ระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลสำหรับอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะ

นายปิติพงศ์ พงศ์ภัทรานนท์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2555

Transcoding Cache for Smart Phones

Mr. Pitiphong Phongpattranont

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering Program in Computer Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2012

หัวข้อวิทยานิพนธ์	ระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลสำหรับอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะ
โดย	นายปิติพงศ์ พงศ์ภัทรานนท์
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์.ดร. เกริก ภิรมย์โสภา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์. ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ณัฐวุฒิ หนูไพโรจน์)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ผู้ช่วยศาสตราจารย์ ดร. เกริก ภิรมย์โสภา)

..... กรรมการภายนอกมหาวิทยาลัย
(ดร.พงศ์ธวัช ชีพพิมลชัย)

ปิติพงศ์ พงศ์ภัทรานนท์: ระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลสำหรับอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะ (TRANSCODING CACHE FOR SMART PHONES) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ.ดร. เกริก ภิรมย์โสภา, 44 หน้า.

งานวิจัยนี้ เราได้นำเสนอระบบแคชแบบปรับเปลี่ยนรหัสข้อมูล ซึ่งเป็นระบบแคชสำหรับอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะสำหรับแคชข้อมูลจากบริการต่างๆ บนระบบเครือข่ายอินเตอร์เน็ต โดยระบบแคชนี้มีแนวคิดใหม่สำหรับการแคชแฟ้มข้อมูลเพื่อให้ได้ประสิทธิภาพในการทำงานที่มากขึ้น โดยไม่ได้เปลี่ยนอัลกอริทึมการแทนที่ แต่ระบบแคชแบบปรับเปลี่ยนการเข้ารหัสข้อมูลจะใช้วิธีการปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูลที่ต้องการแคช ให้เหมาะสมกับพฤติกรรมการใช้งานของผู้ใช้ จากผลการทดลองของเราพบว่าระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลมีประสิทธิภาพดีกว่าระบบแคชทั่วไป ซึ่งระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลนี้ สามารถลดการส่งข้อมูลผ่านระบบเครือข่ายได้มากถึง 15 % เมื่อเทียบกับระบบแคชแบบทั่วไป

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต.....

สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อ.ที่ปรึกษาวิทยานิพนธ์หลัก.....

ปีการศึกษา 2555

5270392021: MAJOR COMPUTER ENGINEERING

KEYWORDS: Mobile / Cache; Transcoding / Web Cache / Web / Image / Web Service

PITIPHONG PHONGPATTRANONT : TRANSCODING CACHE FOR SMART PHONES. ADVISOR : ASST.PROF. KRERK PIROMSOPA, 44 pp.

We present Transcoding Cache (TC), a new way for caching data in smart phone application using web service. Transcoding Cache adapts the cache data for better utilization of limited space. Without changing the cache algorithm, TC works by transcoding the cache data in the disk cache. Our experiments show that TC performs better than the normal cache system. In particular, we can reduce the data transmitted through the mobile network up to 15% comparing to the legacy cache system.

Department: Computer Engineering Student's Signature .....

Field of Study: Computer Engineering Advisor's Signature .....

Academic Year: 2012

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยความอนุเคราะห์อย่างยิ่งของอาจารย์ ผศ. เกริก ภิรมย์โสภา อาจารย์ที่ปรึกษา ซึ่งท่านได้ให้ความรู้ แนะนำแนวทางการวิจัย ตรวจสอบให้คำแนะนำ และสนับสนุนเป็นอย่างดี จนทำให้การวิจัยในครั้งนี้สำเร็จออกมาด้วยดี

ขอขอบพระคุณ ผศ. ดร. ณัฐวุฒิ หนูไพโรจน์ และอาจารย์ ดร. พงศ์ธวัช ชีพพิมลชัย กรรมการสอบวิทยานิพนธ์ ที่กรุณาเสียสละเวลา ให้คำแนะนำ ตรวจสอบ และแก้ไขวิทยานิพนธ์ฉบับนี้

ขอขอบคุณเพื่อนร่วมงานที่บริษัท ทรูเวฟ (ประเทศไทย) จำกัด ที่ให้ความช่วยเหลือในการเก็บข้อมูลจากผู้ใช้งานเพื่อใช้ทดลองในงานวิจัยนี้

ท้ายที่สุด ผู้เสนอวิทยานิพนธ์ขอขอบคุณเพื่อน ๆ ทุก ๆ คน รวมทั้งครอบครัว ที่คอยติดตาม ให้กำลังใจ และสนับสนุนการทำงานในงานวิจัยนี้ รวมถึงท่านอื่น ๆ ที่มิได้กล่าวชื่อไว้ ณ ที่นี้ที่มีส่วนช่วยให้วิทยานิพนธ์สำเร็จได้ด้วยดี

สารบัญ

หน้า

หน้า

หน้า

สารบัญตาราง

หน้า

สารบัญรูป

หน้า

หน้า

บทที่ 1
บทนำ

ในปัจจุบัน อุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะ (Smart phone) กำลังได้รับความนิยมมากขึ้นในหมู่ผู้บริโภค เนื่องจากอุปกรณ์ดังกล่าวมีความสามารถที่สูงขึ้น อีกทั้งส่วนติดต่อผู้ใช้มีการใช้งานที่ง่ายขึ้นกว่าเมื่อก่อนมาก จึงทำให้อุปกรณ์โทรศัพท์มือถืออัจฉริยะเข้าถึงผู้บริโภคได้มากขึ้น และเป็นที่ต้องการมากขึ้น โดยบริษัท Gartner, Inc. ได้คาดการณ์ไว้ว่า ตลาดของอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะ มีการเติบโตไปอีกอย่างน้อย 5 ปี ผิดกับตลาดของเครื่องคอมพิวเตอร์ส่วนบุคคล ที่มีแนวโน้มว่าจะมีขนาดเล็กลงเรื่อยๆ เนื่องจากการเติบโตของอุปกรณ์แท็บเล็ต [1]

เหตุที่อุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะมีอัตราการเติบโตที่สูงมาก เนื่องจากในปัจจุบัน ตลาดอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะมีการแข่งขันที่ค่อนข้างสูง ผู้ผลิตอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะรายต่างๆ ต่างพยายามพัฒนาอุปกรณ์ของตนให้มีความสามารถใหม่ๆขึ้นมาเพื่อให้อุปกรณ์ของตนโดดเด่นเหนือกว่าอุปกรณ์จากผู้ผลิตรายอื่น ซึ่งการแข่งขันที่สูงมากนี่เอง ทำให้ตัวอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะนั้น มีอัตราการพัฒนาที่สูงมาก ทำให้อุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะมีความสามารถต่างๆ ที่อำนวยความสะดวกแก่ผู้ใช้มากขึ้น เช่น การใส่อุปกรณ์เชื่อมต่อเครือข่ายไร้สาย เพื่อให้อุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะสามารถเชื่อมต่อกับเครือข่ายไร้สาย ภายในบ้านของผู้ใช้ หรือเครือข่ายสาธารณะ เป็นต้น โดยเฉพาะหน่วยประมวลผลกลางของอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะ ซึ่งมีอัตราการพัฒนาที่สูงกว่าหน่วยประมวลผลกลางในเครื่องคอมพิวเตอร์ส่วนบุคคล

แต่ถึงแม้ชิปประมวลผลในอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะจะมีประสิทธิภาพที่สูงขึ้นอย่างต่อเนื่อง แต่แบตเตอรี่กลับมีการพัฒนาที่ช้ามาก ทั้งในด้านของความจุ ความสามารถ/ความเร็วในการประจุไฟ ฯลฯ [2] ซึ่งไม่สอดคล้องกับการพัฒนาของอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะ ทั้งที่แบตเตอรี่เอง เป็นหนึ่งในทรัพยากรที่สำคัญมากในอุปกรณ์เคลื่อนที่

การดึงข้อมูลต่างๆผ่านระบบเครือข่าย เป็นหนึ่งในกิจกรรมที่กินพลังงานมากที่สุดเป็นอันดับต้นๆในอุปกรณ์โทรศัพท์เคลื่อนที่ และยังใช้พลังงานมากเมื่อเทียบกับการใช้พลังงานในการประมวลผลต่างๆ [3] อีกทั้งผู้ใช้บริการเครือข่ายโทรศัพท์เคลื่อนที่เก็บค่าบริการการใช้เครือข่ายโดยคำนวนจากปริมาณข้อมูลที่ผู้ใช้ใช้ผ่านระบบเครือข่าย ดังนั้นหากเราสามารถลดค่าใช้จ่ายของผู้ใช้ด้วยการลดปริมาณการใช้ข้อมูลผ่านระบบเครือข่ายได้

อีกทั้ง จากการสังเกตพฤติกรรมการใช้งานของผู้ใช้โดยทั่วไป ในแฟ้มข้อมูลบางประเภทนั้น ผู้ใช้มักดูแฟ้มข้อมูลเพียงบางส่วนหรือข้อมูลโดยรวมของแฟ้มข้อมูล ซึ่งทำให้เกิดแนวคิดที่ว่าเราใช้ปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูลเพื่อลดขนาดของแฟ้มข้อมูลที่จะแคชได้ ทำให้ระบบแคชสามารถแคชข้อมูลได้จำนวนมากขึ้น ซึ่งจะทำให้ระบบแคชมีประสิทธิภาพมากขึ้น ซึ่งทำให้ผู้ใช้งานประหยัดทรัพยากรของอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะและค่าใช้จ่ายของผู้ใช้ได้

โครงงานวิจัยนี้จึงได้พัฒนา ระบบแคชแบบใหม่ โดยประยุกต์ใช้วิธีการปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูลเข้ากับระบบแคช เพื่อเพิ่มประสิทธิภาพการทำงานของระบบแคช และลดการใช้ทรัพยากรต่างๆในระบบของอุปกรณ์โทรศัพท์เคลื่อนที่

วัตถุประสงค์ของงานวิจัย

โครงงานวิจัยนี้มีวัตถุประสงค์ดังนี้

1. ทำการออกแบบและพัฒนาระบบแคชที่ออกแบบมาทำงานบนอุปกรณ์เคลื่อนที่โทรศัพท์มือถืออัจฉริยะ
2. นำระบบแคชที่พัฒนามาทดสอบประสิทธิภาพในการทำงานของระบบแคชแบบปรับเปลี่ยนรหัสข้อมูล

ลักษณะและขอบเขตของงานวิจัย

ลักษณะและขอบเขตของงานวิจัย มีลักษณะดังนี้

1 ลักษณะของงานวิจัย

1.1 ภาพรวมของระบบแคชแบบปรับเปลี่ยนการเข้ารหัสสำหรับโทรศัพท์แบบฉลาด

โครงงานนี้นำเสนอระบบแคชแบบปรับเปลี่ยนการเข้ารหัสสำหรับอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะ ซึ่งเป็นระบบแคชที่ถูกออกแบบให้ทำงานบนเครื่องอุปกรณ์เคลื่อนที่อัจฉริยะ ซึ่งมีสภาพแวดล้อมการใช้งาน ต่างจากเครื่องคอมพิวเตอร์ส่วนบุคคลโดยทั่วไป โดยระบบแคชที่เราออกแบบ มีคุณลักษณะที่ต้องการเป็นดังนี้

1. ระบบแคชถูกออกแบบมาให้ทำงานบนอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะ ซึ่งมีทรัพยากรของระบบที่จำกัด
2. ระบบแคชทำงานบน iOS แพลตฟอร์ม

ระบบแคชที่ออกแบบ จะถูกแบ่งออกเป็น 3 ส่วนใหญ่ คือ

1. ระบบคำนวนการเก็บแฟ้มข้อมูลของผู้ใช้ ระบบส่วนนี้ มีหน้าที่คำนวนและคัดเลือกแฟ้มข้อมูลของผู้ใช้ ว่าแฟ้มข้อมูลใดควรถูกเก็บอยู่ในแคช แฟ้มข้อมูลใดสามารถทิ้งไปได้
2. ระบบปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูล เพื่อลดพื้นที่การใช้งานของแฟ้มข้อมูลลง
3. ระบบเก็บข้อมูลสถิติการทำงานของระบบแคช ระบบส่วนนี้ทำหน้าที่เก็บข้อมูลและสถิติการเข้าถึงแฟ้มข้อมูลของผู้ใช้และระบบแคช เพื่อให้ระบบคำนวนการเก็บแฟ้มข้อมูลของผู้ใช้ใช้ข้อมูลในส่วนนี้ในการคำนวน และเพื่อนำข้อมูลดังกล่าวมาใช้วัดผลการทำงาน เพื่อให้เราสามารถปรับปรุงการทำงานของระบบแคช ให้มีประสิทธิภาพยิ่งขึ้นได้

1.2 ภาพรวมของอัลกอริทึมของระบบแคชแบบปรับเปลี่ยนการเข้ารหัสสำหรับโทรศัพท์แบบฉลาด

อัลกอริทึมของระบบแคชแบบปรับเปลี่ยนการเข้ารหัสสำหรับโทรศัพท์แบบฉลาดนั้น ใช้ข้อมูลต่างๆที่เกี่ยวข้องกับการใช้งานแฟ้มข้อมูลของผู้ใช้ มาประมวลผลการทำงานของระบบแคช เพื่อให้ได้ผลลัพท์ที่เหมาะสมและดีที่สุดสำหรับการใช้งานของผู้ใช้ระบบ

ข้อมูลที่เกี่ยวข้องกับการใช้งานระบบแคชดังกล่าว ได้แก่

1. ความถี่ในการเรียกดูแฟ้มข้อมูล
2. ขนาดของแฟ้มข้อมูล
3. ลักษณะของแฟ้มข้อมูล

โดยเมื่อระบบแคชได้รับข้อมูลที่เกี่ยวข้อง ระบบแคชบันทึกข้อมูลดังกล่าวเก็บไว้ เพื่อใช้ในการประมวลผลการทำงานของระบบแคชเมื่อถึงเวลาจำเป็น

1.3 ภาพรวมของกระบวนการปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูล

เมื่อระบบแคชจะแคชข้อมูลของผู้ใช้เข้าสู่ระบบ ระบบแคชจะส่งข้อมูลดังกล่าวมายังระบบการปรับเปลี่ยนการเข้ารหัสของข้อมูล ซึ่งระบบดังกล่าวจะทำการปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูล เมื่อทำการปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูลเสร็จแล้ว ระบบแคชจึงแคชข้อมูลที่ถูกปรับเปลี่ยนรหัสเรียบร้อยแล้ว ซึ่งจะทำให้ประหยัดพื้นที่ที่ใช้แคชข้อมูล และเพิ่มโอกาสที่ระบบแคชจะแคชข้อมูลที่ผู้ใช้ต้องการเข้าถึงอีกในอนาคต

1.4 ภาพรวมของโปรแกรมที่ใช้ทดสอบการทำงานของระบบแคชแบบปรับเปลี่ยนการเข้ารหัสสำหรับโทรศัพท์แบบฉลาด

ในการทดสอบการทำงานของระบบแคชที่ได้สร้างขึ้น ทางผู้จัดทำจะสร้างโปรแกรมทดสอบ โดยมีคุณสมบัติดังนี้

1. โปรแกรมทดสอบ จะติดต่อกับบริการของ Facebook ซึ่งโปรแกรมดังกล่าว จะสามารถแสดงรูปภาพต่างๆของผู้ใช้ได้
2. โปรแกรมดังกล่าว จะใช้ระบบแคชแบบปรับเปลี่ยนการเข้ารหัสสำหรับโทรศัพท์แบบฉลาดที่ได้สร้างขึ้นมา แทนที่ระบบแคชทั่วไป เพื่อทดสอบการทำงานของระบบแคช

2 ขอบเขตของงานวิจัย

ขอบเขตของงานวิจัย มีดังนี้

1. สร้างและทดสอบการทำงานของระบบแคชแบบปรับเปลี่ยนการเข้ารหัสสำหรับอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะ

2. สร้างโปรแกรมที่ทำงานบน iOS platform สำหรับทดสอบและทดลองใช้งานระบบแคชสำหรับอุปกรณ์โทรศัพท์มือถือเคลื่อนที่

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

การปรับเปลี่ยนรหัสข้อมูล

การปรับเปลี่ยนรหัสข้อมูล สามารถทำได้หลายวิธี ขึ้นอยู่กับวัตถุประสงค์ของการปรับเปลี่ยนรหัสข้อมูล เช่น

1. เพื่อลดขนาดของแฟ้มข้อมูล
2. เพื่อเพิ่มความปลอดภัยของการเข้าถึงข้อมูล
3. เพื่อเพิ่มประสิทธิภาพในการประมวลผลของข้อมูล

ซึ่งในรายงานฉบับนี้ เราจะนำเสนอการปรับเปลี่ยนข้อมูลที่เกี่ยวข้องกับงานวิจัย โดยมีดังต่อไปนี้

1 การบีบอัดข้อมูล

การบีบอัดข้อมูล เป็นกระบวนการที่ใช้สำหรับลดขนาดของแฟ้มข้อมูล เพื่อจุดประสงค์ต่างๆเช่น ลดขนาดของข้อมูลที่ใช้เก็บบันทึก ลดขนาดของข้อมูลที่ใช้ส่งผ่านระบบเครือข่าย เป็นต้น ซึ่งการบีบอัดข้อมูลสามารถแบ่งได้เป็น 2 ประเภท ดังนี้

1.1 การบีบอัดแบบไม่มีความสูญเสียของข้อมูล

การบีบอัดแบบไม่มีความสูญเสียของข้อมูลเป็นกระบวนการบีบอัดข้อมูลที่ไม่มีการสูญเสียข้อมูลในกระบวนการบีบอัดเลย ทำให้เมื่อเราคลายการบีบอัดข้อมูลแล้ว เราจะได้ข้อมูลที่เหมือนข้อมูลต้นฉบับทุกประการ

ในปัจจุบัน มีกระบวนการบีบอัดแบบไม่มีความสูญเสียของข้อมูลอยู่หลายวิธี ขึ้นอยู่กับประเภทของแฟ้มข้อมูลที่ต้องการบีบอัด เช่น

- แฟ้มข้อมูลประเภทรูปภาพ ได้แก่ การบีบอัดแบบ Portable Network Graphics (PNG) [4] การบีบอัดแบบ Tagged Image File Format (TIFF) [5] การบีบอัดแบบ JPEG2000 [6] แบบไม่สูญเสียข้อมูล เป็นต้น
- แฟ้มข้อมูลประเภทวีดีโอ ได้แก่ การบีบอัดแบบ H.264 [7] แบบไม่สูญเสียข้อมูล การบีบอัดแบบ JPEG2000 แบบไม่สูญเสียข้อมูล เป็นต้น
- แฟ้มข้อมูลประเภทเสียง ได้แก่ การบีบอัดแบบ Free Lossless Audio Codec (FLAC) [8] การบีบอัดแบบ Apple Lossless Audio Codec (ALAC) [9] เป็นต้น
- แฟ้มข้อมูลทั่วไป ได้แก่ การบีบอัดแบบ DEFLATE [10] เป็นต้น

1.2 การบีบอัดแบบมีความสูญเสียของข้อมูล

การบีบอัดแบบมีความสูญเสียของข้อมูลเป็นกระบวนการบีบอัดข้อมูลที่มีการสูญเสียข้อมูลในกระบวนการบีบอัด ซึ่งข้อมูลที่ถูกทิ้งไป มักเป็นข้อมูลที่มีความสำคัญน้อย ซึ่งความสำคัญของข้อมูลนั้น ขึ้นอยู่กับประเภทของข้อมูล และการทำงานของอัลกอริทึมการบีบอัด

ในปัจจุบัน มีกระบวนการบีบอัดแบบมีความสูญเสียของข้อมูลอยู่หลายวิธี ขึ้นอยู่กับประเภทของแฟ้มข้อมูลที่ต้องการบีบอัด เช่น

- แฟ้มข้อมูลประเภทรูปภาพ ได้แก่ การบีบอัดแบบ Joint Photographic Experts Group (JPEG) การบีบอัดแบบ JPEG2000 เป็นต้น
- แฟ้มข้อมูลประเภทวีดีโอ ได้แก่ การบีบอัดแบบ H.264 เป็นต้น
- แฟ้มข้อมูลประเภทเสียง ได้แก่ การบีบอัดแบบ MPEG-2 Audio Layer III (MP3) [11] การบีบอัดแบบ Advanced Audio Coding (AAC) [12] เป็นต้น

2 การบีบอัดแฟ้มข้อมูลรูปภาพแบบ JPEG

การบีบอัดแฟ้มข้อมูลรูปภาพแบบ JPEG เป็นวิธีการบีบอัดแฟ้มข้อมูลประเภทรูปภาพที่เป็นที่นิยมมาก มีอุปกรณ์และโปรแกรมต่างๆรองรับมากมาย เนื่องจาก การบีบอัดแบบ JPEG สามารถลดขนาดของแฟ้มข้อมูลได้ถึง 10 เท่า โดยสูญเสียคุณภาพของรูปภาพที่มองเห็นได้เพียงเล็กน้อย

การทำงานของการบีบอัดแฟ้มข้อมูลรูปภาพแบบ JPEG มีหลักการทำงานโดยสังเขปดังนี้

1. กระบวนการบีบอัดแฟ้มข้อมูลรูปภาพแบบ JPEG ใช้การบีบอัดที่มีรากฐานมาจากกระบวนการ Discrete Cosine Transform (DCT)
2. การบีบอัดแฟ้มข้อมูลรูปภาพแบบ JPEG สามารถปรับแต่งการบีบอัดได้ เพื่อกำหนดผลลัพธ์ของรูปภาพหลังการบีบอัดได้ ว่าต้องการให้ภาพที่ได้มีคุณภาพที่ดี หรือเพื่อให้ได้แฟ้มข้อมูลขนาดเล็กได้
3. การบีบอัดแฟ้มข้อมูลรูปภาพแบบ JPEG จะแปลงข้อมูลจากข้อมูลขอบเขตข้อมูล 2 มิติเป็นข้อมูลขอบเขตของความถี่
4. การบีบอัดแฟ้มข้อมูลรูปภาพแบบ JPEG ใช้หลักการบีบอัดที่อิงกับการทำงานของระบบการมองเห็นของมนุษย์ ซึ่งระบบการมองเห็นของมนุษย์ละทิ้งข้อมูลที่มีความถี่สูงในของเขตของความถี่ ทำให้เราสามารถละทิ้งข้อมูลในส่วนนี้ได้
5. การบีบอัดแฟ้มข้อมูลรูปภาพแบบ JPEG เปลี่ยนข้อมูลของแฟ้มข้อมูลรูปภาพ ให้อยู่ในรูปข้อมูลของสีและรายละเอียดของรูป
6. การบีบอัดแฟ้มข้อมูลรูปภาพแบบ JPEG บีบอัดข้อมูลของสีมากกว่าข้อมูลรายละเอียดของรูป เนื่องจาก การมองเห็นของมนุษย์รับรู้ข้อมูลรายละเอียดของรูปได้ดีกว่าข้อมูลสีของรูป

7. ในส่วนของข้อมูลรายละเอียดของรูป การบีบอัดแฟ้มข้อมูลรูปภาพแบบ JPEG จะแบ่งข้อมูลรายละเอียดเป็น 2 แบบ คือ ข้อมูลรายละเอียดส่วนหยาบ และข้อมูลรายละเอียดส่วนละเอียด โดยการบีบอัดแฟ้มข้อมูลรูปภาพแบบ JPEG จะละทิ้งข้อมูลรายละเอียดส่วนละเอียดก่อน เนื่องจาก การมองเห็นของมนุษย์จะรับรู้ข้อมูลรายละเอียดส่วนหยาบ ได้ดีกว่าข้อมูลรายละเอียดส่วนละเอียด
8. กระบวนการการบีบอัดแฟ้มข้อมูลรูปภาพแบบ JPEG จะแบ่งรูปภาพออกรูปเล็กๆ มีขนาด 8x8 pixels ซึ่งแต่ละรูปขนาดเล็ก จะมีการบีบอัดข้อมูลแยกจากกัน ดังนั้น เมื่อเราบีบอัดภาพด้วยการบีบอัดแฟ้มข้อมูลรูปภาพแบบ JPEG โดยกำหนดค่าการบีบอัดให้มีการบีบอัดสูง จะทำให้ภาพที่ได้มีคุณภาพที่ไม่ดี โดยจะพบรูปแบบการบีบอัดสี่เหลี่ยมขนาดเล็กๆ ในรูปภาพ

จากคุณสมบัติต่างๆของการบีบอัดแฟ้มข้อมูลรูปภาพแบบ JPEG และประสิทธิภาพของการบีบอัดรูปภาพ ทำให้การบีบอัดแฟ้มข้อมูลรูปภาพแบบ JPEG เป็นกระบวนการบีบอัดแฟ้มข้อมูลประเภทรูปภาพที่ได้รับความนิยมอย่างมาก

3 การเปลี่ยนขนาดรูปภาพ

การเปลี่ยนขนาดรูปภาพเป็นกระบวนการสำหรับเปลี่ยน/ย่อ/ขยายรูปภาพที่ทำให้เกิดการเปลี่ยนแปลงขนาดของรูป ซึ่งทำให้เกิดการเปลี่ยนแปลงของ pixel ในรูป เพื่อจุดประสงค์ต่างๆ เช่น การนำแฟ้มข้อมูลรูปภาพไปแสดงบนอุปกรณ์แสดงผลขนาดต่างๆ เป็นต้น

3.1 การลดขนาดของรูปภาพ

การลดขนาดของรูปภาพเป็นกระบวนการที่ใช้ย่อขนาดของรูปภาพ เพื่อแสดงภาพขนาดใหญ่บนอุปกรณ์แสดงผลขนาดเล็กหรือเพื่อสร้างแฟ้มข้อมูลรูปภาพขนาดเล็กเพื่อใช้สำหรับเป็นรูปภาพเพื่อแสดงผลขนาดเล็ก (Thumbnail) ซึ่งเป็นกระบวนการสำคัญในแอพพลิเคชันต่างๆ หรือแม้กระทั่งในบริการต่างๆบนเครือข่ายอินเตอร์เน็ต

3.2 การเพิ่มขนาดของรูปภาพ [13]

การเพิ่มขนาดของรูปภาพเป็นกระบวนการที่ใช้ขยายขนาดของรูปภาพ เพื่อแสดงภาพขนาดเล็กบนอุปกรณ์แสดงผลขนาดใหญ่ การเพิ่มขนาดของรูปภาพอาจทำให้รูปภาพนั้นลดความคมชัดลง หรืออาจทำให้เกิดรอยหยักในรูป

เนื่องจากการเพิ่มขนาดของรูป มักจะลดคุณภาพของรูปภาพลง จึงมีการคิดกระบวนการเพิ่มขนาดของรูปหลายวิธี ดังนี้

3.2.1 Nearest Neighbor Interpolation

วิธี Nearest Neighbor Interpolation เป็นวิธีการเพิ่มขนาดของรูปภาพที่ง่ายที่สุด โดยข้อมูลสีของ pixel ตำแหน่งต่างๆ เป็นการนำข้อมูลสีของ pixel ที่ใกล้กับตำแน่งเดิมมากที่สุดในภาพต้นฉบับ

เนื่องจากวิธี Nearest Neighbor Interpolation ใช้ข้อมูลสีเดิม โดยไม่มีการคำนวนค่าสีใหม่ ทำให้วิธีการนี้ทำให้เกิดรอยหยักขึ้นในรูปภาพที่ขยายออกมา

3.2.2 Bilinear interpolation

วิธี Bilinear interpolation นั้น ข้อมูลสีของ pixel ตำแหน่งต่างๆ จะถูกคำนวนจากค่าเฉลี่ยที่ถ่วงน้ำหนักไว้ของข้อมูลสีรอบ pixel ที่สนใจ จำนวน 4 pixels ด้วยรูปแบบ 2x2 pixels โดยรอบ pixel ที่สนใจ

เนื่องจากวิธี Bilinear interpolation คำนวนข้อมูลสีสำหรับตำแหน่ง pixel ต่างๆบนรูป โดยคำนวนจากค่าเฉลี่ยของ pixel โดยรอบ ทำให้รูปที่ได้มีรอยหยักในรูปภาพลดลง ทำให้เส้นต่างๆในภาพนั้นเรียบเนียนมากขึ้น

3.2.3 Bicubic interpolation

วิธี Bicubic interpolation นั้น จะคล้ายกับวิธี Bilinear interpolation แต่จะเพิ่มจำนวน pixels จาก 4 (2x2) pixels รอบข้าง เป็น 16 pixels ด้วยรูปแบบ 4x4 pixels รอบ pixel ที่สนใจ

เนื่องจากวิธี Bicubic interpolation นั้น เพิ่มจำนวนข้อมูลที่ใช้คำนวนจากวิธี Bilinear interpolation ทำให้ภาพที่ได้ มีรอยหยักลดลง และได้ภาพที่เส้นต่างๆในภาพนั้นเรียบเนียนมากยิ่งขึ้นเมื่อเทียบกับวิธี Bilinear interpolation

และเพราะประสิทธิภาพที่ดีของวิธีนี้อ ทำให้วิธี Bicubic interpolation มักถูกนำไปใช้ในแอพพลิเคชันตกแต่งรูปภาพ ปรินเตอร์ กล้องถ่ายรูปดิจิตอล ในการขยายรูปภาพ

นอกจากวิธีข้างต้นแล้ว ยังมีวิธีการขยายขนาดของรูปที่ออกแบบมาสำหรับการขยายขนาดของรูปที่อยู่เกมสมัยก่อน (Pixel art) เนื่องจาก เครื่องเกมสมัยก่อน มีพลังในการประมวลผลที่น้อยมากเมื่อเทียบกับปัจจุบัน ทำให้รูปต่างๆที่ใช้ภายในเกมและโปรแกรมต้องมีขนาดเล็ก เพื่อประหยัดพลังประมวลผลของเครื่องเกม

แต่ในปัจจุบัน เครื่องเกมต่างๆหรือแม้กระทั่งเครื่องคอมพิวเตอร์ส่วนบุคคลมีพลังในการประมวลผลสูงขึ้นมาก อีกทั้งยังมีคนนิยมเล่นและชื่นชอบเกมสมัยก่อนอยู่มากมาย ทำให้มีคนคิดค้นกระบวนการขยายภาพในเกมสมัยก่อน ให้มีขนาดใหญ่ขึ้นโดยเฉพาะ เช่น วิธี Depixelizing Pixel Art [14] เป็นต้น

แคช

แคช [15] เป็นองค์ประกอบหนึ่งในระบบคอมพิวเตอร์ที่มีความสำคัญ แคชทำหน้าที่เก็บข้อมูลที่มีการเข้าถึงไว้ในองค์ประกอบที่สามารถเข้าถึงข้อมูลได้เร็วกว่าส่วนที่เก็บข้อมูลจริง ทำให้เมื่อมีการเข้าถึงข้อมูลดังกล่าวอีกครั้งในอนาคต แคชสามารถคืนข้อมูลที่ถูกเก็บไว้ ซึ่งแคชคืนข้อมูลได้เร็วกว่าการเข้าถึงข้อมูลจากส่วนที่เก็บข้อมูลจริง ทำให้ประสิทธิภาพโดยรวมของระบบดีขึ้น

ในระบบคอมพิวเตอร์ แคชจะทำงานในส่วนต่างๆ เช่น

1 หน่วยประมวลผลกลาง

ในหน่วยประมวลผลกลางของระบบคอมพิวเตอร์ มีแคชอยู่หลายส่วน เพื่อใช้เก็บข้อมูลหรือชุดคำสั่งที่เพิ่งมีการประมวลผลไว้ในหน่วยประมวลผลกลางเอง ทั้งนี้เนื่องจากหน่วยประมวลผลกลางมีการทำงานที่เร็วกว่าหน่วยความจำหลักมาก ทำให้เมื่อหน่วยประมวลผลกลางต้องการเข้าถึงข้อมูลในหน่วยความจำหลัก หน่วยประมวลผลกลางต้องรอข้อมูลจากหน่วยความจำหลักหลายสิบหรือหลายร้อยสัญญาณนาฬิกา ผิดกับการเข้าถึงข้อมูลในแคชของหน่วยประมวลผลกลางซึ่งใช้เวลาเพียงไม่กี่สัญญาณนาฬิกาเท่านั้น

ในหน่วยประมวลผลกลางของระบบคอมพิวเตอร์ในปัจจุบันเอง มีแคชอยู่หลายระดับ โดยในระดับที่ใกล้กับหน่วยประมวลผลกลางมากที่สุดมีขนาดเล็กที่สุด แต่ก็มีความเร็วสูงที่สุดด้วยเช่นกัน โดยในหน่วยประมวลผลกลางบางรุ่น มีแคชอยู่ถึง 3 ระดับ

แคชในหน่วยประมวลผลกลางจะถูกออกแบบรวมอยู่ในวงจรการทำงานของหน่วยประมวลผลกลางโดยตรง ทำให้โดยปกติแล้ว เราไม่สามารถเข้าถึงข้อมูลในแคชของหน่วยประมวลผลกลางโดยตรงได้

2 หน่วยความจำ

ในระบบคอมพิวเตอร์ มีหน่วยความจำอยู่หลายระดับ เช่น หน่วยความจำหลักที่ทำหน้าที่เก็บข้อมูลการทำงานของระบบคอมพิวเตอร์ และหน่วยความจำสำรองที่ทำหน้าที่เก็บข้อมูลต่างๆของผู้ใช้รวมถึงตัวระบบปฏิบัติการ

โดยทั่วไป หน่วยความจำหลักมีความเร็วในการเข้าถึงข้อมูลในหน่วยความจำสำรองหลายร้อยถึงหลักพันเท่า ดังนั้น ในระบบปฏิบัติการบางตัวมีการแคชข้อมูลของหน่วยความจำสำรองที่ถูกเรียกใช้ไว้ในหน่วยความจำหลัก เพื่อให้ผู้ใช้สามารถเข้าถึงข้อมูลในหน่วยความจำสำรองได้เร็วขึ้น [16]

แต่ในบางครั้ง หน่วยความจำสำรองเองก็สามารถทำหน้าที่เป็นแคชให้กับข้อมูลที่ถูกเก็บในหน่วยความจำที่มีความเร็วต่ำกว่ามากๆได้ เช่น ข้อมูลที่ถูกเก็บอยู่ในระบบเครือข่าย ข้อมูลที่ถูกเก็บอยู่ในหน่วยความจำสำรองแบบเทป เป็นต้น

3 เว็บแคช

เว็บแคชเป็นระบบแคชที่ถูกใช้งานในระบบเครือข่ายอินเตอร์เน็ต ซึ่งมีอยู่ในทุกระดับชั้นของระบบเครือข่าย ทั้งนี้ เนื่องจากการเข้าถึงข้อมูลในระบบเครือข่ายอินเตอร์เน็ต ต้องผ่านระบบเครือข่ายซึ่งมีการส่งข้อมูลที่ช้า และอาจไม่เสถียรในบางแห่ง และทรัพยากรระบบเครือข่ายเป็นทรัพยากรที่มีอยู่อย่างจำกัด ทำให้เว็บแคชเป็นองค์ประกอบหนึ่งที่สำคัญในระบบเครือข่ายอินเตอร์เน็ต ทั้งนี้ ระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลในงานวิจัยนี้ก็ถือเป็นเว็บแคชแบบหนึ่ง

โดยปกติเว็บแคชทำงานในส่วนต่างๆของระบบเครือข่ายอินเตอร์เน็ต เช่น

1. ในเว็บเบราว์เซอร์เองมีการแคชข้อมูลเว็บไซต์ที่ผู้ใช้เรียกดู โดยมีการควบคุมอายุของเนื้อหาเว็บไซต์ที่ถูกแคชผ่านโปรโตคอล HTTP

2. ในองค์กรที่มีผู้ใช้ระบบเครือข่ายอินเตอร์เน็ตจำนวนมาก ทางองค์กรมักมีระบบเว็บแคชแบบเครื่องบริการแทน (Proxy server) ติดตั้งในระบบเครือข่ายขององค์กร เพื่อลดปริมาณข้อมูลที่จะใช้ออกสู่ระบบเครือข่ายขององค์กร ทำให้ประหยัดค่าใช้จ่ายสำหรับองค์กรเอง
3. ในระบบเครือข่ายระดับประเทศหรือภูมิภาค ผู้ให้บริการอินเตอร์เน็ต มักติดตั้งระบบเว็บแคชแบบเครื่องบริการแทนเอาไว้ในระบบเครือข่าย เพื่อลดปริมาณข้อมูลที่ออกสู่ระบบเครือข่ายภายนอก เนื่องจากผู้ให้บริการอินเตอร์เน็ตมักจะมีจุดเชื่อต่อออกสู่ระบบเครือข่ายภายนอกที่จำกัด ทำให้ต้องการลดปริมาณข้อมูลที่ถูกส่งเข้า/ออกไปนอกเครือข่ายของตัวเอง
4. ผู้ให้บริการค้นหา (Search engine) มักแคชหน้าเว็บไซต์ต่างๆที่ตนเองได้ทำดรรชนี (Index) เอาไว้ด้วยเช่นกัน สำหรับกรณีที่ผู้ใช้ต้องการเรียกดูเว็บไซต์ที่ปิดบริการลงไปแล้วหรือผู้ให้บริการไม่สามารถให้บริการหน้าเว็บไซต์นั้นได้

อัลกอริทึมการแทนที่

เป็นอัลกอริทึมสำหรับเลือกข้อมูลที่ต้องถูกแทนที่ด้วยข้อมูลใหม่ในแคช ซึ่งมีอัลกอริทึมที่นิยมใช้ดังนี้

1 Least Recently Used (LRU)

อัลกอริทึม Least Recently Used [17] เป็นอัลกอริทึมที่เลือกทิ้งข้อมูลที่ถูกเรียกใช้งานเก่าที่สุดทิ้งไป ซึ่งการเขียนอัลกอริทึมนี้โดยทั่วไป มักจะใช้วิธีการระบุอายุของข้อมูลหรือเวลาที่ผู้ใช้เข้าถึงข้อมูล

2 Universal Mobile Cache (UMC)

อัลกอริทึม Universal Mobile Cache [18] เป็นอัลกอริทึมที่ใช้คุณสมบัติหลายๆอย่าง เป็นเกณฑ์ในการเลือกข้อมูลทิ้ง โดยจัดอันดับของแฟ้มข้อมูลด้วยค่าใช้จ่ายในการดึงแฟ้มข้อมูล (Cost of retrieval) และแทนที่แฟ้มข้อมูลด้วยแฟ้มข้อมูลที่มีค่าใช้จ่ายในการดึงที่น้อยที่สุด ซึ่งสูตรที่ใช้หาค่าใช้จ่ายในการดึงแฟ้มข้อมูลที่ใช้ในงานวิจัยนี้เป็นดังนี้

$$H(p) = w_L(\log(L(p)) + w_C \log(C(p)) - w_B \log(B(p))$$

โดย UMC จะใช้คุณสมบัติของแฟ้มข้อมูลดังนี้

1. L(p) คือ ความน่าจะเป็นที่จะมีการเข้าถึงแฟ้มข้อมูล
2. C(p) คือ ค่าใช้จ่ายที่ใช้ในการดึงแฟ้มข้อมูล
3. B(p) คือ ผลประโยชน์ที่ได้เมื่อมีการทิ้งแฟ้มข้อมูล

ในงานวิจัยนี้ เราใช้ เราใช้คุณสมบัติต่อไปนี้ในการคำนวนหาค่าใช้จ่ายในการดึงแฟ้มข้อมูล

1. ความถี่ในการเข้าถึงแฟ้มข้อมูลเป็นค่าความน่าจะเป็นที่จะมีการเข้าถึงแฟ้มข้อมูล
2. ขนาดของแฟ้มข้อมูลต้นฉบับเป็นค่าใช้จ่ายในการดึงแฟ้มข้อมูล
3. ขนาดของแฟ้มข้อมูลที่ถูกปรับเปลี่ยนรหัสข้อมูลเป็นผลประโยชน์ที่ได้เมื่อมีการทิ้งแฟ้มข้อมูล

บทที่ 3
การออกแบบ

แนวคิดของการออกแบบระบบแคช

จากการสังเกตการใช้งานของผู้ใช้ส่วนใหญ่ พบว่า โดยปกติเมื่อผู้ใช้ดูแฟ้มข้อมูลประเภทรูปภาพ ผู้ใช้มักดูรูปภาพทั้งรูปบนหน้าจอ ซึ่งโดยปกติ ขนาดของรูปภาพมักจะมีขนาดใหญ่กว่าขนาดของหน้าจอเครื่องโทรศัพท์เคลื่อนที่

ดังนั้น หากเราต้องการแคชแฟ้มข้อมูลรูปภาพดังกล่าว เราสามารถย่อขนาดของรูปภาพลงให้มีขนาดพอดีกับหน้าจอของอุปกรณ์โทรศัพท์เคลื่อนที่ เพื่อลดขนาดของแฟ้มข้อมูลที่จะแคชได้ ซึ่งช่วยเพิ่มจำนวนข้อมูลในระบบแคชและสามารถเพิ่มโอกาสที่ระบบแคชจะแคชข้อมูลที่ผู้ใช้ต้องการเข้าถึงในอนาคตได้ จึงเป็นที่มาของงานวิจัยนี้

ภาพรวมของระบบแคช

เราสามารถแบ่งระบบแคชออกเป็นระบบย่อยได้ 3 ระบบย่อย ดังนี้

1 ระบบการจัดการการแคชแฟ้มข้อมูล

ระบบการจัดการการแคชข้อมูล เป็นระบบที่ทำหน้าที่จัดการและประมวลผลข้อมูลที่เกี่ยวข้องกับการแคชข้อมูลที่ผู้ใช้เข้าถึง ซึ่งรับข้อมูลการเข้าถึงแฟ้มข้อมูลของผู้ใช้ ประมวลผลข้อมูลดังกล่าวเพื่อตรวจสอบดูว่าระบบสามารถคืนแฟ้มข้อมูลที่ได้แคชไว้ได้หรือไม่ ถ้าไม่ได้ ระบบนี้จะดึงข้อมูลที่ผู้ใช้ต้องการผ่านระบบเครือข่ายคืนข้อมูลให้ผู้ใช้ และส่งข้อมูลให้กับระบบปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูล เพื่อปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูลก่อนทำการแคชข้อมูลดังกล่าว แต่ถ้าได้ ระบบจะคืนข้อมูลที่ถูกแคชไว้ในระบบให้กับผู้ใช้

2 ระบบปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูล

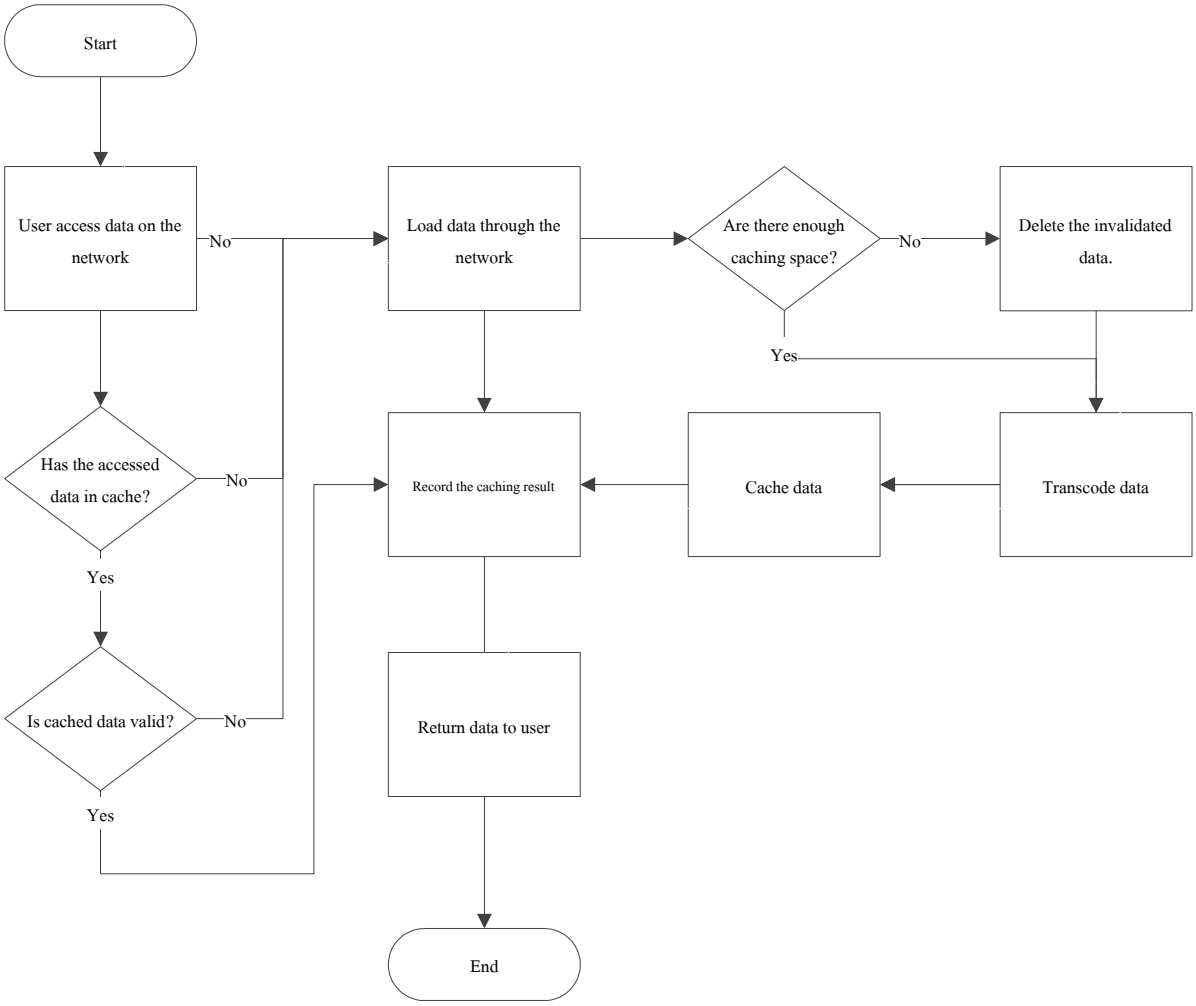
ระบบปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูล เป็นระบบที่ทำหน้าที่ปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูลที่ระบบแคชต้องการแคช ซึ่งรายละเอียดของการปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูลจะอธิบายโดยละเอียดในหัวข้อ การปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูลที่ใช้ในงานวิจัย

3 ระบบเก็บสถิติการทำงานของระบบแคชข้อมูล

ระบบเก็บสถิติการทำงานของระบบแคชข้อมูลมีหน้าที่เก็บข้อมูลการทำงานของระบบแคช เพื่อให้ระบบคำนวนการเก็บแฟ้มข้อมูลของผู้ใช้ใช้ข้อมูลในส่วนนี้ในการคำนวนและเพื่อใช้ในการประเมินผลการทำงานของระบบและงานวิจัย

ขั้นตอนการทำงานของระบบแคช

ระบบแคชมีขั้นตอนการทำงานดังแผนภาพต่อไปนี้



รูปที่ 1 แผนภาพแสดงการทำงานของระบบแคช

จากรูปที่ 1 แผนภาพแสดงการทำงานของระบบแคช เราพบข้อแตกต่างของระบบแคชแบบปรับเปลี่ยนการเข้ารหัสข้อมูลกับระแบบแคชทั่วไปดังนี้

1. เมื่อผู้ใช้ต้องการเข้าถึงข้อมูลต้นฉบับ ทางผู้ใช้หรือแอพพลิเคชันต้องระบุในคำขอเข้าถึงข้อมูลด้วยว่าผู้ใช้ต้องการเข้าถึงข้อมูลต้นฉบับ
2. ในระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลที่ใช้กระบวนการปรับเปลี่ยนรหัสข้อมูลบางวิธี เมื่อผู้ใช้ต้องการเข้าถึงข้อมูลใดๆ หากผู้ใช้ระบุว่าต้องการเข้าถึงข้อมูลต้นฉบับ ระบบแคชถือว่าระบบไม่สามารถส่งต่อข้อมูลดังกล่าวให้กับผู้ใช้ได้ และดึงข้อมูลต้นฉบับผ่านทางเครือข่ายทันที
3. เมื่อระบบจะแคชข้อมูลใดๆ ระบบปรับเปลี่ยนการเข้ารหัสข้อมูลของข้อมูลดังกล่าวก่อนทำการแคช

การปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูล

1 การปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูล

การปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูลสามารถทำได้หลายรูปแบบ ขึ้นอยู่กับประเภทของแฟ้มข้อมูลนั้น ดังนี้

1.1 แฟ้มข้อมูลประเภทรูปภาพ

การปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูลประเภทรูปภาพ สามารถทำได้หลายวิธี ได้แก่

1. การเปลี่ยนขนาดของรูปภาพ
2. การลดจำนวนข้อมูลรหัสสี
3. การเปลี่ยนกระบวนการบีบอัด
4. การปรับเปลี่ยนพารามีเตอร์ของการบีบอัด

1.2 แฟ้มข้อมูลประเภทวีดีโอ

การปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูลประเภทวีดีโอ สามารถทำได้หลายวิธี ได้แก่

1. การเปลี่ยนขนาดของวีดีโอ
2. การเปลี่ยนอัตราข้อมูลในแฟ้มข้อมูลวีดีโอ
3. การตัดบางส่วนของวีดีโอ

1.3 แฟ้มข้อมูลประเภทข้อความ

การปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูลประเภทข้อความ สามารถทำได้หลายวิธี ได้แก่

1. การบีบอัดแฟ้มข้อมูล

2 การปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูลที่ใช้ในงานวิจัย

ในงานวิจัยนี้ เรามุ่งเน้นไปที่แฟ้มข้อมูลประเภทรูปภาพ ซึ่งเราได้ทดลองการปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูลประเภทรูปภาพด้วยวิธีต่างๆ ดังนี้

1. การเปลี่ยนขนาดของรูปภาพ
2. การลดจำนวนข้อมูลรหัสสี
3. การปรับเปลี่ยนพารามีเตอร์ของการบีบอัด

โดยการทดสอบ ได้ผลดังนี้

รูปที่ 2 ภาพต้นฉบับ

รูปที่ 3 ภาพที่ถูกย่อขนาดลงเหลือ 640x960 pixels

รูปที่ 4 ภาพที่ถูกลดจำนวนข้อมูลรหัสสี

รูปที่ 5 ภาพที่เพิ่มอัตราการบีบอัดของข้อมูล

จากผลการทดสอบ เราพบว่า ผู้ใช้สังเกตเห็นความแตกต่างระหว่าง ภาพที่ถูกลดขนาดของภาพกับภาพต้นฉบับได้น้อยที่สุด รองลงมาคือภาพที่เพิ่มอัตราการบีบอัดของข้อมูล ส่วน ภาพที่ถูกลดจำนวนข้อมูลรหัสสีนั้น มีความแตกต่างกับภาพต้นฉบับมากจนผู้ใช้สามารถสังเกตเห็นได้ตั้งแต่เริ่มต้น

ในงานวิจัยนี้เราเลือกการบีบอัดแฟ้มข้อมูลรูปภาพแบบ JPEG เป็นวิธีที่ใช้บีบอัดแฟ้มข้อมูล เนื่องจากการบีบอัดแฟ้มข้อมูลรูปภาพแบบ JPEG เป็นวิธีการบีบอัดแฟ้มข้อมูลรูปภาพที่เป็นที่นิยม มีประสิทธิภาพสูง และเหมาะสำหรับแฟ้มข้อมูลรูปภาพที่เป็นภาพถ่าย ซึ่งเป็นรูปภาพที่มีมากในเครือข่ายอินเตอร์เน็ต

ปัจจัยที่ส่งผลกับขนาดของข้อมูลที่ถูกบีบอัดด้วยการบีบอัดแฟ้มข้อมูลรูปภาพแบบ JPEG มีดังนี้

1. ขนาดของรูปภาพ
2. พารามิเตอร์ของการบีบอัด

ดังนั้น ในงานวิจัยนี้ เราจึงเลือกใช้วิธีการปรับเปลี่ยนการเข้ารหัสข้อมูลของแฟ้มข้อมูลรูปภาพ ดังนี้

1. ลดขนาดของภาพ และเพิ่มอัตราการบีบข้อมูลเป็นบีบอัดระดับปานกลาง
2. เพิ่มอัตราการบีบอัดข้อมูลเป็นบีบอัดระดับค่อนข้างมาก

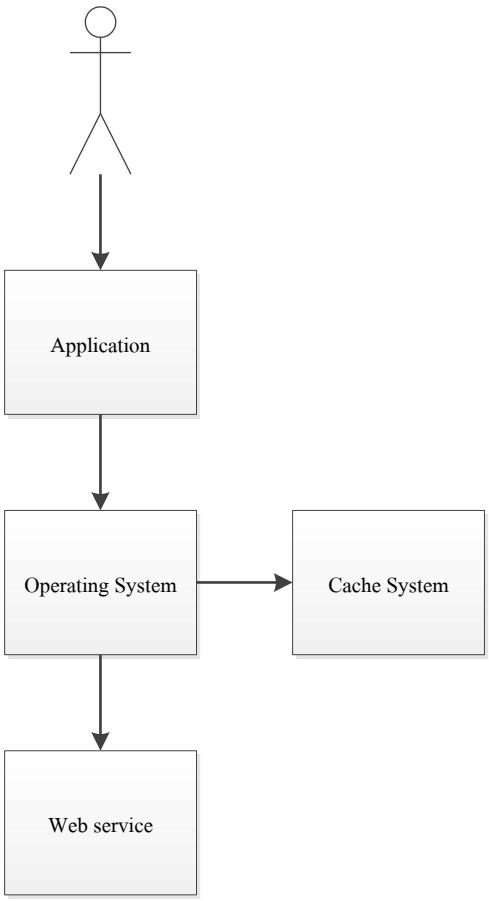
บทที่ 4
การอิมพลิเมนต์

ลักษณะโครงสร้างการทำงานของระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลบนแพลตฟอร์ม iOS

ในงานวิจัยนี้ เราได้อิมพลิเมนต์ระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลบนแพลตฟอร์ม iOS ซึ่งทำงานบนระบบปฏิบัติการ iOS เวอร์ชัน 6.0.0

นอกจากระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลแล้ว เราได้พัฒนาโปรแกรมที่ใช้ทดสอบการทำงานของระบบแคชด้วย โดยโปรแกรมดังกล่าว จะแสดงรายการรูปของผู้ใช้ ที่เก็บไว้ที่บริการเครือข่ายสังคม และสามารถใช้เรียกดูรูปของผู้ใช้ได้

โดยระบบแคชและโปรแกรมที่ใช้ทดสอบมีโครงสร้างการทำงานดังนี้



รูปที่ 6 โครงสร้างการทำงานของระบบแคชและโปรแกรมทดสอบ

โปรแกรมทดสอบ จะดึงแฟ้มข้อมูลรูปภาพของผู้ใช้ผ่านระบบปฏิบัติการ และระบบปฏิบัติการตรวจสอบข้อมูลกับระบบแคชก่อน ว่าสามารถคือแฟ้มข้อมูลให้กับผู้ใช้ได้หรือไม่ เพื่อให้ระบบปฏิบัติการสามารถคืนข้อมูลที่ผู้ใช้ต้องการเข้าถึงให้กับผู้ใช้ได้

การทำงานของระบบจัดการการแคชแฟ้มข้อมูล

ระบบจัดการการแคชแฟ้มข้อมูลเป็นระบบที่จัดการเรื่องการแคชแฟ้มข้อมูล โดยระบบนี้ทำงานร่วมกับระบบปฏิบัติการ

โดยเมื่อผู้ใช้ต้องการเข้าถึงแฟ้มข้อมูลบนระบบเครือข่าย ระบบปฏิบัติการสอบถามระบบแคชก่อน ว่ามีข้อมูลที่ผู้ใช้ต้องการเข้าถึงอยู่ในระบบแคชหรือไม่ ถ้าพบ ระบบปฏิบัติการคืนแฟ้มข้อมูลที่ถูกแคชไว้ในระบบแคชให้กับผู้ใช้ทันที โดยไม่มีการดึงแฟ้มข้อมูลจากระบบเครือข่าย

แต่ถ้าระบบแคชไม่พบข้อมูลที่ผู้ใช้ต้องการเข้าถึง ระบบปฏิบัติการดึงแฟ้มข้อมูลที่ผู้ใช้ต้องการผ่านระบบเครือข่าย เมื่อได้ข้อมูลแล้ว ระบบปฏิบัติการส่งแฟ้มข้อมูลให้กับระบบแคชก่อนคืนแฟ้มข้อมูลให้กับผู้ใช้

ระบบเก็บข้อมูลที่เกี่ยวข้องกับการทำงานของระบบแคช

ระบบแคชเก็บข้อมูลการเข้าถึงแฟ้มข้อมูลและข้อมูลของแฟ้มข้อมูล เพื่อใช้ในการทำงานของระบบแคช และเพื่อใช้ประเมินประสิทธิภาพการทำงานของระบบ โดยจะเก็บข้อมูลดังต่อไปนี้

1. ขนาดของแฟ้มข้อมูลต้นฉบับ
2. ขนาดของแฟ้มข้อมูลที่ถูกปรับเปลี่ยนรหัสข้อมูลแล้ว
3. เวลาที่ผู้ใช้ต้องการเข้าถึงแฟ้มข้อมูล
4. การเข้าถึงแฟ้มข้อมูล ผู้ใช้ต้องการเข้าถึงแฟ้มข้อมูลต้นฉบับหรือไม่

ทั้งนี้ เนื่องจากระบบแคชต้องการทราบว่า ผู้ใช้ต้องการเข้าถึงแฟ้มข้อมูลต้นฉบับหรือไม่ ดังนั้น เมื่อผู้ใช้ต้องการเข้าถึงแฟ้มข้อมูลต้นฉบับ แอพพลิเคชันเองต้องระบุมาในคำขอเข้าถึงข้อมูลด้วย

การทำงานของระบบปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูล

ระบบปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูลมีหน้าที่ปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูล ในงานวิจัยนี้ เราพิจารณาเฉพาะกรณีของแฟ้มข้อมูลประเภทรูปภาพเท่านั้น

เนื่องจาก ในการทำงานของระบบปฏิบัติการ iOS ได้มีการปรับเปลี่ยนขนาดของรูปภาพตลอดเวลาเพื่อใช้ในการแสดงผลแก่ผู้ใช้ ไม่ว่าในระดับของตัวระบบปฏิบัติการเอง หรือระดับแอพพลิเคชันที่ทำงานบนระบบปฏิบัติการ ดังนั้น ระบบปฏิบัติการ iOS จึงเตรียมเอพีไอสำหรับการเปลี่ยนขนาดของรูปภาพให้กับผู้พัฒนาโปรแกรม ซึ่งในงานวิจัยนี้ เราจะเรียกใช้เอพีไอที่ระบบปฏิบัติการเตรียมไว้ให้ เนื่องจากทางผู้ผลิตระบบปฏิบัติการ ได้ปรับแต่งการทำงานของเอพีไอนี้ให้ได้ผลลัพท์ที่ดี และทำงานได้อย่างรวดเร็ว ซึ่งเหมาะสมกับการใช้ในงานวิจัยนี้

ในระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลที่ใช้วิธีเปลี่ยนขนาดของรูปภาพและบีบอัดแฟ้มข้อมูลรูปภาพเป็นวิธีการปรับเปลี่ยนรหัสแฟ้มข้อมูล ระบบแคชใช้กระบวนการปรับเปลี่ยนขนาดรูปภาพของระบบปฏิบัติการ เมื่อแฟ้มข้อมูลรูปภาพถูกปรับเปลี่ยนขนาดของรูปภาพเสร็จแล้ว ระบบปรับเปลี่ยนรหัสข้อมูลจะบีบอัดแฟ้มข้อมูลรูปภาพ ด้วยวิธีการบีบอัดแบบ JPEG ด้วยค่าอัตราการบีบอัดปานกลาง แล้วแคชแฟ้มข้อมูลดังกล่าวเข้าสู่ระบบ เนื่องจากวิธีปรับเปลี่ยนรหัสข้อมูลนี้มีการเปลี่ยนขนาดของรูปภาพในกระบวนการปรับเปลี่ยนรหัสข้อมูล เมื่อผู้ใช้ต้องการเข้าถึงแฟ้มข้อมูลต้นฉบับ ระบบแคชถือว่าไม่สามารถส่งคืนแฟ้มข้อมูลที่ถูกแคชให้ผู้ใช้ได้

ในระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลที่ใช้วิธีบีบอัดแฟ้มข้อมูลรูปภาพเป็นวิธีการปรับเปลี่ยนรหัส ระบบปรับเปลี่ยนรหัสข้อมูลบีบอัดแฟ้มข้อมูลรูปภาพ ด้วยวิธีการบีบอัดแบบ JPEG ด้วยค่าอัตราการบีบอัดค่อนข้างสูง แล้วแคชแฟ้มข้อมูลดังกล่าวเข้าสู่ระบบ เนื่องจากวิธีปรับเปลี่ยนรหัสข้อมูลนี้ไม่มีการเปลี่ยนขนาดของรูปภาพในกระบวนการปรับเปลี่ยนรหัสข้อมูล เมื่อผู้ใช้ต้องการเข้าถึงแฟ้มข้อมูลต้นฉบับ ระบบแคชถือว่าระบบสามารถส่งคืนแฟ้มข้อมูลที่ถูกแคชให้ผู้ใช้ได้

การทำงานของโปรแกรมทดสอบระบบแคช

โปรแกรมสำหรับทดสอบการทำงานของระบบแคช เป็นโปรแกรมที่ถูกสร้างมาเพื่อเก็บข้อมูลการใช้งานของผู้ใช้ มาใช้จำลองการใช้งานของผู้ใช้ในการทดสอบระบบ

โดยโปรแกรมทดสอบเชื่อมต่อเข้ากับบริการของเครือข่ายสังคม Facebook และแสดงรายชื่อรูปภาพทั้งหมดของผู้ใช้ และสามารถเปิดดูรูปภาพแต่ละรูปของผู้ใช้ได้

เมื่อผู้ใช้ต้องการเข้าถึงแฟ้มข้อมูลรูปภาพของตนในบริการเครือข่ายสังคม Facebook โปรแกรมทดสอบเข้าถึงข้อมูลรูปภาพผ่านระบบปฏิบัติการ ซึ่งทำงานร่วมกับระบบแคช และแสดงภาพที่ผู้ใช้ต้องการเข้าถึงบนหน้าจอของอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะ

บทที่ 5
การประเมินผลการทำงานของระบบ

ในการประเมินผลการทำงานของระบบ เราใช้อัตราการเจอข้อมูลในแคช และปริมาณข้อมูลที่ส่งผ่านเครือข่ายเป็นข้อมูลในการประเมินประสิทธิภาพในการทำงานของระบบแคช โดยมีรายละเอียดการทดลองดังนี้

ลักษณะการทดลอง

1 ลักษณะของข้อมูลที่ใช้ทดสอบ

ผู้จัดทำ ได้ทำการเก็บบันทึกข้อมูลการเข้าถึงแฟ้มข้อมูลรูปภาพของผู้ใช้ โดยบันทึกจากการใช้งานจากอาสาสมัคร ซึ่งมีรายละเอียดดังนี้

1. จำนวนรูปภาพทั้งหมด มีจำนวน 473 รูปภาพ
2. แฟ้มข้อมูลรูปภาพทั้งหมด เป็นแฟ้มรูปภาพประเภท JPEG
3. แฟ้มข้อมูลรูปภาพ มีขนาดของรูปไม่เกิน 2048x2048 pixels
4. แฟ้มข้อมูลรูปภาพ มีขนาดทั้งหมดประมาณ 242.2 MB
5. การเข้าถึงข้อมูลทั้งหมด มีจำนวน 600 ครั้ง
6. การเข้าถึงข้อมูลดังกล่าว มีการเข้าถึงข้อมูลที่ผู้ใช้ต้องการเข้าถึงข้อมูลต้นฉบับอยู่จำนวน 20%

2 ลักษณะของอุปกรณ์ที่ใช้ทดสอบ

อุปกรณ์เครื่องโทรศัพท์เคลื่อนที่อัจฉริยะที่ใช้ในการทดลอง มีคุณสมบัติดังนี้

1. อุปกรณ์เครื่องโทรศัพท์เคลื่อนที่อัจฉริยะ มีขนาดหน้าจอ 320x480 pixels และ 640x960 pixels
2. อุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะ ทำงานบนระบบปฏิบัติการ iOS เวอร์ชัน 6.0.0

3 ลักษณะของระบบแคชที่ใช้ในการทดสอบ

ระแบบแคชที่ใช้ในการทดลอง มีคุณสมบัติดังนี้

1. ระบบแคช จะใช้อัลกอริทึมการแทนที่แบบ LRU กับแบบ UMC เป็นอัลกอริทึมการแทนที่
2. ระบบแคชจะกำหนดขนาดของพื้นที่เก็บข้อมูลแคชไว้ 5 ขนาดดังนี้
    1. 8 MB
    2. 16 MB
    3. 32 MB
    4. 64 MB
    5. 128 MB
3. ระบบแคชจะใช้วิธีการปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูล 2 วิธี ดังนี้
    1. ลดขนาดของภาพ และเพิ่มอัตราการบีบข้อมูลเป็นบีบอัดระดับปานกลาง

2. เพิ่มอัตราการบีบอัดข้อมูลเป็นบีบอัดระดับค่อนข้างมาก

ผลการทดลอง

1 ผลการทดลองการใช้พลังงานของระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูล

ตารางที่ 1 ตารางแสดงปริมาณพลังงานที่ใช้ในกิจกรรมต่างๆ

กิจกรรม	ดึงแฟ้มข้อมูลผ่านระบบเครือข่ายไร้สาย	ปรับเปลี่ยนรหัสข้อมูลด้วยวิธีลดขนาดของภาพและเพิ่มอัตราการบีบข้อมูล	ปรับเปลี่ยนรหัสข้อมูลด้วยวิธีเพิ่มอัตราการบีบข้อมูล
ปริมาณพลังงานที่ใช้ (mAh)	0.149048626	0.013305322	0.011554622

ตารางที่ 1 แสดงปริมาณพลังงานที่ใช้ในกิจกรรมที่เกี่ยวข้องกับการปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลรูปภาพ จากผลการทดลองพบว่า การดึงแฟ้มข้อมูลผ่านระบบเครือข่ายไร้สายใช้ปริมาณพลังงานมากกว่าการปรับเปลี่ยนรหัสข้อมูลด้วยวิธีการลดขนาดของภาพและเพิ่มอัตราการบีบข้อมูลมากถึง 11.202181052 เท่าและมากกว่าการเพิ่มอัตราการบีบข้อมูลถึง 12.899480918 เท่า

2 ผลการทดลองของระบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูล

ตารางที่ 2 ตารางแสดงขนาดของแฟ้มข้อมูลที่ปรับเปลี่ยนรหัสข้อมูลแล้วโดยเฉลี่ย

วิธีการปรับเปลี่ยนรหัสแฟ้มข้อมูล	วิธีลดขนาดรูปภาพเหลือขนาด 320x480 pixels และเพิ่มอัตราการบีบข้อมูล	วิธีลดขนาดรูปภาพเหลือขนาด 640x960 pixels และเพิ่มอัตราการบีบข้อมูล	วิธีเพิ่มอัตราการบีบข้อมูลอย่างเดียว
Cached File Size (%)	6.36	21.27	48.62

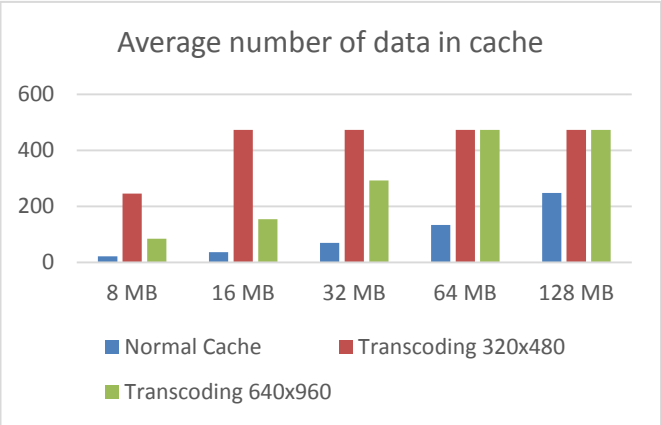
ตารางที่ 2 แสดงขนาดของแฟ้มข้อมูลที่ปรับเปลี่ยนรหัสข้อมูล เทียบกับขนาดข้อมูลของแฟ้มข้อมูลดั้งเดิม จากผลการทดลองพบว่า เมื่อทดลองกับอุปกรณ์โทรศัพท์เคลื่อนที่ที่มีขนาดหน้าจอ 320x480 pixels ระบบแคชสามารถลดขนาดของแฟ้มข้อมูลได้ถึงกว่า 97% อุปกรณ์โทรศัพท์เคลื่อนที่ที่มีขนาดหน้าจอ 640x960 pixels ระบบแคชสามารถลดขนาดของแฟ้มข้อมูลได้ 78% และในกระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลที่ใช้การบีบอัดข้อมูลของแฟ้มข้อมูลรูปภาพเพียงอย่างเดียว สามารถลดขนาดของแฟ้มข้อมูลได้ 51%

ตารางที่ 3 ตารางแสดงเวลาที่ใช้ในการปรับเปลี่ยนรหัสแฟ้มข้อมูลโดยเฉลี่ย

วิธีการเข้ารหัส	วิธีลดขนาดรูปภาพเหลือขนาด 320x480 pixels และเพิ่มอัตราการบีบข้อมูล	วิธีลดขนาดรูปภาพเหลือขนาด 640x960 pixels และเพิ่มอัตราการบีบข้อมูล	วิธีเพิ่มอัตราการบีบข้อมูล
เวลาที่ใช้โดยเฉลี่ย (วินาที)	0.07207	0.17787619	0.133711104

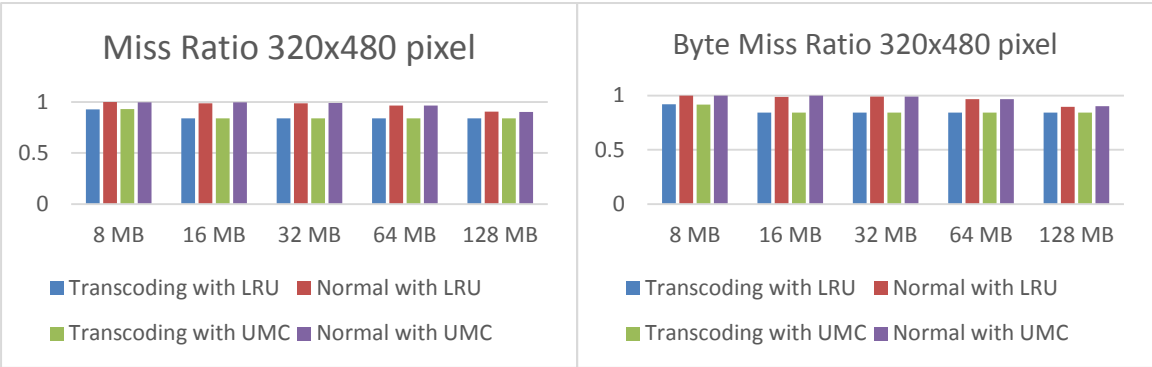
ตารางที่ 3 แสดงเวลาที่ใช้ในการปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลประเภทรูปภาพด้วยวิธีต่างๆ จากผลการทดลอง เราพบว่า การปรับเปลี่ยนการเข้ารหัสข้อมูลด้วยวิธีลดขนาดของภาพและบีบอัดแฟ้มข้อมูลใช้เวลา 0.072 วินาทีในระบบที่มีหน้าจอขนาด 320x480 pixels และ 0.178 วินาทีในระบบที่มีหน้าจอขนาด 640x960 pixels และวิธีบีบอัดแฟ้มข้อมูลเพียงอย่างเดียว ใช้เวลา 0.134 วินาที โดยเฉลี่ย

3 ผลการทดลองของระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลด้วยวิธีลดขนาดของรูปภาพและวิธีบีบอัดแฟ้มข้อมูลรูปภาพ



รูปที่ 7 จำนวนแฟ้มข้อมูลที่ถูกแคช

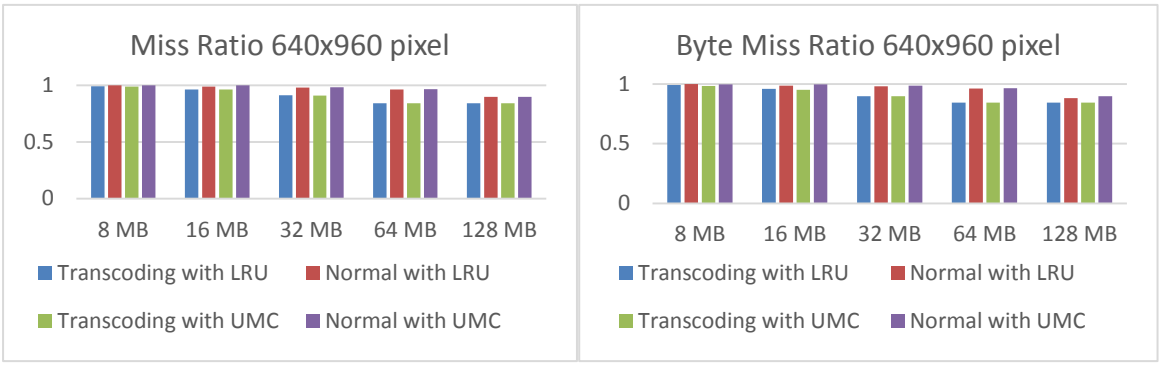
รูปที่ 7 จำนวนแฟ้มข้อมูลที่ถูกแคชแสดงจำนวนของแฟ้มข้อมูลที่ถูกแคชได้ในระบบ จากผลการทดลองเราพบว่า ระบบแคชแบบปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูล ช่วยเพิ่มจำนวนของแฟ้มข้อมูลที่ถูกแคชได้เป็นจำนวนมาก ซึ่งในการทดลองบนอุปกรณ์ที่มีขนาดหน้าจอ 320x480 pixels เราพบว่า ระบบแคชสามารถแคชแฟ้มข้อมูลทั้งหมดที่ผู้ใช้เคยเข้าถึงได้ในระบบแคชที่กำหนดขนาดของพื้นที่สำหรับเก็บแฟ้มข้อมูลเพียง 16 MB เท่านั้น ส่วนในการทดลองบนอุปกณณ์ที่มีขนาดหน้าจอ 640x960 pixels นั้น ระบบแคชสามารถแคชข้อมูลที่ผู้ใช้เคยเข้าถึงทั้งหมด เมื่อเรากำหนดขนาดพื้นที่สำหรับเก็บแฟ้มข้อมูลที่ 64 MB



รูปที่ 8 อัตราความผิดพลาดของการแคชข้อมูลในอุปรณ์ที่มีหน้าจอขนาด 320x480 pixels

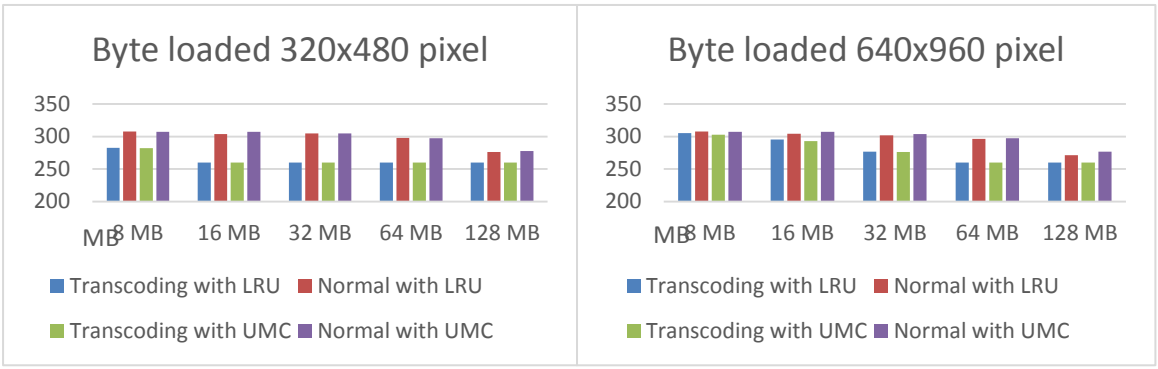
ในรูปที่ 8 อัตราความผิดพลาดของการแคชข้อมูลในอุปรณ์ที่มีหน้าจอขนาด 320x480 pixels เราพบว่า ระบบแคชแบบปรับเปลี่ยนรหัสแฟ้มข้อมูล สามารถทำงานได้ดีกว่าระบบแคชแบบปกติ ในทุกกรณี และเนื่องจาก ระบบแคชแบบปรับเปลี่ยนรหัสแฟ้มข้อมูลสามารถแคชแฟ้มข้อมูลทั้งหมดของผู้ใช้ได้ เมื่อกำหนดขนาดพื้นที่สำหรับแคชแฟ้มข้อมูลไว้ที่ตั้งแต่ 16 MB ขึ้นไป ดังนั้น เมื่อเปรียบเทียบการทำงานของทั้งสองระบบ

ระบบแคชแบบปรับเปลี่ยนรหัสแฟ้มข้อมูลมีประสิทธิภาพสูงกว่าระแบบแคชแบบปกติมากที่สุดในระบบที่กำหนดขนาดพื้นที่สำหรับแคชแฟ้มข้อมูลไว้ที่ 16 MB และมีประสิทธิภาพต่างกันลดลงเมื่อเราเพิ่มขนาดพื้นที่สำหรับแคชแฟ้มข้อมูล



รูปที่ 9 อัตราความผิดพลาดของการแคชข้อมูลในอุปรณ์ที่มีหน้าจอขนาด 640x960 pixels

ในรูปที่ 9 อัตราความผิดพลาดของการแคชข้อมูลในอุปรณ์ที่มีหน้าจอขนาด 640x960 pixels เราพบว่า ระบบแคชแบบปรับเปลี่ยนรหัสแฟ้มข้อมูลทำงานได้ดีกว่าระบบแคชแบบปกติในทุกกรณีเช่นกัน โดยระบบแคชแบบปรับเปลี่ยนรหัสแฟ้มข้อมูลมีประสิทธิภาพสูงกว่าระบบแคชแบบทั่วไปมากขึ้นเมื่อเราเพิ่มขนาดพื้นที่สำหรับแคชแฟ้มข้อมูล และมีประสิทธิภาพสูงกว่าระบบแคชแบบทั่วไปมากที่สุดเมื่อเรากำหนดขนาดพื้นที่สำหรับแคชแฟ้มข้อมูลไว้ที่ 64 MB และมีประสิทธิภาพที่ต่างกันลดลงเมื่อเราเพิ่มขนาดพื้นที่สำหรับแคชแฟ้มข้อมูลมากขึ้น

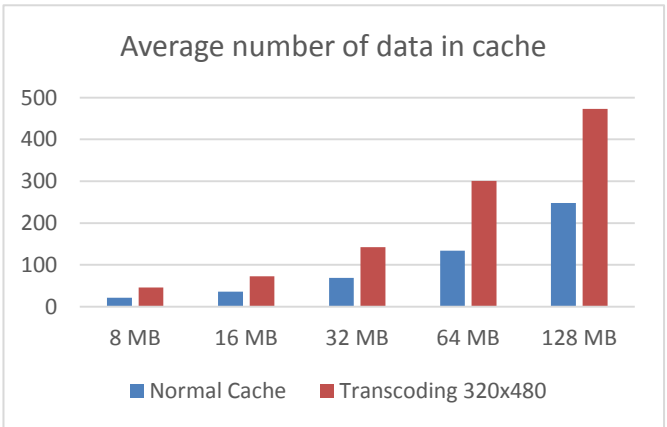


รูปที่ 10 ปริมาณข้อมูลที่ดึงผ่านระบบเครือข่าย

รูปที่ 11 ปริมาณข้อมูลที่ถูกดึงผ่านระบบเครือข่ายที่ระบบแคชช่วยลด

ใน รูปที่ 10 และรูปที่ 11 แสดงถึงปริมาณข้อมูลที่ดึงผ่านระบบเครือข่ายของระบบแคชทั้งสอบแบบและปริมาณข้อมูลที่ถูกดึงผ่านระบบเครือข่ายที่ระบบแคชช่วยลดเมื่อเปลี่ยนมาใช้ระบบแคชแบบปรับเปลี่ยนรหัสแฟ้มข้อมูล ซึ่งปริมาณข้อมูลที่ถูกดึงผ่านระบบเครือข่าย สอดคล้องกับอัตราความผิดพลาดของการแคชข้อมูล กล่าวคือ ระบบแคชแบบปรับเปลี่ยนรหัสแฟ้มข้อมูลนั้น สามารถลดปริมาณข้อมูลที่ถูกส่งผ่านระบบเครือข่ายได้ในทุกกรณี โดยจะสามารถลดปริมาณข้อมูลที่ถูกส่งผ่านระบบเครือข่ายได้มากที่สุดเมื่อเรากำหนดขนาดพื้นที่สำหรับแคชแฟ้มข้อมูลไว้ที่ 16 MB สำหรับอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะที่มีขนาดหน้าจอ 320x480 pixels และเมื่อเรากำกำหนดขนาดพื้นที่สำหรับแคชแฟ้มข้อมูลไว้ที่ 64 MB สำหรับอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะที่มีขนาดหน้าจอ 640x960 pixels ซึ่งขนาดดังกล่าวคือขนาดที่ระบบแคชแบบปรับเปลี่ยนรหัสแฟ้มข้อมูลจะสามารถแคชแฟ้มข้อมูลทั้งหมดของผู้ใช้ได้

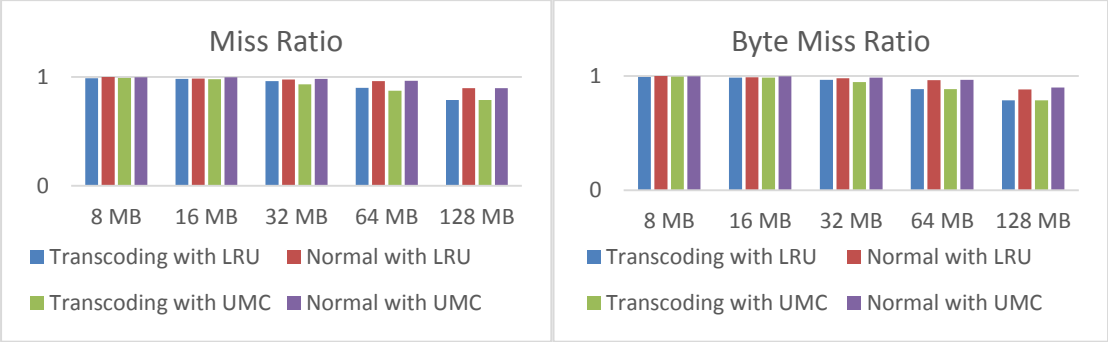
4 ผลการทดลองของระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลด้วยวิธีบีบอัดแฟ้มข้อมูลรูปภาพ



รูปที่ 12 จำนวนแฟ้มข้อมูลที่ถูกแคชในระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลด้วยวิธีบีบอัดแฟ้มข้อมูล

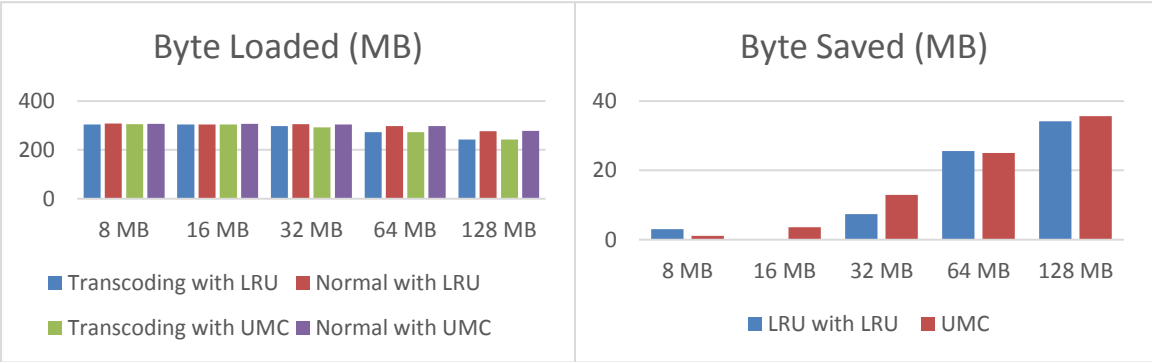
รูปที่ 12 จำนวนแฟ้มข้อมูลที่ถูกแคชในระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลด้วยวิธีบีบอัดแฟ้มข้อมูล แสดงจำนวนของแฟ้มข้อมูลที่ถูกแคชได้ในระบบ จากผลการทดลอง เราพบว่า ระบบแคชแบบปรับเปลี่ยนการ

เข้ารหัสของแฟ้มข้อมูล ช่วยเพิ่มจำนวนของแฟ้มข้อมูลที่ถูกแคช โดยเราพบว่าระบบแคชสามารถแคชแฟ้มข้อมูลทั้งหมดที่ผู้ใช้เคยเข้าถึงได้ในระบบแคชที่กำหนดขนาดของพื้นที่สำหรับเก็บแฟ้มข้อมูลที่ 128 MB



รูปที่ 13 อัตราความผิดพลาดของการแคชข้อมูลในระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลด้วยวิธีบีบอัดแฟ้มข้อมูล

ในรูปที่ 13 อัตราความผิดพลาดของการแคชข้อมูลในระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลด้วยวิธีบีบอัดแฟ้มข้อมูล เราพบว่า ระบบแคชแบบปรับเปลี่ยนรหัสแฟ้มข้อมูล สามารถทำงานใกล้เคียงกับระบบแคชแบบปกติในระบบแคชที่กำหนดขนาดพื้นที่สำหรับแคชไว้ที่ 8 MB และ 16 MB และสามารถทำงานดีกว่าระบบแคชแบบปกติในระบบแคชที่กำหนดขนาดพื้นที่สำหรับแคชไว้ที่ 32 MB ขึ้น และมีประสิทธิภารสูงกว่าระบบแคชแบบทั่วไปมากที่สุดในระบบแคชที่กำหนดขนาดพื้นที่สำหรับแคชไว้ที่ 128 MB ซึ่งเป็นขนาดที่ระบบแคชแบบปรับเปลี่ยนรหัสแฟ้มข้อมูลด้วยวิธีบีบอัดข้อมูลสามารถแคชข้อมูลทั้งหมดของผู้ใช้ได้



รูปที่ 14 ปริมาณข้อมูลที่ดึงผ่านระบบเครือข่ายและปริมาณข้อมูลที่ระบบแคชแบบปรับเปลี่ยนรหัสแฟ้มข้อมูลด้วยวิธีบีบอัดข้อมูลช่วยลด

ในรูปที่ 14 แสดงถึงปริมาณข้อมูลที่ดึงผ่านระบบเครือข่ายของระบบแคชทั้งสอบแบบและปริมาณข้อมูลที่ถูกดึงผ่านระบบเครือข่ายที่ระบบแคชช่วยลดเมื่อเปลี่ยนมาใช้ระบบแคชแบบปรับเปลี่ยนรหัสแฟ้มข้อมูลด้วยวิธีบีบอัดข้อมูล ซึ่งปริมาณข้อมูลที่ถูกดึงผ่านระบบเครือข่ายสอดคล้องกับอัตราความผิดพลาดของการแคชข้อมูล กล่าวคือ ในระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลที่กำหนดขนาดพื้นที่สำหรับแคชข้อมูลไว้ที่ 8 MB และ 16 MB นั้น ระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลสามารถลดปริมาณข้อมูลที่ถูกส่งผ่านระบบเครือข่ายได้น้อย และในระบบแคชที่กำหนดขนาดพื้นที่สำหรับแคชข้อมูลไว้ที่ 32 MB ขึ้นไป ระบบแคชสามารถช่วยลดปริมาณ

ข้อมูลที่ถูกส่งผ่านระบบเครือข่ายได้ โดยสามารถลดปริมาณข้อมูลที่ถูกส่งผ่านระบบเครือข่ายได้มากที่สุดเมื่อเรากำหนดขนาดพื้นที่สำหรับแคชแฟ้มข้อมูลไว้ที่ 128 MB ซึ่งขนาดดังกล่าวคือขนาดที่ระบบแคชแบบปรับเปลี่ยนรหัสแฟ้มข้อมูลจะสามารถแคชแฟ้มข้อมูลทั้งหมดของผู้ใช้ได้

5 ผลการทดลองความเร็วที่ใช้เข้าถึงข้อมูลของระบบแคชโดยเฉลี่ย

ตารางที่ 4 ตารางแสดงเวลาที่ใช้เข้าถึงข้อมูลของระบบแคชโดยเฉลี่ย

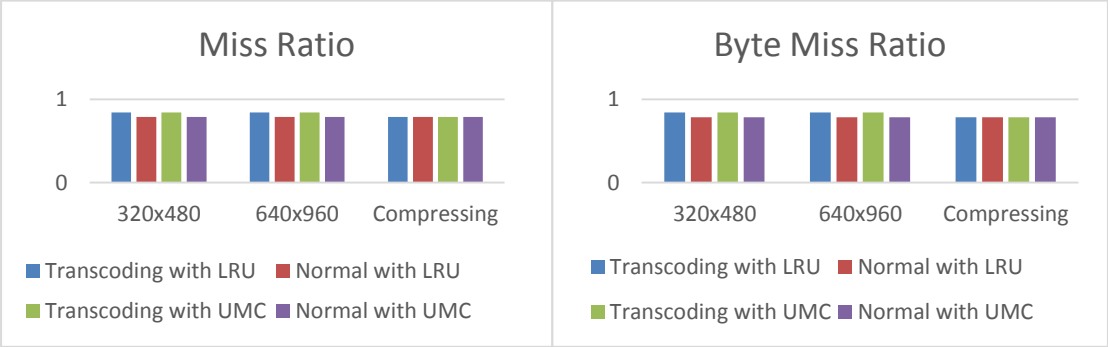
ระบบแคช	ระบบแคชแบบทั่วไป		ระบบแคชแบบปรับเปลี่ยนรหัสแฟ้มข้อมูลด้วยวิธีลดขนาดภาพและบีบอัดข้อมูล		ระบบแคชแบบปรับเปลี่ยนรหัสแฟ้มข้อมูลด้วยวิธีบีบอัดข้อมูล	
	ไม่พบข้อมูล	พบข้อมูล	ไม่พบข้อมูล	พบข้อมูล	ไม่พบข้อมูล	พบข้อมูล
เวลาที่ใช้เข้าถึงข้อมูลโดยเฉลี่ย	2.15134	0.01202	2.39943	0.00627	2.2637	0.00996

ตารางที่ 4 ตารางแสดงเวลาที่ใช้เข้าถึงข้อมูลของระบบแคชโดยเฉลี่ย จากผลการทดลองเราพบว่า ในกรณีที่ไม่พบข้อมูลที่ผู้ใช้ต้องการเข้าถึงในระบบแคช ระบบแคชทั้งสามแบบใช้เวลาในการเข้าถึงข้อมูลที่ใกล้เคียงกัน ทั้งนี้เวลาที่ใช้แตกต่างกันไปขึ้นกับปัจจัยต่างๆ โดยเฉพาะปัจจัยของระบบเครือข่ายที่ใช้อยู่ ณ ขณะนั้น เช่น ความเร็วในการรับส่งข้อมูลของระบบเครือข่าย (bandwidth) ความเร็วในการเข้าถึงข้อมูลของระบบเครือข่าย (latency) เป็นต้น

ในกรณีที่พบข้อมูลที่ผู้ใช้ต้องการเข้าถึงในระบบแคช ระบบแคชแบบปรับเปลี่ยนรหัสแฟ้มข้อมูลด้วยวิธีลดขนาดภาพและบีบอัดข้อมูลใช้เวลาเข้าถึงข้อมูลของผู้ใช้น้อยที่สุด เนื่องจากแฟ้มข้อมูลที่ถูกแคชในระบบแคชแบบปรับเปลี่ยนรหัสแฟ้มข้อมูลด้วยวิธีลดขนาดภาพและบีบอัดข้อมูลนั้นมีขนาดเล็ก ทำให้ระบบแคชใช้เวลาดึงข้อมูลจากหน่วยความจำสำรองที่ใช้เก็บแฟ้มข้อมูลที่ถูกแคชอยู่น้อยกว่าระบบแคชแบบอื่น และด้วยเหตุผลเดียวกันระบบแคชแบบทั่วไปใช้เวลาเข้าถึงข้อมูลของผู้ใช้มากที่สุด เนื่องจากแฟ้มข้อมูลที่ถูกแคชในระบบแคชแบบทั่วไปมีขนาดของแฟ้มข้อมูลที่ถูกแคชใหญ่ที่สุด

6 ผลการทดลองของระบบแคชเมื่อระบบแคชสามารถแคชข้อมูลทั้งหมดของผู้ใช้ได้

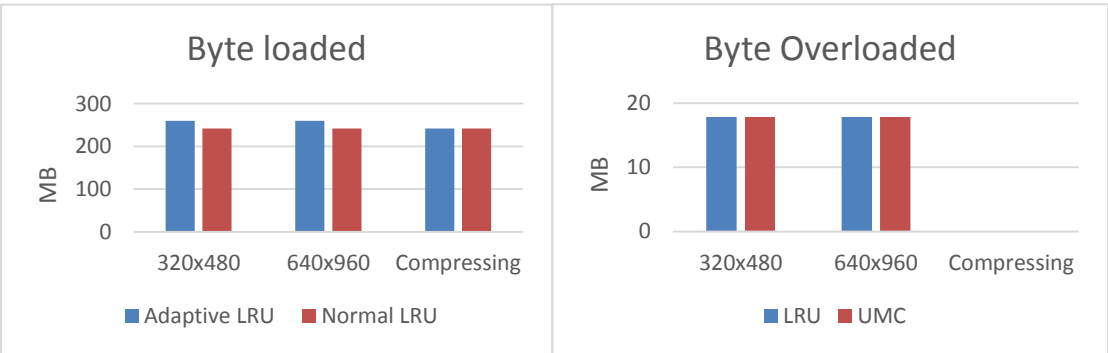
ผู้จัดทำได้ทดลองการทำงานของระบบแคชแบบต่างๆในกรณีที่ระบบแคชสามารถแคชข้อมูลทั้งหมดของผู้ใช้ได้ โดยกำหนดขนาดพื้นที่สำหรับแคชแฟ้มข้อมูลของผู้ใช้ไว้ที่ 16 MB สำหรับระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลด้วยวิธีลดขนาดของภาพและบีบอัดข้อมูลบนอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะที่มีหน้าจอขนาด 320x480 pixels ขนาด 64 MB สำหรับระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลด้วยวิธีลดขนาดของภาพและบีบอัดข้อมูลบนอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะที่มีหน้าจอขนาด 640x960 pixels ขนาด 128 MB สำหรับระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลด้วยวิธีบีบอัดข้อมูล และ 256 MB สำหรับระบบแคชแบบทั่วไป โดยได้ผลการทดลองดังนี้



รูปที่ 15 อัตราความผิดพลาดของการแคชข้อมูลในระบบแคชในกรณีที่ระบบแคชสามารถแคชข้อมูลทั้งหมดของผู้ใช้ได้

ในรูปที่ 15 แสดงถึงอัตราความผิดพลาดของการแคชข้อมูลในระบบแคชในกรณีที่ระบบแคชสามารถแคชข้อมูลทั้งหมดของผู้ใช้ได้ จากผลการทดลองเราพบว่า ระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลด้วยวิธีลดขนาดของภาพและบีบอัดข้อมูลมีประสิทธิภาพน้อยกว่าระบบแคชแบบทั่วไป เนื่องจากเมื่อผู้ใช้ต้องการเข้าถึงข้อมูลต้นฉบับ ระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลด้วยวิธีลดขนาดของภาพและบีบอัดข้อมูลจะไม่สามารถคืนข้อมูลต้นฉบับให้ผู้ใช้ได้ ในขณะที่ระบบแคชแบบทั่วไปสามารถคืนข้อมูลต้นฉบับให้ผู้ใช้ได้ ทำให้ระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลด้วยวิธีลดขนาดของภาพและบีบอัดข้อมูลมีประสิทธิภาพโดยรวมน้อยกว่าระบบแคชแบบทั่วไปในกรณีนี้

ในระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลด้วยวิธีบีบอัดข้อมูลนั้นมีประสิทธิภาพเท่ากับระบบแคชแบบทั่วไป เนื่องจากระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลด้วยวิธีบีบอัดข้อมูลมีการทำงานในกรณีที่ผู้ใช้ต้องการเข้าถึงข้อมูลต้นฉบับเหมือนกับระบบแคชแบบทั่วไป ทำให้ระบบแคชระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลด้วยวิธีบีบอัดข้อมูลมีประสิทธิภาพการทำงานเท่ากับระบบแคชแบบทั่วไปในกรณีนี้



รูปที่ 16 ปริมาณข้อมูลที่ดึงผ่านระบบเครือข่ายและปริมาณข้อมูลที่ระบบใช้มากกว่าในกรณีที่ระบบแคชสามารถแคชข้อมูลทั้งหมดของผู้ใช้ได้

ในรูปที่ 16 แสดงถึงปริมาณข้อมูลที่ดึงผ่านระบบเครือข่ายและปริมาณข้อมูลที่ระบบใช้มากกว่าในกรณีที่ระบบแคชสามารถแคชข้อมูลทั้งหมดของผู้ใช้ได้ จากผลการทดลองเราพบว่า ระบบแคชแบบปรับเปลี่ยน

รหัสข้อมูลของแฟ้มข้อมูลด้วยวิธีลดขนาดของภาพและบีบอัดข้อมูลดึงข้อมูลผ่านระบบเครือข่ายมากกว่าระบบแคชแบบทั่วไป และระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลด้วยวิธีลดบีบอัดข้อมูลดึงข้อมูลผ่านระบบเครือข่ายเท่ากับระบบแคชแบบทั่วไป ซึ่งสอดคล้องกับผลการทดลองอัตราความผิดพลาดของการแคชข้อมูลของระบบแคช

7 การคำนวนการใช้พลังงานของระบบแคช

ในส่วนนี้ ผู้จัดทำเสนอการคำนวนพลังงานที่ใช้ในระบบแคช ซึ่งปริมาณพลังงานที่ใช้ระบบแคชสามารถคำนวนได้ด้วยสูตรดังต่อไปนี้

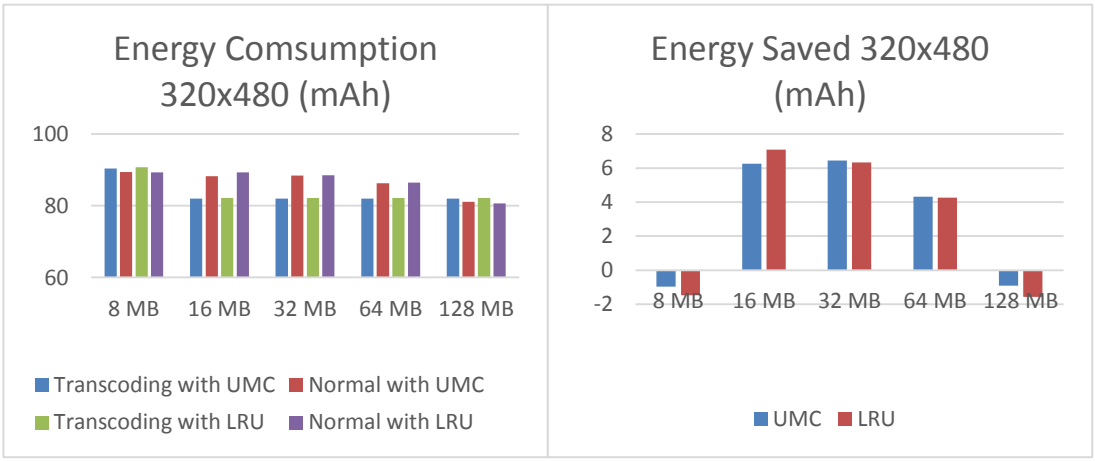
$$E_{ts} = M_r \; x \; (E_t + \; E_n) \; x \; C_r$$

$$E_{ns} = M_r \; x \; E_t \; x \; C_r$$

โดยตัวแปรต่างๆในสูตรคำนวนมีดังนี้

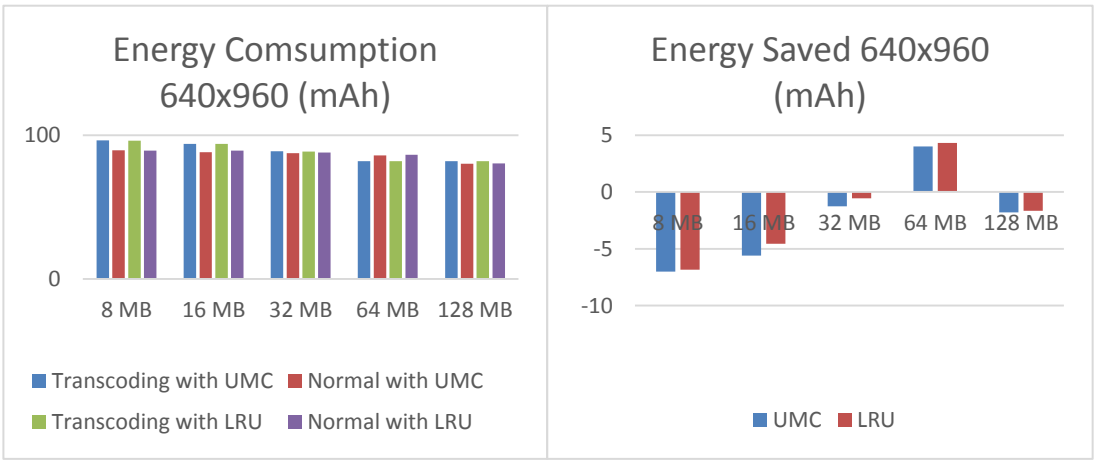
- E_{ts} คือปริมาณพลังงานที่ระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลใช้ทำงาน
- E_{ns} คือปริมาณพลังงานที่ระบบแคชแบบทั่วไปใช้ทำงาน
- M_r คืออัตราความผิดพลาดของระบบแคช
- E_t คือปริมาณพลังงานที่ระบบแคชใช้ปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูล
- E_n คือปริมาณพลังงานที่ระบบแคชใช้ดึงแฟ้มข้อมูลผ่านระบบเครือข่าย
- C_r คือจำนวนการเข้าถึงข้อมูลของผู้ใช้

ซึ่งการคำนวน ได้ผลดังนี้



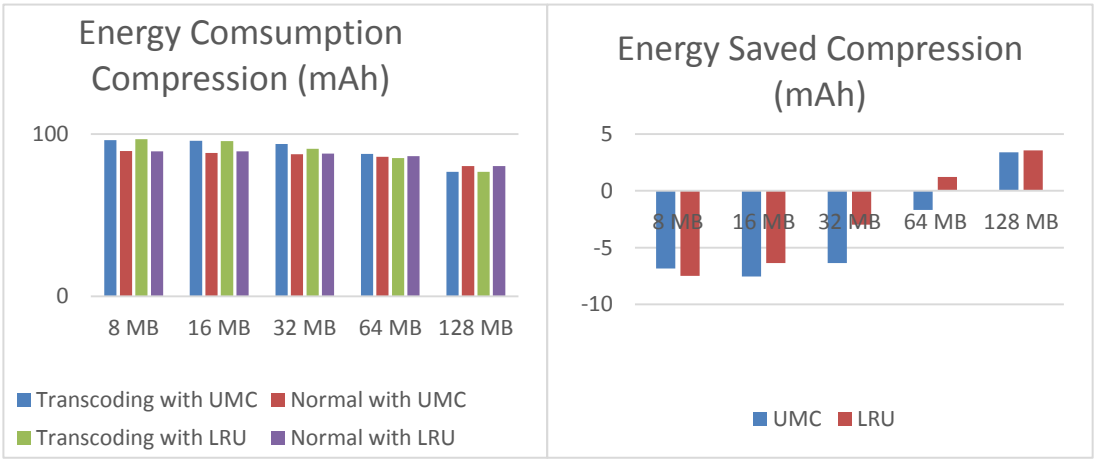
รูปที่ 17 ปริมาณพลังงานที่ใช้ทำงานสำหรับระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลด้วยวิธีลดขนาดของภาพและบีบอัดข้อมูลบนอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะที่มีขนาดหน้าจอขนาด 320x480 pixels และปริมาณพลังงานของระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลช่วยลดเมื่อเทียบระบบแคชแบบทั่วไป

ในรูปที่ 17 แสดงถึงปริมาณพลังงานที่ใช้ทำงานสำหรับระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลด้วยวิธีลดขนาดของภาพและบีบอัดข้อมูลบนอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะที่มีขนาดหน้าจอขนาด 320x480 pixels และปริมาณพลังงานของระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลช่วยลดเมื่อเทียบระบบแคชแบบทั่วไป จากผลการทดลองพบว่า ระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลสามารถลดปริมาณการใช้พลังงานของอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะเมื่อเทียบกับการใช้พลังงานของระบบแคชแบบทั่วไปเกือบทุกกรณี โดยมีระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลที่กำหนดขนาดพื้นที่สำหรับแคชข้อมูลไว้ที่ 8 MB และ 128 MB ที่ใช้พลังงานมากกว่าระบบแคชแบบทั่วไปเล็กน้อย และสามารถลดปริมาณการใช้พลังงานได้มากที่สุดในระบบแคชที่กำหนดขนาดสำหรับแคชแฟ้มข้อมูลของผู้ใช้ไว้ที่ 16 MB



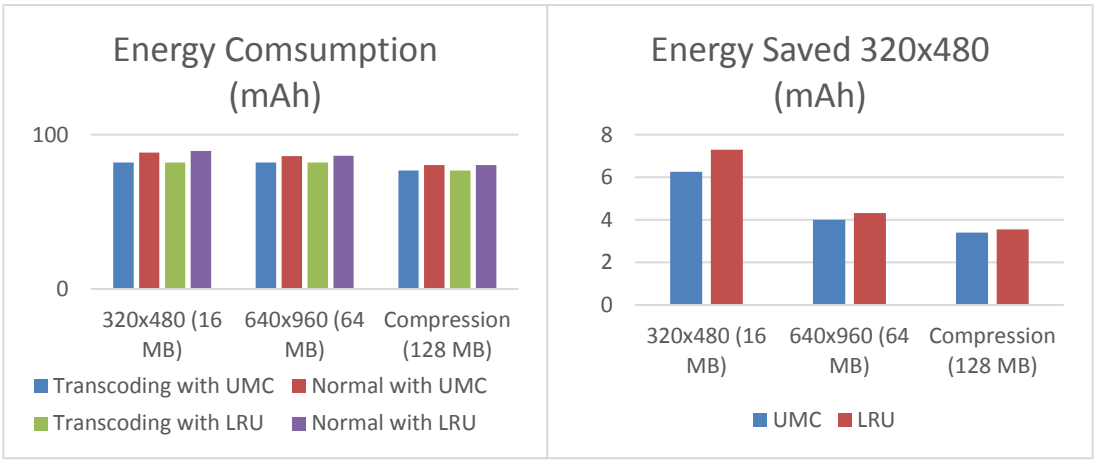
รูปที่ 18 ปริมาณพลังงานที่ใช้ทำงานสำหรับระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลด้วยวิธีลดขนาดของภาพและบีบอัดข้อมูลบนอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะที่มีหน้าจอขนาด 640x960 pixels และปริมาณพลังงานของระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลช่วยลดเมื่อเทียบระบบแคชแบบทั่วไป

ในรูปที่ 18 แสดงถึงปริมาณพลังงานที่ใช้ทำงานสำหรับระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลด้วยวิธีลดขนาดของภาพและบีบอัดข้อมูลบนอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะที่มีหน้าจอขนาด 640x960 pixels และปริมาณพลังงานของระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลช่วยลดเมื่อเทียบระบบแคชแบบทั่วไป จากผลการทดลองพบว่า ระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลใช้ปริมาณพลังงานของอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะมากกว่าเมื่อเทียบกับการใช้พลังงานของระบบแคชแบบทั่วไปเล็กน้อยในหลายกรณี แต่ระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลสามารถลดปริมาณการใช้พลังงานได้มากที่สุดในระบบแคชที่กำหนดขนาดสำหรับแคชแฟ้มข้อมูลของผู้ใช้ไว้ที่ 64 MB



รูปที่ 19 ปริมาณพลังงานที่ใช้ทำงานสำหรับระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลด้วยวิธีบีบอัดข้อมูล และปริมาณพลังงานของระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลช่วยลดเมื่อเทียบระบบแคชแบบทั่วไป

ในรูปที่ 19 แสดงถึงปริมาณพลังงานที่ใช้ทำงานสำหรับระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลด้วยวิธีบีบอัดข้อมูล และปริมาณพลังงานของระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลช่วยลดเมื่อเทียบระบบแคชแบบทั่วไป จากผลการทดลองพบว่า ระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลใช้พลังงานมากกว่าระบบแคชแบบทั่วไปเล็กน้อยในหลายกรณี และสามารถลดปริมาณการใช้พลังงานของอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะเมื่อเทียบกับการใช้พลังงานของระบบแคชแบบทั่วไปในระบบแคชที่กำหนดขนาดสำหรับแคชแฟ้มข้อมูลของผู้ใช้ไว้ที่ 128 MB



รูปที่ 20 ปริมาณพลังงานที่ใช้ทำงานสำหรับระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลสามารถลดได้สูงสุด และปริมาณพลังงานของระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลช่วยลดเมื่อเทียบระบบแคชแบบทั่วไป

ในรูปที่ 20 แสดงปริมาณพลังงานที่ใช้ทำงานสำหรับระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลสามารถลดได้สูงสุด และปริมาณพลังงานของระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลช่วยลดเมื่อเทียบระบบแคชแบบทั่วไป จากผลการทดลองพบว่า ระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลด้วยวิธีบีบอัดข้อมูลมีการใช้พลังงานน้อยที่สุด

และ ระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลด้วยวิธีลดขนาดของภาพและบีบอัดข้อมูลบนอุปกรณ์โทรศัพท์ที่เคลื่อนที่อัจฉริยะที่มีหน้าจอขนาด 320x480 pixels สามารถประหยัดพลังงานจากระบบแคชแบบทั่วไปมากที่สุดในระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลต่างๆ

บทที่ 6
สรุปผลงานวิจัย

สิ่งที่ได้จากงานวิจัย

งานวิจัยนี้ เราได้วิจัยถึงวิธีการใหม่ของการแคชข้อมูลในอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะ โดยเราได้ทดลองปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลก่อนถูกแคช ซึ่งวิธีการปรับเปลี่ยนรหัสข้อมูลดังกล่าว ถูกอิงจากพฤติกรรมของผู้ใช้ โดยการปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลนั้นจะช่วยลดขนาดของแฟ้มข้อมูลที่ถูกเก็บ และเพิ่มจำนวนของแฟ้มข้อมูลที่ถูกแคชในระบบได้

ผลของงานวิจัยโดยสรุป ระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลนั้น มีประสิทธิภาพดีกว่าระบบแคชแบบทั่วไปเมื่อพิจารณาในแง่ปริมาณข้อมูลที่ถึงดึงผ่านระบบเครือข่ายและปริมาณพลังงานที่ใช้ โดยมีความแตกต่างของประสิทธิภาพในการทำงานของระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูล เมื่อเทียบกับระบบแคชแบบทั่วไปขึ้นอยู่กับกระบวนการปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูล ขนาดของหน้าจอของอุปกรณ์โทรศัพท์เคลื่อนที่อัจฉริยะ และขนาดของพื้นที่สำหรับแคชแฟ้มข้อมูล

โดยวิธีลดขนาดของรูปภาพและบีบอัดข้อมูลทำให้ระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลลดปริมาณการดึงข้อมูลผ่านระบบเครือข่าย และวิธีบีบอัดข้อมูลใช้ปริมาณพลังงานในการทำงานน้อยที่สุด แต่วิธีทั้งสองมีความแตกต่างของปริมาณพลังงานใช้ทำงานต่างกันน้อยมาก และวิธีลดขนาดของรูปภาพและบีบอัดข้อมูลดึงข้อมูลผ่านระบบเครือข่ายน้อยที่สุด ซึ่งเมื่อเราคำนึงถึงค่าบริการระบบเครือข่ายที่ผู้ใช้ต้องเสีย และผลการทดลองในระบบแคชที่กำหนดขนาดพื้นที่สำหรับแคชข้อมูลที่มีขนาดเล็กด้วยแล้ว ทำให้วิธีลดขนาดของภาพและบีบอัดข้อมูลจึงเหมาะสำหรับนำมาประยุกต์ใช้ในระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูลมากที่สุด

งานที่สามารถต่อยอดได้ในอนาคต

ระบบแคชแบบปรับเปลี่ยนรหัสข้อมูลสามารถต่อยอดได้ เพื่อเพิ่มประสิทธิภาพของการทำงานของระบบ โดยมีแนวทางดังนี้

1. ออกแบบอัลกอริทึมการแทนที่ ที่คำนึงถึงลักษณะการทำงานของระบบแคชแบบปรับเปลี่ยนรหัสข้อมูล เพื่อให้ระบบแคชทำงานได้อย่างมีประสิทธิภาพสูงสุด
2. ในงานวิจัยนี้ เราใช้กระบวนการปรับเปลี่ยนรหัสข้อมูลที่มีอยู่ในระบบปฏิบัติการ เราสามารถเปลี่ยนกระบวนการปรับเปลี่ยนรหัสข้อมูล เพื่อให้ได้ผลของการปรับเปลี่ยนรหัสข้อมูลที่ดีขึ้น [19]
3. เราสามารถเปลี่ยนวิธีการบีบอัดข้อมูลของแฟ้มข้อมูล เพื่อลดขนาดของแฟ้มข้อมูลลง ในขณะที่ยังคงคุณภาพของแฟ้มข้อมูลได้ เช่น การใช้อัลกอริทึมการบีบอัดแฟ้มข้อมูลรูปภาพแบบ JPEG2000 แทน JPEG [20] [21]

รายการอ้างอิง

[1] Gartner, Inc. Gartner, Inc. [Online]. 2013, April Available from: http://www.gartner.com/newsroom/id/2408515 [2013, April]

[2] Weihua Wu. Mobile OS Architecture Trends. [Online]. 2013, Jan. Available from: http://software.intel.com/en-us/articles/mobile-os-architecture-trends [2013, April]

[3] Gernot Heiser Aaron Carroll, "An Analysis of Power Consumption in a Smartphone," in *USENIX Association Berkeley*, CA, 2010.

[4] W3C. Portable Network Graphics (PNG) Specification. [Online]. 2003, November Available from: http://www.w3.org/TR/PNG [2012, December]

[5] Adobe Systems Incorporated. TIFF. [Online]. 2009 Available from: http://partners.adobe.com/public/developer/tiff/index.html [2012, December]

[6] Elysium Ltd. JPEG 2000. [Online]. 2007 Available from: http://www.jpeg.org/jpeg2000/ [2012, December]

[7] Overview of the H.264/AVC Video Coding Standard. [Online]. 2003, July Available from: http://ip.hhi.de/imagecom_G1/assets/pdfs/csvt_overview_0305.pdf [2012, December]

[8] Josh Coalson. FLAC Documentation. [Online]. 2008 Available from: http://flac.sourceforge.net/documentation.html [2012, December]

[9] Apple Inc. Apple Lossless Audio Codec. [Online]. 2011, October Available from: http://alac.macosforge.org/ [2012, December]

[10] Salomon David, "Data Compression: The Complete Reference," in *Data Compression: The Complete Reference*., 2007, p. 241.

[11] Matthias Rose. The mp3 History. [Online]. Available from: http://www.mp3-history.com/ [2012, December]

[12] KARLHEINZ BRANDENBURG. MP3 AND AAC EXPLAINED. [Online]. Available from: http://telos-systems.com/techtalk/hosted/Brandenburg_mp3_aac.pdf [2012, December]

[13] Vincent Bockaert. Interpolation. [Online]. Available from:

http://www.dpreview.com/glossary/digital-imaging/interpolation [2012, December]

[14] Johannes Kopf and Dani Lischinski, "Depixelizing Pixel Art," *ACM Transactions on Graphics*, vol. 30, no. 4, pp. 99:1-99:8, 2011.

[15] Mark Nottingham. Caching Tutorial. [Online]. 2012, February Available from: http://www.mnot.net/cache_docs/ [2012, December]

[16] Ulrich Drepper. What Every Programmer Should Know About Memory. [Online]. 2007, November Available from: http://www.unilim.fr/sci/wiki/_media/cali/cpumemory.pdf [2012, December]

[17] Yuanyuan Zhou. Definitions of various cache algorithms. [Online]. 2001, Apr. Available from: http://static.usenix.org/events/usenix01/full_papers/zhou/zhou_html/node3.html [2013, Jan.]

[18] Ahmed Amer, Panos K. Chrysanthis Ganesh Santhanakrishnan, "Towards Universal Mobile Caching," , Maryland, 2005.

[19] Ariel Shamir Shai Avidan, "Seam carving for content-aware image resizing," *ACM Transactions on Graphics*, vol. 26, no. 3, 2007.

[20] R. Colin Johnson. JPEG2000 wavelet compression spec approved. [Online]. 1999, December Available from: http://eetimes.com/electronics-news/4168829/JPEG2000-wavelet-compression-spec-approved [2013, January]

[21] Raphaël Grosbois, Touradj Ebrahimi Diego Santa-Cruz, "JPEG 2000 performance evaluation and assessment," *Signal Processing: Image Communication*, vol. 17, no. 1, pp. 113-130, January 2002.

ภาคผนวก

ภาคผนวก ก ซอร์สโคดของระบบแคชปรับเปลี่ยนรหัสข้อมูลของแฟ้มข้อมูล

ก.1 ซอร์สโคดส่วนของการเชื่อมต่อการทำงานกับระบบปฏิบัติการ

```
//
// PBImageURLLoadingProtocol.m
// Photo Browser
//
// Created by Pitiphong Phongpattranont on 10/11/55 BE.
// Copyright (c) 2555 Pitiphong Phongpattranont. All rights reserved.
//

#import "PBImageURLLoadingProtocol.h"

#import "PBPhoto.h"
#import "PBCacheManager.h"


@interface PBImageURLLoadingProtocol ()

- (void)finishLoadingWithData:(NSData *)loadedData
         didLoadFromNetwork:(BOOL)didLoadFromNetwork
                   response:(NSURLResponse *)response
                      error:(NSError *)error;

+ (NSOperationQueue *)downloadImageQueue;

@end


@implementation PBImageURLLoadingProtocol

+ (BOOL)canInitWithRequest:(NSURLRequest*)theRequest
{
  return [NSURLProtocol propertyForKey:PBPhotoIDURLKey inRequest:theRequest] != nil;
}

+ (NSURLRequest *)canonicalRequestForRequest:(NSURLRequest *)theRequest
{
  return theRequest;
}

+ (BOOL)requestIsCacheEquivalent:(NSURLRequest *)a toRequest:(NSURLRequest *)b {
  return NO;
}

- (void)startLoading
{
  NSString *photoID = [NSURLProtocol propertyForKey:PBPhotoIDURLKey inRequest:self.request];
  NSNumber *isAccessFullSize = [NSURLProtocol propertyForKey:PBAccessingFullSizeURLKey inRequest:self.request];
  NSData *cachedData = [PBCacheManager.defaultManager cachedDataForPhotoID:photoID
                                                    accessFullSize:isAccessFullSize.boolValue];
  if (cachedData) {
    NSURLResponse *response = nil;
    response = [[NSURLResponse alloc] initWithURL:[self.request URL]
                                         MIMEType:@"image/jpeg"
                            expectedContentLength:cachedData.length
                                 textEncodingName:nil];
    [self finishLoadingWithData:cachedData
             didLoadFromNetwork:NO
                       response:response
                          error:nil];
  } else {
    NSMutableURLRequest *imageHTTPRequest = [self.request mutableCopy];
    [NSURLProtocol removePropertyForKey:PBPhotoIDURLKey inRequest:imageHTTPRequest];
    [NSURLConnection sendAsynchronousRequest:imageHTTPRequest
                                       queue:self.class.downloadImageQueue
                           completionHandler:^(NSURLResponse *response, NSData *data, NSError *error) {
                             [PBCacheManager.defaultManager setData:data
                                                       forPhotoID:photoID];
                             [self finishLoadingWithData:data
                                      didLoadFromNetwork:YES
                                                response:response
                                                   error:error];
```

```
                    }];
  }
}

- (void)stopLoading
{
}


#pragma mark - Private methods

- (void)finishLoadingWithData:(NSData *)loadedData
          didLoadFromNetwork:(BOOL)didLoadFromNetwork
                    response:(NSURLResponse *)response
                       error:(NSError *)error {
  if (loadedData) {
    NSString *photoID = [NSURLProtocol propertyForKey:PBPhotoIDURLKey inRequest:self.request];
    NSNumber *accessingFullSize = [NSURLProtocol propertyForKey:PBAccessingFullSizeURLKey
                                                      inRequest:self.request];
    [PBCacheManager.defaultManager accessPhotoWithPhotoID:photoID
                                        didLoadData:didLoadFromNetwork
                                           fullSize:accessingFullSize.boolValue];
    [[self client] URLProtocol:self didReceiveResponse:response
          cacheStoragePolicy:NSURLCacheStorageNotAllowed];
    [[self client] URLProtocol:self didLoadData:loadedData];
  } else {
    [self.client URLProtocol:self didFailWithError:error];
  }

  [[self client] URLProtocolDidFinishLoading:self];
}


#pragma mark - Class methods

+ (NSOperationQueue *)downloadImageQueue {
  static NSOperationQueue *downloadImageQueue;
  static dispatch_once_t onceToken;
  dispatch_once(&onceToken, ^{
    downloadImageQueue = [[NSOperationQueue alloc] init];
    [downloadImageQueue setMaxConcurrentOperationCount:8];
  });

  return downloadImageQueue;
}


@end
```

ก.2 ซอร์สโคดส่วนระบบการทำงานของระบบแคชแบบปรับเปลี่ยนการเข้ารหัสของแฟ้มข้อมูล

```
//
//  PBCacheManager.m
//  Photo Browser
//
//  Created by Pitiphong Phongpattranont on 8/11/55 BE.
//  Copyright (c) 2555 Pitiphong Phongpattranont. All rights reserved.
//

#import "PBCacheManager.h"

#import "PBAccessEntry.h"
#import "PBCacheEntry.h"

#import "UIImage+PBResizing.h"
#import "PPBLRUCache.h"
#import "PPBUMCCache.h"
#import "PPBCompressTranscoding.h"
#import "PPBResizeCompressTranscoding.h"
#import "PPBNoneTranscoding.h"

#import <mach/mach_time.h>
#import <mach/mach.h>


const unsigned long long PBDefaultCacheFileSize = 128 * 1024 * 1024;
```

```
NSString * const PBFullSizedImageSuffixName                 = @"-full";
NSString * const PBCacheSizedImageSuffixName                = @"-cached";

inline BOOL         cacheManagerIsCachedDataNotValidated(id<PPBTranscoding> transcoding, PBCacheMode mode) {
 return !transcoding.isTranscodeDataValidForOriginal && (mode == PBCacheAdaptiveLRUMode || mode == PBCacheAdaptiveUMCMode);
};

inline NSURL * cacheDataURLForMode(id<PPBTranscoding> transcoding, PBCacheMode mode, NSInteger cacheSize) {
 NSURL *cachingDataStoreURL = [[[NSFileManager defaultManager] URLsForDirectory:NSDocumentDirectory
                                                      inDomains:NSUserDomainMask] lastObject];

 NSString *cacheName = @"Photo_Browser_Cache";
 NSString *modeSuffix = nil;
 switch (mode) {
   case PBCacheAdaptiveLRUMode:
     modeSuffix = @"-Adaptive-LRU";
     break;
   case PBCacheNormalLRUMode:
     modeSuffix = @"-Normal-LRU";
     break;
   case PBCacheAdaptiveUMCMode:
     modeSuffix = @"-Adaptive-UMC";
     break;
   case PBCacheNormalUMCMode:
     modeSuffix = @"-Normal-UMC";
     break;

   default:
     break;
 }

 cacheName = [[NSString stringWithFormat:@"%@-%@-%d%@", transcoding.transcodingName, cacheName, cacheSize, modeSuffix] stringByAppendingPathExtension:@"sqlite"];
 cachingDataStoreURL = [cachingDataStoreURL URLByAppendingPathComponent:cacheName];

 return cachingDataStoreURL;
};


@interface PBCacheManager ()

@property (strong)   id<PPBCacheAlgorithm> algorithm;
@property (strong)   NSManagedObjectContext *context;
@property (readonly, strong, nonatomic) NSManagedObjectModel *managedObjectModel;
@property (readonly, strong, nonatomic) NSPersistentStoreCoordinator *persistentStoreCoordinator;
@property (readonly, strong, nonatomic) NSMutableArray *resizingTimes;

@property (strong) id<PPBTranscoding> trandcoding;

- (PBCacheEntry *)cacheEntryForPhotoID:(NSString *)photoID;

+ (NSString *)photoCachePath;

@end


@implementation PBCacheManager

@synthesize managedObjectModel = _managedObjectModel;
@synthesize persistentStoreCoordinator = _persistentStoreCoordinator;

- (void)setCacheMode:(PBCacheMode)cacheMode {
 [self setCacheMode:cacheMode resetCacheData:NO];
}

- (void)setCacheMode:(PBCacheMode)cacheMode resetCacheData:(BOOL)resetCacheData {
 if (self->_cacheMode != cacheMode) {
   self->_cacheMode = cacheMode;

   id<PPBCacheAlgorithm> algorithm = nil;
   switch (self.cacheMode) {
     case PBCacheAdaptiveLRUMode:
     case PBCacheNormalLRUMode: {
       algorithm = [[PPBLRUCache alloc] init];
       break;
     }

     case PBCacheAdaptiveUMCMode:
     case PBCacheNormalUMCMode: {
       algorithm = [[PPBUMCCache alloc] init];
```

```
    }

    default:
      break;
  }

  self.algorithm = algorithm;

  [self.context performBlockAndWait:^{
    [self.context reset];
  }];

  for (NSPersistentStore *store in self.persistentStoreCoordinator.persistentStores) {
    [self.persistentStoreCoordinator removePersistentStore:store error:NULL];
  }

  NSDictionary *options = @{ NSInferMappingModelAutomaticallyOption : @YES,
                             NSMigratePersistentStoresAutomaticallyOption: @YES };

  NSError *error = nil;
  if (![self.persistentStoreCoordinator addPersistentStoreWithType:NSSQLiteStoreType
                                                     configuration:nil
                                                               URL:cacheDataURLForMode(self.trandcoding, self.cacheMode, self.cacheSize)
                                                           options:options
                                                             error:&error]) {
    NSLog(@"Adding persistent store error: %@", error);
  }
 }

 if (resetCacheData) {
  [NSFileManager.defaultManager removeItemAtPath:self.class.photoCachePath
                                           error:NULL];
  [NSFileManager.defaultManager createDirectoryAtPath:self.class.photoCachePath
                           withIntermediateDirectories:YES
                                           attributes:nil
                                                error:NULL];

 }
}

- (NSManagedObjectModel *)managedObjectModel
{
 if (_managedObjectModel != nil) {
  return _managedObjectModel;
 }
 NSURL *modelURL = [[NSBundle mainBundle] URLForResource:@"Caching_Data" withExtension:@"momd"];
 _managedObjectModel = [[NSManagedObjectModel alloc] initWithContentsOfURL:modelURL];
 return _managedObjectModel;
}

- (NSPersistentStoreCoordinator *)persistentStoreCoordinator
{
 if (self->_persistentStoreCoordinator != nil) {
  return self->_persistentStoreCoordinator;
 }

 self->_persistentStoreCoordinator = [[NSPersistentStoreCoordinator alloc] initWithManagedObjectModel:[self managedObjectModel]];


 return self->_persistentStoreCoordinator;
}

- (id)init {
 self = [super init];
 if (self) {
  self.trandcoding = [[PPBResizeCompressTranscoding alloc] init];
  self.cacheSize = PBDefaultCacheFileSize;
  self->_cacheMode = -1;
  self.cacheMode = PBCacheNormalLRUMode;
  self->_resizingTimes = [[NSMutableArray alloc] initWithCapacity:500];
 }

 return self;
}

- (CFAbsoluteTime)averageResizingTimes {
 return [[self.resizingTimes valueForKeyPath:@"@avg.self"] doubleValue];
}
```

```
#pragma mark - Caching methods

- (NSData *)cachedDataForPhotoID:(NSString *)photoID
                  accessFullSize:(BOOL)accessFullSize {
  NSString *cacheName = [photoID stringByAppendingString:(!cacheManagerIsCachedDataNotValidated(self.trandcoding, self.cacheMode) || accessFullSize) ? PBFullSizedImageSuffixName : PBCacheSizedImageSuffixName];
  NSString *cachePath = [self.class.photoCachePath stringByAppendingPathComponent:cacheName];
  NSData *cachedData = nil;
  if ([NSFileManager.defaultManager fileExistsAtPath:cachePath]) {
    cachedData = [[NSData alloc] initWithContentsOfFile:cachePath
                                                options:NSDataReadingUncached
                                                  error:NULL];
  }

  return cachedData;
}

- (void)setData:(NSData *)data forPhotoID:(NSString *)photoID {
  [self setData:data forPhotoID:photoID resizing:YES];
}

- (void)setData:(NSData *)data forPhotoID:(NSString *)photoID resizing:(BOOL)resize {
  NSData *savingData = data;
  CGSize imageSize = CGSizeZero;
  if (resize) {
    CFTimeInterval        start;
    CFTimeInterval        end;
    CFTimeInterval        elapsed;
    start = CACurrentMediaTime();

    savingData = [self.trandcoding transcodedDataForImageData:data];
    end = CACurrentMediaTime();

    elapsed = end - start;
    [self.resizingTimes addObject:@(elapsed)];
  }

  [self.context performBlock:^{
    PBCacheEntry *cacheEntry = [self cacheEntryForPhotoID:photoID];

    cacheEntry.cacheFileSize = @(savingData.length);
    cacheEntry.fullFileSize = @(data.length);
    cacheEntry.width = @(imageSize.width);
    cacheEntry.height = @(imageSize.height);

    BOOL shouldEvict = [self shouldEvictData];
    if (shouldEvict) {
      NSArray *evictingPhotoCacheEntries = [self evictingPhotoCacheEntries];
      [evictingPhotoCacheEntries enumerateObjectsUsingBlock:^(NSString *evictingPhotoID, NSUInteger idx, BOOL *stop) {
        NSString *name = [evictingPhotoID stringByAppendingString:cacheManagerIsCachedDataNotValidated(self.trandcoding, self.cacheMode) ? PBCacheSizedImageSuffixName : PBFullSizedImageSuffixName];
        NSString *pathName = [self.class.photoCachePath stringByAppendingPathComponent:name];
        if ([NSFileManager.defaultManager fileExistsAtPath:pathName] &&
            [NSFileManager.defaultManager removeItemAtPath:pathName error:NULL] &&
            ![self shouldEvictData]) {
          *stop = YES;
        }
      }];
    }

    NSError *saveError = nil;
    [self.context save:&saveError];
  }];

  NSString *cacheName = [photoID stringByAppendingString:(resize && cacheManagerIsCachedDataNotValidated(self.trandcoding, self.cacheMode)) ?
                         PBCacheSizedImageSuffixName : PBFullSizedImageSuffixName];
  NSString *cachePath = [self.class.photoCachePath stringByAppendingPathComponent:cacheName];
  [savingData writeToFile:cachePath atomically:YES];
}

- (void)accessPhotoWithPhotoID:(NSString *)photoID didLoadData:(BOOL)loadedData fullSize:(BOOL)isFullSize {
  [self.context performBlock:^{
    PBCacheEntry *cacheEntry = [self cacheEntryForPhotoID:photoID];
    PBAccessEntry *accessEntry =  [NSEntityDescription insertNewObjectForEntityForName:@"AccessEntry"
                                                 inManagedObjectContext:self.context];
    accessEntry.accessFullSize = @(isFullSize);
    accessEntry.loadDataLength = loadedData ? cacheEntry.fullFileSize : @0;
    accessEntry.entry = cacheEntry;
```

```
    NSError *saveError = nil;
    [self.context save:&saveError];
  }];
}

- (CGSize)preferredSizeForPhotoWithID:(NSString *)photoID {
  __block CGSize photoSize = CGSizeZero;
  [self.context performBlockAndWait:^{
    PBCacheEntry *photoCacheEntry = [self cacheEntryForPhotoID:photoID];
    photoSize = (CGSize) {
      photoCacheEntry.width.floatValue,
      photoCacheEntry.height.floatValue
    };
  }];

  return photoSize;
}

- (BOOL)shouldEvictData {
  NSDirectoryEnumerator *dirEnum = [NSFileManager.defaultManager enumeratorAtPath:self.class.photoCachePath];

  unsigned long long fileSize = 0;
  NSString *fileName = nil;
  while (fileName = [dirEnum nextObject]) {
    if ([fileName hasSuffix:PBFullSizedImageSuffixName] ||
        [fileName hasSuffix:PBCacheSizedImageSuffixName]) {
      fileSize += [dirEnum.fileAttributes[NSFileSize] unsignedLongLongValue];
    }
  }

  return fileSize > self.cacheSize;
}

- (NSArray *)evictingPhotoCacheEntries {
  __block NSArray *evictingPhotoCacheEntries = nil;
  [self.context performBlockAndWait:^{
    NSFetchRequest *fetchRequest = [NSFetchRequest fetchRequestWithEntityName:@"CacheEntry"];
    fetchRequest.returnsObjectsAsFaults = NO;

    NSArray *allCachedEntries = [self.context executeFetchRequest:fetchRequest
                                                             error:NULL];
    evictingPhotoCacheEntries = [self.algorithm evictingItemFromEntry:allCachedEntries];
  }];

  return evictingPhotoCacheEntries;
}


#pragma mark - Private methods

- (PBCacheEntry *)cacheEntryForPhotoID:(NSString *)photoID {
  NSFetchRequest *fetchRequest = [[NSFetchRequest alloc] initWithEntityName:@"CacheEntry"];
  NSPredicate *predicate = [NSPredicate predicateWithFormat:@"dataID == %@", photoID];
  fetchRequest.predicate = predicate;

  NSError *error = nil;
  PBCacheEntry *cacheEntry = [[self.context executeFetchRequest:fetchRequest
                                                          error:&error] lastObject];
  if (!cacheEntry) {
    cacheEntry = [NSEntityDescription insertNewObjectForEntityForName:@"CacheEntry"
                                               inManagedObjectContext:self.context];
    cacheEntry.dataID = photoID;
  }

  return cacheEntry;
}


#pragma mark - Class methods

+ (PBCacheManager *)defaultManager {
  static PBCacheManager *defaultManager;
  static dispatch_once_t onceToken;
  dispatch_once(&onceToken, ^{
    defaultManager = [[PBCacheManager alloc] init];
    defaultManager->_context = [[NSManagedObjectContext alloc] initWithConcurrencyType:NSPrivateQueueConcurrencyType];
    [defaultManager->_context setPersistentStoreCoordinator:defaultManager.persistentStoreCoordinator];
  });
```

```
  return defaultManager;
}

+ (NSString *)photoCachePath {
  static NSString *photoCachePath;
  static dispatch_once_t onceToken;
  dispatch_once(&onceToken, ^{
    photoCachePath = [[NSSearchPathForDirectoriesInDomains(NSCachesDirectory, NSUserDomainMask, YES) lastObject] stringByAppendingPathComponent:@"Photo"];
    if (![NSFileManager.defaultManager fileExistsAtPath:photoCachePath]) {
      [NSFileManager.defaultManager createDirectoryAtPath:photoCachePath
                               withIntermediateDirectories:YES
                                                attributes:nil
                                                     error:NULL];
    }
  });

  return photoCachePath;
}


@end


//
//  PPBResizeCompressTranscoding.m
//  Photo Browser
//
//  Created by Pitiphong Phongpattranont on 5/5/56 BE.
//  Copyright (c) 2556 Pitiphong Phongpattranont. All rights reserved.
//

#import "PPBResizeCompressTranscoding.h"

#import "UIImage+PBResizing.h"

@implementation PPBResizeCompressTranscoding

- (NSString *)transcodingName {
  return @"Resizing+Compressing";
}

- (NSData *)transcodedDataForImageData:(NSData *)data {
  UIImage *fullSizeImage = [[UIImage alloc] initWithData:data
                                                   scale:UIScreen.mainScreen.scale];
  UIImage *scaledImage = [fullSizeImage resizedImageToFitScreen];
  return UIImageJPEGRepresentation(scaledImage, 0.5f);
}

- (BOOL)isTranscodeDataValidForOriginal {
  return NO;
}

@end

//
//  PPBCompressTranscoding.m
//  Photo Browser
//
//  Created by Pitiphong Phongpattranont on 5/5/56 BE.
//  Copyright (c) 2556 Pitiphong Phongpattranont. All rights reserved.
//

#import "PPBCompressTranscoding.h"

@implementation PPBCompressTranscoding

- (NSString *)transcodingName {
  return @"Compressing";
}

- (NSData *)transcodedDataForImageData:(NSData *)data {
  UIImage *fullSizeImage = [[UIImage alloc] initWithData:data
                                                   scale:UIScreen.mainScreen.scale];
  return UIImageJPEGRepresentation(fullSizeImage, 0.3f);
}

- (BOOL)isTranscodeDataValidForOriginal {
  return YES;
}
```

```
@end


@implementation PPBLRUCache

- (NSArray *)evictingItemFromEntry:(NSArray *)cacheEntrys {
  NSArray *accessingData = [cacheEntrys valueForKey:@"accesses"];
  NSMutableArray *data = [[NSMutableArray alloc] initWithCapacity:accessingData.count];
  for (NSMutableOrderedSet *accessData in accessingData) {
    if (accessData.lastObject) {
      [data addObject:accessData.lastObject];
    }
  }

  NSSortDescriptor *lastAccessSortDescriptor = [[NSSortDescriptor alloc] initWithKey:@"accessDate"
                                                                           ascending:NO];
  [data sortUsingDescriptors:@[lastAccessSortDescriptor]];

  return [data valueForKeyPath:@"entry.dataID"];
}

@end
```

ประวัติผู้เขียนวิทยานิพนธ์

นายปิติพงศ์ พงศ์ภัทรานนท์ เกิดเมื่อวันที่ 12 พฤษภาคม พ.ศ. 2530 ที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ จากภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2551 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ ที่ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2552 โดยมีผลงานตีพิมพ์ชื่อ Transcoding Cache for Smart Phones ตีพิมพ์ในงานสัมนาวิชาการ The International Conference on E-Technologies and Business on the Web (EBW2013) ขณะนี้กำลังทำงานที่บริษัท ทรูเวฟ (ประเทศไทย) จำกัด ในตำแหน่ง ผู้พัฒนาโปรแกรมบนระบบ iOS