังนั้นในการวิจัยครั้งนี้จึงได้เลือกใช้เทคนิคในการคลายข้อมูลที่เครื่องผู้ใช้งานเพราะจะทำให้การทำงานของเครื่องบริการเว็บ มีประสิทธิภาพมากยิ่งขึ้น โดยเครื่องมือที่นำมาใช้ในการทดสอบประสิทธิภาพการทำงานในขั้นตอนนี้มีคุณสมบัติดังนี้

- ฮาร์ดแวร์ เครื่องคอมพิวเตอร์ Notebook ประกอบไปด้วย ซีพียู Intel Core i3-2410M @ 2.30 GHz หน่วยความจำหลัก 4 GB ฮาร์ดดิสก์ขนาด 500 GB

- ซอฟต์แวร์ ระบบปฏิบัติการ Windows 10 Professional 32 Bit โปรแกรมสำหรับโหลดไฟล์และการคลายข้อมูล (API) พัฒนาด้วย Visual Studio.Net 2019 ด้วยภาษา C#, Browser Chrome Version 14

2.3 **การรับ-ส่งไฟล์ข้อมูลผ่านเว็บแอปพลิเคชัน** เป็นขั้นตอนในการถ่ายโอนข้อมูลจากเครื่องบริการเว็บไปยังเครื่องผู้ใช้งาน ในการถ่ายโอนข้อมูลจะออกแบบโปรแกรมเป็น Web Client โดยสามารถแสดงรายละเอียดขั้นตอนการทำงานได้ ดังรูปที่ 2



รูปที่ 2 ขั้นตอนการรับ-ส่งไฟล์ข้อมูลผ่านเว็บแอปพลิเคชัน

จากรูปที่ 2 แสดงขั้นตอนในการถ่ายโอนข้อมูลจากเว็บแอปพลิเคชันไปที่เครื่องผู้ใช้งาน โดยเริ่มต้นการทำงานจากผู้ใช้งานทำการคลิกเลือกไฟล์ข้อมูลที่ต้องการดาวน์โหลด จากรายการไฟล์ที่แสดงบนเว็บแอปพลิเคชัน จากนั้นโปรแกรมจะทำการส่งค่าพารามิเตอร์ (URL) ให้กับโปรแกรมดาวน์โหลดไฟล์เอกสาร (API) เมื่อโปรแกรมทำการดาวน์โหลดไฟล์จากเครื่องบริการเว็บลงมาจัดเก็บที่เครื่องผู้ใช้งานเสร็จเรียบร้อย โปรแกรมจะทำการคลายการบีบอัดข้อมูล และบันทึกจัดเก็บไฟล์ข้อมูลที่เครื่องผู้ใช้งาน

3. **การประเมินประสิทธิภาพการทำงาน** สำหรับงานวิจัยนี้ได้ทำการแบ่งแยกขั้นตอนการทำงานออกเป็น 2 กระบวนการ คือ 1) กระบวนการอัปโหลดไฟล์และบีบอัดข้อมูล และวัดประสิทธิภาพการบีบอัด 2) กระบวนการดาวน์โหลดไฟล์ การคลายข้อมูล และการรับ-ส่งไฟล์ผ่านเว็บแอปพลิเคชัน และวัดประสิทธิภาพการทำงาน

ในวิจัยฉบับนี้จะวัดประสิทธิภาพการทำงานจำนวน 4 ตัวชี้วัด ดังนี้

3.1 **การวัดความเร็วการบีบอัดข้อมูล (Compression Speed)** [9] - [10] คือการวัดความเร็วในการบีบอัดข้อมูลซึ่งเป็นอัตราส่วนระหว่างปริมาณข้อมูลและเวลาที่ใช้ในการบีบอัดข้อมูล มีหน่วยเป็นไบต์ต่อวินาทีดังสมการที่ (1)

$$\text{Compression Speed} = \frac{\text{File Size}}{\text{Compression Time}} \quad (1)$$

- File Size คือ ขนาดข้อมูลก่อนการบีบอัด มีหน่วยเป็นไบต์
- Compression Time คือ เวลาที่ใช้ในการบีบอัดข้อมูล มีหน่วยเป็นวินาที

3.2 **การวัดความเร็วการคลายข้อมูล (Decompression Speed)** [11] - [12] คือการวัดความเร็วในการคลายข้อมูล ซึ่งเป็นอัตราส่วนระหว่างปริมาณข้อมูลและเวลาที่ใช้ในการคลายข้อมูลมีหน่วยเป็นไบต์ต่อวินาที ดังสมการที่ (2)

$$\text{Decompression Speed} = \frac{\text{File Size1}}{\text{Compression Time}} \tag{2}$$

- File Size1 คือ ขนาดข้อมูลหลังการบีบอัด มีหน่วยเป็นไบต์
- Decompression Time คือ เวลาที่ใช้ในการคลายข้อมูล มีหน่วยเป็นวินาที

3.3 **การวัดอัตราส่วนการบีบอัดข้อมูล (Compression Ratio)** [13] ซึ่งแสดงถึงจำนวนข้อมูลที่เหลือหลังจากการบีบอัด ความสัมพันธ์ดังกล่าวจะอยู่ในรูปของร้อยละขนาดของข้อมูลดั้งเดิม (File Size2) และขนาดของข้อมูลที่ถูกบีบอัดแล้ว (Compression Size) ดังสมการที่ (3)

$$\text{Compression Ratio} = \frac{\text{Compression Size}}{\text{File Size2}} * 100 \tag{3}$$

3.4 **เวลารวมทุกกระบวนการ (Overall Time)** [6], [14] คือเวลารวมของระบบที่ใช้ในการส่งข้อมูลจากเครื่องต้นทางไปยังเครื่องปลายทาง ดังสมการที่ (4)

$$\begin{aligned}\text{Overall Times} = &(\text{Compression Time} + \text{Processing Time}) + \\ &(\text{Decompression Time} + \text{Transmission Time})\end{aligned} \tag{4}$$

- Processing Time คือ เวลาที่เครื่องบริการเว็บใช้ในการประมวลผล
- Transmission Time คือ เวลาที่ใช้ในการส่งข้อมูลจากเครื่องต้นทางไปยังปลายทาง

ผลการวิจัย

จากการออกแบบและพัฒนาเว็บแอปพลิเคชันและทดสอบวัดประสิทธิภาพการบีบอัดข้อมูล โดยใช้ไฟล์ข้อมูลในการทดสอบทั้งหมด 150 ไฟล์ข้อมูล และจัดแยกไฟล์ตามหมวดหมู่ไฟล์ ดังนี้ 1) หมวดไฟล์เอกสาร จำนวน 60 ไฟล์ข้อมูล 2) หมวดไฟล์รูปภาพจำนวน 48 ไฟล์ข้อมูล 3) หมวดไฟล์มัลติมีเดียจำนวน 42 ไฟล์ข้อมูล จากนั้นทำการทดสอบเพื่อวัดประสิทธิภาพการทำงาน ด้วยการวัดความเร็วในการบีบอัด ความเร็วในการคลายข้อมูล อัตราส่วนการบีบอัด และเวลารวมการทำงานทุกกระบวนการ โดยสามารถแสดงผลการทดสอบได้ดังนี้

1. **ผลการเปรียบเทียบความเร็วการบีบอัด (Compression Speed)**

เป็นการวัดหาความเร็วที่ใช้ในการบีบอัดข้อมูล จากเริ่มต้นจนสิ้นสุดการบีบอัดข้อมูล ซึ่งผลลัพธ์ที่ได้จะมีหน่วยเป็นไบต์ต่อวินาที โดยสามารถแสดงผลการทดสอบได้ดังตารางที่ 1

ตารางที่ 1 ความเร็วการบีบอัดข้อมูล

อัลกอริทึม	ความเร็วการบีบอัดข้อมูล (Byte / S)			
	Multimedia	**Images**	**Data**	**ค่าเฉลี่ยรวม**
LZ4	133,034,743.74	65,835,663.25	42,744,585.52	80,538,330.84
Deflate	23,073,507.59	32,384,160.89	20,504,606.93	25,320,758.47
Huffman Coding	2,371,039.79	2,303,321.63	2,120,797.02	2,265,052.81
BZip2	1,023,002.04	1,608,261.90	1,408,694.51	1,346,652.82
LZMA	830,718.02	840,928.95	783,964.60	818,537.19

จากตารางที่ 1 แสดงผลลัพธ์การวัดความเร็วในการบีบอัดไฟล์ข้อมูล จากผลการทดลองพบว่า อัลกอริทึม LZ4 สามารถทำความเร็วในการบีบอัดข้อมูลดีที่สุดที่ความเร็วเฉลี่ย 80,538,330.84 ไบต์ต่อวินาที และยังพบว่าอัลกอริทึม LZ4 สามารถทำความเร็วได้ดีทั้ง 3 หมวดหมู่ไฟล์ โดยหมวดไฟล์มัลติมีเดีย มีความเร็วเฉลี่ย 133,034,743.74 ไบต์ต่อวินาที หมวดไฟล์เอกสารความเร็วเฉลี่ย 65,835,663.25 ไบต์ต่อวินาที หมวดไฟล์รูปภาพความเร็วเฉลี่ย 42,744,585.52 ไบต์ต่อวินาที จะเห็นได้ว่า อัลกอริทึม LZ4 มีความเร็วในการบีบอัดที่สูงกว่าอัลกอริทึมอื่น ๆ เป็นอย่างมาก โดยมีค่าเฉลี่ยมากกว่าอัลกอริทึม Deflate ที่มีลำดับถัดมาที่ความเร็วเฉลี่ย 25,320,758.47 ไบต์ต่อวินาที

2. **ผลการเปรียบเทียบความเร็วการคลายข้อมูล (Decompression Speed)**

เป็นการวัดหาความเร็วที่ใช้ในการคลายข้อมูล จากเริ่มต้นจนสิ้นสุดการคลายข้อมูล ซึ่งผลลัพธ์ที่ได้จะมีหน่วยเป็นไบต์ต่อวินาที โดยสามารถแสดงผลการทดลองได้ดังตารางที่ 2

ตารางที่ 2 ความเร็วการคลายข้อมูล

อัลกอริทึม	ความเร็วการคลายข้อมูล (Byte / S)			
	Multimedia	**Images**	**Data**	**ค่าเฉลี่ยรวม**
LZ4	763,951,843.31	104,543,522.90	297,746,985.22	388,747,450.48
Deflate	162,223,009.27	51,296,907.76	87,952,600.45	100,490,839.16
BZip2	3,541,681.15	1,291,306.45	2,569,048.82	2,467,345.47
LZMA	3,022,228.02	1,607,089.16	2,382,631.09	2,337,316.09
Huffman Coding	161,432.40	83,929.44	90,452.61	111,938.15

จากตารางที่ 2 แสดงผลลัพธ์การวัดความเร็วในการคลายข้อมูล จากผลการทดลองพบว่า อัลกอริทึม LZ4 สามารถทำความเร็วในการคลายข้อมูลดีที่สุดที่ความเร็วเฉลี่ย 388,747,450.48 ไบต์ต่อวินาที และยังพบว่าอัลกอริทึม LZ4 สามารถทำความเร็วได้ดีทั้ง 3 หมวดหมู่ไฟล์ โดยหมวดไฟล์มัลติมีเดีย มีความเร็วเฉลี่ย 763,951,843.31 ไบต์ต่อวินาที หมวดไฟล์เอกสารความเร็วเฉลี่ย 297,746,985.22 ไบต์ต่อวินาที หมวดไฟล์รูปภาพความเร็วเฉลี่ย 104,543,522.90 ไบต์ต่อวินาที และจากผลการทดลองในตารางที่ 1 และ 2 จะเห็นได้ว่า อัลกอริทึม LZ4 จะมีความเร็วในการบีบอัดและคลายข้อมูลดีที่สุด จากผลการทดลอง

ความเร็วในการคลายข้อมูล อัลกอริทึม LZ4 มีค่าเฉลี่ยมากกว่าอัลกอริทึม Deflate ที่มีลำดับถัดมาที่ความเร็วเฉลี่ย 100,490,839.16 ไบต์ต่อวินาที เนื่องจากอัลกอริทึม LZ4 มีความเร็วในการบีบอัดและคลายข้อมูลที่สูง ยิ่งไฟล์ที่มีขนาดใหญ่ตั้งแต่ 10 เมกะไบต์ขึ้นไป การทำงานของอัลกอริทึมก็จะมีความรวดเร็วมากยิ่งขึ้น อีกทั้งเครื่องที่ใช้ในการทดลองมีซีพียูที่เป็นแบบมัลติคอร์ อัลกอริทึม LZ4 สามารถทำความเร็วได้อีกเท่าตัว จึงส่งผลทำให้ผลการทดลองพบว่า อัลกอริทึม LZ4 มีความเร็วเฉลี่ยที่ดีที่สุดทั้งในด้านการบีบอัดและการคลายข้อมูล จากผลลัพธ์ในการวัดความเร็วในการบีบอัดและการคลายข้อมูลมีความสอดคล้องกับผลการวิจัยเรื่อง A Critical Evaluation of Lossless Algorithm and Its Applications [15] ซึ่งได้ทำการทดสอบเปรียบเทียบการบีบอัดแบบไม่สูญเสียข้อมูล โดยการใช้งานการบีบอัดข้อมูลในระบบงานซึ่งมีการแลกเปลี่ยนข้อมูลกันผ่านโทรศัพท์มือถือ ซึ่งได้ผลลัพธ์เช่นเดียวกับบทความวิจัยฉบับนี้

3. **ผลการเปรียบเทียบอัตราส่วนการบีบอัดข้อมูล (Compression Ratio)**

เป็นการวัดหาอัตราส่วนพื้นที่ขนาดไฟล์คงเหลือหลังจากผ่านขั้นตอนการบีบอัดไฟล์ข้อมูลมาแล้ว ซึ่งผลลัพธ์การเปรียบเทียบอัตราส่วนการบีบอัดไฟล์ข้อมูลที่ได้จะมีหน่วยเป็นร้อยละของขนาดไฟล์ข้อมูลดังเดิม โดยสามารถสรุปผลการทดลองได้ดังตารางที่ 3

ตารางที่ 3 อัตราส่วนการบีบอัดข้อมูล

อัลกอริทึม	อัตราส่วนการบีบอัดข้อมูล (%)			
	Data	**Images**	**Multimedia**	**ค่าเฉลี่ยรวม**
LZMA	73.05	71.03	94.72	79.60
BZip2	76.47	76.51	94.82	82.60
Deflate	76.21	77.20	96.81	83.41
LZ4	80.01	85.99	98.77	88.26
Huffman Coding	111.43	109.10	98.65	106.39

จากตารางที่ 3 แสดงข้อมูลอัตราส่วนการบีบอัดข้อมูล ผลการทดลองเปรียบเทียบอัตราส่วนการบีบอัดข้อมูลระหว่างอัลกอริทึม LZMA, BZip2, Deflate, LZ4, และ Huffman Coding พบว่าอัลกอริทึม LZMA มีอัตราส่วนการบีบอัดที่น้อยที่สุดใน 3 หมวดไฟล์ โดยเฉลี่ย 79.60% อัลกอริทึม BZip2 โดยเฉลี่ย 82.60% อัลกอริทึม Deflate โดยเฉลี่ย 83.41% อัลกอริทึม LZ4 โดยเฉลี่ย 88.26% และอัลกอริทึม Huffman Coding โดยเฉลี่ย 106.39% และพบว่าหมวดไฟล์ที่มีอัตราส่วนการบีบอัดที่น้อยที่สุดคือ หมวดไฟล์รูปภาพโดยมีค่าเฉลี่ย 71.03% ซึ่งหมายความว่าเมื่อนำไฟล์เอกสารใด ๆ มาผ่านกระบวนการบีบอัดข้อมูลจะทำให้มีขนาดที่เล็กลง ยิ่งมีอัตราส่วนการบีบอัดที่น้อยก็จะทำให้ขนาดไฟล์เล็กลงไปเรื่อย ๆ ดังนั้นหากพิจารณาในด้านการลดขนาดของไฟล์ อัลกอริทึม LZMA จึงมีประสิทธิภาพดีที่สุด จากผลการทดลองยังพบว่าอัลกอริทึม Huffman Coding มีอัตราส่วนการบีบอัดที่มีค่ามากที่สุดโดยค่าเฉลี่ย 106.39 % ทั้งในหมวดไฟล์ข้อมูล หมวดไฟล์รูปภาพ และหมวดไฟล์มัลติมีเดีย และเมื่อไฟล์มีขนาดใหญ่ถ้าทำการบีบอัดไฟล์จะพบว่าขนาดไฟล์หลังจากบีบอัดจะมีขนาดที่ใหญ่ขึ้นกว่าไฟล์ต้นฉบับ หรือขนาดไฟล์ลดลง

แต่ไม่มากนัก เพราะฉะนั้นอัลกอริทึม Huffman Coding จึงไม่ควรนำมาบีบอัดข้อมูลที่เน้นลดขนาดไฟล์ จากผลลัพธ์ในการวัดอัตราส่วนการบีบอัดข้อมูลมีความสอดคล้องกับผลการวิจัยเรื่อง The Application of LZMA Algorithm in ISCS Based on Pretreatment [4] ซึ่งได้นำการบีบอัดแบบไม่สูญเสียข้อมูล โดยใช้การบีบอัดข้อมูลในระบบ IoT (Internet of Things) ซึ่งมีการแลกเปลี่ยนข้อมูลจากฝั่งไคลเอ็นต์และฝั่งเซิร์ฟเวอร์ของการขนส่งทางรถไฟเดิมเป็นแบบ XML มาเป็น JSON ซึ่งได้ผลลัพธ์เช่นเดียวกับบทความวิจัยฉบับนี้

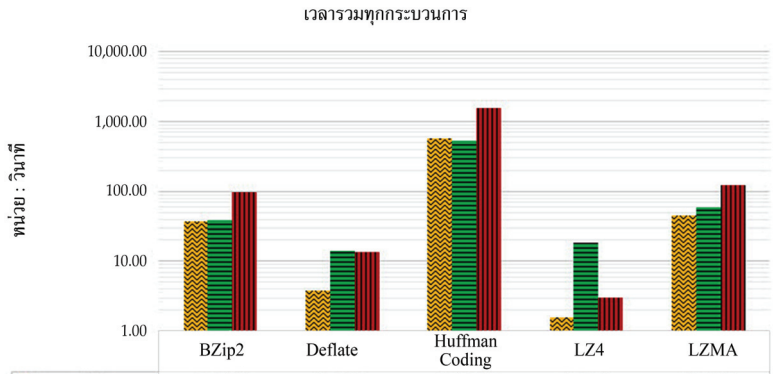
ตารางที่ 4 อัตราส่วนการบีบอัดข้อมูลแยกตามช่วงขนาดไฟล์ข้อมูล

ช่วงขนาดไฟล์	อัตราส่วนการบีบอัดข้อมูล (%)				
	LZMA	**BZip2**	**Deflate**	**LZ4**	**Huffman Coding**
1 - 5 KB	60.09	102.31	60.86	69.53	379.54
5 - 10 KB	74.03	81.35	74.21	79.42	124.56
10 - 50 KB	48.05	53.49	49.87	53.26	78.21
50 - 200 KB	76.73	79.76	77.75	80.30	97.37
200 - 500 KB	79.74	82.01	81.65	84.17	95.40
500 - 800 KB	95.06	96.37	95.77	96.90	100.23
800 - 1000 KB	76.94	78.37	80.47	84.46	94.43
1 - 5 MB	70.31	71.71	73.92	77.92	89.35
5 - 10 MB	75.86	79.83	82.16	86.79	91.82
10 - 50 MB	85.14	85.56	88.00	92.02	97.26
50 - 60 MB	89.37	88.83	92.65	98.13	98.65
60 - 70 MB	83.28	83.22	90.29	97.85	98.32
70 - 80 MB	89.81	85.39	91.65	98.28	98.25
80 - 100 MB	79.91	80.73	86.57	91.98	93.53
100 MB ขึ้นไป	83.61	83.47	89.55	97.42	98.17
ค่าเฉลี่ยรวม	77.86	82.16	81.02	85.90	115.67

จากผลการทดลองในตารางที่ 4 แสดงอัตราส่วนการบีบอัดข้อมูลโดยทำการแบ่งแยกตามขนาดช่วงไฟล์ 15 ช่วงไฟล์ พบว่าไฟล์ขนาด 1 กิโลไบต์ - 50 เมกะไบต์ อัลกอริทึม LZMA มีการบีบอัดข้อมูลได้มากกว่าอัลกอริทึมอื่น ๆ อัลกอริทึม LZMA สามารถบีบอัดไฟล์ข้อมูลที่มีเนื้อหาเป็นข้อความธรรมดา (Plain Text) ได้ดี โดยไฟล์ที่ใช้ในการทดลองมีเนื้อหาใกล้เคียงกัน จึงทำให้การบีบอัดด้วยอัลกอริทึม LZMA จึงมีขนาดเล็กที่สุด เมื่อขนาดของไฟล์มีขนาดมากกว่า 50 เมกะไบต์ กลับพบว่าอัลกอริทึม BZip2 มีการบีบอัดข้อมูลได้มากกว่าอัลกอริทึมอื่น ๆ แต่เมื่อดูค่าเฉลี่ยรวมที่มีอัตราส่วนการบีบอัดดีที่สุด เป็นอัลกอริทึม LZMA ที่มีค่าเฉลี่ยรวมดีที่สุดที่ 77.86%

4. **ผลการเปรียบเทียบประสิทธิภาพเวลารวมทุกกระบวนการ (Overall Time)**

การวัดเวลารวมทุกกระบวนการทำงาน (Overall Time) โดยจะวัดประสิทธิภาพการทำงานตั้งแต่เริ่มต้นการบีบอัดข้อมูล การรับ-ส่งข้อมูลส่งผ่านเว็บแอปพลิเคชัน จนถึงขั้นตอนการคลายข้อมูล



	BZip2	Deflate	Huffman Coding	LZ4	LZMA
Group_Data	37.3420	3.7885	574.3526	1.5360	45.5844
Group_Images	39.0747	13.8792	528.8272	18.4899	58.3489
Group_Multimedia	97.9539	13.4263	1,567.8997	3.0337	121.0335

รูปที่ 3 ความเร็วรวมทุกกระบวนการเริ่มบีบอัดข้อมูลจนถึงกระบวนการคลายข้อมูล

จากผลการทดลองในรูปที่ 3 พบว่า อัลกอริทึม LZ4 มีค่าเฉลี่ย Overall Time น้อยที่สุด โดยมีค่าเฉลี่ย 7.6865 วินาที และพบว่ามีค่าเฉลี่ยที่ต่ำในหมวดไฟล์ข้อมูลโดยเฉลี่ย 1.5360 วินาที และหมวดไฟล์มัลติมีเดียโดยเฉลี่ย 3.0337 วินาที แต่พบว่าในหมวดไฟล์รูปภาพ อัลกอริทึมที่มีค่าเฉลี่ยที่น้อยที่สุด คือ อัลกอริทึม Deflate โดยเฉลี่ย 13.8792 วินาที เมื่อดูค่า Overall Time โดยเฉลี่ยรวมทั้งหมด กลับพบว่า อัลกอริทึม LZ4 มีค่าเฉลี่ย Overall Time น้อยที่สุด ทำให้การส่งข้อมูลผ่านเว็บแอปพลิเคชันมีประสิทธิภาพและมีความเร็วเพิ่มขึ้น จากตารางที่ 3 จะพบว่าอัลกอริทึม LZMA จะมีอัตราส่วนการบีบอัดที่ดีที่สุด แต่เมื่อเทียบประสิทธิภาพรวมทุกกระบวนการ อัลกอริทึม LZMA ยังเป็นรองอัลกอริทึม LZ4 อยู่มาก จากรูปที่ 3 จะพบว่าอัตราส่วนเฉลี่ยค่า Overall Time อัลกอริทึม LZ4 จะมีค่าเฉลี่ยน้อยกว่า อัลกอริทึม LZMA โดยเฉลี่ย 63.4977 วินาที จากผลลัพธ์ในการทดสอบยังมีความสอดคล้องกับผลการวิจัยเรื่อง Data Compression Device Based on Modified LZ4 Algorithm [13] ซึ่งได้นำการบีบอัดข้อมูลมาใช้การเพิ่มประสิทธิภาพการจัดเก็บและอายุการใช้งานใน Solid State Drive (SSD) ทำให้มีอัตราการอ่านเขียนข้อมูลมีประสิทธิภาพมากขึ้น ซึ่งได้ผลลัพธ์เช่นเดียวกับบทความวิจัยฉบับนี้

การอภิปรายผล

งานวิจัยนี้มีวัตถุประสงค์เพื่อเปรียบเทียบอัลกอริทึมการบีบอัดแบบไม่สูญเสียข้อมูลเพื่อลดขนาดไฟล์ข้อมูลให้เล็กลงก่อนทำการส่งไฟล์ผ่านเครือข่ายอินเทอร์เน็ต โดยใช้อัลกอริทึมการบีบอัดจำนวน 5 อัลกอริทึม ได้แก่ Huffman Coding, Deflate, BZip2, LZMA, และ LZ4 จะเห็นได้ว่าอัลกอริทึม Huffman Coding มีรูปแบบการบีบอัดเป็นการนำตัวอักษรมาไล่เรียงเป็นแบบต้นไม้ (Binary Tree) เมื่อไฟล์มีขนาดที่ค่อนข้างใหญ่ เวลาที่ใช้ก็จะเพิ่มขึ้นอย่างมาก จึงทำให้มีความเร็วลดลง อัลกอริทึม BZip2 และ

LZMA เป็นอัลกอริทึมที่ถูกพัฒนามาจากอัลกอริทึม LZ77 ซึ่งมีรูปแบบเป็นการระบุสัญลักษณ์ที่ซ้ำกันแทนด้วยชุดตัวอักษรที่เป็นสัญลักษณ์และตัวเลข ทำให้พื้นที่ในการจัดเก็บหลังการบีบอัดมีขนาดที่เล็กลง จึงทำให้อัลกอริทึม LZMA และ BZip2 มีอัตราส่วนในการบีบอัดที่ดีที่สุดตามลำดับ ยิ่งชุดข้อมูลมีเนื้อหาที่ใกล้เคียงกันอัตราส่วนการบีบอัดก็จะสูงตามไปด้วย อัลกอริทึม Deflate เป็นอัลกอริทึมที่รวมการทำงานของอัลกอริทึม Huffman Coding กับอัลกอริทึม LZ77 โดยมีขนาดบล็อกในการจัดเก็บข้อมูลเพิ่มขึ้น จึงทำให้มีความเร็วในการบีบอัดและคลายข้อมูลที่เร็วกว่าอัลกอริทึม Huffman Coding, BZip2, และ LZMA แต่พบว่าอัตราส่วนการบีบอัดยังเป็นรองอัลกอริทึม LZMA และ BZip2 อัลกอรึม LZ4 เป็นอัลกอริทึมที่มุ่งเน้นความเร็วของการบีบอัดและถอดรหัสข้อมูล โดยการค้นหาคำที่มีเนื้อหาซ้ำซ้อนกันและจะใช้ Hash Table เข้ามาช่วยในการจัดเก็บและค้นหาข้อมูล ทำให้มีความรวดเร็วกว่ารูปแบบต้นไม้ ส่งผลให้อัลกอริทึม LZ4 มีความเร็วในการบีบอัด และความเร็วในการคลายข้อมูลได้ดีที่สุด

สรุปผลการทดลอง

จากการทดลองเมื่อพิจารณาจาก Compression Ratio, Compression Time, Decompress Time, และ Overall Time พบว่าการใช้การบีบอัดข้อมูลโดยใช้อัลกอริทึมแบบ LZ4 ได้ผลลัพธ์ที่ดีในด้านความเร็วในการบีบอัด ความเร็วในการคลายข้อมูล และความเร็วรวมทุกกระบวนการที่มีความเร็วทั้งในหมวดไฟล์เอกสาร ไฟล์รูปภาพ และไฟล์มัลติมีเดีย จึงเหมาะสำหรับการนำมาใช้ในการบีบอัดข้อมูลและรับ-ส่งข้อมูลผ่านเว็บแอปพลิเคชัน อัลกอริทึม Huffman Coding ไม่ควรนำมาพิจารณาในการบีบอัดข้อมูล จากผลการทดลองพบว่าประสิทธิภาพในด้านความเร็วของการบีบอัดและการคลายข้อมูลใช้เวลานาน และเมื่อไฟล์มีขนาดใหญ่ตั้งแต่ 10 เมกะไบต์ขึ้นไป เวลาที่ใช้ในการบีบอัดและคลายข้อมูลก็จะมากกว่าอัลกอริทึมอื่น ๆ อย่างมาก ถ้าหากพิจารณาอัลกอริทึมบีบอัดข้อมูลที่เน้นเรื่องการลดขนาดพื้นที่การจัดเก็บไฟล์ พื้นที่ในการสำรองไฟล์ อัลกอริทึม LZMA จะเหมาะสมกับวิธีการนี้เป็นที่สุด

References

[1] Silawong, C. and Anusasamornkul, T. (2013). A Comparative Study of Compression Algorithms for Each Data Type. In **2013 International Computer Science and Engineering Conference (ICSEC 2013)**. pp. 435-440

[2] Pranveenit, S. and Chanchio, K. (2016). The Performance Analysis of Compression Techniques for Thread-Based Live Migration of Virtual Machine. In **ICMSIT 2016: International Conference on Management Science, Innovation, and Technology**. Faculty of Management Science, Suan Sunandha Rajabhat University. pp. 103-114

[3] Uthayakumar, J., Vengattaraman, T., and Dhavachelvan, P. (2019). A New Lossless Neighborhood Indexing Sequence (NIS) Algorithm for Data Compression in Wireless Sensor Networks. **Ad Hoc Networks**. Vol. 83, pp. 149-157. DOI: 10.1016/j.adhoc.2018.09.009

[4] Xudong, X. and Yiran, L. (2018). The Application of LZMA Algorithm in ISCS Based on Pretreatment. In **2018 5th International Conference on Systems and Informatics (ICSAI)**. pp. 521-525. DOI: 10.1109/ICSAI.2018.8599491

[5] Uthayakumar, J. and Vengattaraman, T. (2018). Performance Evaluation of Lossless Compression Techniques: An Application of Satellite Images. In **2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)**. pp. 750-754. DOI: 10.1109/ICECA.2018.8474759

[6] Arshad, R., Saleem, A., and Khan, D. (2016). Performance Comparison of Huffman Coding and Double Huffman Coding. In **2016 Sixth International Conference on Innovative Computing Technology (INTECH)**. pp. 361-364. DOI: 10.1109/INTECH.2016.7845058

[7] Tariq, Z. B., Arshad, N., and Nabeel, M. (2015). Enhanced LZMA and BZIP2 for Improved Energy Data Compression. In **2015 International Conference on Smart Cities and Green ICT Systems (SMARTGREENS)**. pp. 1-8. DOI:10.5220/0005454202560263

[8] Harnik, D., Khaitzin, E., Sotnikov, D., and Taharlev, S. (2014). A Fast Implementation of Deflate. **Data Compression Conference**. pp. 223-232. DOI:10.1109/DCC.2014.66

[9] Lan, C., Xu, J., Wenjun, Z., and Wu, F. (2015). Compound Image Compression Using Lossless and Lossy LZMA in HEVC. In **2015 IEEE International Conference on Multimedia and Expo (ICME)**. pp. 1-6. DOI: 10.1109/ICME.2015.7177430

[10] Zhou, B., Jin, H., and Zheng, R. (2014). A High Speed Lossless Compression Algorithm Based on CPU and GPU Hybrid Platform. In **2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications**. pp. 693-698. DOI: 10.1109/TrustCom.2014.90

[11] Zhu, W., Xu, J., Ding, W., Shi, Y., and Yin, B. (2013). Adaptive LZMA-Based Coding for Screen Content. In **2013 Picture Coding Symposium (PCS)**. pp. 373-376. DOI: 10.1109/PCS.2013.6737761

[12] Sundaresan, M. and Devika, E. (2012). Image Compression Using H.264 and Deflate Algorithm. In **International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME-2012)**. pp. 242-245. DOI: 10.1109/ICPRIME.2012.6208351

[13 Liu, W., Mei, F., Wang, C., O'Neill, M., and Swartzlander, E. E. (2018). Data Compression Device Based on Modified LZ4 Algorithm. **IEEE Transactions on Consumer Electronics**. Vol. 64, Issue 1, pp. 110-117. DOI: 10.1109/TCE.2018.2810480

[14] Li, H., Tuo, X., Shen, T., Henderson, M. J., Courtois, J., and Yan, M. (2017). An Improved Lossless Group Compression Algorithm for Seismic Data in SEG-Y and MiniSEED File Formats. **Computers & Geosciences**. Vol. 100, pp. 41-45. DOI: 10.1016/ j.cageo.2016.11.017

[15] Preet, S. and Bagga, A. (2018). Lempel-Ziv-Oberhumer: A Critical Evaluation of Lossless Algorithm and Its Applications. In **2018 4th International Conference on Computing Sciences (ICCS)**. pp.175-182. DOI:10.1109/ICCS.2018.00036