กลไกซอฟต์แวร์แคชสำหรับวิดีโอสตรีมมิ่งโดยใช้ดัชนีความนิยม

A Software Cache Mechanism for Video Streaming Applying Popularity Index

เตาฟีก หลำสุบ

Taofik Lamsub

วิทยานิพนธ์นี้สำหรับการศึกษาตามหลักสูตรปริญญา

วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

มหาวิทยาลัยสงขลานครินทร์

A Thesis Submitted in Fulfillment of the Requirement for the Degree of

Master of Engineering in Computer Engineering

Prince of Songkla University

2562

กลไกซอฟต์แวร์แคชสำหรับวิดีโอสตรีมมิ่งโดยใช้ดัชนีความนิยม

A Software Cache Mechanism for Video Streaming Applying Popularity Index

เตาฟีก หลำสุบ

Taofik Lamsub

วิทยานิพนธ์นี้สำหรับการศึกษาตามหลักสูตรปริญญา

วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

มหาวิทยาลัยสงขลานครินทร์

A Thesis Submitted in Fulfillment of the Requirement for the Degree of

Master of Engineering in Computer Engineering

Prince of Songkla University

2562

ชื่อวิทยานิพนธ์ กลไกซอฟต์แวร์แคชสำหรับวิดีโอสตรีมมิ่งโดยใช้ดัชนีความนิยม
ผู้เขียน นายเตาฟีก หลำสุบ
สาขาวิชา วิศวกรรมคอมพิวเตอร์

อาจารย์ที่ปรึกษาวิทยานิพนธ์

…….....
(ผู้ช่วยศาสตราจารย์ ดร.พิชญา ตัณฑัยย์)

คณะกรรมการสอบ

……..... ประธานกรรมการ
(ดร.สมชัย หลิมศิโรรัตน์)

……..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.พิชญา ตัณฑัยย์)

……..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ปัญญยศ ไชยกาฬ)

……..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.เทพฤทธิ์ บัณฑิตวัฒนาวงศ์)

บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ อนุมัติให้นับวิทยานิพนธ์ฉบับนี้สำหรับการศึกษา ตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

……..
(ศาสตราจารย์ ดร.ดำรงศักดิ์ ฟ้ารุ่งสาง)
คณบดีบัณฑิตวิทยาลัย

ขอรับรองว่า ผลงานวิจัยนี้มาจากการศึกษาวิจัยของนักศึกษาเอง และได้แสดงความขอบคุณบุคคลที่มีส่วนช่วยเหลือแล้ว

ลงชื่อ…….....
(ผู้ช่วยศาสตราจารย์ ดร.พิชญา ตัณฑัยย์)
อาจารย์ที่ปรึกษาวิทยานิพนธ์

ลงชื่อ…….....
(นายเตาฟีก หลำสุบ)
นักศึกษา

ข้าพเจ้าขอรับรองว่า ผลงานวิจัยนี้ไม่เคยเป็นส่วนหนึ่งในการอนุมัติปริญญาในระดับใดมาก่อน และไม่ได้ถูกใช้ในการยื่นขออนุมัติปริญญาในขณะนี้

ลงชื่อ…..…….....

(นายเตาฟีก หลำสุบ)

นักศึกษา

ชื่อวิทยานิพนธ์ กลไกซอฟต์แวร์แคชสำหรับวิดีโอสตรีมมิ่งโดยใช้ดัชนีความนิยม
ผู้เขียน นายเตาฟีก หลำสุบ
สาขาวิชา วิศวกรรมคอมพิวเตอร์
ปีการศึกษา 2561

บทคัดย่อ

ในปัจจุบันมีความต้องการใช้งานข้อมูลไฟล์วิดีโอที่สูงขึ้นและมีแนวโน้มเพิ่มมากขึ้นเรื่อย ๆ เนื่องจากผู้ใช้สามารถเข้าถึงระบบเครือข่ายอินเตอร์เน็ตได้อย่างง่ายดาย ข้อมูลไฟล์วิดีโอมักมีขนาดใหญ่และเป็นข้อมูลที่เรียงตัวอย่างต่อเนื่อง ต้องใช้ทรัพยากรมากในการประมวลผลและสื่อสารข้อมูลระหว่างไคลเอนต์และเซิร์ฟเวอร์ แคชถูกนำมาใช้ในการเพิ่มประสิทธิภาพและลดระยะเวลาในการอ่านข้อมูลจากแหล่งจัดเก็บที่มีอัตราการอ่านข้อมูลที่ต่ำ

วิทยานิพนธ์นี้มุ่งนำเสนอระบบแคชที่นำค่าความนิยมไฟล์วิดีโอมาใช้ในนโยบายการแทนที่แคชเพื่อเพิ่มประสิทธิภาพให้กับแคชของระบบให้บริการไฟล์วิดีโอสตรีมมิ่งผ่านเอชทีทีพีโปรโตคอลและเพิ่มกระบวนการลดทอนค่าความนิยมตามเวลาที่เปลี่ยนแปลงไป เพื่อเพิ่มความสามารถของแคชโดยการเพิ่มอัตราการพบข้อมูลที่ต้องการในแคชและลดอัตราการดึงข้อมูลจากแหล่งจัดเก็บข้อมูลเบื้องหลังทำให้ลดกระบวนการและระยะเวลาในการส่งข้อมูลจากเซิร์ฟเวอร์เอชทีทีพีไปยังไคลเอนต์

Thesis Title	A Software Cache Mechanism for Video Streaming Applying Popularity Index
Author	Mr. Taofik Lamsub
Major Program	Computer Engineering
Academic Year	2018

ABSTRACT

The present, there are growing demands for video streaming which are likely to increase as the users can easily access the Internet. Video file data is usually large and continuously sorted. It requires a lot of resources to process and communicate data between clients and the server. Cache is used to increase the efficiency and to reduce the time to read data from the storage which has a slow reading rate.

This thesis introduces a cache system that exploits the video file popularity in the cache replacement policy in order to increase the efficiency of the cache system that serves video files via HTTP streaming, and also add the popularity attenuation process that decrease the popularity as the time passes by. It helps increase the cache efficiency by increasing the cache hit rate, reducing the latency of data retrieval from the backend storage, and as a result for reducing the process time for transmit from the HTTP server to the client.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยความช่วยเหลืออย่างดียิ่งจากผู้ช่วยศาสตราจารย์ ดร.พิชญา ตัณฑัยย์ อาจารย์ที่ปรึกษาที่ได้ความอนุเคราะห์และสละเวลาอันมีค่ายิ่งในการแนะนำแนวทางการทำวิทยานิพนธ์และให้ข้อคิดเห็นที่เป็นประโยชน์อย่างยิ่งตลอดระยะเวลาการทำวิจัยเสมอมา

ขอขอบพระคุณคณะกรรมการควบคุมการสอบวิทยานิพนธ์ ซึ่งประกอบด้วย ดร.สมชัย หลิมศิโรรัตน์ ผู้ช่วยศาสตราจารย์ ดร.ปัญญยศ ไชยกาพ และผู้ช่วยศาสตราจารย์ ดร. เทพฤทธิ์ บัณฑิตวัฒนาวงศ์และคณาจารย์ทุกท่าน

ขอขอบคุณพระคุณคณาจารย์และบุคลากรทุกท่านของภาควิชาวิศกรรมคอมพิวเตอร์ และคณะวิศวกรรมศาสตร์ และบัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ ที่ให้ความกรุณาอำนวยความสะดวกให้การทำวิจัยนี้สำเร็จลุล่วง โดยเฉพาะอย่างยิ่งคณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์ ที่ได้มอบทุนการศึกษาในการศึกษาวิจัยครั้งนี้ และกขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ กฤษณ์วรา รัตนโอภาส ในความช่วยเหลือที่ดียิ่งตลอดมา

นอกจากนี้ขอขอบพระคุณบิดา มารดา ผู้ให้กำเนิดและให้ความช่วยตลอดระยะเวลาที่ผ่านมา ทุก ๆ ความดีงามทั้งหมดผู้วิจัยขอมอบแต่บุพการีทั้งสองท่าน ขอขอบพระคุณบุคคลที่รักในครอบครัวทุกท่าน โดยเฉพาะอย่างยิ่ง ผู้ช่วยศาสตราจารย์ ดินาถ หลำสุบ พี่สาวผู้คอยอยู่เคียงข้างให้การช่วยเหลือและเป็นกำลังใจให้ตลอดเวลา

เตาฟีก หลำสุบ

สารบัญ

บทคัดย่อ ..... (5)
ABSTRACT ..... (6)
กิตติกรรมประกาศ ..... (7)
สารบัญตาราง ..... (11)
สารบัญรูปภาพ ..... (12)
บทที่ 1 บทนำ 1
1.1 ที่มาและความสำคัญ 1
1.2 วัตถุประสงค์ของการวิจัย 4
1.3 ขอบเขตของการวิจัย 5
1.4 ประโยชน์ที่คาดว่าจะได้รับ 5
บทที่ 2 งานวิจัยและวรรณกรรมที่เกี่ยวข้อง 6
2.1 วิดีโอสตรีมมิ่ง 6
2.2 วิธีการสตรีมมิ่ง (Streaming Method) 7
2.2.1 HTTP Live Streaming (HLS) 8
2.2.2 Dynamic Adaptive Streaming over HTTP (DASH) 9
2.3 แคชที่ใช้งานในระบบสตรีมมิ่ง (Streaming Cache) 11
2.4 นโยบายการแทนที่แคช 12
2.4.1 นโยบายการแทนที่แคชแบบ Least-Recently-Used (LRU) 13
2.4.2 นโยบายการแทนที่แคชแบบ Least-Frequency-Used (LFU) 14
2.4.3 นโยบายการแทนที่แคชที่อ้างอิงค่า Popularity 15
2.5 พฤติกรรมของผู้ใช้งานระบบให้บริการไฟล์วิดีโอ 16
2.5.1 พฤติกรรมการใช้งานแบบ Uniform 16
2.5.2 พฤติกรรมการใช้งานแบบ Zipf-like 17
2.5.3 การเปลี่ยนแปลงความถี่ของการถูกใช้งานของไฟล์วิดีโอในระบบ HTTP สตรีมมิ่ง 19
2.6 การใช้งาน Nginx 20
2.7 ซอฟต์แวร์แคช Redis 21
2.8 เฟรมเวิร์ค OpenResty 23
บทที่ 3 วิธีดำเนินการวิจัย 24

3.1 ระบบ DASH Streaming และการทำงานของ DASH Client 24
3.2 พฤติกรรมของผู้ใช้งาน (User Behavior) 26
3.3 การออกแบบแคช 28
3.4 โครงสร้างและส่วนประกอบของระบบ (System Architecture) 33
3.4.1 Client 33
3.4.2 HTTP Proxy 34
3.4.3 ByteRangeHandler 34
3.4.4 QualityProxy 34
3.4.5 แคช (Cache) 35
3.4.6 Backend storage 35
3.5 อัลกอริทึม Dynamic Popularity Caching 35
3.6 การทำงานของนโยบายการแทนที่แคช Dynamic Popularity Caching 37
3.6.1 Popularity Calculation Process 38
3.6.2 Eviction Process 38
3.7 การลดปัญหาความสอดคล้องกันของข้อมูล (cache coherence) ภายในแคช 42
3.8 การนำนโยบายการแทนที่แคชที่ออกแบบไปใช้กับซอฟต์แวร์แคช Redis 43
3.9 การอ้างอิง segment ในซอฟต์แวร์แคช Redis 43
3.10 การเขียนโปรแกรมภาษา Lua เพื่อทำงานใน Nginx 44
3.11 ข้อมูลไฟล์วิดีโอที่ใช้ในการทดสอบ (Video dataset) 44
3.12 คุณลักษณะของระบบให้บริการไฟล์วิดีโอที่ใช้ในการทดลอง 46
3.12.1 ซอฟต์แวร์ 46
3.12.2 ฮาร์ดแวร์ 46
3.13 พารามิเตอร์ที่ใช้ในการวัดประสิทธิภาพการทำงานของระบบ 47
3.13.1 Hit rate 47
3.13.2 Miss rate 47
3.13.3 Cache effective access time 48
บทที่ 4 การทดลองและวิเคราะห์ผล 49
4.1 การออกแบบการทดลอง 49
4.2 รูปแบบพฤติกรรมที่ของผู้ใช้ 50
4.3 การเปรียบเทียบประสิทธิภาพการทำงานของระบบ 51

4.3.1 การวัดประสิทธิภาพของการตอบสนองข้อมูลของ Redis แคช....51
4.3.2 การวัดความเร็วของระบบเมื่อขนาดของ segment duration ต่างกัน....52
4.4 การวัดประสิทธิภาพการทำงานของนโยบายการแทนที่แคชตามพฤติกรรมของผู้ใช้งาน ...54
4.4.1 ผลการวัดประสิทธิภาพการทำงานของแคชด้วยพฤติกรรมการใช้งานของผู้ใช้แบบ Uniform 54
4.4.2 การวัดประสิทธิภาพการทำงานของแคชด้วยพฤติกรรมการใช้งานของผู้ใช้แบบ Zipf-Like... 57
บทที่ 5 สรุปผลการวิจัย อภิปรายผล และข้อเสนอแนะ....60
5.1 สรุปผลการวิจัย....60
5.2 อภิปรายผลการวิจัย....62
5.3 ข้อเสนอแนะ....65
5.3.1 ข้อเสนอแนะการเขียนโปรแกรมภาษา Lua เพื่อใช้งานกับ Nginx โดยใช้ไลบรารีของ OpenResty....65
5.3.2 การวัดประสิทธิภาพระบบโดยใช้โปรแกรมจำลองสร้างภาระงานให้กับระบบ (Workload tester) 66
5.4 การนำไปใช้....66
5.5 แนวทางการพัฒนาต่อยอด....67
บรรณานุกรม....69
ภาคผนวก ก....75
ภาคผนวก ข....76
ภาคผนวก ค....78
ภาคผนวก ง....79
ประวัติผู้เขียน....86

สารบัญตาราง

ตารางที่ 2.1 พารามิเตอร์ที่เกี่ยวข้องกับการจัดแคชของ Redis 22
ตารางที่ 2.2 นโยบายแทนที่แคชของ Redis 22
ตารางที่ 3.1 ข้อมูลวิดีโอที่ใช้ในการวัดประสิทธิภาพของระบบ.....45
ตารางที่ ข-1 โมดูลย่อยของ Nginx ที่ใช้ในการพัฒนาระบบ.....76

สารบัญรูปภาพ

รูปที่ 2.1 การให้บริการ streaming ผ่านเครือข่ายอินเตอร์เน็ต 7
รูปที่ 2.2 กระบวนการดาวน์โหลดไฟล์วิดีโอ streaming.....9
รูปที่ 2.3 โครงสร้างภายในของไฟล์ Manifest 10
รูปที่ 2.4 กระบวนการสื่อสารข้อมูลในระบบให้บริการไฟล์วิดีโอตามมาตรฐาน MPEG-DASH [21] 11
รูปที่ 2.5 ผังงานการทำงานนโยบายแคช Least Recently Used.....13
รูปที่ 2.6 นโยบายแทนที่แคชแบบ Least Frequency Used 14
รูปที่ 2.7 อันดับของไฟล์วิดีโอเมื่อเรียงตามจำนวนผู้ใช้งานในช่วงระยะเวลาต่าง ๆ.....18
รูปที่ 2.8 ความนิยมของไฟล์วิดีโอเรียงตามช่วงเวลาของวิดีโอ.....18
รูปที่ 2.9 การเปลี่ยนแปลงความถี่ของการถูกใช้งานไฟล์วิดีโอของ YouTube เมื่อแบ่งตามวันที่ไฟล์ถูกเผยแพร่ในระบบ 19
รูปที่ 2.10 การเปลี่ยนแปลงความถี่ของการถูกใช้งานที่มีผลมาจากปัจจัยความต้องการไฟล์วิดีโอเกี่ยวกับการซ่อมเครื่องแอร์คอนดิชันเนอร์ของ YouTube 20
รูปที่ 2.11 โครงสร้างการทำงานของ Nginx 21
รูปที่ 3.1 การทำงานของแคชในระบบ video streaming แบบ byte range.....28
รูปที่ 3.2 ตัวอย่างการลดทอนค่าความนิยม 31
รูปที่ 3.3 การเปลี่ยนแปลงค่าความนิยมด้วยการลดทอนความนิยมของ segment ไฟล์วิดีโอตามสมการที่ (5).....32
รูปที่ 3.4 ส่วนประกอบและของสร้างของระบบ.....33
รูปที่ 3.5 แผนผังการทำงานของนโยบายการแทนที่แคชที่ผู้วิจัยออกแบบ 39
รูปที่ 3.6 แผนผังการทำงานของโปรเซสย่อยภายในนโยบายการแทนที่แคช.....40
รูปที่ 3.7 การเข้าถึงข้อมูลใน EvictList ของเทรดย่อย.....41
รูปที่ 3.8 การอ้างอิงข้อมูล segment ภายในแคชตามโครงสร้างข้อมูลแบบ Set ของ Redis 44
รูปที่ 3.9 แผนผังไฟล์วิดีโอในระบบของ Netflix.....45
รูปที่ 4.1 การเชื่อมต่อเครือข่ายอินเตอร์เน็ต (LAN) ของระบบในการทดลอง 49
รูปที่ 4.2 เวลาที่ใช้ในการตอบสนองข้อมูลของเซิร์ฟเวอร์เมื่อเทียบกับขนาดข้อมูลในกรณีทั้ง Hit และ Miss 52

รูปที่ 4.3 ผลของขนาด segment duration ที่ 4 วินาที (บน) และ 10 วินาที (ล่าง) ต่อเวลาในการตอบสนองของระบบ 53
รูปที่ 4.4 Hit rate ของแคชตามนโยบายการแทนที่แคชในรูปแบบพฤติกรรมการใช้งานแบบ Uniform.....55
รูปที่ 4.5 Miss rate ของแคชตามนโยบายการแทนที่แคชในรูปแบบพฤติกรรมการใช้งานแบบ Uniform.....56
รูปที่ 4.6 Miss rate ของแคชตามนโยบายการแทนที่แคชในรูปแบบพฤติกรรมการใช้งานแบบ Zipf-Like 58
รูปที่ 4.7 Hit rate ของแคชตามนโยบายการแทนที่แคชในรูปแบบพฤติกรรมการใช้งานแบบ Zipf-Like 58
รูปที่ ก-1 เฟสการทำงานของ Nginx ในการประมวลผลการร้องขอจากไคลเอนต์.....75
รูปที่ ข-1 เฟสของ Nginx ที่สามารถเพิ่มส่วนการทำงานได้……..…77

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

ในปัจจุบันมีผู้ใช้งานวิดีโอบนอินเตอร์เน็ต หรือ video streaming เป็นจำนวนมาก เนื่องจากการเข้าถึงระบบอินเตอร์เน็ตที่รวดเร็ว อีกทั้งยังมีความต้องการใช้งานวิดีโอผ่านทางอุปกรณ์ต่าง ๆ เช่น การใช้งานผ่านทางคอมพิวเตอร์และโทรศัพท์มือถือประเภทสมาร์ทโฟนที่หลากหลาย รวมถึงยังสามารถรับชมผ่านโทรทัศน์ที่เชื่อมต่อระบบอินเตอร์เน็ตได้หรือ smart TV เป็นต้น โดยสามารถเรียกใช้งานผ่านเครือข่ายอินเตอร์เน็ต ซึ่งมีหลายรูปแบบด้วยกัน รูปแบบหนึ่งที่เป็นที่นิยมคือระบบการให้บริการวิดีโอแบบ streaming ผ่าน HTTP (Hypertext Transfer Protocol) ซึ่งทำให้ผู้ใช้สามารถเข้าใช้งานได้อย่างสะดวกไม่ว่าจะเป็นการใช้งานผ่านเว็บบราวเซอร์ หรือการใช้งานผ่านโปรแกรมประยุกต์ต่าง ๆ ที่เชื่อมต่อกับอินเตอร์เน็ตได้ อีกทั้งในปัจจุบันมีการใช้งาน social network อย่างแพร่หลาย ระบบให้บริการวิดีโอถูกนำไปใช้ร่วมกับการให้บริการผ่าน social media ทำให้วิดีโอถูกใช้งานมากขึ้น ประกอบกับความต้องการระบบ video streaming ที่ให้บริการอยู่ในปัจจุบัน เช่น YouTube, Netflix, Hulu, HBO, Amazon Prime หรือแม้แต่กระทั่งบน Facebook หรือ Twitter ก็ตาม ซึ่งผู้ใช้จำนวนมากสามารถเข้าถึงและเรียกใช้บริการได้อย่างสะดวก ทำให้การใช้งานอินเตอร์เน็ตเพื่อ video streaming มีปริมาณที่สูง ประกอบกับจำนวนผู้ใช้งานที่มีจำนวนมาก จากผลการศึกษาของ Cisco ใน Cisco Visual Networking Index: Forecast and Trends, 2017-2022 White Paper [1] ซึ่งได้ทำการศึกษาปริมาณการใช้งานอินเตอร์เน็ตโดยแบ่งตามประเภทข้อมูลที่ใช้ ปรากฏว่า video streaming เป็นประเภทข้อมูลที่มีปริมาณการใช้งานบนอินเตอร์เน็ตมากที่สุดและยังคงเพิ่มขึ้นตลอดเวลาเมื่อเทียบกับข้อมูลประเภทอื่น ๆ ทั้งปริมาณผู้ใช้และปริมาณข้อมูล ทำให้ระบบให้บริการจำเป็นต้องรองรับภาระงาน (Workload) ที่สูงมาก

ระบบให้บริการไฟล์วิดีโอนั้นประกอบไปด้วย เครื่องแม่ข่ายให้บริการไฟล์วิดีโอ หรือ HTTP streaming server เครื่องผู้ใช้หรือไคลเอนต์ (client) ที่เปิดเล่นไฟล์วิดีโอ และ backend storage ทำหน้าที่เป็นแหล่งจัดเก็บข้อมูลอยู่เบื้องหลัง HTTP streaming server โดยเรียกรวมๆ เป็นเซิร์ฟเวอร์ (server) การรับส่งข้อมูลระหว่างไคลเอนต์และเซิร์ฟเวอร์ผ่านช่องทางการสื่อสารข้อมูลด้วยโปรโตคอล HTTP ทำให้ผู้ใช้สามารถเข้าใช้ได้สะดวกและใช้งานผ่านอุปกรณ์ได้หลากหลาย การให้บริการ video streaming นั้นต้องใช้ทรัพยากรคอมพิวเตอร์ที่ค่อนข้างสูงอันเนื่องมาจากขนาด

ของไฟล์วิดีโอ (File size) ความละเอียดและความคมชัด (Bitrate) ของภาพวิดีโอ ประกอบกับการเรียกใช้งานจากผู้ใช้เป็นการร้องขอไฟล์จากอย่างต่อเนื่อง ทำให้การอ่านไฟล์ของเซิร์ฟเวอร์ต้องอ่านอย่างต่อเนื่องเช่นกัน ซึ่งสัมพันธ์กับอัตราเร็วของข้อมูลที่วิดีโอเล่นและขนาดของไฟล์วิดีโอ อัตราเร็วภาพ (เฟรมต่อวินาที หรือ Frames per second) ทั้งหมดนี้มีผลต่อการทำงานทั้งทางฝั่งผู้ใช้และทางฝั่งเครื่องแม่ข่ายที่ให้บริการไฟล์วิดีโอ ซึ่งความเร็วในการรับและส่งข้อมูลเป็นตัวแปรหลักที่ทำให้การออกแบบระบบจำเป็นต้องคำนึงถึงประสิทธิภาพการทำงานเป็นสำคัญ

การให้บริการ HTTP สตรีมมิ่งนั้นมีการเชื่อมต่อกับไคลเอนต์จำนวนมาก เซิร์ฟเวอร์ต้องรองรับการร้องขอของไคลเอนต์เหล่านั้น จึงจำเป็นต้องแยกแหล่งจัดเก็บข้อมูลออกมา เพื่อให้เซิร์ฟเวอร์ทำงานได้อย่างคล่องตัวและให้ backend storage ทำงานการอ่านและเขียนข้อมูล การแยกการทำงานด้วยโครงสร้างแบบนี้ทำให้ระบบสามารถขยายตัวได้ในอนาคต แต่ยังมีข้อจำกัดเรื่องความเร็วของการอ่านเขียนข้อมูล การร้องขอไฟล์วิดีโอของผู้ใช้แต่ละครั้งต้องอ่านข้อมูลจากแหล่งจัดเก็บข้อมูล หากจำเป็นต้องอ่านข้อมูลจากแหล่งจัดเก็บทุกครั้งจะทำให้ backend storage ทำงานหนัก และทำให้การอ่านข้อมูลระหว่าง server ภายในระบบมีอัตราที่สูง และปัญหาการอ่านข้อมูลชุดเดิม ๆ ซ้ำ ๆ บ่อยครั้งตามการร้องขอข้อมูลจากไคลเอนต์ซึ่งต้องใช้ระยะเวลานานในการอ่านข้อมูลแต่ละครั้ง เป็นผลให้เกิดความล่าช้าในการให้บริการ จึงมีการนำแคช (Cache) มาใช้เป็นแหล่งจัดเก็บข้อมูลชั่วคราวที่มีความเร็วสูงแต่มีขนาดที่จำกัดเนื่องจากราคาแพง แคชถูกจัดวางไว้ระหว่าง HTTP server และ backend storage เพื่อลดภาระงานของ backend storage เพิ่มความคล่องตัวและความรวดเร็วให้กับ HTTP server โดยไฟล์วิดีโอจะถูกแบ่งส่งเป็นส่วนย่อย ๆ ไปยังไคลเอนต์ เพื่อให้ไคลเอนต์สามารถเปิดเล่นได้ทันทีโดยไม่จำเป็นต้องรอดาวน์โหลดจนจบไฟล์ โดยหากข้อมูลที่ไคลเอนต์ร้องขอนั้นอยู่ภายในแคช HTTP server สามารถส่งข้อมูลไปให้ไคลเอนต์ได้ทันทีโดยไม่ต้องอ่านจาก backend storage ทำให้ลดขั้นตอนการทำงานลงและช่วยให้ HTTP server ทำงานได้รวดเร็วขึ้น และลดจำนวนครั้งในการอ่านข้อมูลจาก backend storage ลง ทำให้สามารถรองรับการร้องขอได้มากขึ้น ทั้งนี้ขึ้นอยู่กับข้อมูลที่อยู่ภายในแคชตรงตามความต้องการของไคลเอนต์ หรือไม่มากน้อยเพียงใด

ข้อมูลที่อยู่ในแคชสามารถอ่านและส่งไปยังไคลเอนต์ได้ทันทีโดยไม่ต้องทำการร้องขอข้อมูลซ้ำจาก backend storage ทำให้ลดเวลาการทำงานลงอย่างมาก ทั้งนี้ขึ้นอยู่กับขนาดพื้นที่จัดเก็บของแคช เนื่องจากไฟล์วิดีโอเป็นไฟล์ขนาดใหญ่และในระบบการให้บริการไฟล์วิดีโอนั้นมีไฟล์วิดีโอเป็นจำนวนมาก จึงเป็นไปไม่ได้ที่จะทำการจัดเก็บไฟล์วิดีโอทั้งหมดลงในแคช ซึ่งเป็นแหล่งจัดเก็บข้อมูลที่มีความรวดเร็วในการอ่านและเขียนข้อมูลสูงแต่มีราคาแพงเมื่อเทียบกับ

backend storage ที่ขนาดความจุเท่ากัน ไฟล์ข้อมูลจำนวนมากนี้มักจัดเก็บในแหล่งจัดเก็บข้อมูลประเภท hard disk drive, solid state drive หรือ cache ตามความเหมาะสมกับขนาดจัดเก็บข้อมูลที่ต้องการ เมื่อพิจารณาที่ความเร็วของแหล่งจัดเก็บข้อมูลทั้งสามแล้ว cache เป็นแหล่งจัดเก็บข้อมูลที่มีความเร็วในการอ่านและเขียนสูงที่สุด

เนื่องด้วยขนาดของแคชที่จำกัดประกอบกับขนาดของไฟล์วิดีโอที่มีขนาดใหญ่ รวมไปถึงผู้ใช้งานมีจำนวนมาก ระบบให้บริการวิดีโอจึงจำเป็นต้องใช้แคชอย่างมีประสิทธิภาพ นโยบายการแทนที่แคช หรือ cache replacement policy มีหน้าที่จัดการการนำเข้า (Insertion) และตัดสินใจขับออก (Eviction) ข้อมูลจากแคชในกรณีที่มีพื้นที่ว่างไม่เพียงพอต่อการเขียนเข้าข้อมูลใหม่สู่แคชนั้น นโยบายการแทนที่แคชจะเป็นตัวคอยตัดสินว่าข้อมูลนั้นจะอยู่ในแคชได้นานเพียงใด ซึ่งขึ้นอยู่กับการใช้งานระบบในขณะนั้นโดยพิจารณาจากปัจจัยต่าง ๆ ของข้อมูลที่อยู่ในแคช เช่น ความล่าสุดของการถูกใช้งาน (Recency) หรือความถี่ของการถูกใช้ (Frequency) เป็นต้น ซึ่งนโยบายการแทนที่แคชที่เป็นที่นิยมคือ Least-Recently-Used (LRU) [2] และ Least-Frequency-Used (LFU) [3] ซึ่งมีข้อดีข้อเสียที่แตกต่างกัน โดยนโยบายการแทนที่แคช LRU จะพิจารณาความล่าสุดของการถูกใช้งานโดยข้อมูลต่อไปจะที่อยู่ในแคชนานที่สุดจะเป็นข้อมูลที่ถูกใช้งานล่าสุดโดยไม่พิจารณาถึงความถี่หรือความนิยมของการถูกใช้งาน ในขณะที่นโยบายการแทนที่แคช LFU จะให้ความสำคัญกับข้อมูลที่มีความถี่สูงที่สุดให้สามารถจะอยู่ในแคชได้นานที่สุดซึ่งเป็นความถี่สะสมของการถูกใช้งาน

นโยบายการแทนที่แคชที่เลือกใช้จะทำงานได้อย่างมีประสิทธิภาพเมื่อนโยบายสอดคล้องกับพฤติกรรมการใช้งานซึ่งขึ้นอยู่กับพฤติกรรมของผู้ใช้และรูปแบบของข้อมูล พฤติกรรมการใช้งานของผู้ใช้และรูปแบบของข้อมูลในระบบให้บริการไฟล์วิดีโอนั้นแตกต่างจากการใช้งานไฟล์ข้อมูลอื่น ๆ เช่น ไฟล์เอกสารทั่วไป เนื่องจากปัจจัยต่าง ๆ เช่น ความนิยมของผู้ใช้ที่มีต่อไฟล์จากวิดีโอนั้น ๆ หรือจำนวนผู้ใช้ที่ต้องการไฟล์วิดีโอนั้น ๆ พร้อม ๆ กันและช่วงเวลาที่ใช้ ซึ่งนโยบายการแทนที่แคชมีผลอย่างมากต่อประสิทธิภาพของระบบให้บริการไฟล์วิดีโอ แต่ข้อมูลที่อยู่ในแคชอาจไม่ได้เป็นข้อมูลที่สำคัญและเป็นที่ต้องการที่สุดในระบบในนโยบาย LRU และ LFU ซึ่งไม่ได้ให้ความสำคัญกับค่าความนิยม (Popularity) ของไฟล์วิดีโอ ซึ่งเกี่ยวข้องกับรูปแบบการใช้งานของผู้ใช้และส่งผลต่อประสิทธิภาพของระบบให้บริการไฟล์วิดีโอ

ได้มีการศึกษาพฤติกรรมของผู้ใช้งานของระบบให้บริการไฟล์วิดีโอพบว่าความนิยมของผู้ใช้งานที่มีต่อไฟล์วิดีโอนั้นมีผลต่อรูปแบบพฤติกรรมการใช้งานระบบ [4,5] และมีผลต่อประสิทธิภาพการทำงานของนโยบายแทนที่แคช ซึ่งนโยบายแทนที่แคชที่สอดคล้องกับพฤติกรรม

การใช้งานจะทำให้ระบบทำงานได้อย่างเต็มประสิทธิภาพ ได้มีการปรับใช้ค่าความนิยมกับนโยบายแทนที่แคช [6] โดยพิจารณาให้ความสำคัญกับช่วงข้อมูลวิดีโอที่กำลังเปิดเล่นอยู่และเพิ่มความสำคัญให้กับช่วงวิดีโอส่วนถัดไป [7] แต่จากพฤติกรรมของผู้ใช้ส่วนใหญ่มักจะเปิดรับชมเพียงช่วงต้นของไฟล์วิดีโอ ถึงแม้จะมีผู้ใช้จำนวนหนึ่งรับชมวิดีโอจนจบก็ตาม นอกจากนี้ความนิยมของแต่ช่วงเวลาภายในไฟล์วิดีโอเรียงตัวไม่ต่อเนื่องกัน ความนิยมในช่วงต้นของไฟล์จะมากกว่าเสมอและอาจจะมีบางช่วงที่มีความนิยมสูง รวมถึงความนิยมในอดีตมักลดลงเมื่อเวลาผ่านไปถึงแม้จะมีความถี่สะสมค่าสูงก็ตาม ซึ่งทำให้นโยบาย LFU ไม่สอดคล้องกับพฤติกรรม

ในงานวิจัยที่ผ่านมาได้มีการนำเสนอการจัดระดับความสำคัญของความนิยมโดยให้ระยะเวลาของความนิยมนั้นเป็นตัวบ่งบอกความน่าเชื่อถือของระดับความนิยมนั้น แต่จะต้องใช้ระยะเวลานานเป็นเดือนหรือเป็นปีในการสะสมข้อมูลการใช้งานเพื่อให้ได้ความแม่นยำ จึงไม่สามารถตอบสนองต่อการเปลี่ยนแปลงได้ในเวลาจริง ในขณะที่พฤติกรรมการใช้งานระบบมีการเปลี่ยนแปลงอยู่ตลอดเวลา บางช่วงเวลาอาจจะมีความนิยมเพิ่มขึ้นและบางช่วงเวลาอาจจะลดลงแล้วเพิ่มขึ้นอีก โดยค่าความนิยมของข้อมูลไม่ได้อยู่ในรูปแบบที่ต่อเนื่องกันตลอดทั้งไฟล์ ซึ่งทำให้การคำนวณค่าความนิยมอาจมีความคลาดเคลื่อนและไม่ได้เป็นความนิยมที่ถูกต้องในขณะนั้น ดังนั้นการพิจารณาความนิยมของข้อมูลในระดับช่วงข้อมูลย่อย (segment) ณ เวลาการใช้งานจริง (real time) จะทำให้ระบบทำงานได้ดีขึ้น

วิทยานิพนธ์นี้มุ่งนำเสนอระบบแคชที่นำค่าความนิยมไฟล์วิดีโอมาใช้ใน cache replacement policy เพื่อเพิ่มประสิทธิภาพให้กับแคชของระบบให้บริการไฟล์วิดีโอผ่าน HTTP streaming และเพิ่มกระบวนการลดทอนค่าความนิยม (Popularity attenuation) ตามเวลาที่เปลี่ยนแปลงไป เพื่อเพิ่มความสามารถของแคชโดยการเพิ่มอัตราการพบข้อมูลที่ต้องการในแคช (Cache hit) และลดอัตราการดึงข้อมูลจาก backend storage ทำให้ลดกระบวนการและระยะเวลาในการส่งข้อมูลจาก HTTP server ไปยังไคลเอนต์

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อออกแบบโครงสร้างการทำงานของแคชสำหรับสตรีมมิ่งวิดีโอ เพื่อทำงานในระบบให้บริการไฟล์วิดีโอผ่านโปรโตคอล HTTP

1.2.2 เพื่อเพิ่มประสิทธิภาพของแคช โดยการเพิ่ม hit rate และลด miss rate ด้วยนโยบายการแทนที่แคชที่พิจารณาค่าดัชนีความนิยมของข้อมูลที่อยู่ในแคช และการลดทอนค่าดัชนี

ความนิยมซึ่งแปรผกผันกับเวลาที่เปลี่ยนแปลงไป เพื่อให้สอดคล้องกับพฤติกรรมการใช้งาน video streaming ของผู้ใช้งานในระบบการให้บริการวิดีโอแบบ MPEG-DASH

1.3 ขอบเขตของการวิจัย

1.3.1 สร้างแคชให้กับ video streaming ด้วยมาตรฐานการ streaming ผ่าน HTTP แบบ MPEG-DASH

1.3.2 พัฒนานโยบายการแทนที่แคชที่นำค่าดัชนีความนิยม (Popularity Index) มาใช้ พร้อมทั้งอัตราการลดทอนของค่าความนิยม (Popularity attenuation) ในกระบวนการขับออก (Eviction) ของนโยบายการแทนที่แคช

1.3.3 ทดสอบระบบและเปรียบเทียบกับนโยบาย LRU และ LFU โดยจำลองการทำงานของระบบด้วยวิดีโอ streaming แบบ MPEG-DASH

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 ได้โครงสร้างการทำงานของแคชพร้อมทั้งนโยบายแทนที่แคชที่เหมาะสมกับพฤติกรรมการใช้งานผู้ใช้ในระบบให้บริการไฟล์วิดีโอผ่านโปรโตคอล HTTP ด้วยมาตรฐานการให้บริการวิดีโอ streaming แบบ MPEG-DASH

1.4.2 แคชสามารถทำงานได้อย่างมีประสิทธิภาพสอดคล้องกับพฤติกรรมการใช้งานของผู้ใช้ โดยเพิ่มอัตราการพบข้อมูลในแคช (hit rate) และลดอัตราการร้องขอข้อมูลไปยัง backend storage เมื่อไม่พบข้อมูลในแคช (miss rate) ซึ่ง hit rate ระยะเวลาการรอข้อมูลของผู้ใช้และระยะเวลาการทำงานของระบบโดยรวมจะลดลง

บทที่ 2
งานวิจัยและวรรณกรรมที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงเทคโนโลยีวิธีการให้บริการและงานวิจัยที่เกี่ยวข้องกับระบบให้บริการไฟล์วิดีโอสตรีมมิ่ง

2.1 วิดีโอสตรีมมิ่ง

การให้บริการไฟล์วิดีโอนั้นมีอยู่ด้วยกันหลัก ๆ สองประเภทด้วยกันโดยแบ่งแยกตามลักษณะการให้บริการดังนี้คือคือ

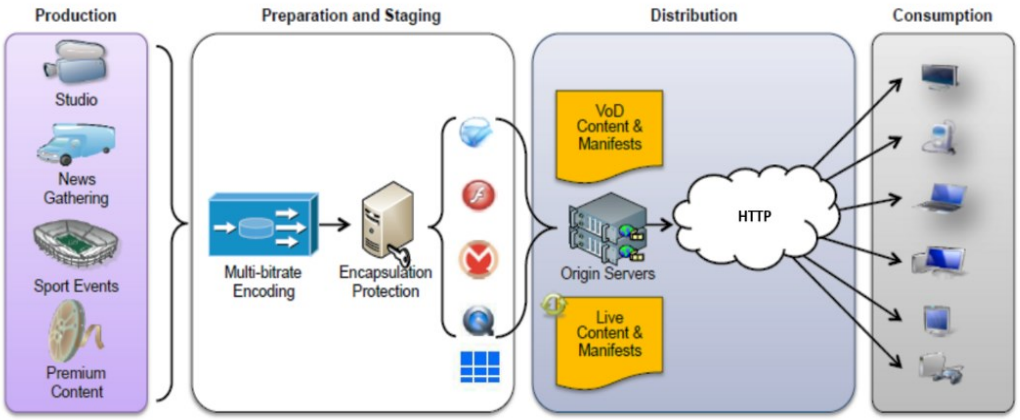
Video on demands (VoD) Streaming ซึ่งเป็นรูปแบบระบบการเรียกดูไฟล์วิดีโอตามความต้องการ และอำนวยความสะดวกให้ผู้ใช้งานสามารถเลือกดูวิดีโอเมื่อใดก็ได้ตามที่ผู้ใช้ต้องการ และสามารถย้อนกลับไปมา (seek) เพื่อรับชมบางช่วงบางตอนของวิดีโอได้ตามต้องการ เช่น ภาพยนตร์ วิดีโอสอนออนไลน์ (online course) ไฟล์เสียงดนตรีและเพลงต่าง ๆ หรือวิดีโอบนเว็บไซต์ YouTube, Netflix และ Coursera หรือ edX เป็นต้น ซึ่งประกอบด้วยไฟล์วิดีโอจำนวนมาก และไฟล์วิดีโอเหล่านั้นได้ถูกจัดเตรียมและเข้ารหัสวิดีโอเอาไว้ล่วงหน้าโดยผู้ให้บริการ

Live Streaming หรือการถ่ายทอดสดเป็นรูปแบบระบบการเรียกดูไฟล์วิดีโอตามช่วงเวลาที่มีการถ่ายทอดเท่านั้น เช่น การถ่ายทอดสดการแข่งขันกีฬา หรือเหตุการณ์สำคัญอื่น ๆ โดยที่ผู้ใช้สามารถรับชมได้เฉพาะในช่วงเวลาการถ่ายทอดเท่านั้น หากผ่านพ้นเวลาถ่ายทอดสดไปแล้ว ผู้ใช้จะไม่สามารถรับชมในลักษณะถ่ายทอดสดได้อีก ซึ่งการเข้ารหัสวิดีโอของและการส่งข้อมูลจากเซิร์ฟเวอร์ไปยังไคลเอนต์นั้นต้องกระทำไปพร้อม ๆ กันในช่วงเวลาที่ถ่ายทอด ซึ่งทำให้เซิร์ฟเวอร์ที่ให้บริการต้องทำงานหนักมากในกรณีที่มีผู้ใช้งานจำนวนมาก และผู้ใช้ไม่สามารถย้อนกลับไปมาเพื่อรับชมบางช่วงของวิดีโอได้

งานวิจัยนี้มุ่งเน้นระบบให้บริการไฟล์วิดีโอในรูปแบบ VoD เป็นหลัก ซึ่งผู้ใช้ที่เข้าใช้ระบบสามารถเลือกรับชมไฟล์วิดีโอได้ตามต้องการ โดยเป็นอิสระจากกันและสามารถย้อนกลับไปมาเพื่อรับชมบางช่วงบางตอนของไฟล์วิดีโอนั้น ๆ ได้ตามต้องการ รูปแบบการสื่อสารข้อมูลผ่านเครือข่ายอินเตอร์เน็ตระหว่าง video server กับไคลเอนต์นั้นมีทั้งผ่านโปรแกรมประยุกต์และเว็บบราวเซอร์

2.2 วิธีการสตรีมมิ่ง (Streaming Method)

การส่งผ่านข้อมูลระหว่างเซิร์ฟเวอร์กับไคลเอนต์ในระบบ video streaming นั้นใช้การสื่อสารข้อมูลผ่านโปรโตคอล HTTP ซึ่งเรียกวิธีการส่งข้อมูลวิดีโอที่ผู้ใช้สามารถเปิดดูไปพร้อมๆ กับการเปิดรับชมผ่าน HTTP นี้ว่า HTTP Streaming Method ซึ่งเป็นการส่งผ่านข้อมูลแบบ pseudo streaming [8] ที่แบ่งส่งข้อมูลไฟล์วิดีโอออกเป็นส่วนเล็ก ๆ ที่ละส่วนไปยังไคลเอนต์ โดยเริ่มจากเริ่มต้นไฟล์ไปจนจบไฟล์ การส่งข้อมูลลักษณะนี้ทำให้ไคลเอนต์สามารถเปิดเล่นไฟล์วิดีโอได้ทันทีเมื่อได้ส่วนของไฟล์ที่ต้องการ โดยไม่จำเป็นต้องรอให้ได้ข้อมูลครบทั้งไฟล์ ซึ่งเป็นข้อดีเพราะไฟล์วิดีโอมักมีขนาดใหญ่ หากไคลเอนต์ต้องรอให้เซิร์ฟเวอร์ส่งข้อมูลมาให้จนครบทั้งไฟล์ก่อน (File download, progressive download [8]) จะต้องเสียเวลารอ โดยในระหว่างการรอการดาวน์โหลดไคลเอนต์ไม่สามารถเปิดเล่นไฟล์วิดีโอได้ อีกทั้งทางฝั่งเซิร์ฟเวอร์ต้องส่งข้อมูลให้เสร็จทั้งไฟล์ก่อน ทำให้เซิร์ฟเวอร์มีภาระงานสูงอยู่ตลอดเวลา การแบ่งส่วนของข้อมูลไฟล์วิดีโอส่งจากเซิร์ฟเวอร์ไปยังไคลเอนต์นั้น ทำให้เซิร์ฟเวอร์มีภาระงานน้อยลงและทำให้ไคลเอนต์แยกส่วนไฟล์จากการดาวน์โหลดและส่วนที่กำลังเปิดเล่นอยู่ทำให้เกิดความยืดหยุ่นและคล่องตัวในการทำงานกับไฟล์ที่มีขนาดใหญ่และใช้ทรัพยากรระบบมากอย่างไฟล์วิดีโอ



รูปที่ 2.1 การให้บริการ streaming ผ่านเครือข่ายอินเตอร์เน็ต[9]

การส่งข้อมูลในลักษณะนี้จำเป็นต้องมีกระบวนการเพิ่มเติมเพื่อให้การส่งข้อมูลจากเซิร์ฟเวอร์ไปยังไคลเอนต์สามารถกระทำได้อย่างรวดเร็วและเป็นไปโดยราบรื่น กล่าวคือก่อนที่ไคลเอนต์จะเริ่มดาวน์โหลดไฟล์วิดีนั้น ไคลเอนต์ต้องรู้ก่อนว่าส่วนย่อย ๆ ของไฟล์วิดีโอนั้นจัดเรียงตัวอย่างไรและไฟล์วิดีโอดังกล่าวเป็นไฟล์ประเภทใด ถูกเข้ารหัสวิดีโอ (video encoding) ประเภท

ใด เช่น MP4, webm หรือ vp9 เป็นต้น เพื่อให้ไคลเอนต์ได้รู้ว่าต้องเปิดไฟล์วิดีโอและถอดรหัสวิดีโอ (video decode) อย่างไร ซึ่งจะทำให้ไคลเอนต์สามารถเปิดไฟล์วิดีโอเล่นได้ทันทีเมื่อดาวน์โหลดเพียงส่วนย่อยของไฟล์วิดีโอมาโดยไม่ต้องรอให้ดาวน์โหลดเสร็จทั้งไฟล์ การทำงานของระบบการให้บริการไฟล์วิดีโอผ่านเครือข่ายอินเตอร์เน็ตประกอบไปด้วยส่วนต่าง ๆ ดังรูปที่ 2.1 แสดงโครงสร้างระบบการให้บริการไฟล์วิดีโอผ่านโปรโตคอล HTTP ซึ่งแยกส่วนประกอบออกเป็นส่วนต่าง ๆ ได้ดังนี้

1. Production เป็นส่วนของการเก็บภาพหรือการถ่ายทำ
2. Preparation and Staging เป็นส่วนกระบวนการเตรียมไฟล์วิดีโอและเข้ารหัสไฟล์
3. Distribution เป็นส่วนของแพร่ภาพหรือให้บริการไฟล์ผ่านระบบเครือข่าย
4. Consumption ส่วนของการเปิดไฟล์รับชมของผู้ใช้งาน

การให้บริการไฟล์วิดีโอแบบสตรีมมิ่งที่ใช้ในประเภทการรับชมแบบ VoD นั้น ไฟล์วิดีโอที่ใช้งานในระบบต้องเข้ารหัส ให้อยู่ในรูปแบบไฟล์ประเภท fMP4 (Fragmented MP4) ซึ่งมีอยู่ด้วยกันหลายประเภทการเข้ารหัส เช่น VP9[10], webm [10] เป็นต้น ซึ่งต้องใช้โปรแกรมประยุกต์การเข้ารหัสเช่น FFMPEG [11], Bento4 [12], MP4Box [13] เป็นต้น ซึ่งผู้ให้บริการไฟล์วิดีโอต้องทำการเข้ารหัสไฟล์ไว้ก่อนล่วงหน้าแล้ว จึงสามารถเปิดให้ผู้ใช้งานสามารถดาวน์โหลดไฟล์วิดีโอพร้อมทั้งเปิดรับชมได้

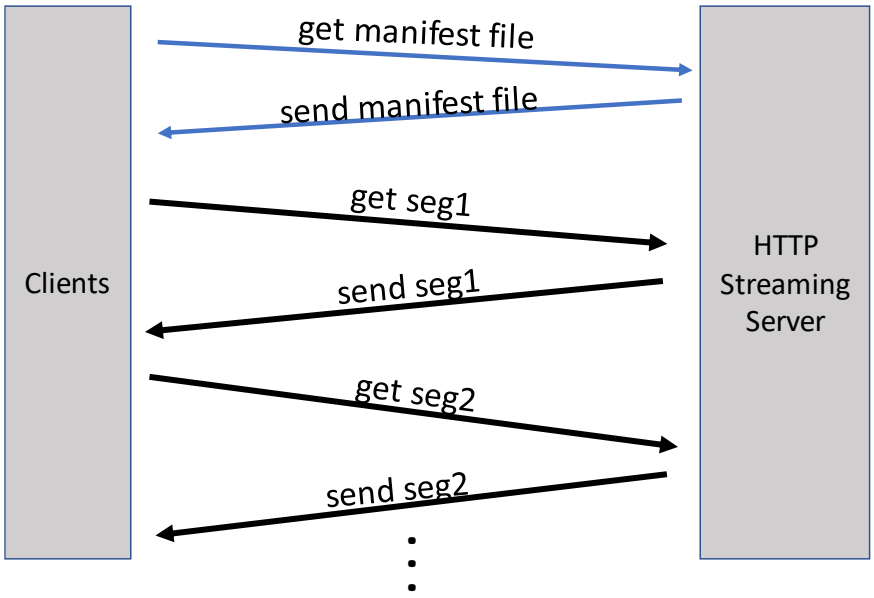
กระบวนการที่สำคัญในการให้บริการไฟล์วิดีโอ streaming คือกระบวนการแพร่ภาพหรือการให้บริการไฟล์ผ่านระบบเครือข่าย (Distribution) หรือวิธีการ streaming นั่นเอง (streaming method) โดยมักส่งผ่านข้อมูลโดยผ่านโปรโตคอล HTTP ซึ่งในปัจจุบันมีมาตรฐานของกระบวนการทำวิดีโอสตรีมมิ่งผ่านโปรโตคอล HTTP ที่เป็นที่ยอมรับและใช้กันอย่างแพร่หลาย เช่น HDS (HTTP Dynamic Streaming) [14], Smooth Streaming [15], HLS (HTTP Live Streaming) [16] และ DASH (Dynamic Adaptive Streaming over HTTP) [17,18,19] เป็นต้น ซึ่งในวิทยานิพนธ์นี้จะขอกล่าวถึงเพียงมาตรฐานที่เปิดให้นักวิจัยใช้งานได้ คือ HLS และ DASH

2.2.1 HTTP Live Streaming (HLS)

HLS เป็นมาตรฐาน streaming video ที่ใช้กับอุปกรณ์ของบริษัท Apple เป็นหลัก และเปิดให้ใช้ได้กับทุกแฟลตฟอร์มโทรศัพท์สมาร์ทโฟนและเว็บบราวเซอร์ หลักการทำงานคือไคลเอนต์จะต้องดาวน์โหลดไฟล์รายการเล่นหรือไฟล์ playlist เรียกว่า index file หรือ manifest ซึ่งเป็นไฟล์ที่ระบุว่าไฟล์วิดีโอที่กำลังจะเริ่มต้นเปิดนั้นเป็นไฟล์ประเภทใด จะต้องจัดเรียงการดาวน์

โหลดอย่างไร และถูกเข้ารหัสวิดีโออย่างไร โดยในมาตรฐาน HLS ไฟล์ index นั้นเป็นไฟล์มีนามสกุลไฟล์เป็น m3u8 หรือ .m3u

การดาวน์โหลดไฟล์วิดีโอของไคลเอนต์จะเริ่มจากการร้องขอไฟล์ index เพื่อให้ไคลเอนต์รับรู้ถึงรายละเอียดของไฟล์วิดีโอที่กำลังจะเริ่มเปิดเล่น ดังรูปที่ 2.2 หลังจากที่ไคลเอนต์ได้รับไฟล์ index เรียบร้อยแล้ว ไคลเอนต์จะทำการเริ่มกระบวนการดาวน์โหลดส่วนของไฟล์วิดีโอโดยเรียกจากต้นไฟล์ไปจนจบไฟล์

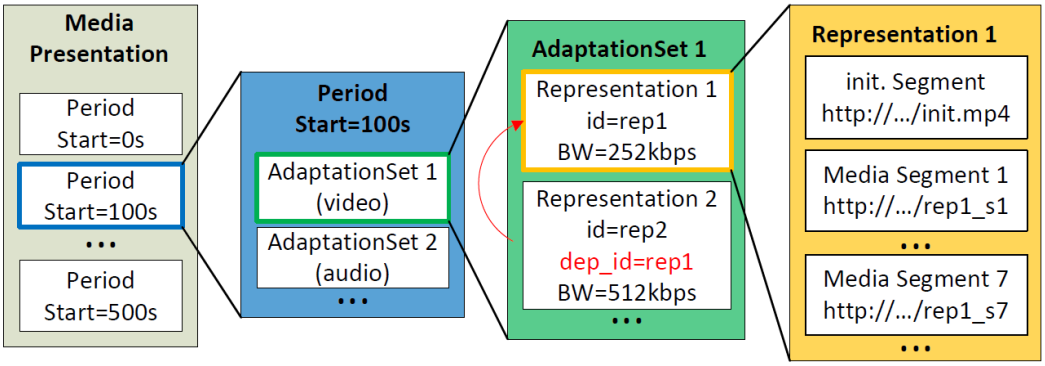


รูปที่ 2.2 กระบวนการดาวน์โหลดไฟล์วิดีโอ streaming

2.2.2 Dynamic Adaptive Streaming over HTTP (DASH)

DASH หรือ MPEG-DASH เป็นมาตรฐาน streaming video ที่กำลังถูกนำมาใช้อย่างแพร่หลายในปัจจุบัน เนื่องจากความคล่องตัวในการใช้โดยคำนึงถึงคุณภาพในการรับชมของผู้ใช้งาน (Quality of Experience: QoE) การเชื่อมต่อข้อมูลระหว่างไคลเอนต์และเซิร์ฟเวอร์ผ่านโปรโตคอล HTTP ให้บริการไฟล์และการร้องขอไฟล์สามารถระบุช่วงไฟล์หรือ byte range request [20] ที่ต้องการได้ ซึ่งเป็นคุณลักษณะหนึ่งของมาตรฐาน MPEG-DASH โดยไฟล์ที่ให้บริการจะต้องเข้ารหัสวิดีโอ (Encode) ให้อยู่ในประเภทไฟล์ fragmented MP4 ในกระบวนการเข้ารหัสไฟล์วิดีโอนั้น จะต้องกำหนดพารามิเตอร์ต่าง ๆ เพื่อให้ได้ไฟล์ผลลัพธ์ในรูปแบบที่ต้องการ เช่น ประเภทไฟล์, คุณภาพของวิดีโอ (Quality) [21]หรือ bitrate การทำ index ให้กับไฟล์เพื่อจัดการในลักษณะ byte range เป็นต้น ไฟล์ข้อมูลผลลัพธ์จากการเข้ารหัสจะประกอบไปด้วยไฟล์วิดีโอที่แยกออกเป็นแต่ละ

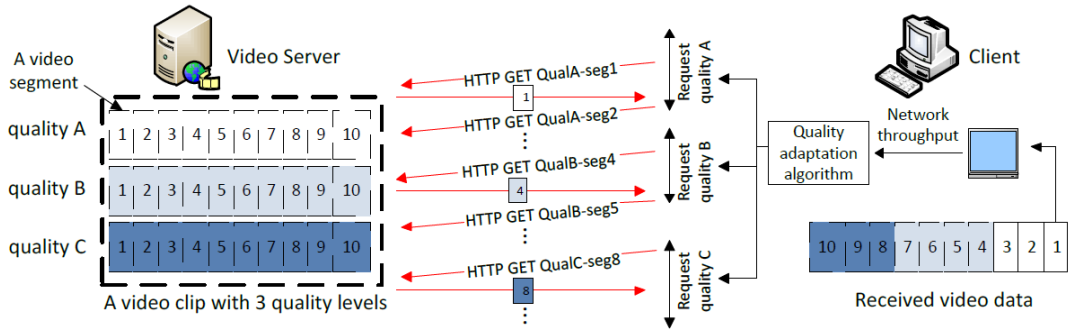
คุณภาพไฟล์ที่ต้องการ จากการกำหนดพารามิเตอร์ในขั้นตอนการเข้ารหัส และสามารถแยกระหว่างไฟล์ภาพวิดีโอและไฟล์เสียงออกจากกัน โดยการเข้ารหัสไฟล์วิดีโอของ MPEG-DASH นั้นจะได้ไฟล์ index เรียกว่า manifest [22] หรือไฟล์ mpd ซึ่งเป็นไฟล์ที่มีรายละเอียดการจัดเรียงและส่วนประกอบของไฟล์วิดีโอรวมทั้งหมดระบุไว้ โครงสร้างของไฟล์ manifest ดังรูปที่ 2.3



รูปที่ 2.3 โครงสร้างภายในของไฟล์ Manifest [23]

ไฟล์ manifest เป็นไฟล์ที่มีโครงสร้างการจัดเก็บข้อมูลแบบ XML [24] (Extensible Markup Language) ประกอบด้วยโครงสร้างระดับชั้นของข้อมูลรายละเอียดไฟล์วิดีโอและระบุคุณภาพของไฟล์รวมถึงการจัดเรียงของไฟล์วิดีโอ ในการร้องขอไฟล์วิดีโอจากไคลเอนต์แต่ละครั้งไคลเอนต์จะดาวน์โหลดไฟล์ manifest ก่อนที่จะเริ่มต้นดาวน์โหลดไฟล์ และไคลเอนต์จะใช้รายละเอียดภายในไฟล์ manifest ในการอ้างอิงการดาวน์โหลดแต่ละส่วนของไฟล์วิดีโอ นอกจากนี้ไฟล์ manifest ยังระบุประเภทของการถอดรหัสไฟล์วิดีโอ เพื่อให้ไคลเอนต์สามารถเปิดเล่นไฟล์วิดีโอได้อย่างถูกต้อง

ในกระบวนการเข้ารหัสไฟล์ต้องระบุค่าของพารามิเตอร์เวลาของวิดีโอในแต่ละช่วงของวิดีโอคือค่า segment duration [19,21] ซึ่งเป็นค่าที่แบ่งไฟล์วิดีโอออกเป็นช่วงเล็ก ๆ ที่เรียกว่า segment และค่า segment duration มีค่าเท่ากันทุก segment โดยแต่ละ segment ในไฟล์วิดีโอเดียวกันไม่ว่าจะเป็นระดับ bitrate ใดก็ตามจะมีค่า segment duration ที่เท่ากันทั้งหมด แต่ขนาดข้อมูลในหน่วยไบต์ (byte) จะไม่เท่ากันขึ้นอยู่กับความแตกต่างของข้อมูลวิดีโอในแต่ละ segment ซึ่งทำให้ไม่สามารถระบุขนาดของแต่ละ segment สามารถทำได้เพียงระบุค่า segment duration



รูปที่ 2.4 กระบวนการสื่อสารข้อมูลในระบบให้บริการไฟล์วิดีโอตามมาตรฐาน MPEG-DASH [21]

Segment duration เป็นค่าที่บ่งบอกว่าวิดีโอถูกแบ่งย่อยออกเป็น segment เล็ก ๆ ด้วยช่วงเวลาเท่าไร และเป็นค่าบ่งบอกให้ไคลเอนต์ทำการดาวน์โหลดช่วงวิดีโอได้อย่างถูกต้อง กล่าวคือ หาก segment duration มีค่า 4 วินาทีหมายถึงทุก ๆ segment ย่อยจะมีวิดีโอบรรจุอยู่ 4 วินาที ไคลเอนต์จะส่งการร้องขอออกไปทุก ๆ 4 วินาที กระบวนการดาวน์โหลดของไคลเอนต์ของระบบให้บริการไฟล์วิดีโอ streaming แบบ MPEG-DASH จะเหมือนกับระบบให้บริการไฟล์วิดีโอแบบ HLS โดยเริ่มจากร้องขอไฟล์ manifest หลังจากนั้นจะเริ่มการร้องขอ segment ย่อยและเปิดเล่นวิดีโอไปด้วย ดังรูปที่ 2.4 ในระหว่างการเปิดเล่นวิดีโออยู่นั้น ไคลเอนต์สามารถเปลี่ยนแปลงระดับคุณภาพของวิดีโอ (bitrate) ที่กำลังดาวน์โหลดอยู่ได้โดยการเปลี่ยนไปร้องขอวิดีโอที่มีเนื้อหาเดียวกันแต่ต่างไฟล์ในช่วง segment duration เดียวกัน ทำให้การเปิดเล่นไฟล์วิดีโอของผู้ใช้งานเป็นไปอย่างราบรื่น ในกรณีที่มีผู้ใช้งานจำนวนมากพร้อม ๆ กัน การร้องขอของไฟล์วิดีโอของไคลเอนต์มักต้องการความรวดเร็วในการตอบสนองของเซิร์ฟเวอร์ แต่ไฟล์วิดีโอเป็นไฟล์ข้อมูลที่ต้องเปิดเล่นต่อเนื่องและมักจะเป็นไฟล์ที่มีขนาดใหญ่ เซิร์ฟเวอร์ต้องใช้ระยะเวลาในการอ่านข้อมูลเพื่อตอบสนองความต้องการของไคลเอนต์ ทำให้ต้องมีระบบที่เข้ามาลดภาระการอ่านและเขียนข้อมูลของเซิร์ฟเวอร์

2.3 แคชที่ใช้งานในระบบสตรีมมิ่ง (Streaming Cache)

เพื่อลดภาระงานของระบบในการอ่านเขียนข้อมูลของ backend storage นั้นได้มีแนวคิดในการปรับปรุงประสิทธิภาพของแหล่งจัดเก็บข้อมูลที่สามารถอ่านและเขียนโดยมีการนำแคชมาใช้กับระบบให้บริการไฟล์วิดีโอแบบสตรีมมิ่ง [25] กล่าวคือใช้แหล่งจัดเก็บข้อมูลที่มีความเร็วของการอ่านและเขียนข้อมูลสูงกว่าแหล่งจัดเก็บข้อมูลหลักโดยในการอ่านข้อมูลจะอ่าน

ข้อมูลที่อยู่ในแคชก่อน หากพบข้อมูลอยู่ในแคช (cache hit) ก็จะสามารถส่งข้อมูลกลับไปยังผู้ร้องขอได้อย่างรวดเร็ว แต่หากไม่พบข้อมูลอยู่ในแคช (cache miss) จำเป็นต้องอ่านข้อมูลจาก backend storage จากนั้นเขียนข้อมูลที่อ่านมาลงในแคชและลบข้อมูลออกจากแคชเมื่อมีพื้นที่เหลือไม่เพียงพอในการจัดเก็บข้อมูลชุดใหม่ที่จะเขียนลงไป ซึ่งแคชจะช่วยให้ระบบให้บริการไฟล์วิดีโอสามารถทำงานได้อย่างรวดเร็วขึ้นและสามารถรองรับการเชื่อมต่อกับผู้ใช้งานได้มากขึ้นด้วยเช่นกัน ในปัจจุบันได้มีการพัฒนาระบบแคชเพื่อใช้งานร่วมกับการใช้บริการไฟล์วิดีโอผ่านเครือข่ายอินเตอร์เน็ต เช่น CDN (Content Delivery Network) [26,27], OCA (Open Connect Appliance) [28], Edge Cache [29] หรืออาจจะใช้แหล่งจัดเก็บข้อมูลประเภทหน่วยความจำหลัก (main memory) เป็นต้น โดยแคชจะมีพื้นที่จัดเก็บน้อยกว่าแหล่งจัดเก็บข้อมูลหลัก ประกอบกับพฤติกรรมการใช้งานของผู้ใช้ทำให้มีความต้องการอ่านและเขียนข้อมูลของแคชในอัตราที่สูง ดังนั้นจึงจำเป็นต้องมีระบบจัดการการทำงานของแคชที่มีประสิทธิภาพ โดยมีนโยบายการแทนที่แคช เป็นตัวคอยจัดการตัดสินใจการนำเข้าข้อมูลเข้าสู่แคชและการตัดสินใจขับข้อมูลออกจากแคชตามสถานการณ์และเงื่อนไขที่ให้ความสำคัญ

2.4 นโยบายการแทนที่แคช

ในกรณีที่มีความจำเป็นต้องลบข้อมูลที่อยู่ในแคชเนื่องจากแคชเต็มหรือกรณีอื่น ๆ ระบบจำเป็นต้องลบข้อมูลที่มีความสำคัญน้อยที่สุดเมื่อเทียบกับข้อมูลที่อยู่ในแคชทั้งหมดในขณะนั้น ซึ่งต้องจัดการอย่างมีประสิทธิภาพด้วยนโยบายการแทนที่แคช (cache replacement policy) ซึ่งจะช่วยกลั่นกรองข้อมูลที่สำคัญให้อยู่ในแคชเป็นระยะเวลาที่ยาวนานกว่าข้อมูลที่สำคัญน้อยกว่าตามเงื่อนไขที่นโยบายนั้นให้ความสำคัญโดยสามารถแบ่งประเภทแคชตามลักษณะของตัวแปรที่นโยบายการแทนที่แคชให้ความสำคัญในการพิจารณาในขั้นตอนการแทนที่แคชดังนี้

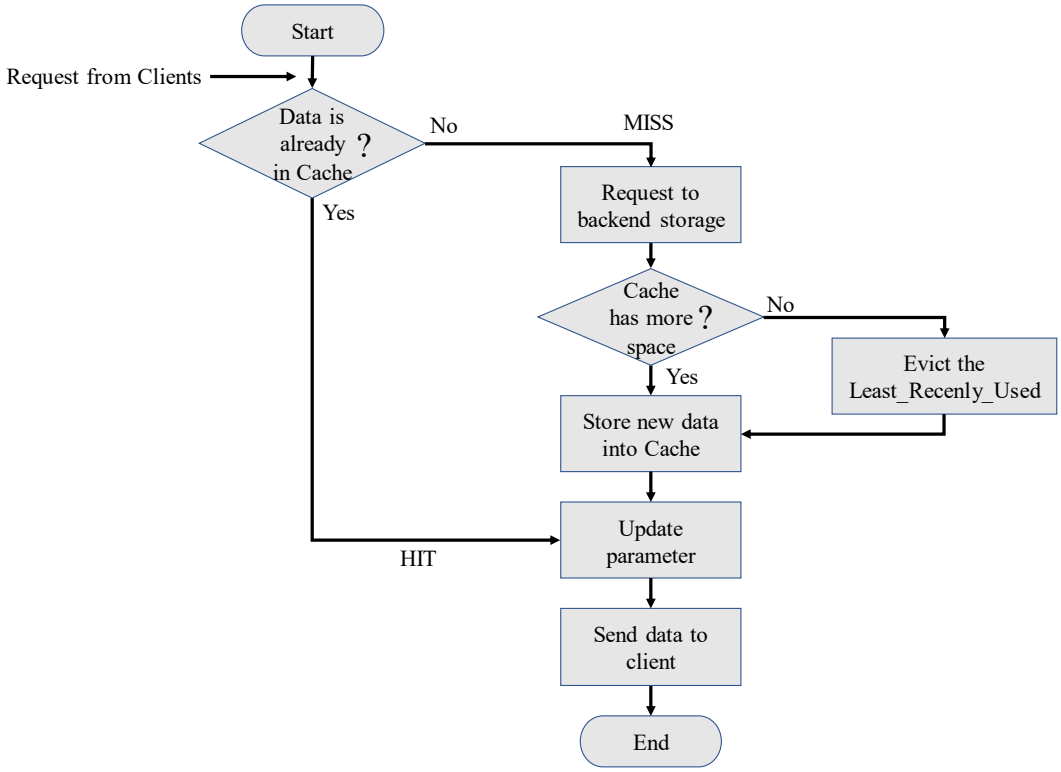
1. Age based
2. Renewal-frequency based
3. Server-load based

นโยบายแคชในประเภท Age based จะใช้ตัวแปรอายุของข้อมูลหรือความล่าสุดของการถูกใช้งานในการเปรียบเทียบความสำคัญของข้อมูล เช่น นโยบายแทนที่แคช Least Recently Used (LRU) [2] ในส่วนของนโยบายการแทนที่แคชประเภท Renewal-frequency based ใช้ความความถี่ของการใช้งานเป็นตัวแปรเพื่อคอยจัดลำดับความความสำคัญของข้อมูลภายในแคช

เช่น นโยบายแทนที่แคช Least Frequency Used (LFU) [3] ส่วนนโยบายแทนที่แคชประเภท Server-load based จะให้ความสำคัญกับภาระงานของเซิร์ฟเวอร์ในขณะนั้นเป็นตัวแปรหลักในการตัดสินใจของนโยบายแทนที่แคช เช่น WAVE [30] ซึ่งมีภาระงานในการคำนวณสูงและไม่เหมาะสมในการใช้งานกับระบบให้บริการไฟล์วิดีโอแบบ MPEG DASH อีกทั้งยังไม่คล่องตัวในการใช้งาน

2.4.1 นโยบายการแทนที่แคชแบบ Least-Recently-Used (LRU)

นโยบายการแทนที่แคชแบบ LRU จะให้ความล่าสุดของการถูกใช้งาน (recency) เป็นตัวแปรหลักในการพิจารณาระดับความสำคัญของข้อมูลที่อยู่ในแคชเมื่อมีความจำเป็นต้องลบข้อมูลออกเพื่อให้มีพื้นที่ว่างให้กับข้อมูลใหม่ โดยจะลบข้อมูลที่มีระดับความสำคัญน้อยที่สุด ซึ่งนโยบาย LRU จะลบข้อมูลที่ไม่ถูกใช้งานนานที่สุดก่อนดังรูปที่ 2.5 ข้อมูลที่จะอยู่ในแคชต่อไปอีกนานที่สุดในขณะนั้นคือข้อมูลที่เพิ่งถูกใช้งานไปล่าสุดนั่นเอง ซึ่งพารามิเตอร์ที่ใช้ในการเปรียบเทียบคือ เวลาที่ถูกเข้าถึง โดยทุกครั้งที่มีการเข้าถึงข้อมูล จะกำหนดค่าเวลาการเข้าถึงใหม่ให้กับข้อมูลที่ถูกเข้าถึงนั้นเสมอ ข้อมูลที่อยู่ในแคชนานที่สุดโดยไม่ได้ถูกใช้งานเมื่อเทียบกับข้อมูลอื่น ๆ ในขณะนั้นจะถูกลบออกจากแคชหากเกิดกรณีแคชเต็มดังรูปที่ 2.5 ซึ่งข้อมูลที่ถูกใช้งานอยู่ตลอดเวลาจะเป็นข้อมูลที่อยู่ในแคชเสมอ

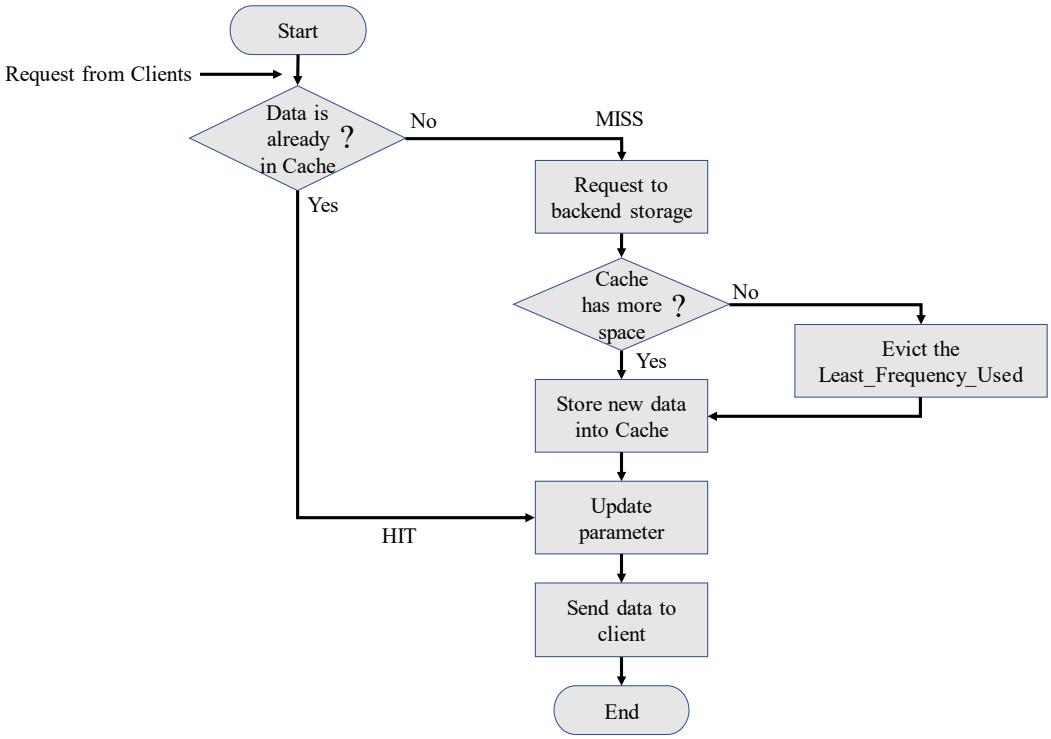


รูปที่ 2.5 ผังงานการทำงานนโยบายแคช Least Recently Used

นโยบายแทนที่แคช LRU เหมาะสมกับข้อมูลที่มีความเคลื่อนไหวอยู่ตลอดเวลา และไม่สามารถชี้ชัดได้ว่าข้อมูลชุดใดได้รับความนิยมมากกว่ากัน ซึ่งนโยบาย LRU จะเปลี่ยนไปให้ความสำคัญกับกลุ่มข้อมูลที่มีการใช้งานล่าสุดเสมอ ดังนั้นหากข้อมูลกลุ่มใดเพิ่งถูกใช้งานไปในระยะเวลาที่ผ่านไปไม่นานก็จะยังคงเป็นชุดข้อมูลที่อยู่ในแคช แต่หากผ่านการใช้งานไปเป็นระยะเวลานานแล้วและถูกเรียกกลับมาใช้ใหม่นั้น ข้อมูลดังกล่าวอาจถูกลบออกไปจากแคชแล้วก่อนหน้าที่จะถูกเรียกใช้อีกครั้ง

2.4.2 นโยบายการแทนที่แคชแบบ Least-Frequency-Used (LFU)

นโยบายแทนที่แคช LFU เป็นโยบายที่ใช้ความถี่ของการถูกใช้งานของข้อมูลเป็นตัวแปรสำคัญเพื่อพิจารณาลำดับความสำคัญของข้อมูล ในกรณีที่แคชถูกใช้ไปจนเต็มความจุและข้อมูลใหม่ที่ไม่ได้อยู่ในแคชถูกเรียกใช้ ข้อมูลนี้จะถูกเขียนลงไปในแคช ซึ่งนโยบาย LFU จะเลือกลบข้อมูลที่มีความถี่ (frequency) ของการถูกใช้งานน้อยที่สุดในขณะนั้นออกจากแคช เพื่อให้มีพื้นที่ว่างเพียงพอต่อการเขียนข้อมูลชุดใหม่ลงไป ดังรูปที่ 2.6 ข้อมูลที่มีความถี่ของการใช้งานสูงสุดจะอยู่ในแคชนานที่สุด



รูปที่ 2.6 นโยบายแทนที่แคชแบบ Least Frequency Used

ความถี่ของการถูกใช้งานจะเพิ่มขึ้นเรื่อย ๆ และเป็นค่าความถี่สะสม หากข้อมูลเป็นข้อมูลชุดเดิม และข้อมูลที่ถูกใช้งานบ่อยครั้งเป็นข้อมูลชุดเดียวกันตลอด นโยบาย LFU จะสามารถทำงานอย่างมีประสิทธิภาพ แต่หากมีการเปลี่ยนแปลงพฤติกรรมการใช้งานข้อมูล เช่น กลุ่มที่เคยถูกใช้งานบ่อยครั้งที่สุดกลับไม่ถูกใช้อีกเลย ความถี่สะสมเดิมจะเป็นความถี่ที่เชื่อถือไม่ได้อีกต่อไป เช่น อาจเกิดกรณีที่ข้อมูลหนึ่งถูกใช้งานด้วยความถี่สูงมากในช่วงระยะเวลาสั้นๆ จนทำให้ข้อมูลนั้นเป็นข้อมูลที่มีความถี่ของการใช้งานสูงที่สุด แต่หลังจากนั้นไม่ถูกเรียกใช้งานอีกเลย ข้อมูลดังกล่าวจะยังคงอยู่ในแคชจนกว่าความถี่สะสมที่เคยมีค่าสูงนั้น จะเป็นความถี่สะสมต่ำที่สุดเมื่อเทียบกับข้อมูลอื่น ๆ ในแคช และถูกลบออกจากแคชเมื่อแคชเต็ม ซึ่งต้องใช้ระยะเวลานานกว่าข้อมูลดังกล่าวจะถูกลบออกจากแคช

หากมีการเปลี่ยนแปลงรูปแบบการใช้งานบ่อยครั้งหรือผู้ใช้งานเปลี่ยนความสนใจไปใช้ข้อมูลชุดอื่นอยู่ตลอดเวลา จะทำให้นโยบาย LFU ต้องใช้ระยะเวลานานในการเปลี่ยนแปลงชุดข้อมูลภายในแคช เพื่อให้ความถี่สะสมเป็นความถี่ที่ถูกต้อง ซึ่งขึ้นอยู่กับลักษณะการใช้งาน และในระบบให้บริการไฟล์วิดีโอนั้นพฤติกรรมของผู้ใช้งานเป็นตัวแปรสำคัญ หากระบบทำงานไม่สอดคล้องกับการใช้งานหรือมีการเปลี่ยนแปลงพฤติกรรมอยู่บ่อยครั้ง เช่นภาพยนตร์ที่ได้รับความสนใจหรือผู้ใช้นิยมภาพยนตร์เรื่องใดเรื่องหนึ่ง หรือวิดีโอเพียงบางช่วงบางตอนอยู่เป็นระยะเวลาสั้น ๆ นั้น แต่มีความถี่การใช้งานที่สูงในระยะเวลาสั้น ๆ ทำให้ความถี่สะสมมีค่าสูงสุดเมื่อเทียบกับข้อมูลอื่น ๆ ที่อยู่ในแคช จะกลายเป็นข้อมูลที่อยู่ในแคชนานที่สุดแม้ไม่ถูกร้องขอใช้งานอีกเลย หากมีข้อมูลที่ถูกใช้งานในรูปแบบพฤติกรรมดังกล่าวเป็นจำนวนมาก จะทำให้พื้นที่ของแคชถูกใช้ไปโดยเปล่าประโยชน์เป็นระยะเวลานาน และทำให้ข้อมูลที่เพิ่งเข้ามาในแคชถูกลบออกไปและอาจถูกนำเข้ามาใหม่และลบออกอยู่หลายรอบ (repeat evict) กว่าจะมีความถี่สะสมที่เพียงพอ

2.4.3 นโยบายการแทนที่แคชที่อ้างอิงค่า Popularity

นโยบายการแทนที่แคชที่พิจารณาค่าของความนิยมเช่น File-based popularity policy [31] จะพิจารณาค่าความนิยมที่ได้จากการคำนวณของแต่ละไฟล์ แต่เนื่องจากไฟล์วิดีโอมีขนาดใหญ่ไม่สามารถบรรจุไฟล์จำนวนมากลงในแคชได้ และผู้ใช้งานส่วนมากมักไม่ได้รับชมวิดีโอทั้งไฟล์และให้ความสนใจกับบางช่วงของไฟล์เป็นพิเศษที่มีเนื้อหาวิดีโอส่วนที่สนใจอยู่ ทำให้การพิจารณาค่าความนิยมในระดับไฟล์นั้นไม่ตรงตามลักษณะการใช้งาน จึงจำเป็นต้องแยกพิจารณาในระดับส่วนย่อยของไฟล์ดังเช่นในนโยบาย Chunked-based policy [7] ที่แยกการคำนวณค่าความนิยมของไฟล์วิดีโอในระดับส่วนย่อย (chunk) และสร้างความเชื่อมโยงระหว่างส่วนย่อยนั้นด้วยการกำหนดให้ส่วนย่อยถัดไปจากส่วนย่อยที่กำลังถูกร้องขออยู่มีค่าความนิยมเพิ่มขึ้นและเกี่ยว

เนื่องกัน ซึ่งในความเป็นจริงผู้ใช้งานอาจไม่ได้ต้องการรับชมส่วนย่อยถัดไปโดยมีความเป็นอิสระจากกัน ทำให้ค่าความนิยมที่ได้นั้นคลาดเคลื่อนจากความเป็นจริง ได้มีการนำเสนอนโยบายการแทนที่แคชที่อาศัยหลักการทางเศรษฐศาสตร์มาปรับใช้คือนโยบายการแทนที่แคช Pareto-based policy [32] โดยจะพิจารณาอันดับ (rank) ค่าความนิยมของข้อมูลโดยอ้างอิงกฎของ Pareto [33] ซึ่งจะแบ่งจำนวนข้อมูลออกเป็นสัดส่วน 80/20 และจะให้ความสำคัญกับข้อมูล 20% แรกของอันดับค่าความนิยมซึ่งจัดเป็นกลุ่มข้อมูลที่มีความสำคัญที่สุดให้อยู่ในแคชและอีก 80% ของข้อมูลซึ่งมีจำนวนข้อมูลมากซึ่งมองว่ามีค่าความนิยมน้อยและมีผลต่อระบบน้อย แต่ในความเป็นจริงจำนวนสัดส่วนของกลุ่มข้อมูลอาจมีการเปลี่ยนแปลงได้ตลอดเวลา และกลุ่มข้อมูลที่ได้รับความนิยมสูงอาจมีการเปลี่ยนแปลงตลอดเนื่องจากพฤติกรรมของผู้ใช้งาน ดังนั้นนโยบายการแทนที่แคชแบบ Pareto-based ไม่มีความยืดหยุ่นในการทำงานและอาจจะไม่สอดคล้องกับพฤติกรรมการใช้งานของผู้ใช้ที่มีต่อระบบให้บริการไฟล์วิดีโอสตรีมมิ่ง ซึ่งพฤติกรรมของผู้ใช้งานมีผลอย่างยิ่งต่อการทำงานของนโยบายการแทนที่แคช

2.5 พฤติกรรมของผู้ใช้งานระบบให้บริการไฟล์วิดีโอ

ระบบให้บริการไฟล์วิดีโอเป็นระบบที่ต้องให้บริการไฟล์ที่มักมีขนาดใหญ่และผู้ใช้จะเปิดใช้งานไฟล์อย่างต่อเนื่อง พฤติกรรมการใช้งานของผู้ใช้มีผลอย่างยิ่งต่อระบบ การเพิ่มแคชเข้ามาช่วยให้ระบบสามารถทำงานได้อย่างมีประสิทธิภาพมากยิ่งขึ้น แต่ยังไม่เพียงพอทั้งนี้ต้องพิจารณาถึงพฤติกรรมการใช้งานของผู้ใช้ ขึ้นอยู่กับว่านโยบายแคชสอดคล้องกับพฤติกรรมการใช้งานของผู้ใช้มากน้อยเพียงใด การเปลี่ยนแปลงพฤติกรรมและความสนใจของผู้ใช้งานที่มีต่อไฟล์วิดีโอ เช่น เปลี่ยนแปลงไปตามกระแสความนิยมอาจจะเป็นเพียงระยะเวลาสั้นๆ หรือมีแนวโน้มที่เพิ่มขึ้นเรื่อย ๆ หรือหลังจากเพิ่มขึ้นแล้วมีแนวโน้มที่จะถูกใช้อีกในระยะเวลาอันใกล้ ซึ่งนโยบาย LRU และ LFU ไม่สามารถตอบรับได้ทุกพฤติกรรมการใช้งาน ในงานวิจัยนี้จะให้ความสนใจพฤติกรรมการใช้งานสองแบบดังนี้

2.5.1 พฤติกรรมการใช้งานแบบ Uniform

พฤติกรรมการใช้งานแบบ Uniform [33,34] คือพฤติกรรมการใช้งานที่ผู้ใช้ทั้งหมดให้ความสนใจและร้องขอข้อมูลแต่ละชุดในจำนวนครั้งที่เท่ากันทุกชุดข้อมูล ซึ่งทำให้ทุกชุดข้อมูลถูกใช้งานด้วยความถี่ที่เท่ากันทั้งหมดและมีโอกาสที่เท่ากันในการถูกจัดเก็บในแคช ซึ่งสามารถบรรยายได้ด้วยสมการที่ (1)

$$P(x) = \frac{1}{N} \quad (1)$$

โดยที่

N คือจำนวนข้อมูลทั้งหมด

จากสมการที่ (1) พฤติกรรมแบบ Uniform จะมีการแข่งขันกันของความต้องการของผู้ใช้ข้อมูลและการจัดการนำข้อมูลลงสู่แคชสูงเพราะทุก ๆ ข้อมูลมีความสำคัญเท่าเทียมกัน

2.5.2 พฤติกรรมการใช้งานแบบ Zipf-like

ไฟล์วิดีโอหรือภาพยนตร์จำนวนหนึ่งจะได้รับความสนใจและเปิดรับชมโดยผู้ใช้จำนวนมาก ในขณะที่อีกหลายไฟล์จะได้รับความสนใจน้อย ทั้งนี้ขึ้นอยู่กับกระแสความนิยมที่ส่งผลต่อความสนใจของผู้ใช้งานและมีผลทำให้ผู้ใช้งานเลือกรับชมไฟล์ตามความนิยมนั้น ๆ เช่น ข้อมูลการรับชมไฟล์วิดีโอใน YouTube [35] หรือภาพยนตร์ที่ได้รับความนิยมเนื่องมาจากกระแสทางสังคมสื่อออนไลน์ หรืออาจจะเป็นเพราะความชื่นชมในตัวนักแสดง หรือเป็นเรื่องที่กำลังได้รับความสนใจอย่างมาก มีการกระจายตัวของความนิยมต่อไฟล์วิดีโอในรูปแบบ Zipf-Like Distribution [33,34] ซึ่งเป็นไปตามสมการที่ (2)

$$P(x) = \frac{1}{x^{\alpha} \sum_{i=1}^{N} (1/i)^{\alpha}} \quad (2)$$

โดยที่

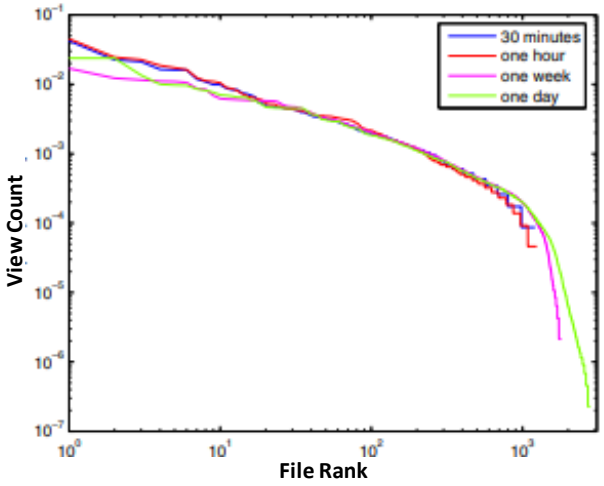
N เป็นจำนวนข้อมูลทั้งหมด

$x = 1, 2, 3, \ldots, N$

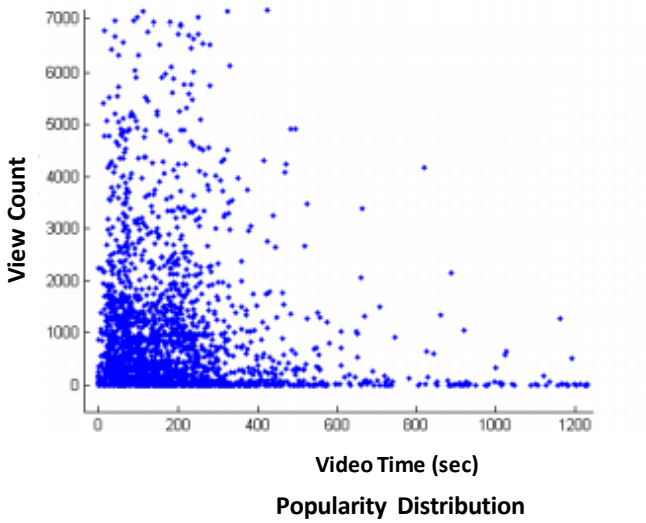
และ $\alpha \geq 1$

ได้มีการศึกษาวิจัยรูปแบบพฤติกรรมการใช้งานและรับชมไฟล์วิดีโอของระบบ YouTube [35] ทั้งในระบบให้บริการไฟล์วิดีโออื่น ๆ [36] เมื่อพิจารณาลำดับความนิยมของไฟล์วิดีโอพบว่าผู้ใช้งานส่วนใหญ่จะให้ความสนใจและเปิดรับชมไฟล์วิดีโอที่มีความนิยมในลำดับต้นๆ เป็นไปตามการกระจายตัวของความนิยมแบบ Zipf-like ดังรูปที่ 2.7 แต่จะลดลงอย่างรวดเร็วในช่วงลำดับท้ายๆ ซึ่งเป็นผลมาจากไฟล์วิดีโอในระบบมีเป็นจำนวนมากและมีไฟล์วิดีโอจำนวนหนึ่งที่ไม่ได้รับความนิยมเลย (unpopular) ถูกเรียกใช้งานน้อยมากหรืออาจไม่ถูกเรียกใช้งานเลย ทำให้การกระจายตัวของความนิยมจะเป็นแบบเชิงเส้น (linear) และในตอนท้ายจะเป็นแบบฟังก์ชันเลขชี้

กำลัง (exponential) เมื่อเวลาผ่านไปไฟล์วิดีโอในกลุ่มที่ได้รับนิยมสูงอาจเป็นที่ต้องการน้อยลง หรือไม่เป็นที่ต้องการ เกิดการลดทอนความนิยม (Popularity attenuation) ในทางกลับกันไฟล์วิดีโอในกลุ่มที่ไม่ได้รับความนิยมอาจได้รับความนิยมสูงขึ้น ทั้งนี้ขึ้นอยู่กับปัจจัยที่มีผลต่อความนิยม



รูปที่ 2.7 อันดับของไฟล์วิดีโอเมื่อเรียงตามจำนวนผู้ใช้งานในช่วงระยะเวลาต่าง ๆ [36]



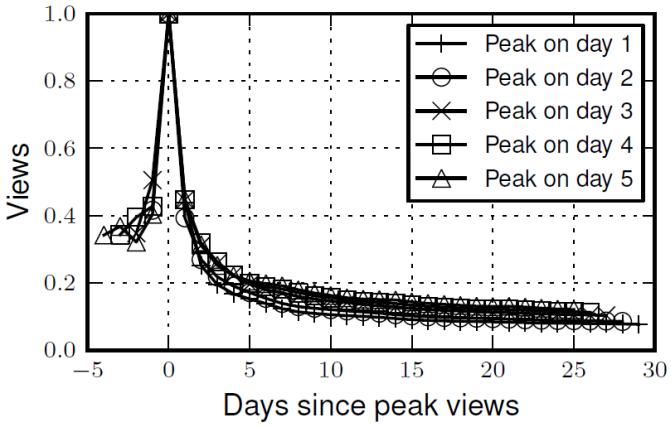
รูปที่ 2.8 ความนิยมของไฟล์วิดีโอเรียงตามช่วงเวลาของวิดีโอ [36]

นอกจากนี้ ยังพบว่าความนิยมต่อไฟล์วิดีโอนั้น มีการจัดเรียงแบบ Zipf-like ทั้งในระดับไฟล์และระดับช่วงเวลาวิดีโอภายในไฟล์ดังรูปที่ 2.8 โดยผู้ใช้งานมักจะเปิดรับชมในช่วงต้นของวิดีโอแต่วิดีโอในช่วงหลังจากนั้นจะไม่ได้รับความนิยมเมื่อเปรียบเทียบกันภายในไฟล์วิดีโอ

เดียวกันและการให้ความสำคัญกับวิดีโอช่วงถัดไปที่กำลังดาวน์โหลดนั้น อาจไม่สอดคล้องกับพฤติกรรมของผู้ใช้งานระบบ

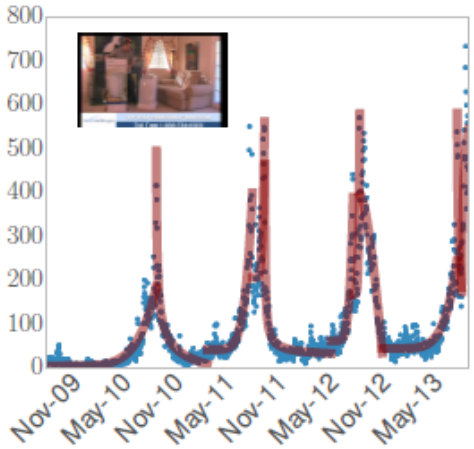
2.5.3 การเปลี่ยนแปลงความถี่ของการถูกใช้งานของไฟล์วิดีโอในระบบ HTTP สตรีมมิ่ง

พฤติกรรมการใช้งานระบบของผู้ใช้ทำให้สามารถแจกแจงระดับความถี่ของการถูกใช้งานของไฟล์วิดีโอในระบบได้ ในนโยบายแทนที่แคชที่ใช้ความถี่และความนิยมในการจัดจัดการ มักใช้ค่าความถี่สะสมของการถูกใช้งานมาพิจารณา และเมื่อเวลาผ่านไปความถี่สะสมของการถูกใช้งานนั้นจะเพิ่มขึ้นโดยไม่ลดลง เป็นการรวมความถี่สะสมในอดีตที่ผ่านมาทั้งหมด จากผลการศึกษาใน [37] เมื่อจำแนกความถี่ของการถูกใช้งานในแต่ละวันที่ผ่านไป (inter arrival time) พบว่าความถี่ของการถูกงานจะเพิ่มขึ้นจนถึงจุดสูงสุดและลดลงหลังจากผ่านจุดสูงสุดไปแล้ว ดังรูปที่ 2.9 จะเห็นว่าเมื่อเวลาผ่านไปความถี่ของการถูกใช้งานจะลดลงตามระยะเวลาที่ผ่านไปหลังจากมีความถี่ของการใช้งานสูงสุด



รูปที่ 2.9 การเปลี่ยนแปลงความถี่ของการถูกใช้งานไฟล์วิดีโอของ YouTube เมื่อแบ่งตามวันที่ไฟล์ถูกเผยแพร่ในระบบ [37]

นอกจากการลดลงของอัตราความถี่ของการใช้งานเมื่อจำแนกเป็นช่วงเวลาแล้วนั้น ความถี่ของการถูกใช้งานไฟล์วิดีโออาจมีการเพิ่มขึ้นได้ด้วยเช่นกัน เนื่องจากปัจจัยอื่น ๆ เช่น ความนิยมเพิ่มขึ้นจากความต้องการใช้งานตามช่วงเวลา ทั้งนี้ขึ้นอยู่กับเนื้อหาของไฟล์วิดีโอนั้น ๆ เช่น ในรูปที่ 2.10 เป็นไฟล์วิดีโอเกี่ยวกับการซ่อมเครื่องแอร์คอนดิชันเนอร์ ซึ่งมีความถี่ของการใช้งานที่เพิ่มขึ้นเนื่องจากมีความต้องการของผู้ใช้สูงสุดในช่วงฤดูร้อนของทุกปีและลดลงต่ำสุดในช่วงฤดูหนาว และจะเพิ่มขึ้นอีกหลังจากเริ่มเข้าสู่ฤดูร้อนอีกครั้ง เป็นต้น ซึ่งเป็นปัจจัยที่มีผลต่อความนิยมที่ทำให้ค่าความถี่ของการถูกใช้งานเปลี่ยนแปลงไป



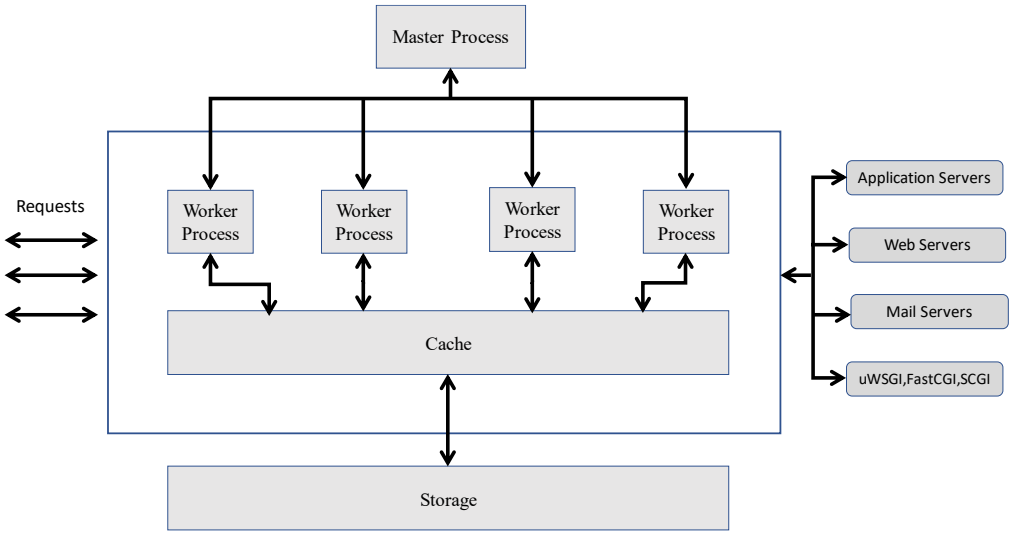
รูปที่ 2.10 การเปลี่ยนแปลงความถี่ของการถูกใช้งานที่มีผลมาจากปัจจัยความต้องการไฟล์วิดีโอเกี่ยวกับการซ่อมเครื่องแอร์คอนดิชันเนอร์ของ YouTube [38]

2.6 การใช้งาน Nginx

Nginx [39] เป็น HTTP server ที่มักถูกใช้เป็นเว็บเซิร์ฟเวอร์ให้บริการแก่ผู้ใช้งานเว็บไซต์หรือให้บริการอื่น ๆ ที่ให้บริการผ่านโปรโตคอล HTTP, HTTPS เป็นต้น ระบบให้บริการไฟล์วิดีโอในปัจจุบันเป็นระบบให้บริการผ่านทางโปรโตคอล HTTP เป็นหลัก เนื่องจากสามารถกระจายการให้บริการและรองรับการใช้งานและเปิดให้ผู้ใช้งานจากหลากหลายประเภทอุปกรณ์สามารถเข้าถึงได้โดยการเชื่อมต่อผ่านโปรโตคอล HTTP ในกรณีที่มีผู้ใช้งานจำนวนมากได้ และระบบให้บริการไฟล์วิดีโอแบบ MPEG-DASH ให้บริการไฟล์ในรูปแบบการร้องขอช่วงของข้อมูล (byte range) และ Nginx รองรับมาตรฐานการร้องขอข้อมูลแบบ HTTP byte range [6] Nginx จะทำงานในลักษณะ master-workers โดยสามารถมี worker ได้หลายตัวร่วมกันทำงานดังรูปที่ 2.11

การทำงานของ Nginx นั้นจะแตกต่างกันไปเพื่อให้บริการแต่ละประเภทการร้องขอจากไคลเอนต์โดยมีรายละเอียดการประมวลผลที่แตกต่างกันการทำงานแต่ละครั้งจะประกอบด้วยเฟสย่อย (phase) ซึ่งเฟสการทำงานของ Nginx ประกอบด้วยเฟสทำงานต่อเนื่องกันแต่แบ่งแยกกันอย่างชัดเจนจะแบ่งแยกตามขั้นตอนการทำงานของ Nginx โดยเปิดให้แทรกการทำงานลงไปในบางเฟสได้ ซึ่งจะกล่าวถึงการแทรกการทำงานนี้ในหัวข้อเฟรมเวิร์ค OpenResty [40]

จากรูปที่ 2.11 จะเห็นว่าการประมวลผลการร้องขอข้อมูลจากไคลเอนต์ของเซิร์ฟเวอร์แต่ละครั้งนั้นประกอบด้วย 11 เฟสย่อยซึ่งสามารถดูรายละเอียดเพิ่มเติมได้ในภาคผนวก ก แต่ละเฟสจะทำงานเป็นเทรดแยกออกจากกันอย่างชัดเจน



รูปที่ 2.11 โครงสร้างการทำงานของ Nginx

Nginx มีระบบจัดการแคชในตัวเอง แต่ยังไม่เพียงพอเนื่องจากขนาดของข้อมูลวิดีโอที่มักมีขนาดใหญ่ทำให้การใช้งานแคชภายในของ Nginx ไม่คล่องตัวต่อการขยายระบบประกอบกับนโยบายแทนที่แคชที่มียังไม่สอดคล้องกับพฤติกรรมการทำงานของระบบ

2.7 ซอฟต์แวร์แคช Redis

ปัจจุบันได้มีการพัฒนาซอฟต์แวร์แคชที่สามารถรองรับการใช้งานการจัดเก็บข้อมูลที่ต้องใช้ความเร็วในการอ่านและเขียน และเป็นข้อมูลที่ถูกใช้งานบ่อยครั้ง ตัวอย่างระบบแคช เช่น Memcached [41], Caffeine [42], Varnish [43] และ Redis [44] เป็นต้น ซึ่งซอฟต์แวร์แคชแต่ละตัวก็มีความโดดเด่นแตกต่างกัน ในวิทยานิพนธ์นี้ ผู้วิจัยได้นำเสนอการใช้ Redis ทำหน้าที่เป็นแคชเก็บข้อมูลไฟล์วิดีโอแบบ streaming ให้กับระบบให้บริการไฟล์วิดีโอผ่านโปรโตคอล HTTP ด้วยการ streaming ตามมาตรฐาน MPEG-DASH ซึ่งเป็นข้อมูลขนาดใหญ่และเป็นข้อมูลที่เรียงตัวต่อเนื่องอย่างหนาแน่น Redis มีโครงสร้างการจัดเก็บข้อมูลที่หลากหลาย ได้แก่ Strings, Sets, Hashes, Lists, Sorted Sets, Bitmaps และ Hyper LogLogs และยังมีคำสั่งต่าง ๆ เพื่อดำเนินการกับ

ข้อมูลที่เก็บไว้ได้อีกด้วย รวมไปถึงสามารถอ่านข้อมูลที่เก็บอยู่ใน Redis ด้วยการอ้างอิงข้อมูลเป็นช่วงไบต์ที่ต้องการได้

Redis เป็น in-memory แคชที่มีโครงสร้างข้อมูลให้เลือกใช้งานหลากหลาย ในงานวิจัยนี้ใช้ Redis รุ่น 4.0.11 โดยมีนโยบายการแทนที่แคชของ Redis ซึ่งสามารถควบคุมโดยการกำหนดค่าพารามิเตอร์ที่เกี่ยวข้องดังนี้

ตารางที่ 2.1 พารามิเตอร์ที่เกี่ยวข้องกับการจัดแคชของ Redis

พารามิเตอร์	ความหมาย
maxmemory	ขนาดความจุข้อมูลของ Redis มีหน่วยเป็นไบต์ (byte)
maxmemory-policy	นโยบายการแทนที่แคชที่ใช้กับข้อมูลที่เก็บอยู่ใน Redis
Expire-Time	อายุของข้อมูลที่อยู่ในแคชโดยกำหนดเป็นระยะเวลาที่ข้อมูลอยู่ในแคช หรือ TTL (Time to Live)

นโยบายการแทนที่แคชของ Redis สามารถใช้งานโดยการกำหนดค่าให้กับพารามิเตอร์ maxmemory-policy เมื่อเกิดกรณีความจุไม่เพียงพอไม่สามารถตอบสนองต่อข้อมูลที่เข้ามาใหม่ได้ โดยอ้างอิงจากค่าพารามิเตอร์ maxmemory ซึ่งค่าพารามิเตอร์ maxmemory-policy [45] มีรายละเอียดดังตารางที่ 2.2

ตารางที่ 2.2 นโยบายแทนที่แคชของ Redis

นโยบายแทนที่แคช	วิธีการเลือกลบข้อมูลเมื่อพื้นที่แคชเหลือไม่เพียงพอ
allkeys-lru	เลือกลบข้อมูลโดยใช้นโยบาย LRU กับข้อมูลทั้งหมดที่มีอยู่ในแคช
volatile-lru	เลือกลบข้อมูลโดยใช้นโยบาย LRU กับข้อมูลที่ถูกกำหนดค่า TTL เอาไว้และลบข้อมูลที่มี TTL เหลือน้อยที่สุดก่อน
allkeys-lfu	เลือกลบข้อมูลใช้นโยบาย LFU กับข้อมูลทั้งหมดที่มีอยู่ในแคช
volatile-lfu	เลือกลบข้อมูลโดยใช้นโยบาย LFU กับข้อมูลที่ถูกกำหนดค่า TTL เอาไว้และลบข้อมูลที่มี TTL เหลือน้อยที่สุดก่อน
allkeys-random	เลือกลบข้อมูลด้วยการสุ่มจากข้อมูลทั้งหมดที่มีอยู่ในแคช
volatile-random	เลือกลบข้อมูลด้วยการสุ่มจากข้อมูลที่มีค่า TTL
volatile-ttl	เลือกลบข้อมูลที่มีค่า TTL เหลือน้อยที่สุดก่อน
Noeviction	ไม่ใช้นโยบายการแทนที่แคชของ Redis

นโยบายแทนที่แคชของ Redis ที่มีอยู่เป็นโยบายพื้นฐานที่ใช้งานกับข้อมูลทั่วไป ดังนั้นในงานวิจัยนี้จะปิดการใช้งานนโยบายแทนที่แคชของ Redis และใช้งานนโยบายแทนที่แคชที่ผู้วิจัยนำเสนอแทน

2.8 เฟรมเวิร์ค OpenResty

OpenResty [40] เป็นเฟรมเวิร์คโอเพนซอร์สของการเขียนโปรแกรมเพื่อควบคุมการทำงานของ Nginx โดยสามารถเขียนด้วยโปรแกรมภาษา Lua [40] และฝัง (embed) โปรแกรมเข้าไปในส่วนต่าง ๆ ของ Nginx เพื่อให้เทรด (thread) ของ Nginx ทำงานตามต้องการ โดยการแบ่งการทำงานของ Nginx ออกเป็นเฟส (phase) และเปิดให้นักวิจัยสามารถเพิ่มส่วนของการทำงานเข้าไปในเฟสนั้น เช่น Initialization phase, Rewrite/Access phase, Contents phase และ Log phase เป็นต้น ซึ่งส่วนของโปรแกรมที่แทรกไปในเฟสการทำงานของ Nginx นั้นจะทำงานเป็นเทรดย่อยของเทรดหลักของ Nginx อีกทีหนึ่ง ซึ่งเทรดย่อยนี้จะมีลักษณะเป็น Light weight threads และทำงานในลักษณะ co-routine [40] และ co-socket [40] ของเทรดหลัก และบางคำสั่งที่สามารถสร้างเทรดย่อยลงไปอีกชั้นหนึ่งเพื่อให้ทำงานไปพร้อม ๆ กันกับเทรดย่อยเดิมและเทรดหลักของ Nginx ได้ เช่น คำสั่ง ngx.timer.at(premature) [40] เป็นต้น

บทที่ 3
วิธีดำเนินการวิจัย

ในบทนี้จะนำเสนอการวิเคราะห์ระบบและการออกแบบแคช พร้อมทั้งการพัฒนานโยบายการแทนที่แคชเพื่อให้ใช้ในระบบให้บริการไฟล์วิดีโอแบบ MPEG-DASH ได้อย่างมีประสิทธิภาพสอดคล้องกับพฤติกรรมการใช้งานระบบ

3.1 ระบบ DASH Streaming และการทำงานของ DASH Client

ผู้ใช้งานระบบให้บริการไฟล์วิดีโอสามารถเข้าใช้งานได้พร้อม ๆ กัน การร้องขอข้อมูลระหว่างไคลเอนต์กับเซิร์ฟเวอร์เป็นแบบ byte range request แต่ละการร้องขอที่ถูกส่งจากไคลเอนต์มายังเซิร์ฟเวอร์จะร้องขอส่วนของไฟล์วิดีโอแต่ละส่วนเรียกว่า segment โดยที่แต่ละ segment คือช่วงข้อมูลย่อย ๆ ของไฟล์วิดีโอนั้น ๆ และขนาดของทุก segment ถูกควบคุมโดยช่วงเวลาของวิดีโอในแต่ละ segment ช่วงเวลาดังกล่าวเรียกว่า segment duration ซึ่งมีค่าเท่ากันในทุก ๆ segment แต่ขนาดข้อมูลของแต่ละ segment นั้นจะไม่เท่ากัน ค่าของ segment duration นี้ถูกกำหนดในขั้นตอนการเข้ารหัสไฟล์วิดีโอตามมาตรฐาน MPEG-DASH และ segment duration จะถูกกำหนดใช้กับทุก ๆ segment ย่อย โดยการระบุช่วงของไฟล์ตาม segment duration นั้น กระบวนการอ่านไฟล์ของเซิร์ฟเวอร์จะอ่านช่วงของไฟล์ตามค่านี้ และจะถูกระบุไว้ในไฟล์ manifest ของทุกไฟล์วิดีโอในระบบ กระบวนการแบ่งการร้องขอไฟล์ของไคลเอนต์จะอ้างอิงค่า segment duration ตามค่าที่ระบุไว้ในไฟล์ manifest ที่ไคลเอนต์ได้ดาวน์โหลดมาเป็นอันดับแรกก่อนเริ่มการดาวน์โหลดและเปิดเล่นไฟล์วิดีโอ

ค่า segment duration ของทุก segment ของไฟล์วิดีโอหนึ่งๆ มีค่าเท่ากันโดยมีหน่วยเป็นวินาที ใช้ในการควบคุมจำนวนเวลาของวิดีโอในแต่ละ segment และใช้ในการกำหนดการดาวน์โหลดแต่ละ segment ของไคลเอนต์ กล่าวคือ หาก segment duration ของไฟล์วิดีโอหนึ่ง มีค่าเท่ากับ 2 วินาที หมายความว่าทุก ๆ การร้องขอส่วนของไฟล์วิดีโอนั้นในแต่ละครั้งจากไคลเอนต์จะร้องขอข้อมูลไฟล์วิดีโอเป็น segment ย่อย ๆ ที่มีความยาว 2 วินาที แต่หากพิจารณาในหน่วยขนาดข้อมูลทุก ๆ segment จะมีขนาดไม่เท่ากันขึ้นอยู่กับคุณสมบัติของไฟล์วิดีโอ ณ วินาทีนั้น ๆ ที่บรรจุในแต่ละ segment ย่อยของไฟล์วิดีโอนั้น คุณสมบัติที่แตกต่างกันของแต่ละ segment ทำให้ไม่สามารถควบคุมขนาดข้อมูลในหน่วย byte ของแต่ละ segment ที่ให้บริการให้เท่ากันได้

เพียงแต่สามารถควบคุม segment duration ให้ตรงกันได้เท่านั้น โดยสามารถกระทำได้ในขั้นตอนการสร้างไฟล์วิดีโอและการสร้างไฟล์ manifest โดยไคลเอนต์จะอ่านค่านี้ได้จากไฟล์ manifest ที่ดาวน์โหลดมาและคำนวณแต่ละ segment ย่อย ๆ ก่อนจะเริ่มดาวน์โหลดไฟล์วิดีโอ

จากคุณลักษณะดังกล่าวของมาตรฐาน MPEG-DASH และด้วยขนาดที่มักจะมีขนาดใหญ่ของข้อมูลวิดีโอ การแบ่งการดาวน์โหลดออกเป็น segment ย่อย ๆ นั้นทำให้สามารถแบ่งการทำงานของเซิร์ฟเวอร์ออกเป็นส่วนเล็ก ๆ ด้วยเช่นกัน ซึ่งทำให้เกิดผลดีคือเซิร์ฟเวอร์ใช้เวลาเพียงระยะสั้น ๆ ในการตอบสนองการร้องขอจากหลาย ๆ ไคลเอนต์ที่เข้ามาพร้อม ๆ กัน ทำให้เซิร์ฟเวอร์สามารถทำงานได้อย่างมีประสิทธิภาพ และยังทำส่งผลให้ไคลเอนต์สามารถดาวน์โหลดไฟล์ที่มีคุณภาพ (bitrate) สูงได้ในระหว่างการรับชมโดยที่การรับชมยังคงราบรื่น จากเหตุผลดังกล่าวทำให้การออกแบบแคชจึงต้องคำนึงถึงการรองรับการทำงานของแคชแบบบางส่วน (Partial Cache) ได้

คุณสมบัติที่สำคัญของการทำงาน ของ DASH client คือการให้ความสำคัญกับคุณภาพของวิดีโอที่เหมาะสมและไคลเอนต์สามารถรับชมโดยไม่ติดขัด ซึ่ง DASH ถูกออกแบบมาให้ไคลเอนต์สามารถเปลี่ยนแปลงคุณภาพของไฟล์วิดีโอหรือเปลี่ยนแปลงจากไฟล์วิดีโอที่เปิดเล่นอยู่ไปดาวน์โหลดไฟล์วิดีโออื่นที่มี bitrate แตกต่างกันได้ตลอดเวลา เพื่อให้เหมาะสมกับสภาวะความพร้อมของไคลเอนต์เอง โดยอาศัยพารามิเตอร์ buffer occupy [46] และ real time bandwidth กล่าวคือ buffer occupy คือความจุของ buffer ของไคลเอนต์ในหน่วยของเวลา ซึ่งไคลเอนต์จะใช้พิจารณาร่วมกับค่าของ bandwidth ที่ไคลเอนต์สามารถดาวน์โหลดได้จริงในขณะนั้น เพื่อใช้ในการพิจารณาเพิ่มหรือลดคุณภาพของไฟล์วิดีโอที่กำลังเปิดดาวน์โหลดอยู่ หาก buffer ของไคลเอนต์มีพื้นที่ว่างเพียงพอ และระดับ bandwidth ที่ได้จากการดาวน์โหลดมีค่าสูง ไคลเอนต์จะเปลี่ยนแปลงไฟล์ที่กำลังดาวน์โหลดอยู่ในการร้องขอ segment ถัดไปเป็นไฟล์วิดีโอเดียวกันที่มี bitrate สูงขึ้นเพื่อให้ผู้ใช้ได้ชมวิดีโอที่มีคุณภาพที่สูงขึ้น ในทางกลับกัน หาก buffer เหลือน้อยและ bandwidth ที่ได้มีค่าน้อย ไคลเอนต์จะปรับลดค่า bitrate ของวิดีโอที่จะดาวน์โหลดในการร้องขอครั้งถัดไป ซึ่งการเปลี่ยนแปลง bitrate ของการร้องขอ segment ไฟล์วิดีโอของไคลเอนต์นี้คือการเปลี่ยนไฟล์ของวิดีโอที่ต้องการไปเป็นไฟล์อื่นที่ bitrate ต่างกันของวิดีโอที่มีเนื้อหาเดียวกัน โดยการเปลี่ยนแปลงนี้จะไม่ทำให้การรับชมของผู้ใช้งานสะดุด เพราะไคลเอนต์สามารถเลือกดาวน์โหลดและระบุได้ว่าไคลเอนต์ต้องการช่วงไฟล์ใด

DASH ถูกออกแบบให้รองรับ Range Request ดังที่ได้อธิบายไว้ในบทที่ 2 และใช้ Adaptation Bitrate Algorithm (ABR) [21] ในการพิจารณาการเปลี่ยนแปลง bitrate ของวิดีโอที่กำลัง

เปิดเล่นอยู่ ดังแสดงใน Algorithm 3.1 [21] ซึ่งเป็นมาตรฐานของ Open-Source Media Framework (OSMF)

Algorithm 3.1 Quality adaptation algorithm หรือ ABR ของ OSMF

1. $t_{lastfrag}$: Time of downloading the last fragment
2. l_{cur}: Current quality level
3. l_{nxt}: Proposed quality level
4. l_{min}: Lowest quality level
5. l_{max}: Highest quality level
6. $b(l)$: Bit rate of quality level l
7. $r_{download} \leftarrow \_\theta/t_{lastfrag}$
8. if $r_{download} < 1$ then
9. if $l_{cur} > l_{min}$ then
10. if $r_{download} < (b(l_{cur} - 1)/b(l_{cur}))$ then
11. $l_{nxt} \leftarrow l_{min}$
12. else
13. $l_{nxt} \leftarrow l_{cur} - 1$
14. end if
15. end if
16. else
17. if $l_{cur} < l_{max}$ then
18. if $r_{download} \geq (b(l_{cur} - 1)/b(l_{cur}))$ then
19. repeat
20. $l_{nxt} \leftarrow l_{nxt} + 1$
21. until ($l_{nxt} = l_{max}$) or ($r_{download} < (b(l_{nxt} + 1)/b(l_{cur}))$)
22. end if
23. end if
24. end if

3.2 พฤติกรรมของผู้ใช้งาน (User Behavior)

พฤติกรรมของผู้ใช้งานนั้นเป็นปัจจัยสำคัญที่ต้องพิจารณาในการออกแบบระบบ เนื่องจากการพิจารณาว่าระบบทำงานได้ดีหรือไม่เพียงใดนั้นขึ้นอยู่กับว่าระบบสามารถรองรับและทำงานกับพฤติกรรมการใช้งานของผู้ใช้งานได้ดีหรือไม่ ซึ่งในขั้นตอนการออกแบบจำเป็นอย่างยิ่งที่จะต้องพิจารณาและวิเคราะห์พฤติกรรมของผู้ใช้งาน เพื่อให้ระบบที่ออกแบบมาสามารถทำงานได้อย่างมีประสิทธิภาพ ซึ่งพฤติกรรมการใช้งานระบบให้บริการไฟล์วิดีโอนั้นมีความแตกต่างจากระบบอื่น ๆ กล่าวคือ เมื่อพิจารณาถึงจำนวนผู้ใช้งานและจำนวนไฟล์วิดีโอในระบบ สิ่งที่ต้องคำนึงถึงเป็นอันดับแรกคือ จำนวนผู้ใช้งานที่ต้องการใช้ไฟล์นั้น ๆ หรือจำนวนผู้ใช้งานต่อไฟล์ ซึ่ง

จะแตกต่างกันขึ้นอยู่กับการเป็นที่นิยมของไฟล์นั้น ๆ ยกตัวอย่างเช่น จำนวนผู้ใช้งานที่เข้ารับชมภาพยนตร์เรื่องหนึ่งในระบบ Netflix หรือ YouTube จะสามารถบ่งบอกถึงความนิยมของผู้ใช้งานที่มีต่อภาพยนตร์เรื่องนั้น ผู้ใช้จำนวนมากจะเปิดรับชมวิดีโอเพียงไม่กี่ไฟล์ที่มีความนิยมสูง หรือในระบบเครือข่ายสังคมออนไลน์ วิดีโอที่กำลังได้รับความนิยมสูงจะถูกรับชมจากผู้ใช้งานจำนวนมาก การกระจายตัวของความถี่ของการถูกใช้งานของแต่ละไฟล์เป็นแบบ Zipf-like [34] กล่าวคือกลุ่มไฟล์วิดีโอที่ได้รับความนิยมสูงเป็นกลุ่มที่มีจำนวนน้อยแต่ผู้ใช้จำนวนมากให้ความสนใจและไฟล์เหล่านั้นจะถูกเรียกใช้บ่อยครั้ง ส่วนไฟล์วิดีโอที่ได้รับความนิยมน้อยซึ่งเป็นส่วนที่เหลือ มักจะมีจำนวนมากและถูกเรียกใช้ด้วยความถี่ต่ำ อันเนื่องมาจากจำนวนผู้ใช้ที่ให้ความสนใจมีน้อย ดังนั้นค่าความนิยมเป็นปัจจัยที่สำคัญอย่างยิ่งต่อการออกแบบแคชให้กับระบบการให้บริการไฟล์วิดีโอ ซึ่งค่าความนิยมหรือ Popularity ของไฟล์หนึ่ง ๆ ในระบบสามารถหาได้จากการคำนวณหาค่าดัชนีความนิยม (Popularity Index) ของไฟล์นั้น ๆ เทียบกับไฟล์อื่น ๆ ทั้งหมดที่มีในระบบ ซึ่งต้องเก็บจำนวนความถี่ของการถูกเรียกใช้ของไฟล์นั้น ๆ วิทยานิพนธ์นี้นำเสนอแนวทางการคำนาณดัชนีค่าความนิยมดังในสมการที่ 3

กำหนดให้

N คือจำนวนไฟล์วิดีโอทั้งหมดในระบบ

j คือลำดับที่ของไฟล์ในจำนวน N ไฟล์

Pop_j คือ ค่าดัชนีความนิยมของไฟล์นั้น ๆ

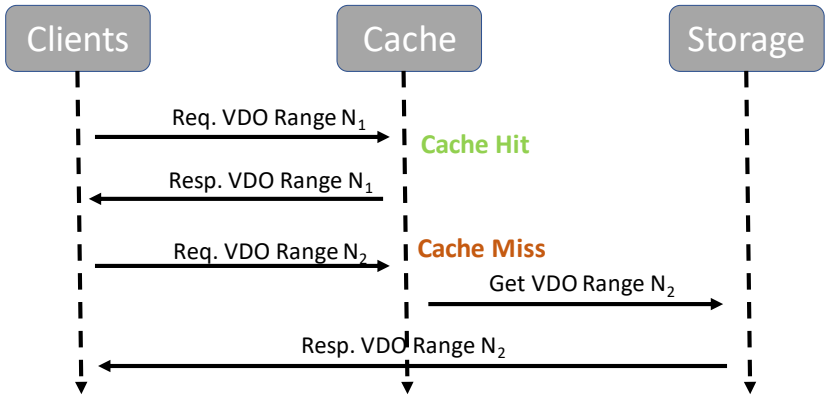
$AccessCount_j$ คือจำนวนครั้งที่ไฟล์ j ถูกใช้งาน

$$Pop_j = \frac{AccessCount_j}{\sum_{j=1}^{N} AccessCount_j} \quad (3)$$

จากสมการที่ 3 เป็นการหาดัชนีค่าความนิยมของไฟล์วิดีโอ ในงานวิจัยนี้ได้เลือกใช้วิธีการให้บริการไฟล์วิดีโอแบบ MPEG-DASH ที่พิจารณาในระดับช่วงของไฟล์หรือ segment ดังนั้นการคำนวณหาดัชนีค่าความนิยมจำเป็นต้องพิจารณาในระดับ segment ย่อย ๆ หรือแต่ละช่วงของข้อมูลไฟล์วิดีโอที่ถูกร้องขอในแต่ละครั้ง ดังรายละเอียดที่กล่าวไปแล้วในหัวข้อที่ 3.1

3.3 การออกแบบแคช

แคชเป็นแหล่งเก็บข้อมูลที่สามารถทำงานได้อย่างรวดเร็วทั้งการอ่านและการเขียน แต่แคชมีราคาแพงเมื่อเทียบกับแหล่งจัดเก็บข้อมูลชนิดอื่น ๆ ในงานวิจัยนี้ได้เลือกแหล่งจัดเก็บข้อมูลประเภทหน่วยความจำหลัก (main memory) หรือ RAM (Random Access Memory) มาทำหน้าที่เป็นแคชของข้อมูลสตรีมมิ่ง ขนาดข้อมูลประเภทสตรีมมิ่งนั้นมักมีขนาดใหญ่เมื่อเทียบกับข้อมูลทั่วไปที่จัดเก็บในแคชซึ่งมักเป็นข้อมูลประเภทข้อความ รูปภาพ หรือไฟล์โค้ดโปรแกรมต่างๆ



รูปที่ 3.1 การทำงานของแคชในระบบ video streaming แบบ byte range

หลักการทำงานของแคชในระบบ streaming เป็นดังรูปที่ 3.1 กล่าวคือเมื่อไคลเอนต์ร้องขอส่วนของไฟล์ข้อมูล ระบบจะมองหาข้อมูลในแคชก่อนเป็นอันดับแรก หากพบว่าข้อมูลที่ต้องการอยู่ภายในแคช จะส่งข้อมูลนั้นไปยังไคลเอนต์ทันที เรียกกรณีนี้ว่า Cache Hit ในทางกลับกันหากไม่พบข้อมูลที่ต้องการภายในแคช จำเป็นต้องอ่านข้อมูลจากแหล่งจัดเก็บข้อมูลภายนอกที่อ่านได้ช้ากว่าทำให้ใช้เวลานานกว่าและมีขั้นตอนมากกว่า คือ backend storage ซึ่งเรียกกรณีนี้ว่า Cache Miss ข้อมูลในระบบวิดีโอสตรีมมิ่ง เป็นข้อมูลที่ต่อเนื่องและมีขนาดใหญ่ มีการแบ่งส่วนการดาวน์โหลดออกเป็นส่วนย่อย ๆโดยไคลเอนต์จะทำการร้องขอทีละส่วนต่อเนื่องกัน

จากการวิเคราะห์การทำงานของ HTTP streaming แบบ DASH ทำให้ทราบว่าขนาดของข้อมูลของไฟล์เดียวกันในแต่ละ segment มักมีขนาดที่ไม่เท่ากัน แต่สิ่งที่เท่ากันคือ segment duration ทำให้การออกแบบแคชไม่สามารถอ้างอิงด้วยขนาดของข้อมูลได้ แต่สามารถอ้างอิงขนาดของแคชด้วยจำนวน segment ที่เก็บภายในแคช ซึ่งขนาดของแคชจะถูกคำนวณได้จากขนาดของข้อมูลทั้งหมด ในงานวิจัยนี้จะใช้จำนวน segment ที่สามารถเก็บได้ในแคชเป็นขนาดความจุของแคช

การออกแบบแคชแบบดั้งเดิมและนโยบายการแทนที่แคชเพื่อเก็บข้อมูลที่ถูกเข้าถึงบ่อยครั้งที่สุด (LFU) หรือถูกเข้าถึงล่าสุด (LRU) นั้นไม่เหมาะสมกับไฟล์วิดีโอโดยเฉพาะในการพิจารณาว่า segment ใดควรอยู่ในแคชและ segment ใดควรถูกนำออกเมื่อแคชไม่มีพื้นที่ว่างเพียงพอ นโยบายการแทนที่แคชควรจะช่วยกลั่นกรองและเพิ่มระยะเวลาการอยู่ในแคชของข้อมูลที่มีความต้องการจากผู้ใช้งานสูง ในงานวิทยานิพนธ์นี้นำเสนอนโยบายการแทนที่แคชโดยอาศัยค่าดัชนีความนิยมในระดับ segment (Segment-based cache replacement policy) ดังนั้นจึงต้องจัดเก็บข้อมูลการใช้งานระบบเพื่อประกอบการคำนวณดัชนีความนิยมของแต่ละ segment ย่อย ๆ ทั้งหมด โดยนำเสนอสมการคำนวณค่าดัชนีความนิยมของแต่ละ segment เป็นดังสมการที่ (4)

กำหนดให้

N คือจำนวนไฟล์ทั้งหมดในแคช

j คือลำดับของไฟล์

M_j คือจำนวน segment ของไฟล์ที่ j

i คือลำดับของ segment ของแต่ละไฟล์

Pop_{ij} คือ ค่าดัชนีความนิยมของ segment ย่อยที่ i ของไฟล์วิดีโอ j ที่อยู่ภายในแคช

$AccessCount_{ij}$ คือ จำนวนครั้งที่ segment ที่ i ของไฟล์วิดีโอ j ถูกร้องขอโดยไคลเอนต์

$$Pop_{ij} = \frac{AccessCount_{ij}}{\sum_{j=1}^{N}\sum_{i=1}^{M_j} AccessCount_{ij}} \quad (4)$$

จากสมการที่ (4) จะเห็นว่าค่าดัชนีความนิยมหรือความนิยมของ segment ที่อยู่ในแคชจะมีการเปลี่ยนแปลงทุกครั้งที่ segment นั้น ๆ ถูกร้องขอโดยไคลเอนต์ ดังนั้นหากยึดเพียงค่าดัชนีความนิยมเพียงอย่างเดียวในบางรูปแบบพฤติกรรมการใช้งานจะทำให้การทำงานของแคชมีรูปแบบที่ใกล้เคียงกับ LFU แคช ลักษณะการใช้งานของผู้ใช้รวมไปถึงแนวโน้มของความนิยมของไฟล์วิดีโอเมื่อเวลาเปลี่ยนแปลงไปมักจะมีค่าลดลงเมื่อเทียบกับความนิยมสูงสุดของไฟล์วิดีโอนั้น เช่น ภาพยนตร์หนึ่งที่ได้รับความนิยมอย่างมากเมื่อปีที่ผ่านมา ณ เวลาปัจจุบันอาจจะพบว่าได้รับความนิยมน้อยลง แต่อัตราการลดลงมักจะสัมพันธ์กับความนิยมเดิมและเวลาที่เปลี่ยนแปลงไป หรือ

กล่าวได้อีกนัยหนึ่งว่าสิ่งที่ได้รับความนิยมในอดีตเมื่อเวลาผ่านไปมักจะได้รับความนิยมลดลง ดังตัวอย่างในรูปที่ 3.2 จะเห็นได้ว่าลักษณะการเปลี่ยนแปลงของความนิยมของสิ่งใด ๆ นั้น จะมีค่าเพิ่มมากขึ้นเรื่อย ๆ ในตอนต้น จนกระทั่งเวลาผ่านไประยะหนึ่งความนิยมจะเพิ่มขึ้นจนถึงจุดสูงสุด หลังจากนั้นจะลดลงเรื่อย ๆ ตามเวลาที่เปลี่ยนแปลงไป การลดลงของความนิยมที่ขึ้นอยู่กับเวลานี้ ในวิทยานิพนธ์นี้นิยามเป็น การลดทอนลงของความนิยม หรือ Popularity attenuation โดยขึ้นอยู่กับระยะเวลาที่เปลี่ยนแปลงไป ซึ่งแตกต่างนโยบายแคช LFU ที่พิจารณาเพียงความถี่ของการถูกใช้งานเพียงอย่างเดียวโดยไม่คำนึงถึงการลดลงของความนิยมเมื่อเวลาเปลี่ยนแปลงไป ดังนั้นจึงนำเสนอสมการการคำนวณหาค่าดัชนีความนิยมของ segment ของวิดีโอที่อยู่ในแคชเป็นดังสมการที่ (5)

กำหนดให้

N คือจำนวน segment ทั้งหมดที่อยู่ในแคช

i คือลำดับของ segment ทั้งหมดที่อยู่ในแคช

Pop_i คือ ค่าดัชนีความนิยมของ segment ย่อยที่ i ที่อยู่ภายในแคช

$AccessCount_i$ คือ จำนวนครั้งที่ segment ที่ i ที่อยู่ในแคช

โดยที่

$\Delta Time_i$ คือระยะเวลาที่ segment นั้น ๆ อยู่ในแคชโดยไม่ถูกร้องขอจากไคลเอนต์

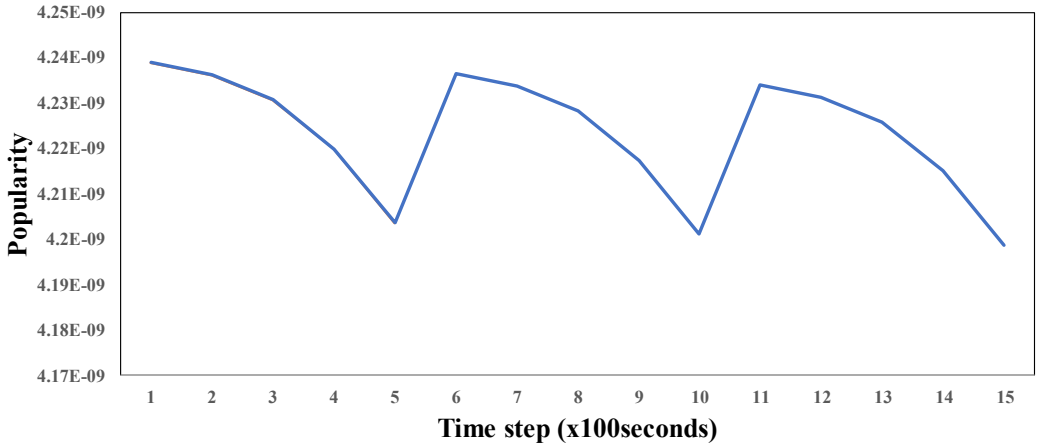
และ $\Delta Time_i = time_{recent} - time_{last\_access}$

$$Pop_i = \frac{AccessCount_i}{\sum_{i=1}^{N} AccessCount_i} + \frac{1}{\Delta Time_i} \quad (5)$$

ซึ่ง

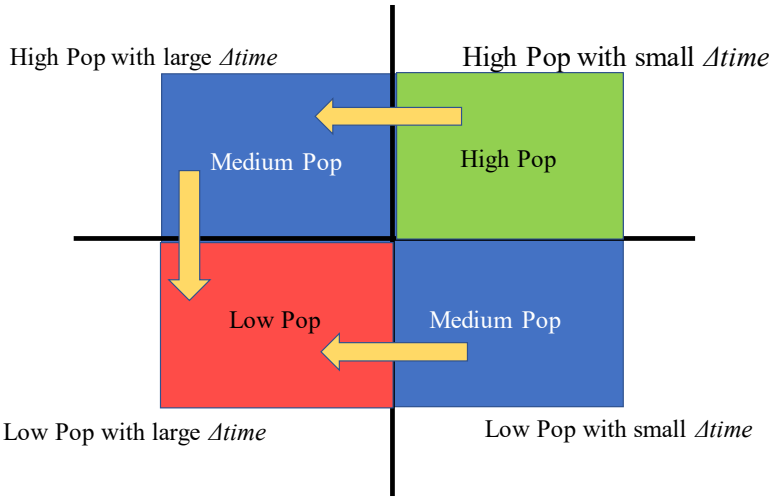
$\frac{1}{\Delta Time_i}$ คืออัตราการลดทอนของดัชนีความนิยมของ segment นั้น ๆ

ในกรณีที่ segment ใดที่อยู่ในแคชถูกร้องขอจากไคลเอนต์ระบบจะเพิ่มค่า access count ประจำ segment นั้นและจะกำหนดค่า $time_{last\_access}$ เป็นค่าใหม่ และกระบวนการลดทอนค่าความนิยมจะเริ่มต้นใหม่อีกครั้งด้วยค่าดัชนีความนิยมค่าใหม่



รูปที่ 3.2 ตัวอย่างการลดทอนค่าความนิยม

ดังตัวอย่างในรูปที่ 3.2 ซึ่งแสดงการลดทอนของดัชนีความนิยมของ segment หนึ่งเมื่อเวลาเปลี่ยนแปลงไป โดยแนวแกน y คือระดับค่าดัชนีความนิยมของ segment และแกน x คือเวลาที่เปลี่ยนแปลงไปมีหน่วยเป็น 100 วินาที ซึ่งในระหว่างที่มีการลดทอนอยู่นั้น หาก segment ถูกเรียกใช้อีกครั้ง กระบวนการลดทอนค่าความนิยมจะถูกกำหนดให้เริ่มต้นใหม่อีกครั้ง การเปลี่ยนแปลงของค่าดัชนีความนิยมด้วยอัตราการลดทอนจะเป็นดังรูปที่ 3.3 กล่าวคือหากค่าดัชนีความนิยมของ segment อยู่ในช่วงใดก็ตาม เมื่อเทียบกับดัชนีความนิยมของ segment อื่น ๆ ที่อยู่ในแคช จะมีค่าลดลงเรื่อย ๆ ตามอัตราการลดทอน และเมื่อเวลาเปลี่ยนแปลงไปอัตราการลดทอนจะยิ่งมากขึ้นทำให้ segment ที่มีดัชนีความนิยมสูง (high pop) จะลดลงไปอยู่ในกลุ่มที่มีดัชนีความนิยมปานกลาง (medium pop) และจะลดลงไปอยู่ในกลุ่มที่มีดัชนีความนิยมต่ำ (low pop) ในที่สุด segment ที่มีค่าดัชนีความนิยมต่ำที่สุดจะถือเป็น segment ที่มีความสำคัญน้อยที่สุด หากในระหว่างนั้นแคชเต็มและมี segment ใหม่ที่ไม่ได้อยู่ในแคชถูกร้องขอจากไคลเอนต์และจำเป็นต้องจัดเก็บไว้ในแคชระยะหนึ่ง (insert) ดังนั้น segment ที่มีดัชนีความนิยมต่ำที่สุดที่อยู่ในแคชจะถูกนำออก (evict) จากแคชเพื่อให้มีพื้นที่ว่างในการเก็บ segment ใหม่เข้าไปในแคช ส่วนการเริ่มต้นลดทอนใหม่อีกครั้งเกิดขึ้นเมื่อ segment ที่อยู่ในแคชถูกร้องขออีกครั้งในขณะที่ดัชนีความนิยมกำลังลดทอนลงตามเวลา โดยจะเริ่มกระบวนการลดทอนใหม่ ณ เวลาที่ถูกร้องขอครั้งล่าสุดตามค่าตัวแปร $time_{last\_access}$



รูปที่ 3.3 การเปลี่ยนแปลงค่าความนิยมด้วยการลดทอนความนิยมของ segment ไฟล์วิดีโอตามสมการที่ (5)

จากรูปที่ 3.3 จะเห็นว่า segment ที่มีค่าดัชนีความนิยมที่ถูกลดทอนแล้วเหลือค่าน้อยที่สุดจะเป็น segment ที่ถูกแทนที่ด้วย segment ใหม่เมื่อเกิดกรณีแคชเต็ม ดังสมการที่ (6)

กำหนดให้

i คือลำดับของ segment ของแต่ละไฟล์

Pop_i คือ ค่าดัชนีความนิยมของ segment ย่อยที่ i ที่อยู่ภายในแคช

$$Pop_{least} = \min(Pop_i) \tag{6}$$

ขนาดของแคช (Cache Capacity) เป็นตัวแปรสำคัญในการออกแบบแคชซึ่งสามารถคำนวณจากขนาดของข้อมูลทั้งหมดของระบบ ซึ่งคิดเป็นเปอร์เซ็นต์ของขนาดข้อมูลทั้งหมด โดยสามารถคำนวณได้จากสมการที่ (7) ดังนี้

กำหนดให้

cache_capacity คือขนาดความจุของแคชหรือจำนวน segment ที่แคชสามารถเก็บได้

DataSize คือขนาดของไฟล์วิดีโอทั้งหมดหรือจำนวน segment ของวิดีโอทั้งหมดในระบบแบ่งตาม segment duration

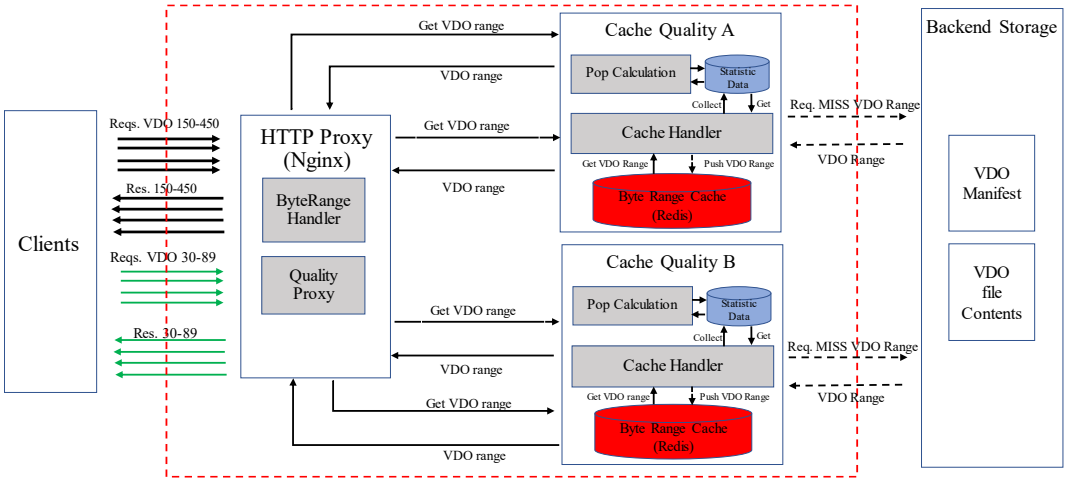
a คืออัตราส่วนของการคำนวณหาขนาดแคชที่เหมาะสมมีค่าอยู่ในช่วง 0 – 1

$$cache\_capacity = a * DataSize \quad (7)$$

โดยอัตราส่วน a จะกำหนดจากขนาดของข้อมูลรวมกันทั้งหมดที่มีในระบบ

3.4 โครงสร้างและส่วนประกอบของระบบ (System Architecture)

จากการวิเคราะห์พฤติกรรมของระบบและพฤติกรรมการใช้งานของผู้ใช้และความสำคัญของดัชนีความนิยมที่มีผลอย่างยิ่งต่อระบบให้บริการไฟล์วิดีโอผ่านเครือข่ายอินเตอร์เน็ตแบบ DASH streaming และสมการการคำนวณหาค่าดัชนีความนิยมของ segment ของไฟล์วิดีโอ นำไปสู่การออกแบบและพัฒนานโยบายการแทนที่แคช และระบบการให้บริการไฟล์วิดีโอ โครงสร้างของระบบที่ออกแบบเป็นดังรูปที่ 3.4 ซึ่งประกอบด้วย



รูปที่ 3.4 ส่วนประกอบและของสร้างของระบบ

3.4.1 Client

ไคลเอนต์ คือ เครื่องลูกข่ายที่ผู้ใช้งานใช้ในการเชื่อมต่ออินเตอร์เน็ตและเปิดรับชมไฟล์วิดีโอซึ่งต้องรองรับ DASH client ในที่นี้ใช้โปรแกรม VLC media player [47] และยังใช้

Google Chrome web browser โดยการติดตั้งส่วนขยาย (extension) [48] เพิ่มเติมเพื่อให้สามารถทำงานเป็น DASH Client ได้

3.4.2 HTTP Proxy

คือเซิร์ฟเวอร์ที่ทำหน้าที่เป็น HTTP web server ที่มีหน้าที่เชื่อมกับ DASH client, รองรับการร้องขอจากไคลเอนต์เป็นตัวเชื่อมต่อกับแคชและส่งข้อมูลกลับไปยังไคลเอนต์ซึ่งในงานวิจัยนี้ ผู้วิจัยได้เลือกใช้ Nginx [39] เป็น HTTP proxy เนื่องจากสามารถเขียนโปรแกรมฝังไว้ในส่วนประกอบของ Nginx (process) โดยแทรกการทำงานเพื่อให้สามารถทำงานนอกเหนือจากงานหลักของ Nginx ได้ การเขียนโปรแกรมเพื่อฝังเข้าไปใน Nginx process นั้น ผู้วิจัยใช้ความสามารถของ OpenResty[40] ซึ่งเป็นเฟรมเวิร์ค (framework) โดยแทรกส่วนของโปรแกรมไว้ในส่วนต่าง ๆ ของ Nginx เรียกว่าช่วง (phase) [40] และเหตุการณ์ (event) [40] ที่เหมาะสมภายในขั้นตอนการทำงานปกติของ Nginx

จากการที่ Nginx เป็นโปรแกรมที่ทำงานในลักษณะ Master-workers ที่ประกอบด้วยหนึ่ง master และหลาย worker ทำให้การจัดการเรื่องของภาวะพร้อมกัน (concurrency) และความสอดคล้องกัน (consistency) ของการทำงานและข้อมูลของระบบรวมไปถึง segment ของวิดีโอภายในแคชเป็นเรื่องสำคัญ ซึ่งจะได้กล่าวในหัวข้อต่อไป ในส่วนภายในของ HTTP proxy นั้นมีส่วนประกอบย่อยสองส่วนด้วยกันคือ

3.4.3 ByteRangeHandler

ทำหน้าที่คำนวณช่วงไบต์ของข้อมูลไฟล์วิดีโอจาก request ของไคลเอนต์และส่งต่อพารามิเตอร์ที่ได้ไปยังแคชในการค้นหาข้อมูลภายในแคช

3.4.4 QualityProxy

ทำหน้าที่พิจารณาเลือกแคชหน่วยที่ทำหน้าที่เก็บข้อมูลไฟล์วิดีโอที่ตรงกับ bitrate ที่ไคลเอนต์ร้องขอมายังเซิร์ฟเวอร์เพื่อรองรับการเปลี่ยนแปลงคุณภาพไฟล์วิดีโอ (bitrate adaptation) ของไคลเอนต์ ซึ่งในวิทยานิพนธ์นี้เน้นการนำเสนอการออกแบบแคชและนโยบายการแทนที่แคชเป็นหลัก จึงกำหนดให้มีการแบ่งช่วง bitrate ของไฟล์วิดีโอออกเป็นสองชุดด้วยกัน คือ แคชสำหรับไฟล์วิดีโอคุณภาพ A ซึ่งมีคุณภาพสูงและแคชสำหรับไฟล์วิดีโอคุณภาพ B ซึ่งมีคุณภาพต่ำกว่า A โดยที่กระบวนการทำงานของแคชทั้งสองหน่วยเหมือนกัน ต่างกันเพียงคุณภาพของ segment ของไฟล์วิดีโอที่ถูกเก็บภายในแคช

3.4.5 แคช (Cache)

มีอยู่ด้วยกัน 2 ชุดตามคุณภาพของไฟล์วิดีโอที่ต้องการทดสอบ ซึ่งมีองค์ประกอบและการทำงานเหมือนกัน แต่ต่างกันที่คุณภาพของวิดีโอที่จัดเก็บ ภายในแคชประกอบด้วย

1.) Byte range cache
ทำหน้าที่เก็บ segment ของไฟล์วิดีโอ ซึ่งในงานวิจัยนี้ผู้วิจัยใช้ Redis cache โดยระงับการทำงานนโยบายการแทนที่แคชของ Redis และปรับใช้นโยบายการแทนที่แคชที่ผู้วิจัยพัฒนาขึ้นเอง

2.) Cache Handler
ทำหน้าที่ควบคุมการทำงานของแคชให้ทำงานอย่างถูกต้องโดยใช้นโยบายการแทนที่แคชที่ปรับใช้ค่าดัชนีความนิยมของข้อมูลที่อยู่ในแคช และการลดทอนของดัชนีความนิยม

3.) Statistic data
เป็นข้อมูลการใช้งานแคช ข้อมูลจำเพาะของแต่ละ segment ที่อยู่ภายในแคช

4.) Pop calculation
ทำหน้าที่คำนวณค่าดัชนีความนิยมที่ใช้ในนโยบายการแทนที่แคช โดยใช้ข้อมูลจากส่วนเก็บข้อมูล Statistic data

3.4.6 Backend storage

เป็นแหล่งจัดเก็บข้อมูลซึ่งแยกออกจาก HTTP Proxy อย่างชัดเจน มีหน้าที่จัดเก็บข้อมูลไฟล์วิดีโอทั้งหมดในระบบและไฟล์ manifest ให้กับระบบ และส่งช่วงข้อมูลไฟล์วิดีโอไปให้ HTTP proxy เมื่อมีการร้องขอไฟล์

3.5 อัลกอริทึม Dynamic Popularity Caching

จากการคำนวณหาค่าดัชนีความนิยมในระดับ segment และการลดทอนของดัชนีความนิยมที่เปลี่ยนแปลงตามระยะเวลา ในสมการที่ (5) Cache replacement policy ของไฟล์วิดีโอ และจากอัลกอริทึมที่ (3.2) ซึ่งแสดงระเบียบวิธีการทำงานของนโยบายการแทนที่แคชที่ผู้วิจัยได้ออกแบบ เพื่อให้การทำงานของโครงสร้างระบบที่นำเสนอทำงานได้อย่างมีประสิทธิภาพ จึงต้องเพิ่มเติมส่วนการทำงานให้สอดคล้องกับระบบที่ภาวะพร้อมกัน (concurrency) และความสอดคล้องกัน (consistency) เข้าไปในกระบวนการทำงานของแคชซึ่งแสดงในอัลกอริทึมที่ 3.2 โดยกำหนด

ขนาดความจุของแคชตามการคำนวณจากขนาดของข้อมูลทั้งหมดที่มี นั่นคือจำนวน segment ทั้งหมดของไฟล์วิดีโอที่มีในระบบ ซึ่งจะได้ขนาดของแคชตามสมการที่ (7) เป็นจำนวน segment ที่แคชสามารถจัดเก็บได้

Algorithm 3.2

```
1. RequestFromClient()
2. Pop = Popularity
3. Cp = Cache capacity
4. N = Number of segments in the cache
5. evictList = Candidate evicted segment
6. evictListSize  = Size of the candidate evicted segment = 10% of the cache
   capacity
7. IF segmentIsInCache THEN
8.    // cache HIT
9.     Get the segment from cache()
10.    IF IsEvictList THEN
11.        Remove from Eviction Queue
12.   Update meta data
13. ELSE
14.   // cache MISS
15.   IF N reaches 95% of Cp OR evictListSize < 5%  THEN
16.       Popularity calculation Eq.(1);
17.       Add evictList to Eviction Queue
18.   IF N = Cp THEN
19.       Evict all segments in the evictList Queue
20.       Update meta data
21.   ELSEIF N reach 99% of Cp THEN
22.       Evict the Least POP segment in Eviction Queue
23.       IF evict not success THEN
24.           Bypass cache
25.       Update meta data
26. ELSE
27.   Get video segment from the backend storage
28.   Store the new segment range in the cache
29. END
30. RETURN
```

จาก Algorithm ที่ 3.2 จะเห็นว่ามีการกำหนดชุดของ segment ที่จะถูกแทนที่ด้วย segmentใหม่ในกรณีที่แคชเต็ม เรียกชุด segment นี้ว่า EvictList ซึ่งเป็นกลุ่มของ segment ที่มีค่าดัชนีความนิยมที่ถูกลดทอนแล้วมีค่าน้อย จากการคำนวณด้วยสมการที่ (5) และ (6) ซึ่ง segment กลุ่มดังกล่าวนี้เรียกอีกชื่อหนึ่งว่า Candidate Evict List โดยมีจำนวนอ้างอิงจากค่าขนาดของแคช (cache capacity) จากสมการที่ (7) ซึ่งสามารถคำนวณขนาดของ EvictList ได้ดังสมการที่ (8)

$$evictList\_size = 0.1 * cache\_capacity \quad (8)$$

ซึ่ง evictList_size จะมีขนาดสูงสุดเท่ากับ 10% ของขนาดความจุแคช เพื่อให้ระบบพร้อมรองรับการทำงานในกรณีที่มีการร้องขอจากไคลเอนต์จำนวนมากและเกิดกรณีที่ต้องจัดเก็บ segment ใหม่ลงในแคชจำนวนมากพร้อมๆ กัน

3.6 การทำงานของนโยบายการแทนที่แคช Dynamic Popularity Caching

เนื่องจาก MPEG-DASH ต้องใช้ HTTP server ในการให้บริการไฟล์วิดีโอให้กับไคลเอนต์ ผู้วิจัยได้ใช้ Nginx ในการทำหน้าที่เป็น HTTP sever ซึ่ง Nginx ทำงานแบบ mater-workers ที่สามารถรองรับการเชื่อมต่อและร้องขอไฟล์จากไคลเอนต์จำนวนมากพร้อม ๆ กัน โดยการแยกการทำงานบางส่วนออกไปเป็นเทรดย่อยเพื่อทำงานให้บริการกับไคลเอนต์และยังต้องเข้าถึงข้อมูลทั้ง segment ของวิดีโอที่อยู่ในแคชและทรัพยากรของระบบ เช่น ข้อมูลจำเพาะของ segment หรือแม้กระทั่งการเรียกใช้งานการคำนวณดัชนีความนิยม ซึ่งอาจทำให้เกิดปัญหาภาวะพร้อมกัน (concurrency) หรือปัญหาความสอดคล้องกันของข้อมูล (consistency) ในขณะที่ระบบกำลังทำงานได้

โดยทั่วไป Nginx นั้นมีกระบวนการทำงานในลักษณะ event-based หรือตามเหตุการณ์ที่เกิดขึ้นคือ จะทำงานตามการร้องขอที่เข้ามาจนเสร็จโดย master จะจัดการเรื่องคิวของการร้องขอข้อมูลและกระจายส่งการร้องขอที่เข้ามานั้นไปยัง workers ที่มีเทรดว่าง หรืออาจจะรอคิวหาก worker ทุกตัวยังไม่สามารถรับงานใหม่ได้ เนื่องจากไม่สามารถแตกเทรด (spawn threads) ออกไปได้อีก จึงต้องรอให้เทรดใดเทรดหนึ่งของ worker ทำงานเสร็จสิ้นก่อนแล้วจึงสามารถสั่งการให้เทรดนั้นรับงานใหม่ไปทำงานต่อได้ แต่ละ worker สามารถแตกเทรดได้ถึง 1024 เทรด [39] ดังนั้น หากมีจำนวน worker ยิ่งมาก จำนวนเทรดที่สามารถมีได้ก็มากขึ้นเช่นกัน ซึ่งทุก ๆ เทรดที่ทำงานอยู่จะสามารถเข้าถึงแคชได้พร้อม ๆ กัน อีกทั้งยังสามารถเข้าถึงข้อมูลการใช้งานแคชที่ใช้ในการคำนวณค่าดัชนีความนิยมได้พร้อม ๆ กัน ดังนั้นการเพิ่มกระบวนการจัดการของการเข้าถึงแคชและข้อมูลการใช้งานนั้นจำเป็นอย่างยิ่ง เพื่อไม่ให้เกิดภาวะการเข้าถึงข้อมูลเดียวกันพร้อมกัน (data concurrency) และข้อมูลของระบบมีความสอดคล้องกัน ผู้วิจัยได้เพิ่มส่วนการจัดการดังกล่าวโดยออกแบบการทำงานในลักษณะคิวให้กับเทรดที่กำลังทำงานอยู่ เพื่อช่วยให้การแทนที่แคชทำงานได้

อย่างถูกต้องและมีประสิทธิภาพ ในงานวิจัยนี้กำหนดแถวคอยหรือคิว (Queue) ในการแทนที่แคช และแยกส่วนต่าง ๆ ออกไปเป็นโปรเซส (Process) ย่อยดังต่อไปนี้

3.6.1 Popularity Calculation Process

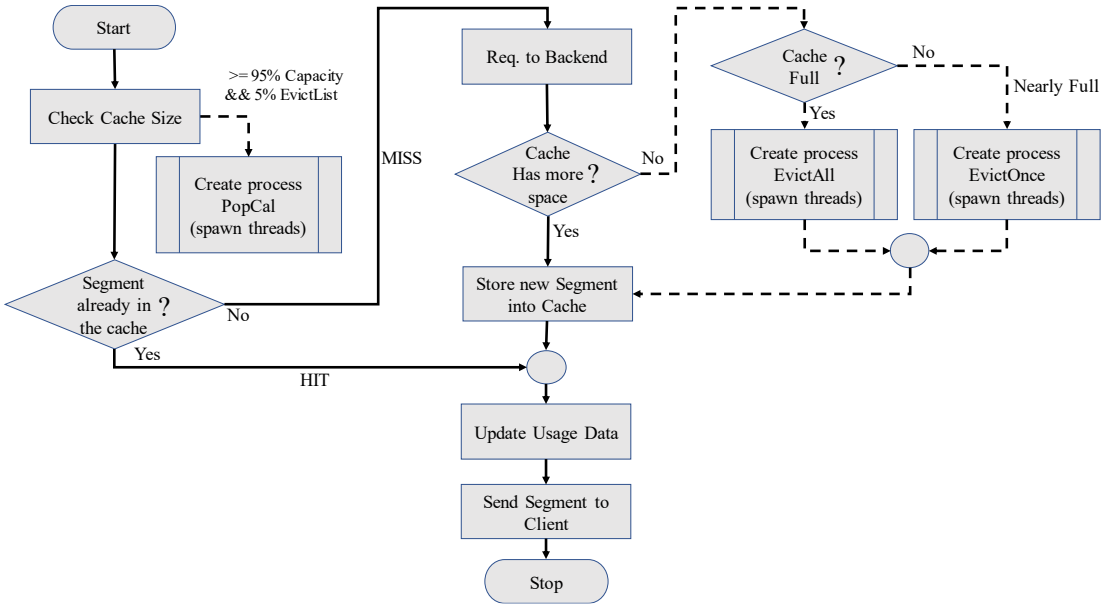
การคำนวณหาค่าดัชนีความนิยมของ segment ที่อยู่ภายในแคช ระบบจะกระทำกระบวนการนี้ก็ต่อเมื่อแคชถูกใช้ไปจนเกือบเต็มหรือมีค่า 95% ของขนาดของแคช (cache capacity) และมี EvictList หรือกลุ่มของ segment ที่จะถูกแทนที่เมื่อแคชเต็ม (Candidate Evict List) เหลืออยู่ 5% ของจำนวน segment ที่จัดอยู่ในกลุ่มที่จะต้องถูกแทนที่เมื่อเกิดกรณีแคชเต็ม ซึ่งสามารถคำนวณขนาด EvictList ได้จากสมการที่ (8) เทียบกับขนาดของแคช เพื่อให้การคำนวณค่าความนิยมเกิดขึ้นเมื่อจำเป็นเท่านั้น โดยจะคำนวณไว้ล่วงหน้าก่อนแคชจะเต็มความจุที่ 95% หรือเหลือจำนวน segment ที่พร้อมจะถูกขับออกจากแคชน้อยกว่า 5% จากสูงสุด 10% ของความจุแคชเพื่อให้ระบบพร้อมในการรองรับการทำงานกับการร้องขอจำนวนมากได้ Popularity Calculation จะถูกแยกการทำงานออกไปโดยการสร้างเทรดจำเพาะในการทำงาน เพื่อให้ไม่ไปรบกวนการทำงานเพื่อรองรับการร้องขอของไคลเอนต์และ Popularity Calculation ใช้การคำนวณตามสมการที่ (5) เมื่อการคำนวณเสร็จสิ้นจะกำหนดแถวคอยของรายชื่อ segment ที่มีค่าดัชนีความนิยมน้อยตามสมการที่ (6) นั่นหมายความว่า segment ดังกล่าวเป็นกลุ่มที่จะถูกแทนที่หากแคชเต็มและมีความจำเป็นต้องแทนที่แคชด้วย segment ใหม่ โดยจะเลือก segment ที่มีดัชนีความนิยมน้อยที่สุดในกลุ่ม ณ ขณะนั้นและเริ่มกระบวนการแทนที่ (eviction process) โดยการสร้างชุดของคิวขึ้นมาจำนวนหนึ่งเรียกว่า EvictListQueue ซึ่งจะมีอยู่ด้วยกันหลายชุด หลังจากนั้นระบบจะกระจาย EvictList ทั้งหมดให้กับ EvictListQueue โดยใช้รูปแบบการกระจาย EvictList แบบ Round Robin [49] กล่าวคือวนสลับการกระจายอย่างเท่าเทียมกันเพื่อกระจายค่าดัชนีความนิยมในแต่ละ EvictListQueue ให้มีค่าใกล้เคียงกัน เพื่อให้ทุก ๆ แถวคอยมีขนาดเท่ากันและได้ EvictList ที่มีค่าดัชนีความนิยมเทียบเท่ากัน

3.6.2 Eviction Process

เมื่อมีการร้องขอจากไคลเอนต์เข้ามาและ master ของ Nginx ส่งภาระงานให้กับ worker ที่มีเทรดว่างและพบว่าข้อมูลที่ต้องการไม่ได้อยู่ในแคช (cache miss) ดังนั้นจึงมีความจำเป็นจะต้องร้องขอข้อมูลต่อไปยัง backend storage หลังจากนั้นเมื่อพบว่าแคชเต็มก็จำเป็นต้องแทนที่แคชด้วย segment ใหม่ เทรดนั้นของ worker จะสร้างโปรเซสใหม่ขึ้นมาเพื่อลบ segment ที่มีดัชนีความนิยมที่ผ่านการลดทอนแล้วค่าน้อยที่สุดออกจากแคช โดยจะตรวจสอบว่า EvictListQueue ชุดใดที่มี segment ดังกล่าวอยู่ ณ เวลานั้นและยังไม่ถูกจองโดยเทรดก่อนหน้า ก็จะนำไป evict ด้วยแถวคอยชุดนั้น การมี EvictListQueue หลายชุดเปรียบเสมือนมีแถวคอยหลายแถวพร้อมให้เทรดใช้

บริการ โดยแต่ละแถวมีความยาวสั้น ๆ เท่าๆ กัน และหัวแถวสามารถเริ่มทำงานได้พร้อม ๆ กันตามจำนวนแถวที่มี หรือจำนวนชุดแถวคอยนั่นเอง ซึ่งจะเป็นผลดีในกรณีที่มีการร้องขอจำนวนมากเข้ามาพร้อม ๆ กัน จะทำให้เทรดรอเวลาที่จะเริ่มทำงานไม่นานนัก และสามารถทำงานพร้อม ๆ กันได้ในทันที การจัดการคิวในขั้นตอนนี้ช่วยให้ทุก ๆ เทรดจากทุก ๆ worker ของ Nginx ที่ต้องทำการแทนที่แคชต้องเข้าแถวคอยรอในแถวคอยของตัวเองเพียงสั้นๆ และเทรดในแต่ละแถวคอยที่เริ่มทำงานจะสามารถทำงานได้พร้อม ๆ กันกับเทรดในแถวคอยอื่น

ในกรณีที่พบ segment ที่ต้องการอยู่ในแคช (cache hit) ระบบจะตรวจสอบทุกครั้งว่า segment ดังกล่าวอยู่ในกลุ่มที่จะถูกแทนที่หรือไม่และ segment นั้นอยู่ในแถวคอยใด หากพบว่าอยู่ในกลุ่มดังกล่าวจะลบ segment นั้นออกจากแถวคอยที่จะนำออกจากแคช (EvictListQueue) นั้นทันที เพื่อให้ segment นั้นยังคงอยู่ในแคชต่อไป และจะเริ่มการลดทอนดัชนีความนิยมใหม่อีกครั้งจากค่าดัชนีความนิยมใหม่ การทำงานของระบบการจัดการแคชจากอัลกอริทึมที่ 3.2 สามารถแสดงในรูปของแผนผังการทำงานได้ดังรูปที่ 3.5

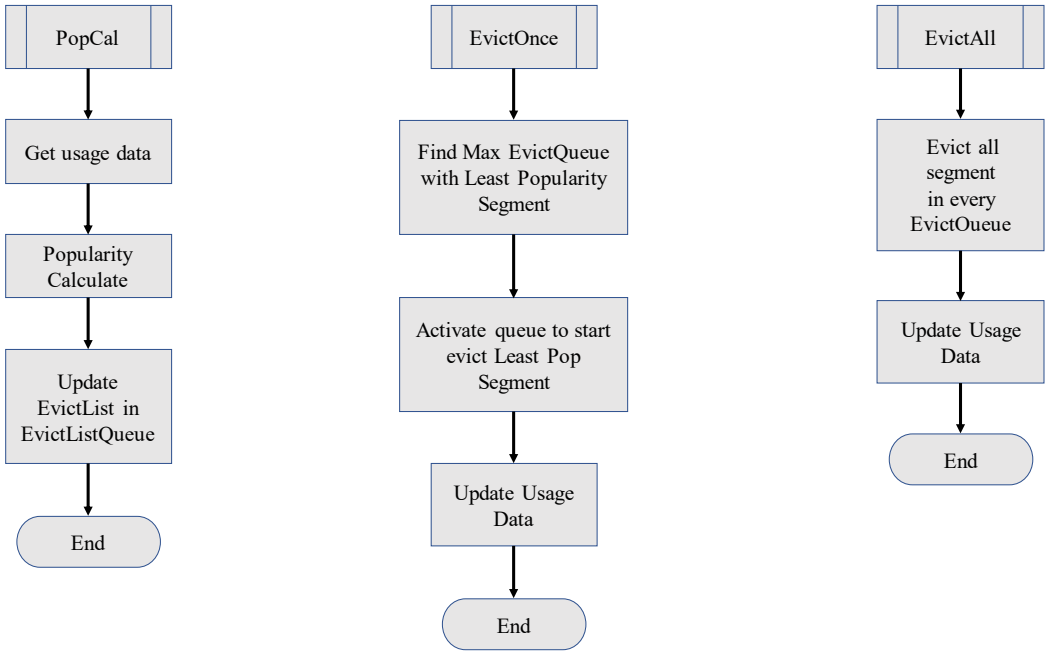


รูปที่ 3.5 แผนผังการทำงานของนโยบายการแทนที่แคชที่ผู้วิจัยออกแบบ

จากรูปที่ 3.5 จะเห็นว่ากระบวนการ PopCal, EvictOnce และ EvictAll ถูกแยกการทำงานออกไปเป็นโปรเซสย่อยที่จะถูกสร้างขึ้นมาให้ทำงานก็ต่อเมื่อมีความจำเป็นเท่านั้น โดยทุก ๆ เทรดของ Nginx สามารถสร้างและเรียกใช้งานทั้งสามโปรเซสย่อยนี้ได้ขึ้นอยู่กับสถานการณ์ของแคชขณะที่รับการร้องขอจากไคลเอนต์ เช่น ในกรณีที่มีการร้องขอ segment จากไคลเอนต์และปรากฏว่าไม่มี segment นั้นในแคชและพบว่าแคชใกล้เต็ม เทรดหลักของ worker ก็จะทำการสร้าง

โปรเซส EvictOnce ขึ้นมาและลบ segment ที่มีค่าดัชนีความนิยมน้อยที่สุดในขณะนั้นออกจากแคช ในขณะเดียวกันนั้นเทรดเดิมใน worker ก็จะยังคงทำงานเดิมต่อเนื่องไปพร้อม ๆ กัน โดยจะทำการร้องขอ segment ของข้อมูลนั้นไปยัง backend storage กล่าวคือการร้องขอ segment ไปยัง backend storage ก็ยังดำเนินต่อไปในระหว่างที่เทรด EvictOnce ทำการลบ (evict) segment ที่มีดัชนีความนิยมที่ผ่านการลดทอนแล้วและมีค่าน้อยที่สุดออกจากแคช ซึ่งการลบ segment ของไฟล์วิดีโอออกจากแคชนั้นสามารถกระทำได้รวดเร็วและจะทำงานเสร็จก่อนกระบวนการเขียนข้อมูลลงในแคชจะเกิดขึ้น ซึ่งเมื่อการลบ segment ออกจากแคชเสร็จสิ้นจะส่งสัญญาณ (Signal) กลับไปบอกเทรดหลักเพื่อให้เทรดหลักทราบ และ EvictOnce จะจบการทำงานไป (Terminate) และเมื่อได้รับข้อมูลตอบกลับจาก backend storage แล้วระบบสามารถเขียน segment ใหม่ลงในแคชได้ทันที ส่วนโปรเซส EvictAll ก็ทำงานในลักษณะเดียวกับ EvictOnce เพียงแต่เป็นการลบ segment ทั้งหมดที่อยู่ในกลุ่มที่จะถูกแทนที่มีอยู่ใน EvictList ซึ่งจะกระทำโปรเซสนี้ก็ต่อเมื่อระบบมีภาระงานสูง

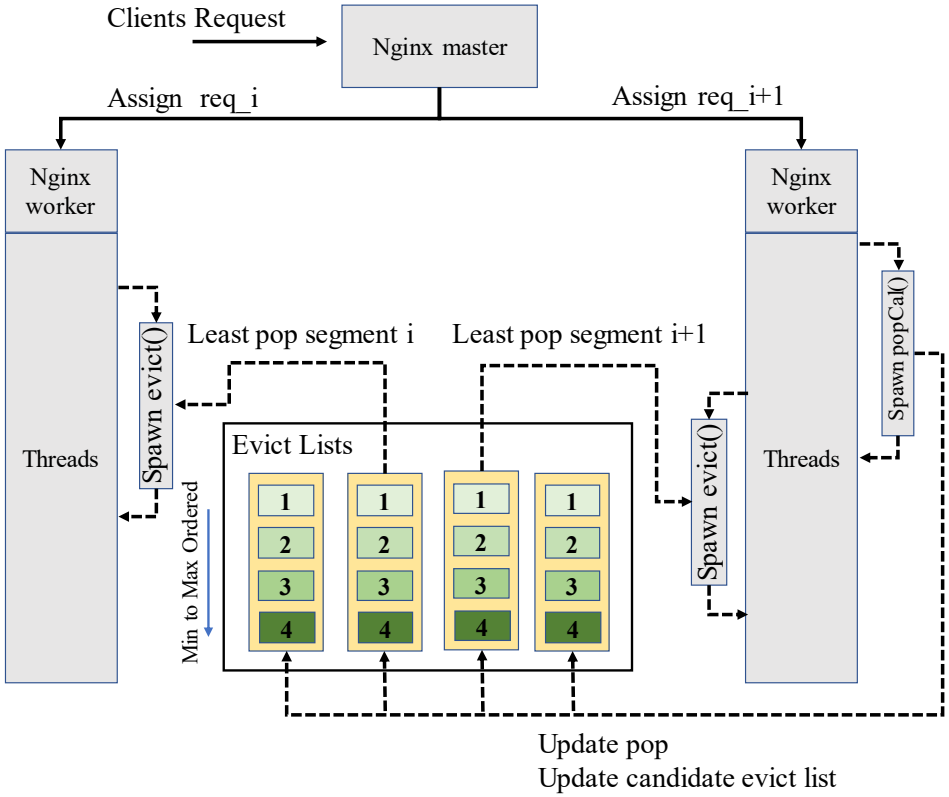
ส่วนโปรเซส PopCal นั้นจะแบ่งการทำงานออกเป็นในลักษณะเทรดย่อยเช่นกัน ซึ่งจะคำนวณหาค่าดัชนีความนิยมเมื่อแคชถูกใช้ไป 95% ของขนาดความจุของแคชและมีจำนวน segment ที่เป็น candidate evict list เหลือน้อยกว่า 5% ของจำนวนสูงสุดที่มีได้ตามสมการที่ (8) กระบวนการทำงานย่อยภายในทั้งสามโปรเซสย่อยเป็นดังรูปที่ 3.6



รูปที่ 3.6 แผนผังการทำงานของโปรเซสย่อยภายในนโยบายการแทนที่แคช

จากรูปที่ 3.6 จะเห็นว่าการทำงานส่วนใหญ่ของระบบจะเป็นการทำงานในส่วนของ Nginx ซึ่งจะต้องควบคุมการทำงานของเทรดย่อย ๆ ให้เป็นไปอย่างถูกต้องโดยไม่ทำให้การทำงานหลักเกิดปัญหา ทุกครั้งที่มีการคำนวณหาค่าดัชนีความนิยมและได้กลุ่มของ EvictList ที่อาจจะถูกแทนที่ก็จะทำการกระจาย EvictList ออกเป็นชุดย่อย ๆ โดยในแต่ละชุดย่อยก็จะเรียงจาก segment ที่มีค่าดัชนีความนิยมน้อยไปมาก การกระจายออกเป็นชุดย่อย ๆ นั้นจะใช้วิธีการแบบ Round Robin ซึ่งเป็นวิธีการกระจาย segment ของแต่ละชุดย่อยให้มีค่าดัชนีความที่นิยมที่กระจายในแต่ละชุดอย่างสม่ำเสมอและใกล้เคียงกันดังที่ได้กล่าวไปแล้ว

ทุก ๆ เทรด EvictOnce จากทุก ๆ การร้องขอสามารถเข้า EvictList ได้พร้อม ๆ กัน แต่เทรดแต่ละตัวจะได้ segment ที่มีค่าความนิยมต่ำสุดไม่เท่ากัน โดยกลไกการจัดแถวคอยให้ทุก ๆ เทรดได้รับ segment ที่มีค่าดัชนีความนิยมน้อยที่สุดในขณะที่เทรดนั้น ๆ เข้ามาใช้งานและพบว่าต้องมี segment หนึ่งถูกแทนที่ โดยการออกแบบให้ EvictList แยกออกเป็นชุดย่อย และในแต่ละชุดย่อยของ EvictList จะเรียง segment ที่มีค่าดัชนีความนิยมจากน้อยไปหามากด้วยเช่นกัน เพื่อเพิ่มช่องทางการเข้าถึงจากทุก ๆ เทรดของ EvictOnce ที่ทำงานอยู่โดยสามารถถูกเข้าถึงได้พร้อมกันดังรูปที่ 3.7



รูปที่ 3.7 การเข้าถึงข้อมูลใน EvictList ของเทรดย่อย

รูปที่ 3.7 แสดงการสร้างเทรดเพื่อให้ทำงานอิสระจากเทรดหลัก และเทรดย่อยจะทำงานอิสระจากกันโดยเทรดที่คำนวณดัชนีความนิยมนั้นจะคำนวณค่าความนิยมของทุก ๆ segment ที่อยู่ในแคชและจะเลือกกลุ่มของ segment ที่มีดัชนีความนิยมน้อยตามอัตราส่วนของขนาดความจุแคช และเพิ่มข้อมูลของ segment กลุ่มดังกล่าวใน EvictList กระจายเป็นชุด ในขณะที่เทรดที่ทำการลบ segment ออกจากแคชจะเข้าถึง EvictList ชุดที่มีข้อมูลอ้างอิงของ segment ที่มีค่าดัชนีความนิยมน้อยที่สุดในขณะนั้นจากแถวคอยทั้งหมด และจะใช้ข้อมูลอ้างอิงของ segment ดังกล่าวลบ segment วิดีโอออกจากแคช พร้อมกับลบข้อมูลอ้างอิงของ segment นั้นออกจาก EvictList และเขียน segment วิดีโอใหม่ลงไปในแคช การเข้าใช้ข้อมูล EvictList ของเทรดย่อยนั้นจะสามารถเข้าใช้งานได้พร้อม ๆ กันตามจำนวนชุดของ EvictList ซึ่งเมื่อเทรดย่อยต้องใช้งาน EvictList เทรดย่อยนั้นจะพิจารณาหาว่าค่าดัชนีต่ำที่สุดอยู่ใน EvictList ชุดใดก็จะดึงข้อมูลอ้างอิงของ segment จาก EvictList ชุดนั้น โดยค่าดัชนีความนิยมของ segment จะถูกเรียงจากน้อยไปหามาก และใช้โครงสร้างข้อมูลแบบ SortedSet [44] ของ Redis ในการเรียงลำดับ ทำให้ไม่ต้องมีส่วนของโปรแกรมจัดเรียงข้อมูลในนโยบายการแทนที่แคชซึ่งลดภาระการทำงานของระบบลงได้อย่างมาก

3.7 การลดปัญหาความสอดคล้องกันของข้อมูล (cache coherence) ภายในแคช

นอกจากการควบคุมการเข้าถึงข้อมูลของระบบและจัดการกระบวนการทำงานของระบบให้เป็นไปอย่างถูกต้องในขั้นตอนการออกแบบและการเขียนโปรแกรมควบคุมแล้ว การเข้าถึงข้อมูล segment วิดีโอภายใน Redis ยังมีการควบคุมภาวะความพร้อมกัน (concurrency) ในการอ่านและเขียนข้อมูลและมีกระบวนการควบคุมความสอดคล้องกันของข้อมูล (data consistency) ภายในของ Redis อีกด้วย

ในส่วนของปัญหาการอยู่ในแคชนานจนเกินไปของ segment ข้อมูลแม้จะไม่ได้รับการร้องขอซ้ำจากไคลเอนต์นั้น ระบบจะลดทอนค่าดัชนีความนิยมด้วยอัตราที่สูงขึ้นเรื่อย ๆ ตามระยะเวลาที่อยู่ในแคชและจะทำให้ค่าดัชนีความนิยมที่ได้มีค่าที่ต่ำมาก ทำให้ในท้ายที่สุด segment ดังกล่าวจะถูกลบออกจากแคชด้วยระบบตามนโยบายแทนที่แคช โดยจะกำหนดค่า TTL ใหม่ให้กับ segment ที่อยู่ในแคชทุกครั้งที่ถูกเข้าถึง หากเกิดกรณีที่ segment มีดัชนีความนิยมสูงมากจนใช้ระยะเวลาในการลดทอนนานมาก การกำหนดค่า TTL ให้กับทุก ๆ segment จะช่วยให้แคชสามารถลบข้อมูลที่ไม่จำเป็นออกจากแคชได้อีกทางหนึ่ง ในการทดลองกำหนดค่า TTL เท่ากับ 24 ชั่วโมง

หรือ 86,400 วินาที แต่ผู้ใช้อาจจะปรับเพิ่มหรือลดตามความเหมาะสมของการใช้งาน และจะกำหนดให้เป็นค่าเริ่มต้นใหม่ทุกครั้งที่ segment นั้นถูกเข้าถึงในแคช

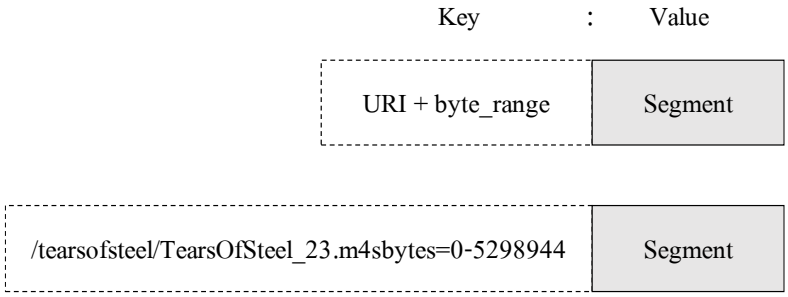
3.8 การนำนโยบายการแทนที่แคชที่ออกแบบไปใช้กับซอฟต์แวร์แคช Redis

ในงานวิจัยนี้ใช้ Redis รุ่น 4.0.11 ซึ่งมีนโยบายการแทนที่แคชของ Redis ที่สามารถควบคุมโดยการกำหนดค่าพารามิเตอร์ที่เกี่ยวข้องเพื่อใช้นโยบายการแทนที่แคชดังรายละเอียดในบทที่ 2 ซึ่งในงานวิจัยนี้ไม่ประสงค์ที่จะใช้นโยบายการแทนที่แคชของ Redis เนื่องจากไม่สามารถควบคุมตัวแปรและวัดค่าพารามิเตอร์ที่สำคัญได้ ดังนั้นจึงกำหนดให้ Redis เป็นเพียงแหล่งจัดเก็บข้อมูลที่อยู่ในหน่วยความจำหลักเท่านั้น และกำหนดพารามิเตอร์ maxmemory-policy ของ Redis ให้มีค่าเป็น “noeviction” ซึ่งจะไม่มีการใช้นโยบายแทนที่แคชของ Redis แต่จะควบคุมด้วยนโยบายแทนที่แคชจากโปรแกรมที่ผู้วิจัยเขียนขึ้นเอง ทั้งนโยบายการแทนที่แคชที่ผู้วิจัยนำเสนอ, นโยบาย LRU และ นโยบายLFU โดยการอ้างอิงความจุของแคชเป็นจำนวนของ segment สูงสุดที่แคชสามารถบรรจุได้แทนและไม่กำหนดค่าพารามิเตอร์ maxmemory ของ Redis ซึ่งเป็นการกำหนดให้พื้นที่จัดเก็บข้อมูลในหน่วยของไบต์ (byte) มีขนาดไม่จำกัด

3.9 การอ้างอิง segment ในซอฟต์แวร์แคช Redis

การร้องขอข้อมูลไฟล์วิดีโอของมาตรฐาน MPEG-DASH จะอ้างอิงเป็นช่วงของข้อมูล (byte range) ในความยาววิดีโอทั้งหมดและแบ่งเป็นช่วงของ segment duration การเก็บข้อมูล segment ของไฟล์วิดีโอในแคชใช้โครงสร้างข้อมูล Set [44] ของ Redis และใช้การอ้างอิงข้อมูลภายในแบบ Key:Value [44] ดังนั้นการอ้างอิงข้อมูล segment จึงใช้การอ้างอิงแบบ Key:Value เช่นกัน โดยกำหนดให้ใช้ชื่อและที่อยู่ของไฟล์วิดีโอโดยรูปแบบเดียวกับ Uniform Resource Identifier (URI) [20] หรือ directory ของไฟล์เป็นไปตามมาตรฐานโปรโตคอล HTTP และตามด้วยช่วงของไบต์ข้อมูลประจำ segment นั้น เป็นคีย์ในการอ้างข้อมูล ดังตัวอย่างในรูปที่ 3.8

จากรูปที่ 3.8 แสดงหลักการอ้างอิง segment ข้อมูลไฟล์วิดีโอที่ถูกจัดเก็บในแคชด้วยโครงสร้างข้อมูลแบบ Set ของ Redis โดยจะใช้ URI ของไฟล์ในการบ่งบอกว่าเป็นไฟล์ใดและช่วงไบต์ข้อมูลจะบ่งบอกว่าเป็นช่วงใดของไฟล์ที่อ่านมาจากไฟล์เดิมที่อยู่ใน backend storage



รูปที่ 3.8 การอ้างอิงข้อมูล segment ภายในแคชตามโครงสร้างข้อมูลแบบ Set ของ Redis

3.10 การเขียนโปรแกรมภาษา Lua เพื่อทำงานใน Nginx

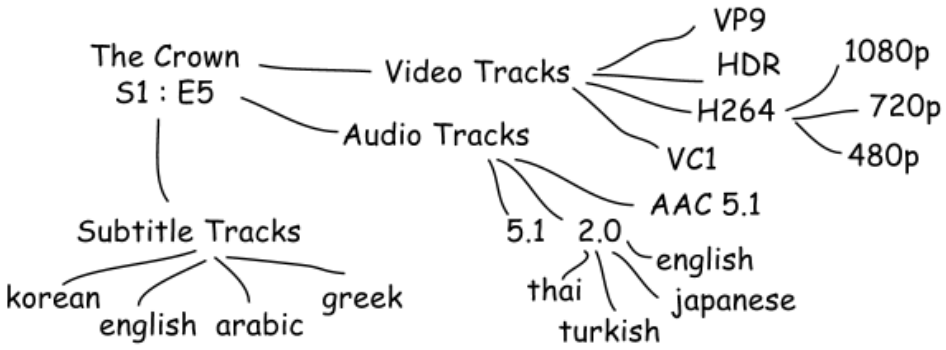
การเขียนโปรแกรมภาษา Lua เพื่อทำงานในระบบของเว็บเซิร์ฟเวอร์ของ Nginx นั้นต้องอาศัยโมดูลโปรแกรมจำนวนมาก ผู้วิจัยได้เลือก OpenResty เป็นเฟรมเวิร์คหนึ่งที่ทำให้นักวิจัยสามารถเขียนโปรแกรมควบคุมการทำงานของ Nginx ได้โดยการแทรกส่วนของโปรแกรมเข้าไปในแต่ละเฟสย่อยของการทำงาน โดยบังคับให้เทรดย่อยของ Nginx ทำงานในส่วนที่เพิ่มเข้าไปโดยไม่ทำให้งานหลักเกิดความเสียหาย ซึ่งการจัดการแคชและนโยบายการแทนที่แคชทั้งที่ผู้วิจัยนำเสนอรวมถึงนโยบาย LRU และ LFU ถูกเขียนด้วยภาษา Lua ทั้งหมด

3.11 ข้อมูลไฟล์วิดีโอที่ใช้ในการทดสอบ (Video dataset)

ข้อมูลวิดีโอที่ใช้ในการทำ streaming ของระบบให้บริการไฟล์ข้อมูลวิดีโอแบบ MPEG-DASH นั้นจำเป็นต้องเข้ารหัสไฟล์ในรูปแบบไฟล์ประเภทที่เรียกว่า fMP4 (fragmented MP4) ซึ่งแตกต่างจากไฟล์ประเภทอื่น ๆ ไฟล์ fMP4 นั้นคือไฟล์ที่สามารถรองรับการอ่านไฟล์โดยอ้างอิงช่วงข้อมูล (byte range) ได้ ซึ่งมีอยู่ด้วยกันหลายประเภทไฟล์ ตัวอย่างโปรแกรมประยุกต์ที่ใช้ในการเข้ารหัสไฟล์วิดีโอในรูปแบบนี้ เช่น FFMPEG, BenTo หรือ MP4box เป็นต้น ในกรณีของระบบแบบ VoD ไฟล์ทั้งหมดต้องทำการเข้ารหัสไว้ก่อนที่จะทำ streaming ได้ เวลาในการเข้ารหัสต้องขึ้นอยู่กับจำนวนไฟล์และผลลัพธ์ตามจำนวนระดับคุณภาพไฟล์ที่ต้องการ

ในข้อมูลวิดีโอหนึ่ง ๆ ที่จะต้องทำ streaming นั้นประกอบด้วยไฟล์ข้อมูลจำนวนมาก ยกตัวอย่างเช่น ไฟล์ข้อมูลที่ต้องเตรียมไว้ล่วงหน้าของ Netflix ดังแสดงรูปที่ 3.9 ไฟล์ในระบบของภาพยนตร์ตอนหนึ่งของเรื่องหนึ่ง ๆ ต้องเตรียมไฟล์เอาไว้เป็นจำนวนมาก แยกออกเป็นไฟล์

ภาพ (Video Tracks), ไฟล์เสียง (Audio Tracks), ไฟล์ข้อความบรรยาย (Subtitle Tracks) และในหมวดหมู่ของไฟล์ภาพมีการเข้ารหัสไฟล์เดียวกันแยกไปหลายไฟล์ตามรูปแบบการเข้ารหัส (video encode หรือ Codec) ต่าง ๆ ในแต่ประเภทการเข้ารหัสไฟล์ก็แยกย่อยไปอีกหลายระดับคุณภาพของไฟล์ภาพ ไฟล์เสียงและไฟล์ข้อความบรรยาย ซึ่งจะแยกย่อยออกไปแต่ละประเภทการเข้ารหัสและแต่ละภาษา จะเห็นว่ามีไฟล์จำนวนมากในระบบ



รูปที่ 3.9 แผนผังไฟล์วิดีโอในระบบของ Netflix [50]

ในงานวิจัยนี้ผู้วิจัยได้เลือกใช้ไฟล์วิดีโอที่เป็น dataset ที่เปิดให้นักวิจัยสามารถนำมาทดสอบระบบจาก MMSys [51,52] ซึ่งเป็นหน่วยวิจัยต่างประเทศที่ทำงานวิจัยทางด้านวิดีโอ streaming โดยเฉพาะงานวิจัยเกี่ยวกับการให้บริการไฟล์ในมาตรฐาน MPEG-DASH และเปิดเป็น open dataset ให้นักวิจัยสามารถนำมาใช้ในงานวิจัยได้ ข้อมูลที่ใช้ในการทดสอบระบบเป็นดังตารางที่ 3.1

ตารางที่ 3.1 ข้อมูลวิดีโอที่ใช้ในการวัดประสิทธิภาพของระบบ

ชื่อวิดีโอ	ความยาววิดีโอ (นาที)	ประเภทวิดีโอ	จำนวนระดับ Bitrate
Big Buck Bunny	9.56	Animation	20
Elephants Dream	10.54	Animation	20
Of Forest and Men	7.33	Documentary	19
Tears of Steel	12.15	SCI-FI Movie	13

ระดับในการทดลองจะเลือกระดับคุณภาพไฟล์วิดีโอหรือระดับ bitrate ของแต่ละไฟล์วิดีโอมาใช้เพียง 2 ระดับที่ใกล้เคียงกันคือ 1.4 Mbps และ 2.3 Mbps โดยไม่ใช้ไฟล์ของตัวอักษรบรรยายซึ่งเป็นไฟล์ขนาดเล็กและมีระดับ bitrate ที่ต่ำ และใช้รูปแบบพฤติกรรมของผู้ใช้สองแบบ

ในการทดสอบ โดยพิจารณาในระดับ segment และอ้างอิงตามรูปแบบการลำดับความนิยม (rank) สองแบบคือ Uniform และ Zipf-Like ซึ่งจะกล่าวถึงในบทถัดไป

3.12 คุณลักษณะของระบบให้บริการไฟล์วิดีโอที่ใช้ในการทดลอง

คุณลักษณะของระบบให้บริการไฟล์วิดีโอที่ใช้ในการทดลองมีดังนี้

3.12.1 ซอฟต์แวร์

เซิร์ฟเวอร์ประกอบด้วยซอฟต์แวร์ที่สำคัญดังต่อไปนี้

1. ระบบปฏิบัติการCentOS รุ่น 7.5.1804
2. Nginx 1.13.6.2
3. OpenResty 1.13.6.2
4. Redis 4.0.11
5. LuaJit รุ่น 2.1
6. Perl รุ่น 5.16.3 ทำหน้าที่รันคำสั่งภาษา Perl ภายในของระบบ OpenResty
7. Lua รุ่น 5.1.4

ไคลเอนต์ประกอบด้วยซอฟต์แวร์ที่สำคัญดังต่อไปนี้

1. Google Chrome รุ่น 74.0.3729.169
2. libdash รุ่น 2.0
3. Native MPEG-Dash + HLS Playback Google Chrome Extension [48]
4. VLC Media Player รุ่น 3.0.6 Vetinari ซึ่งรองรับ libdash [52]
5. JMeter [53]

3.12.2 ฮาร์ดแวร์

1. HTTP Server และแคช
   a. Intel Core i5 3.4 GHz
   b. หน่วยความจำหลัก DDR4 64 กิกะไบต์
   c. ฮาร์ดดิสก์ 1 เทราไบต์
2. Backend Storage

a. Intel Core i5 3.4 GHz
b. หน่วยความจำหลัก DDR3 16 กิกะไบต์
c. ฮาร์ดดิสก์ 500 กิกะไบต์

3. ไคลเอนต์
a. Intel Core i3 M380 2.53 GHz
b. DDR3 8 กิกะไบต์
c. ฮาร์ดดิสก์ 500 กิกะไบต์

3.13 พารามิเตอร์ที่ใช้ในการวัดประสิทธิภาพการทำงานของระบบ

การวัดประสิทธิภาพการทำงานของแคชด้วยนโยบายการแทนที่แคชที่ผู้วิจัยนำเสนอ เปรียบเทียบกับนโยบายการแทนที่แคช LRU และ LFU โดยจะทำการทดสอบเพื่อบันทึกค่าเฉลี่ยของพารามิเตอร์ต่าง ๆ ดังนี้

3.13.1 Hit rate

Hit rate คืออัตราการพบข้อมูลภายในแคชเมื่อคำนวณจากจำนวนการร้องขอทั้งหมดซึ่งมีค่าเป็นดังสมการที่ (9)

กำหนดให้

total_hit เป็นจำนวนครั้งของการพบข้อมูลในแคชจากการร้องขอทั้งหมด

total_request เป็นจำนวนครั้งของการร้องขอข้อมูลทั้งหมด

$$Hit\ rate = \frac{total\_hit}{total\_request} \tag{9}$$

3.13.2 Miss rate

Miss rate คืออัตราการไม่พบข้อมูลภายในแคชเมื่อคำนวณจากจำนวนการร้องขอทั้งหมดจากผู้ใช้และสามารถหาได้จากค่าของ *hit rate* ดังสมการที่ (10)

$$Miss\ rate = 1 - hit\ rate \tag{10}$$

3.13.3 Cache effective access time

Cache effective time คือเวลาที่ในการเข้าถึงข้อมูลภายในแคชของระบบ

กำหนดให้

cache access time เป็นค่าเฉลี่ยเวลาการเข้าถึงข้อมูลภายในแคช

back_end_storage access time เป็นค่าเฉลี่ยระยะเวลาการเข้าถึงข้อมูลที่อยู่ใน back end storage ของระบบ

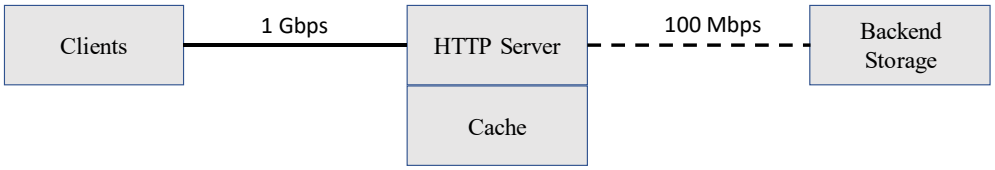
$$cache\ effective\ access\ time = (hit\ rate * cache\ access\ time) + (miss\ rate * back\_end\_storage \quad access\ time) \tag{11}$$

บทที่ 4
การทดลองและวิเคราะห์ผล

ในบทนี้จะกล่าวถึงการทดลองเพื่อวัดประสิทธิภาพการทำงานของแคชในระบบการให้บริการไฟล์วิดีโอแบบ MPEG-DASH และผลของนโยบายการแทนที่แคชที่ผู้วิจัยนำเสนอโดยเปรียบเทียบกับนโยบายการแทนที่แคชที่ใช้กันอยู่ทั่วไปในปัจจุบัน โดยอ้างอิงรูปแบบพฤติกรรมการใช้งานระบบของผู้ใช้งานในระบบการให้บริการไฟล์วิดีโอ พร้อมทั้งวิเคราะห์ผลการทดลองที่ได้

4.1 การออกแบบการทดลอง

จากการวิเคราะห์และออกแบบระบบให้บริการไฟล์วิดีโอ ผู้วิจัยได้เลือกใช้ Redis เพื่อทำหน้าที่เป็นแคชและซอฟต์แวร์จัดการการเข้าถึงข้อมูลภายในแคช โดยให้แคชอยู่ใกล้กับ HTTP server มากที่สุดและแยกส่วนของ backend storage ออกและเชื่อมต่อกันด้วยระบบเครือข่ายอินเตอร์เน็ตแบบสายหรือ LAN (Local Area Network) ระหว่างกันโดยใช้ความเร็วอินเตอร์เน็ตระหว่าง HTTP server และ Backend storage ที่ความเร็ว 100 Mbps (Mega bit per second) และระหว่าง HTTP server และไคลเอนต์ที่ความเร็ว 1 Gbps (Giga bit per second) ในทุก ๆ การทดลองในบทนี้จะอ้างอิงความเร็วของการเชื่อมต่อระหว่างอุปกรณ์หลักดังรูปที่ 4.1 และโครงสร้างการทำงานและรายละเอียดของระบบดังในรูปที่ 3.4 ในบทที่ 3 และทดสอบด้วยไคลเอนต์จำนวนหนึ่งเพื่อวัดประสิทธิภาพการทำงานของแคชด้วยนโยบายที่ผู้วิจัยนำเสนอเปรียบเทียบกับนโยบายการจัดการแคชที่เป็นที่นิยมในปัจจุบันและวัดประสิทธิภาพการทำงานเมื่อขนาดความจุของแคชของเปลี่ยนแปลงไป



รูปที่ 4.1 การเชื่อมต่อเครือข่ายอินเตอร์เน็ต (LAN) ของระบบในการทดลอง

จากการออกแบบนโยบายการแทนที่แคชเพื่อใช้กับระบบบริการไฟล์วิดีโอแบบ DASH ที่แบ่งการดาวน์โหลดออกเป็นช่วงไฟล์ (byte range) หรือ segment ย่อย นโยบายการแทนที่แคชที่ผู้วิจัยออกแบบจะจัดการข้อมูลในแคชในระดับ segment ซึ่งจะแบ่งตามค่า segment duration หรือช่วงเวลาของวิดีโอในแต่ละ segment และทุก ๆ segment ของวิดีโอจะมี segment duration ในระยะเวลาที่เท่ากันและสามารถกำหนดได้ในขั้นตอนการเข้ารหัสไฟล์ แต่ขนาดของข้อมูลในหน่วย byte ของแต่ละ segment จะไม่เท่ากัน ทำให้การออกแบบแคชต้องอ้างอิงขนาดพื้นที่จัดเก็บข้อมูลของแคชเป็นจำนวน segment ที่แคชจะสามารถเก็บได้ แทนที่จะเป็นขนาดของแคชในหน่วยความจุข้อมูล (byte) ผู้วิจัยเลือก segment duration ที่ 4 วินาที เพื่อไม่ให้แต่ละไคลเอนต์ร้องขอข้อมูลบ่อยครั้งจนเกินไปและทำให้ความถี่ของการร้องขอข้อมูลไม่สูงจนเกินไปในกรณีที่มีไคลเอนต์จำนวนมาก

4.2 รูปแบบพฤติกรรมที่ของผู้ใช้

ค่าความนิยมของไฟล์วิดีโอที่นำมาทดสอบจะพิจารณาในระดับ segment และอ้างอิงตามรูปแบบการลำดับความนิยม (rank) สองแบบด้วยกันคือ

1. ความถี่ของการร้องขอข้อมูลจากผู้ใช้เท่ากันทุก segment หรือเรียกว่าการกระจายความถี่แบบ Uniform เป็นไปตามสมการที่ (1) ซึ่งความต้องการของผู้ใช้ต่อไฟล์วิดีโอทุก ๆ ไฟล์กระจายตัวอย่างเท่าเทียมกันโดยทุกไคลเอนต์จะร้องขอไฟล์วิดีโอพร้อมทั้งเปิดเล่นตั้งแต่ต้นจนจบไฟล์ แต่ละไคลเอนต์จะเริ่มร้องขอไฟล์และเปิดรับชมอย่างสุ่มเวลา
2. การกระจายความถี่ของการร้องขอจากผู้ใช้ต่อ segment ข้อมูลแบบ Zipf-Like ซึ่งจำนวนการร้องของแต่ละ segment ไม่เท่ากันโดยเรียงตัวจากมากไปหาน้อยตามลำดับ segment ของไฟล์วิดีโอของทุกไฟล์ และ segment จำนวน 25% แรกในแต่ละไฟล์จะเป็นกลุ่ม segment ที่อัตราความถี่การร้องขอรวมกันคิดเป็น 75% ของจำนวนการร้องขอทั้งหมดในไฟล์เดียวกัน และกลุ่ม segment ที่เหลือในไฟล์เดียวกันคิดเป็นจำนวน 75% ของจำนวน segment จะมีความถี่ของร้องขอข้อมูลคิดเป็น 25% ของจำนวนการร้องขอทั้งหมดต่อไฟล์นั้น ๆ และในแต่ละไฟล์จะอ้างอิงลำดับความถี่เดียวกันดังสมการที่ (2) โดยในงานวิจัยนี้ จะพิจารณาในกรณีที่ Zipf-like เป็นเชิงเส้น กล่าวคือ ข้อมูลไฟล์

วิดีโอทุก segment จะถูกร้องขอจากไคลเอนต์ทั้งหมด แต่ด้วยความนิยมที่ไม่เท่ากัน และจะไม่มี segment ใดไม่ถูกใช้งานเลย

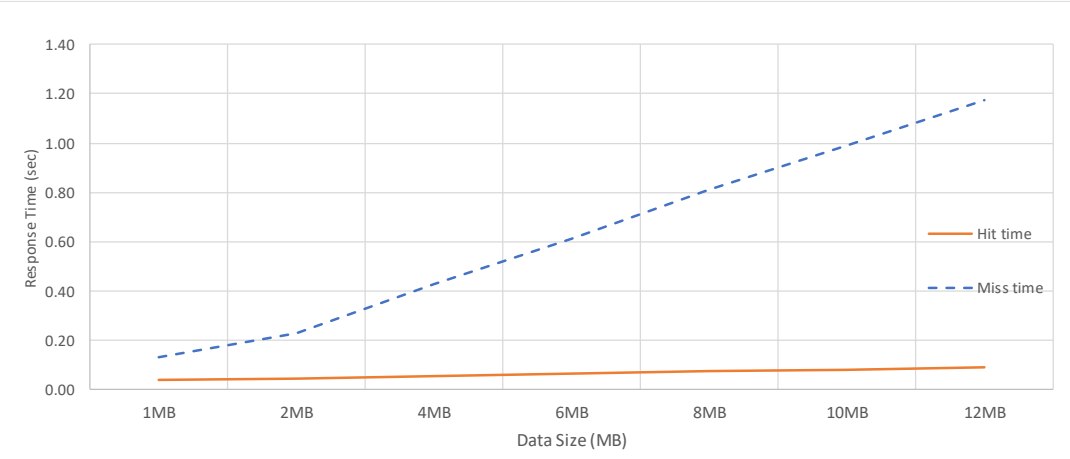
4.3 การเปรียบเทียบประสิทธิภาพการทำงานของระบบ

การทดสอบเพื่อวัดประสิทธิภาพการทำงานของระบบแคชจะวัดค่าความเร็วในหน่วยเวลาของการทำงานของแคชซึ่งใช้ Redis เป็นแคชเก็บข้อมูลโดยการวัดค่าเวลาของขนาดข้อมูลที่แตกต่างกันโดยการกำหนดค่า segment duration ที่ได้จากการร้องขอจากไคลเอนต์จนกระทั่งไคลเอนต์ได้รับข้อมูลตอบกลับจากเซิร์ฟเวอร์โดยเปรียบเทียบช่วงเวลาที่ใช้ในกรณีของการพบข้อมูลในแคช (cache hit) และในกรณีที่ไม่พบข้อมูลในแคช (cache miss)

4.3.1 การวัดประสิทธิภาพของการตอบสนองข้อมูลของ Redis แคช

ในการให้บริการไฟล์วิดีโอแบบ MPEG DASH ด้วยการร้องขอแบบ byte range ทำให้ขนาดข้อมูลที่ไคลเอนต์ร้องขอแต่ละครั้งไม่เท่ากัน ซึ่งทำให้ใช้เวลาในการอ่านข้อมูลแตกต่างกันถือเป็น cost ที่ต้องใช้ในการอ่านข้อมูลของระบบ ผู้วิจัยได้ทดลองเบื้องต้นเพื่อวัดประสิทธิภาพของระบบ โดยเปรียบเทียบเวลาที่ใช้ในการตอบสนองข้อมูลของเซิร์ฟเวอร์ (response time) ซึ่งเป็นระยะเวลาตั้งแต่ไคลเอนต์ร้องขอข้อมูลไปจนกระทั่งไคลเอนต์ได้รับข้อมูลตอบกลับ ทั้งในกรณีพบข้อมูลอยู่ในแคช (cache hit) และกรณีที่ไม่พบข้อมูลในแคช (cache miss) และต้องร้องขอข้อมูลไปยัง backend storage เมื่อขนาดของข้อมูลเปลี่ยนแปลงไป โดยสัมพันธ์กับโดยสัมพันธ์กับคุณภาพของวิดีโอ (bitrate) และ segment duration ที่ใช้ จะมีผลต่อประสิทธิภาพของการทำงานของระบบมากน้อยเพียงใด ซึ่ง response time เป็นการวัดประสิทธิภาพการทำงานของระบบ โดยแยกออกเป็นสองกรณีด้วยพารามิเตอร์ซึ่งเป็นค่าเฉลี่ยในการทดลองเมื่อเปรียบเทียบด้วยขนาดของข้อมูลที่ร้องขอที่เปลี่ยนแปลงไปดังนี้คือ

1. Hit time คือ response time ในการตอบสนองข้อมูลของระบบกรณีพบข้อมูลภายในแคช
2. Miss time คือ response time ในการตอบสนองข้อมูลของระบบกรณีไม่พบข้อมูลภายในแคช



รูปที่ 4.2 เวลาที่ใช้ในการตอบสนองข้อมูลของเซิร์ฟเวอร์เมื่อเทียบกับขนาดข้อมูลในกรณีทั้ง Hit และ Miss

จากรูปที่ 4.2 จะเห็นว่าเวลาเฉลี่ยของการตอบสนองข้อมูลที่ใช้เมื่อขนาดของข้อมูลที่อ่านเปลี่ยนแปลงไปของกรณีพบข้อมูลอยู่ในแคช (hit time) และกรณีที่ไม่พบข้อมูลอยู่ในแคช (miss time) จะเห็นว่าเวลาเฉลี่ยที่ใช้ในการตอบสนองข้อมูลของทั้งสองกรณีมีการเปลี่ยนแปลงที่แตกต่างกัน ในกรณีที่พบข้อมูลอยู่ในแคช เซิร์ฟเวอร์สามารถส่งข้อมูลให้กับไคลเอนต์ได้ทันที ทำให้ใช้ระยะเวลาในการตอบสนองน้อย เมื่อขนาดข้อมูลเพิ่มขึ้นกรณีที่พบข้อมูลอยู่ในแคช ระยะเวลาที่ใช้ในการตอบสนองข้อมูลมีการเปลี่ยนแปลงเพียงเล็กน้อยโดยมีแนวโน้มที่เพิ่มขึ้นด้วยอัตราส่วนที่ต่ำ ในขณะที่กรณีไม่พบข้อมูลอยู่ในแคชระยะเวลาที่ใช้ในการตอบสนองข้อมูลสูงกว่าและมีแนวโน้มที่เพิ่มขึ้นเรื่อย ๆ ตามขนาดข้อมูลที่เปลี่ยนแปลงไป เนื่องจากระยะเวลาที่ใช้ในการตอบสนองข้อมูลในกรณีไม่พบข้อมูลอยู่ในแคช ต้องเพิ่มเวลาในกระบวนการอ่านข้อมูลจาก backend storage ในขณะที่พบข้อมูลในแคชจะไม่มีกระบวนการนี้เกิดขึ้น ทำให้ response ที่ได้แตกต่างกัน หากการร้องขอข้อมูลมีอัตราส่วนของกรณีที่พบข้อมูลอยู่ในแคชสูง จะทำให้มีผลรวมเวลาในการตอบสนองข้อมูลลดลง ทำให้ไคลเอนต์ได้รับข้อมูลรวดเร็วขึ้นและระบบให้บริการไฟล์วิดีโอมีภาระงานน้อยลง เนื่องจากมีการสื่อสารข้อมูลระหว่าง HTTP server กับ backend storage เพื่ออ่านข้อมูลวิดีโอในอัตราส่วนที่ลดลง

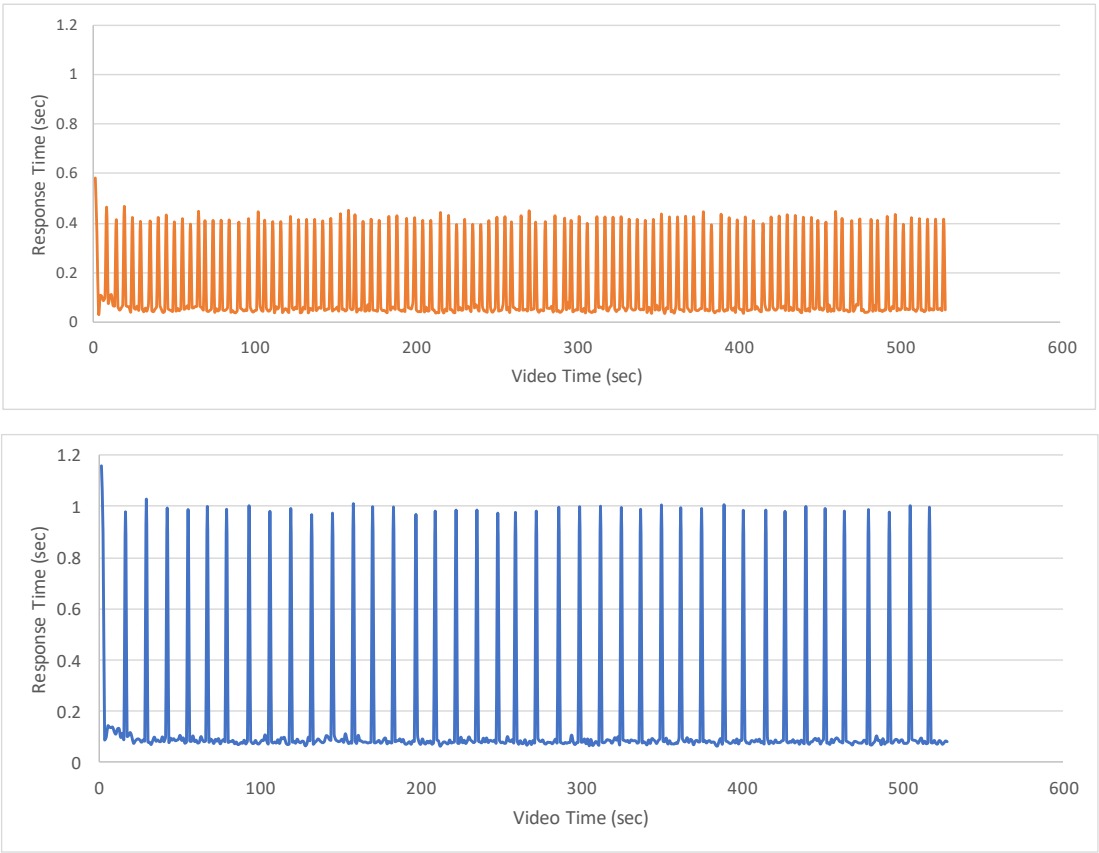
4.3.2 การวัดความเร็วของระบบเมื่อขนาดของ segment duration ต่างกัน

ในการทดลองนี้จะทดสอบความเร็วของการตอบสนองข้อมูลของเซิร์ฟเวอร์ (Response Time) เมื่อคุณภาพของวิดีโอเท่ากัน แต่ระยะของ segment duration ต่างกัน เพื่อดูผลกระทบของขนาด segment duration ต่อการทำงานของระบบ โดยวัดระยะการตอบสนองของระบบตั้งแต่ไคลเอนต์จำนวน 1 เครื่องส่งการร้องขอไปยังเซิร์ฟเวอร์จนกระทั่งไคลเอนต์ได้รับ

segment ข้อมูลตอบกลับจากเซิร์ฟเวอร์ โดยให้ไคลเอนต์จำนวน 1 เครื่องร้องขอไฟล์และเปิดเล่นวิดีโอจากต้นไปกระทั่งจบไฟล์ ซึ่งพารามิเตอร์ในการวัดประสิทธิภาพมีดังนี้

1. Response time
2. Segment duration
3. Video time

ผลการวัดความเร็วของระบบเมื่อขนาดของ segment duration เปลี่ยนแปลงไปดังแสดงในรูปที่ 4.3



รูปที่ 4.3 ผลของขนาด segment duration ที่ 4 วินาที (บน) และ 10 วินาที (ล่าง) ต่อเวลาในการตอบสนองของระบบ

จากรูปที่ 4.3 จะเห็นแกนแนวตั้งเป็นระยะเวลาที่มีหน่วยเป็นวินาทีและแกนแนวนอนคือระยะเวลาเป็นวินาทีของวิดีโอ โดยใช้วิดีโอเดียวกันในการทดสอบ แต่ใช้ segment duration ต่างกันคือ 4 วินาที จะเห็นว่า segment duration ที่นานกว่า (10 วินาที) จะใช้เวลาในการทำงานนานกว่าแต่จำนวนครั้งในการร้องขอข้อมูลจะน้อยกว่าเป็นอัตราส่วนตาม segment duration ที่ต่างกัน

4.4 การวัดประสิทธิภาพการทำงานของนโยบายการแทนที่แคชตามพฤติกรรมของผู้ใช้งาน

เพื่อทดสอบและวัดประสิทธิภาพของแคชที่มีนโยบายการแทนที่แคชที่แตกต่างกันในระบบให้บริการไฟล์วิดีโอแบบ MPEG DASH โดยอ้างอิงพฤติกรรมของผู้ใช้งานเพื่อศึกษาผลกระทบต่อประสิทธิภาพของแคช ผู้วิจัยได้ทำการทดลองโดยแบ่งพฤติกรรมของผู้ใช้งานระบบซึ่งแบ่งออกเป็น 2 แบบดังนี้

1. พฤติกรรมการใช้งานของผู้ใช้แบบ Uniform
2. พฤติกรรมการใช้งานของผู้ใช้เมื่อความต้องการของผู้ใช้กระจายตัวแบบ Zipf-like

ขนาดความจุของแคชที่ใช้ในการทดลองคำนวณเป็นร้อยละของขนาดข้อมูลไฟล์วิดีโอทั้งหมดในระบบซึ่งคิดเป็นจำนวน segment ของไฟล์วิดีโอทั้งหมดในระบบ โดยจะใช้ขนาดของแคชตั้งแต่ 5% ถึง 50% ของจำนวน segment ของไฟล์วิดีโอทั้งหมด และคำนวณขนาดของแคชตามสมการที่ (7) และ (8) โดยเปรียบเทียบผลการทดลองวัดประสิทธิภาพการทำงานของนโยบายการแทนที่แคชเมื่อขนาดของแคชเปลี่ยนแปลงไปทั้งสามนโยบายดังนี้

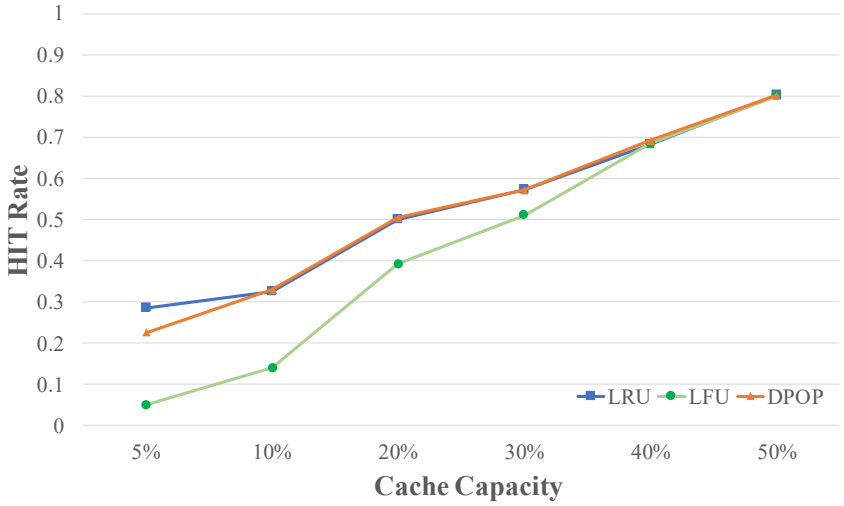
1. LRU (Least Recently Used)
2. LFU (Least Popularity Used)
3. DPOP (Dynamic Popularity)

DPOP หรือ Dynamic Popularity คือนโยบายการแทนที่แคชที่ผู้วิจัยนำเสนอด้วยการลดทอนค่าดัชนีความนิยมของแต่ละ segment ของไฟล์วิดีโอทั้งหมดในระบบ การแทนที่แคชที่ในทั้งสามนโยบายใช้จะพิจารณาข้อมูลในระดับ segment ทั้งหมด พารามิเตอร์ที่ใช้ในการวัดประสิทธิภาพของแคชคืออัตราการพบข้อมูลในแคช (Hit rate) และอัตราการไม่พบข้อมูลในแคช (Miss rate) ซึ่งหากค่า Hit rate สูงและ Miss rate ต่ำ หมายถึงแคชทำงานได้ดีมีประสิทธิภาพสูง ในทางกลับกัน หากค่า Hit rate ต่ำ และ Miss rate สูงหมายถึงแคชมีประสิทธิภาพการทำงานต่ำ โดยปรับเปลี่ยนการใช้งานตามพฤติกรรมการใช้งานของผู้ใช้แบบต่าง ๆ ซึ่งผลการทดลองเป็นดังนี้

4.4.1 ผลการวัดประสิทธิภาพการทำงานของแคชด้วยพฤติกรรมการใช้งานของผู้ใช้แบบ Uniform

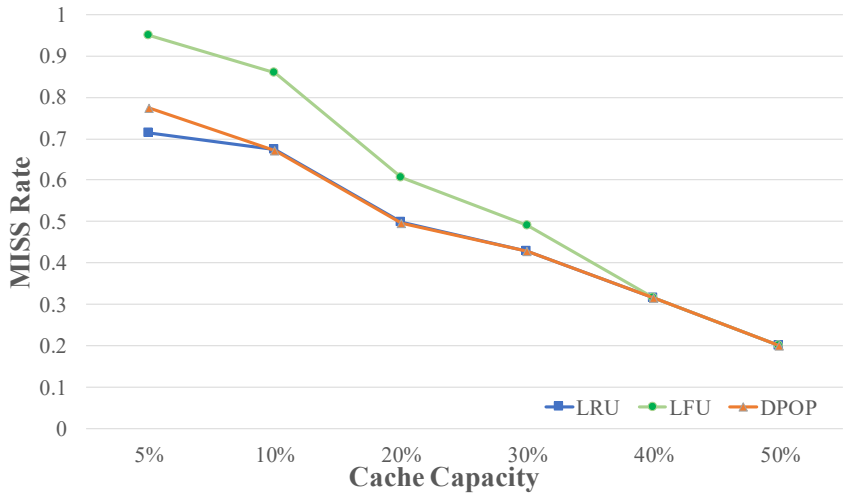
รูปที่ 4.4 จะเห็นว่าจากการทดลองเปรียบเทียบประสิทธิภาพของนโยบายแบบต่าง ๆ ด้วยพฤติกรรมของผู้ใช้งานแบบ uniform หรือในกรณีที่ความนิยมของผู้ใช้งานต่อไฟล์วิดีโอในระบบกระจายตัวอย่างสม่ำเสมอเท่าเทียมกัน ผลที่ได้พบว่านโยบายการแทนที่แคชที่ผู้วิจัยนำเสนอมีประสิทธิภาพเทียบเท่ากับนโยบายการแทนที่แคชแบบ LRU และให้ผลการทำงานที่ดีกว่าเมื่อ

เทียบกับนโยบายการแทนที่แคชแบบ LFU และเมื่อพิจารณาจากขนาดของแคช ทั้งสามนโยบายการแทนที่แคชจะให้ค่า Hit rate ที่สูงขึ้นตามขนาดแคชที่เพิ่มขึ้นและนโยบาย LFU จะให้ Hit rate ที่เท่ากันกับ LRU และ DPOP เมื่อขนาดความจุแคชเท่ากับ 40% ของขนาดข้อมูล ซึ่งอธิบายได้ว่าขนาดของแคชที่เพิ่มขึ้นจะทำให้แคชสามารถรองรับปริมาณการใช้งานระบบและรองรับข้อมูลไฟล์วิดีโอได้มากขึ้น ซึ่งในทางปฏิบัติมักจะใช้แคชเพียง 10%-30% โดยค่าที่มักนำเสนอในงานวิจัยต่างๆ จะอยู่ที่ 20% [41] ของขนาดข้อมูลเนื่องจากแคชมีราคาแพง หากใช้แคชที่มีความจุสูงมากจะให้มีค่าใช้จ่ายที่เพิ่มสูงขึ้น



รูปที่ 4.4 Hit rate ของแคชตามนโยบายการแทนที่แคชในรูปแบบพฤติกรรมการใช้งานแบบ Uniform

จากผลการทดลองในรูปที่ 4.4 พฤติกรรมการใช้งานแบบ Uniform ทำให้ทุก ๆ segment มีความถี่ในการถูกใช้งานเท่ากัน ตามนโยบาย LFU ถือว่าทุก ๆ segment มีความสำคัญเท่ากันทำให้นโยบาย LFU ไม่สามารถลบ segment ที่มีความถี่ต่ำสุดได้อย่างถูกต้อง ส่งผลให้บาง segment ถูกลบออกไปจากแคชและเมื่อ segment ดังกล่าวถูกร้องขอสำหรับใช้งานอีกครั้งการร้องขอนั้นจึงกลายเป็น miss ไปและต้องอ่านเข้ามาใหม่จาก backend storage และเก็บเข้าไปในแคชใหม่อีกครั้งด้วยความถี่ที่เพิ่มขึ้น เหตุการณ์มักเกิดขึ้นบ่อยครั้งเมื่อใช้นโยบาย LFU กับพฤติกรรมการใช้งานแบบ Uniform



รูปที่ 4.5 Miss rate ของแคชตามนโยบายการแทนที่แคชในรูปแบบพฤติกรรมการใช้งานแบบ Uniform

จากรูปที่ 4.5 จะเห็นว่าอัตราส่วนในกรณีไม่พบข้อมูลในแคชหรือ Miss rate จะลดลงเมื่อขนาดของแคชเพิ่มขึ้นในทั้งสามนโยบายการแทนที่แคช ในขณะที่ขนาดของแคชน้อยกว่า 40% นโยบาย LRU และ DPOP จะให้ Miss Rate ที่ต่ำกว่า LFU อย่างชัดเจน ซึ่งค่า miss rate ที่ต่ำหมายถึงแคชมีประสิทธิภาพการทำงานที่สูง

พฤติกรรมของผู้ใช้งานแบบ Uniform จะให้ค่าความถี่ของการเรียกใช้งานแต่ละ segment เท่า ๆ กันในกรณีนี้พบว่านโยบายการแทนที่แคชแบบ LFU มีประสิทธิภาพด้อยกว่านโยบายการแทนที่แคชแบบ LRU และ DPOP ทั้งนี้เมื่อความถี่ของการใช้งานทุก ๆ segment ภายในแคชเท่ากัน นโยบาย LFU จะไม่สามารถจัดการลำดับความสำคัญของแต่ละ segment ได้อย่างถูกต้อง ในขณะที่นโยบาย LRU จะให้ความสำคัญกับลำดับก่อนหลังของการเรียกใช้งานเป็นหลัก ซึ่ง segment ที่ถูกใช้ไปเมื่อนานมาแล้วจะถูกแทนที่ด้วย segment ใหม่เมื่อแคชเต็ม จะทำให้ segment ใหม่ที่สุดจะอยู่ในแคชนานที่สุด ส่วนนโยบาย DPOP จะให้ความสำคัญกับค่าดัชนีความนิยมที่ลดทอนด้วยเวลา ซึ่งในการทดสอบด้วยพฤติกรรมการใช้งานแบบ Uniform นี้ ดัชนีความนิยมของทุก ๆ segment จะมีค่าเท่ากันและให้ผลการจัดลำดับความสำคัญของ segment เช่นเดียวกับการใช้ค่าความถี่แต่เมื่อรวมเข้ากับอัตราการลดทอนของค่าดัชนีความนิยม ซึ่งเปลี่ยนแปลงตามเวลาก่อนหลังทำให้ประสิทธิภาพของนโยบายการการแทนที่แคช DPOP ให้ผลเช่นเดียวกับนโยบายการแทนที่แคชแบบ LRU และดีกว่า LFU เมื่อเทียบกับขนาดของแคชที่เท่ากัน ซึ่งค่าความถี่ของนโยบาย LFU จะไม่ลดค่าความถี่ลง ทำให้ segment ที่มีความถี่ของการใช้งานสูงเมื่อนานมาแล้วจะเป็น

segment ที่อยู่ในแคชนานที่สุดแม้เว้นว่างจากการใช้งานเป็นระยะเวลานาน ซึ่งเป็นข้อด้อยของการทำงานของนโยบาย LFU ในระบบการให้บริการไฟล์วิดีโอ

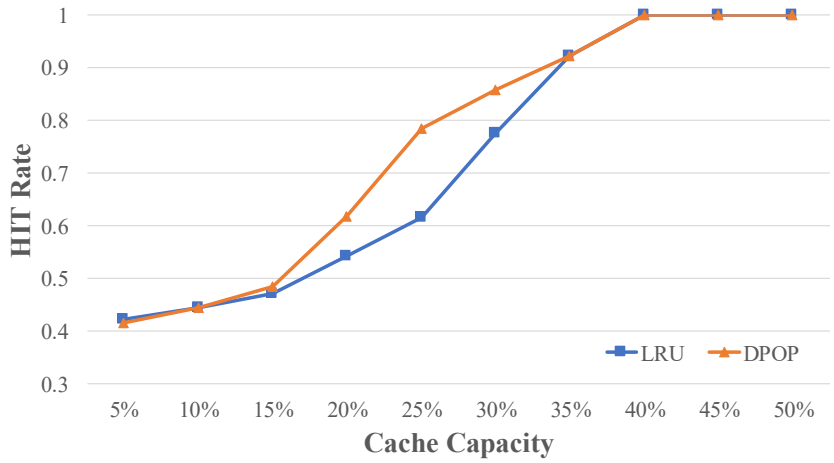
ทั้งนี้เป็นผลมาจากพฤติกรรมผู้ใช้งานแบบ Uniform ที่มีการแข่งขันของ segment เพื่อเข้ามาอยู่ในแคชนั้นสูง เพราะทุก ๆ segment จะมีความถี่ที่เท่ากัน นั่นคือมีความต้องการจากผู้ใช้งานเท่ากัน ทำให้ผลการคำนวณค่าความนิยมของนโยบาย DPOP ให้ค่าที่เท่ากันใน segment ที่มีความถี่เท่ากันในเวลาเริ่มต้น แต่ด้วยค่าการลดทอนดัชนีความนิยมทำให้ ความสำคัญของ segment ต่างกันในขณะที่ความถี่ของการถูกใช้งานสะสมเท่ากัน ซึ่งทำให้นโยบาย DPOP มีประสิทธิภาพที่ดีกว่านโยบาย LFU ในพฤติกรรมการใช้งานแบบ Uniform

ในกรณีที่แคชมีขนาดความจุที่ 5%-10% นโยบาย DPOP มีประสิทธิภาพที่สูงกว่า LFU แต่ก็ยังต่ำกว่า LRU เล็กน้อย ทั้งนี้เป็นเพราะค่าความถี่ที่เท่ากันของ segment นั่นเอง ซึ่งในขนาดความจุแคชต่ำทำให้ความเข้มข้นของการแข่งขันกันอยู่ในแคชของ segment ที่มีความนิยมใกล้เคียงกันสูงหรือแออัดกันมากเกินไปนั่นเอง ปัจจัยอีกประการหนึ่งคือการเข้าใช้งานแบบสุ่มนั้นมีความน่าจะเป็นที่ segment เดียวกันจะถูกเรียกใช้ในระยะเวลาใกล้เคียงกันมีค่อนข้างสูง ทำให้ที่ขนาดแคชต่ำ พารามิเตอร์ความล่าสุด (recency) ของนโยบาย LRU ยังคงใช้งานได้ดี

4.4.2 การวัดประสิทธิภาพการทำงานของแคชด้วยพฤติกรรมการใช้งานของผู้ใช้แบบ Zipf-Like

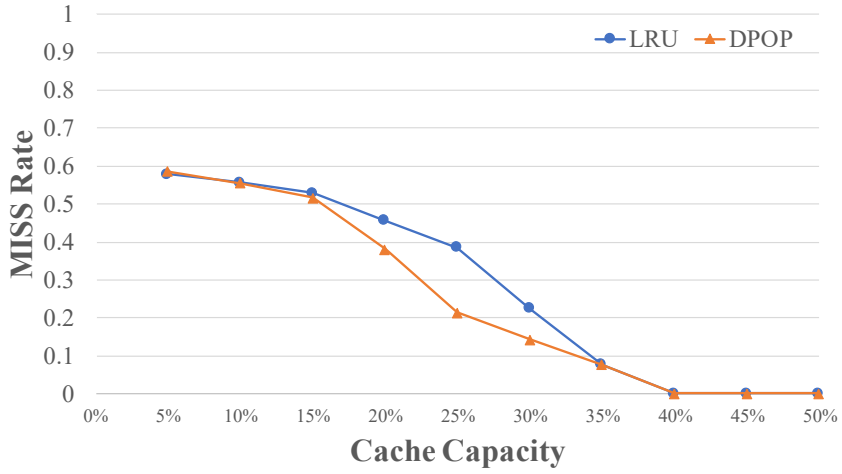
ในการทดลองนี้จะเปรียบเทียบเพียงสองนโยบายการแทนที่แคชคือ LRU และ DPOP เพื่อให้เห็นถึงผลกระทบของการใช้งานดัชนีความนิยมและการลดทอนของดัชนีความนิยมในนโยบายการแทนที่แคชที่ผู้วิจัยนำเสนอเปรียบเทียบกับการใช้ค่าความใหม่หรือการใช้งานล่าสุดของ LRU ซึ่งผลการทดลองเป็นดังนี้

ในรูปที่ 4.6 จะเห็นว่าเมื่อขนาดความจุของแคชน้อยกว่า 15% ประสิทธิภาพของนโยบายการแทนที่แคช DPOP จะเทียบเท่ากับประสิทธิภาพการทำงานของนโยบายการแทนที่แคช LRU และจะเริ่มมี hit rate สูงกว่าที่ความจุแคชตั้งแต่ 15% ไปจนถึง 35% และจะให้ผลที่เท่ากันเมื่อขนาดความจุแคชสูงกว่า 35% โดยนโยบายการแทนที่แคช DPOP ให้ประสิทธิภาพดีที่สุดเมื่อเทียบกับนโยบายแทนที่แคช LRU ที่ขนาดความจุแคชเป็น 25% ของขนาดข้อมูลทั้งหมดที่ใช้ทดสอบระบบ



รูปที่ 4.7 Hit rate ของแคชตามนโยบายการแทนที่แคชในรูปแบบพฤติกรรมการใช้งานแบบ Zipf-Like

ในรูปที่ 4.7 จะเห็นว่า miss rate จะต่ำลงเมื่อขนาดความจุของแคชเพิ่มขึ้น โดยนโยบาย DPOP จะมี miss rate ที่ต่ำกว่า LRU ที่ขนาดของแคชอยู่ระหว่าง 15% ถึง 35% ซึ่งค่า miss rate ที่ต่ำจะทำให้ค่า hit rate สูงซึ่งส่งผลให้แคชมีประสิทธิภาพการทำงานสูง



รูปที่ 4.6 Miss rate ของแคชตามนโยบายการแทนที่แคชในรูปแบบพฤติกรรมการใช้งานแบบ Zipf-Like

สาเหตุที่ทำให้ประสิทธิภาพของนโยบาย DPOP ที่ผู้วิจัยนำเสนอให้ผลที่ดีกว่า LRU คือการอ้างอิงด้วยค่าดัชนีความนิยมที่เกิดขึ้นจริงของ segment ที่อยู่ในแคชและให้ความสำคัญกับอายุ (age) ของค่าดัชนีความนิยมที่มีการเปลี่ยนแปลงอยู่ตลอดเวลา segment ใดที่มีความนิยมสูง

และถูกร้องขอจากผู้ใช้ตลอดเวลา จะเป็น segment ที่อยู่ในแคชนานที่สุด แต่หาก segment ใดไม่ได้ถูกร้องขอเป็นเวลานานทั้ง ๆ ที่มีความถี่สะสมในการเรียกใช้งานบ่อยครั้งที่สุด จะไม่ถือว่าเป็น segment ที่มีความนิยมสูงสุดในเวลาปัจจุบัน segment ดังกล่าวอาจจะถูกแทนที่ในเวลาถัดมา ซึ่งเป็นการให้ค่าความสำคัญของดัชนีความนิยมที่เปลี่ยนแปลงไปตามเวลาด้วยอัตราการลดทอนที่แตกต่างกัน ในขณะที่นโยบายแทนที่แคช LRU ไม่ได้ให้ความสำคัญกับดัชนีความนิยมแต่จะให้ความสำคัญกับ segment ที่เพิ่งถูกร้องขอไปในเวลาล่าสุดถือเป็น segment ที่จะอยู่ในแคชนานที่สุด แต่ในความเป็นจริงพฤติกรรมการใช้งานของผู้ใช้ระบบให้บริการไฟล์วิดีโอนั้นเป็นพฤติกรรมที่อธิบายโดยความนิยมของการใช้งานไฟล์วิดีโอเป็นหลัก ซึ่งเมื่อเวลาผ่านไปค่าความนิยมเดิมจะมีความสำคัญน้อยลงเมื่อเทียบกับค่าความนิยมของการใช้งานที่มีต่อไฟล์ใหม่ที่เพิ่งเข้ามาในระบบ ซึ่งจะมีผลอย่างยิ่งต่อการทำงานของระบบ

บทที่ 5
สรุปผลการวิจัย อภิปรายผล และข้อเสนอแนะ

เนื้อหาในบทนี้จะกล่าวสรุปผลการวิจัยที่ได้นำเสนอในวิทยานิพนธ์นี้ รวมทั้งข้อเสนอแนะต่าง ๆ ที่เกี่ยวข้องกับงานวิจัยด้านการออกแบบและสร้างแคชในระบบวิดีโอ streaming ผ่านเครือข่ายอินเตอร์เน็ต

5.1 สรุปผลการวิจัย

ไฟล์วิดีโอเป็นไฟล์ข้อมูลขนาดใหญ่ที่เรียงตัวต่อเนื่องกัน ซึ่งต้องใช้ทรัพยากรระบบสูงเพื่อทำให้ระบบคอมพิวเตอร์ที่ให้บริการสามารถรองรับการให้บริการได้อย่างมีประสิทธิภาพ การให้บริการไฟล์วิดีโอดังกล่าวจึงต้องคำนึงถึงปัจจัยสำคัญหลายประการ

แคชถูกนำมาใช้เพื่อให้การตอบสนองของระบบเร็วขึ้น แต่การจัดการจะต้องมีประสิทธิภาพ ขนาดไฟล์วิดีโอมักมีขนาดใหญ่และผู้ใช้มักจะเปิดเล่นไฟล์อย่างต่อเนื่อง ระบบที่ให้บริการไฟล์วิดีโอมีรูปแบบการให้บริการไฟล์ผ่านโปรโตคอล HTTP โดยส่งข้อมูลในลักษณะ streaming โดยไคลเอนต์แบ่งการร้องขอออกเป็นส่วน ๆ และเซิร์ฟเวอร์ก็ส่งข้อมูลให้กับไคลเอนต์เป็นส่วนเล็ก ๆ ด้วยเช่นกัน วิธีการสื่อสารข้อมูลระหว่างไคลเอนต์และเซิร์ฟเวอร์ในลักษณะนี้เรียกว่า HTTP Streaming ซึ่งในปัจจุบันประกอบด้วยวิธีการและมาตรฐานการสื่อสารข้อมูล เช่น HDS, HLS และ MPEG-DASH การทำงานและรายละเอียดได้กล่าวไว้แล้วในบทที่ 2 ซึ่งแต่ละมาตรฐานจะมีรายละเอียดที่แตกต่างกัน ผู้วิจัยได้ทดสอบกับระบบการทำงานที่อ้างอิงมาตรฐาน MPG-DASH ซึ่งแบ่งข้อมูลไฟล์วิดีโอออกเป็นส่วนย่อย ๆ ในการสื่อสารข้อมูลระหว่างเซิร์ฟเวอร์และไคลเอนต์

พฤติกรรมการใช้งานระบบเป็นอีกปัจจัยหนึ่งที่สำคัญ ซึ่งระบบให้บริการไฟล์วิดีโอผ่าน HTTP นั้น มีลักษณะที่แตกต่างจากการใช้งานไฟล์แบบอื่น แต่ในระบบที่ให้บริการไฟล์วิดีโอแตกต่างออกไป ค่าความนิยม (Popularity) ที่มีต่อไฟล์วิดีโอนั้นยิ่งจะมีผลอย่างยิ่งต่อประสิทธิภาพการทำงานของระบบ ประกอบกับขนาดไฟล์ที่ใหญ่และเป็นข้อมูลต่อเนื่อง จำเป็นอย่างยิ่งที่จะต้องพัฒนาระบบแคชและนโยบายการแทนที่แคชที่ทำงานสอดคล้องกับลักษณะการใช้งานของระบบ

ในงานวิจัยนี้ ผู้วิจัยได้นำเสนอระบบแคชและนโยบายการแทนที่แคช ที่ให้ความสำคัญกับพฤติกรรมการทำงานของผู้ใช้ที่มีผลต่อระบบ โดยเฉพาะอย่างยิ่งผลกระทบต่อแคชที่แตกต่างออกไปเมื่อปรับใช้นโยบายการแทนที่แคชที่ให้ความสำคัญต่อค่าดัชนีความนิยม โดยได้ทำการออกแบบแคชและนโบายแทนที่แคชให้ทำงานร่วมกับ video streaming แบบ MPEG-DASH ซึ่งให้ความสำคัญของข้อมูลในระดับ segment ย่อยของไฟล์วิดีโอ และได้ทดสอบด้วยพฤติกรรมของผู้ใช้งานสองแบบด้วยกันคือ

1. ความนิยมแบบกระจายตัวต่อ segment ของทุกไฟล์วิดีโออย่างเท่าเทียมกัน หรือการกระจายค่าดัชนีความนิยมแบบ Uniform
2. ความนิยมกระจายตัวต่อ segment ของทุกไฟล์วิดีโอแบบ Zipf-like ซึ่งเป็นการกระจายตัวของความต้องการใช้งานไฟล์วิดีโอในการปฏิบัติจริง

ผู้วิจัยได้ทดลองเปรียบเทียบผลของนโยบายการแทนที่แคชที่ผู้วิจัยนำเสนอ กับนโยบายการแทนที่แคชแบบ LRU และ LFU ด้วยโครงสร้างระบบและข้อมูลไฟล์วิดีโอเดียวกันด้วยพฤติกรรมของผู้ใช้งานตามความนิยมทั้งสองแบบ

นโยบายการแทนที่แคชที่เหมาะสมกับพฤติกรรมการใช้งานไฟล์วิดีโอจะช่วยกลั่นกรองข้อมูลที่ได้รับความนิยมสูง เก็บไว้ในแคชเพื่อให้ผู้ใช้งานที่มีความต้องการใช้ข้อมูลชุดเดียวการได้รับข้อมูลอย่างรวดเร็วและลดภาระงานของระบบให้บริการได้อย่างมีประสิทธิภาพ เพิ่มอัตราการพบข้อมูลในแคชหรือ hit rate ซึ่งผู้ใช้จะสามารถได้รับข้อมูลรวดเร็วขึ้น และลดอัตราการอ่านข้อมูลจาก backend storage หรือ miss rate

ผู้วิจัยได้ออกแบบระบบและสร้างระบบให้บริการไฟล์วิดีโอโดยยึดมาตรฐานการสื่อสารข้อมูล streaming แบบ MPEG-DASH และพัฒนาต่อยอดระบบ HTTP server ของ Nginx ให้สามารถทำงานได้โดยการเพิ่มแคชภายนอกระบบของ Nginx โดยเพิ่มส่วนของโปรแกรมด้วยภาษา Lua เข้าไปแทรกในแต่ละเฟส (phase) การทำงานของ Nginx ซึ่งทำงานโดยการสร้างเทรดจำนวนมากเพื่อรองรับการทำงานของทั้งการร้องขอจากไคลเอนต์และการทำงานย่อยภายใน Nginx เอง โดยคำนึงถึงภาวะความพร้อมกัน (Concurrency) และความสอดคล้องกัน (Consistency) ทั้งในการทำงานของเทรดและข้อมูลระบบ

จากผลการทดลองที่ได้พบว่านโยบายการแทนที่แคชที่อ้างอิงพฤติกรรมของผู้ใช้งานตามความนิยมของไฟล์และในระดับ segment นั้น พบว่าค่าความนิยมส่งผลต่อการทำงานของแคชโดยเฉพาะอย่างยิ่งพฤติกรรมของผู้ใช้งานแบบ Zipf-Like จะเห็นได้จากการทดลอง โดย

ให้ผลอัตราของการพบข้อมูลอยู่ในแคชที่สูงกว่านโยบายแคชแบบ LRU เมื่อพิจารณาในในกรณีขนาดความจุแคชที่เท่ากัน และให้ผลเทียบเท่ากับ LRU และสูงกว่า LFU ในการทดลองด้วยค่าความนิยมของข้อมูลในแคชจากผู้ใช้ที่เท่าเทียมกันหรือแบบ Uniform

5.2 อภิปรายผลการวิจัย

จากผลการทดลองพบว่าดัชนีความนิยมเนื่องมาจากพฤติกรรมของผู้ใช้มีผลอย่างยิ่งต่อการทำงานของระบบ ซึ่งการใช้งานระบบให้บริการไฟล์วิดีผ่านโปรโตคอล HTTP เช่น Netflix, Hulu หรือ YouTube มีรูปแบบการใช้งานตามการกระจายตัวของความนิยมแบบ Zipf-Like ทั้งสิ้น เพื่อให้เห็นข้อแตกต่างของผลกระทบความนิยมที่มีผลต่อการให้บริการไฟล์วิดีโอ ผู้วิจัยได้ทำการทดลองวัดประสิทธิภาพของการทำงานของแคชที่ปรับใช้การคำนวณค่าความนิยมดังแสดงในผลทดลองในบทที่ 4

เมื่อเปรียบเทียบผลการทดลองที่ได้ระหว่างประสิทธิภาพของนโยบายแคชทั้งสามนโยบายในการกระจายความนิยมแบบ Uniform ของผู้ใช้งานต่อไฟล์วิดีโอในระบบในระดับ segment โดยการวัดประสิทธิภาพด้วยอัตราการพบข้อมูลในแคชในขนาดความจุแคชต่าง ๆ พบว่าพฤติกรรมของผู้ใช้งานแบบ Uniform นั้นให้ค่าความถี่ของการถูกเรียกใช้งานของทุก ๆ segment เท่ากัน

นโยบายแคชที่ผู้วิจัยนำเสนอให้ผลการการพบข้อมูลอยู่ในแคชที่อัตราใกล้เคียงกับนโยบายแคช LRU โดยในขนาดพื้นที่แคชน้อย นโยบาย LRU ให้ผลที่ดีกว่าเล็กน้อย ทั้งนี้เป็นผลมาจาก LRU จะให้ความสำคัญค่าของความล่าสุด (recency) ซึ่งในกรณีขนาดพื้นที่แคชน้อยนั้น พื้นที่ไม่เพียงพอต่อความต้องการจัดเก็บข้อมูลที่มีค่าความนิยมกลุ่มของ segment สูง ซึ่งพฤติกรรมการใช้งานแบบ Uniform จะทำให้ทุก ๆ segment มีความถี่ของการเรียกใช้งานเท่ากัน ทำให้ segment ในกลุ่มที่มีความนิยมสูงมีจำนวนมาก ทำให้ความจุแคชไม่เพียงพอต่อความต้องการเมื่อเทียบกับนโยบายแคช LRU แต่ด้วยกระบวนการลดทอนค่าดัชนีความนิยมของนโยบาย DPOP ทำให้ segment ที่มีความถี่เท่ากันมีค่าดัชนีความนิยมที่ถูกลดทอนแล้วไม่เท่ากัน ส่งผลให้นโยบาย DPOP สามารถจัดลำดับความสำคัญของ segment ได้ดีและมีประสิทธิภาพต่ำกว่านโยบาย LRU เพียงเล็กน้อยที่ขนาดความจุแคชต่ำ

แต่เมื่อเทียบกับขนาดความจุแคชที่สูงขึ้นพบว่านโยบายการแทนที่แคชที่ผู้วิจัยนำเสนอให้ผลที่เทียบเท่านโยบายแทนที่แคชแบบ LRU ทั้งนี้เนื่องจากอัตราการลดทอนที่นำเสนอ

นั้น ช่วยให้ความนิยมเดิมที่เท่ากันทุก ๆ segment ของข้อมูลที่อยู่ในแคช ปรับเปลี่ยนค่าไม่เท่ากันขึ้นอยู่กับระยะเวลาที่อยู่ในแคช และเมื่อเปรียบเทียบผลการวัดประสิทธิภาพกับนโยบาย LFU พบว่าผลของนโยบายที่ผู้วิจัยนำเสนอให้ประสิทธิภาพที่ดีกว่าด้วยค่า hit rate ที่สูงกว่า ทั้งนี้เนื่องจากนโยบาย LFU ให้ความสำคัญกับค่าความถี่ของการถูกใช้งานข้อมูลภายในแคชเพื่อคัดเลือกว่าข้อมูลหรือ segment ใดควรจะอยู่ในแคชเมื่อเทียบค่าความถี่ ซึ่งพฤติกรรมการใช้งานของผู้ใช้แบบ Uniform นั้นทำให้ความถี่ของการถูกใช้งานของทุก ๆ segment ที่อยู่ในแคชมีค่าเท่ากัน ซึ่งส่งผลให้ segment ที่อยู่ในแคชที่ใช้นโยบาย LFU มีค่าความถี่ที่เท่ากันและมีการแข่งขันเพื่อจะเข้าไปอยู่ในแคชที่สูง ทำให้แคช LFU ไม่สามารถระบุได้ว่า segment ใดสำคัญกว่ากัน และในกรณีที่ segment ใหม่เพิ่งถูกเขียนไปในแคชครั้งแรก ความถี่เริ่มต้นมีค่าต่ำสุดเมื่อเทียบกับ segment อื่น ๆ ในแคชที่มีอยู่เดิม ทำให้ segment ใหม่ดังกล่าวอาจถูกลบออกไปจากแคชในระยะเวลาสั้น และเมื่อมีการร้องขอ segment ดังกล่าวอีกครั้ง กลับพบว่าไม่ได้อยู่ในแคชแล้วและมี miss rate ที่เพิ่มสูงขึ้น ซึ่งทำให้นโยบาย LFU มีประสิทธิภาพที่ต่ำกว่านโยบายแทนที่แคช DPOP ที่ผู้วิจัยนำเสนอและนโยบาย LRU เมื่อใช้ในรูปแบบพฤติกรรมผู้ใช้งานแบบ Uniform และเมื่อขนาดความจุแคชมีขนาดใหญ่มาก ทุกนโยบายการแทนที่แคชจะมีประสิทธิภาพที่สูงเทียบเท่ากัน เนื่องจากขนาดความจุแคชที่ใหญ่ขึ้นเมื่อเทียบกับขนาดข้อมูลทั้งหมดทำให้แคชมีพื้นที่เพียงพอรองรับความต้องการในการใช้งาน ผลที่ได้คือจะให้ค่า hit rate ที่สูงและ miss rate ที่ต่ำในทุกนโยบาย โดยขนาดความจุของแคชสามารถคำนวณได้จากสมการที่ (7) ในบทที่ 3 และขนาดของแคชที่ให้ประสิทธิภาพดีตอบสนองต่อพฤติกรรมแบบ Uniform คือ 20%-40% ของขนาดข้อมูลทั้งหมดในระบบ

เมื่อเปรียบเทียบผลการวัดประสิทธิภาพการทำงานของนโยบายแทนที่แคชที่ผู้วิจัยนำเสนอกับนโยบาย LRU ในพฤติกรรมการใช้งานของผู้ใช้ไฟล์วิดีในลักษณะ Zipf-Like พบว่านโยบายที่ผู้วิจัยนำเสนอให้ผลการทำงานด้วยประสิทธิภาพสูงกว่านโยบาย LRU ในขนาดความจุแคชระหว่าง 15% - 35% โดยในขนาดความจุแคชที่ต่ำกว่า 15% นโยบายการแทนที่แคชที่ผู้วิจัยนำเสนอและนโยบาย LRU มีประสิทธิภาพที่ใกล้เคียงกัน เนื่องจากขนาดของแคชที่น้อยนั้นไม่สามารถรองรับความต้องการของระบบได้เมื่อเทียบกับขนาดของข้อมูลไฟล์วิดีโอทั้งหมดและทำให้แคชมีอัตราการนำเข้าและการลบออกมากจนเกินไป ทำให้ hit rate น้อยและ miss rate สูง และประสิทธิภาพที่ต่ำ

เมื่อขนาดความจุของแคชสูงขึ้นในช่วง 15% - 35% ดังกล่าวจะเห็นความแตกต่างของประสิทธิภาพของนโยบายการแทนที่แคชทั้งสองอย่างชัดเจน เนื่องจากขนาดมีความจุแคชที่เหมาะสม และนโยบายแทนที่แคชที่ผู้วิจัยนำเสนอได้นำหลักการการคำนวณค่าดัชนีความนิยมมาใช้

พร้อมทั้งเพิ่มความสามารถในการลดทอนค่าดัชนีความนิยม จากผลการวัดประสิทธิภาพผลที่ได้คือ นโยบายการแทนที่แคชที่ผู้วิจัยนำเสนอมี hit rate ที่สูงกว่านโยบาย LRU เมื่อเปรียบเทียบที่ขนาดความจุแคชที่เท่ากัน ซึ่งที่ขนาดความจุแคชต่ำกว่านโยบาย DPOP ที่ผู้วิจัยนำเสนอสามารถให้ประสิทธิภาพที่สูงกว่านโยบาย LRU ที่ขนาดแคชสูงกว่า และที่ขนาดความจุแคชตั้งแต่ 35% เป็นต้นไปการวัดประสิทธิภาพของทั้งสองนโยบายให้ผลเช่นเดียวกัน เนื่องจากขนาดความจุแคชเพียงพอสามารถรองรับปริมาณข้อมูลและจำนวนการร้องขอไฟล์จากผู้ใช้งานได้

การออกแบบแคชและนโยบายการแทนที่แคชนั้นจำเป็นต้องคำนึงถึงขนาดข้อมูลและขนาดความจุแคชด้วย ซึ่งจากผลการทดลองวัดประสิทธิภาพนโยบายการแทนที่แคชนั้นพบว่าขนาดความจุแคชมีผลต่อประสิทธิภาพการทำงานของแคช ซึ่งขนาดความจุแคชที่เหมาะสมจะทำให้ระบบทำงานได้ดีรวมถึงนโยบายการแทนที่แคชก็จะทำงานได้อย่างมีประสิทธิภาพด้วย โดยปกติขนาดความจุแคชที่มักใช้กันจะอยู่ที่ 20% ของขนาดข้อมูลทั้งหมดในระบบ จากผลการทดสอบระบบพบว่า พฤติกรรมการใช้งานส่งผลต่อประสิทธิภาพของนโยบายการแทนที่แคชในขนาดความจุแคชที่แตกต่างกัน ซึ่งจะเห็นว่าในพฤติกรรมการใช้งานแบบ Uniform ขนาดความจุแคชที่ตอบสนองต่อนโยบาย DPOP และ LRU ได้ดีคือ 20% - 40% และนโยบาย LFU ที่ 30% - 40% สำหรับพฤติกรรมการใช้งานแบบ Zipf-Like นั้นขนาดความจุแคชที่ตอบสนองนโยบาย LRU ได้ดีคือ 25% - 35% และสำหรับนโยบาย DPOP ขนาดแคชควรจะอยู่ที่ 20% - 35% ทั้งนี้ขึ้นอยู่กับการกระจายตัวของความนิยมจากการใช้งาน ซึ่งสอดคล้องกับค่าตัวแปร α ในสมการที่ (2)

จากผลการทดลองพบว่าที่ขนาดความจุแคช 15% - 25% นโยบายอทนที่แคชที่ผู้วิจัยนำเสนอทำให้ค่า hit rate มีอัตราเพิ่มสูงขึ้นมากเมื่อเทียบกับที่ขนาดความจุแคชระหว่าง 25% - 35% ซึ่งค่าของ hit rate เพิ่มขึ้นด้วยอัตราที่ลดลง ดังนั้นนโยบายการแทนที่แคชที่ผู้วิจัยนำเสนอจะทำงานได้ดีและให้อัตราการเพิ่มขึ้นของ hit rate สูงสุดหรือที่ขนาดความจุแคชที่ 25% ของขนาดข้อมูลทั้งหมด โดยนโยบายแทนที่แคช DPOP ให้ผลที่ดีกว่านโยบาย LRU 1.28 เท่าหรือนโยบายการแทนที่แคช DPOP ให้ประสิทธิภาพการทำงานที่ดีกว่านโยบายการแทนที่แคช LRU คิดเป็น 28%

นโยบายการแทนที่แคชที่ผู้วิจัยนำเสนอสามารถทำงานสอดคล้องรับกับพฤติกรรมการใช้งานแบบ Zipf-Like ที่ผู้ใช้จำนวนมากจะเข้าใช้งานไฟล์วิดีโอที่มีความนิยมสูงและผู้ใช้จำนวนน้อยจะกระจายเข้าใช้งานไฟล์วิดีโอส่วนที่เหลือซึ่งมีจำนวนไฟล์มาก ทำให้ไฟล์จำนวนมากมีค่าความนิยมต่ำและไฟล์จำนวนน้อยมีค่าความนิยมสูง ซึ่งความนิยมในระดับ segment ของแต่ละไฟล์ก็เรียงตัวเช่นเดียวกัน กล่าวคือผู้ใช้จำนวนมากมักเปิดไฟล์วิดีโอเพื่อรับชมเฉพาะบางช่วงของ

ไฟล์ที่ตนเองสนใจ โดยเฉพาะอย่างยิ่งช่วงต้นของไฟล์วิดีโอ ในกรณีของ MPEG-DASH ซึ่งแยกไฟล์ออกเป็น segment ย่อย ๆ โดย segment แรก ๆ ของไฟล์วิดีโอจะถูกเข้าใช้งานบ่อยกว่า segment อื่น ๆ

เมื่อ segment ใดมีค่าความนิยมสูงก็จะอยู่ในแคชด้วยระยะเวลาที่นานกว่า segment อื่น ๆ แต่หาก segment อยู่ในแคชเป็นระยะเวลานานด้วยค่าดัชนีความนิยมสูงโดยไม่ถูกเรียกใช้เป็นระยะเวลานาน ค่าดัชนีความนิยมของ segment นั้นก็จะถูกลดทอนด้วยอัตราที่สูงด้วยเช่นกัน ด้วยค่า $\Delta time$ ซึ่งหากผ่านไปเป็นระยะเวลานาน segment ดังกล่าวก็จะถูกแทนที่ด้วย segment ใหม่ไปในที่สุด เนื่องจากค่าดัชนีความนิยมที่ถูกลดทอนแล้วมีค่าต่ำ

ในภาพรวมในงานวิจัยนี้ ผู้วิจัยได้นำเสนอองค์ความรู้ดังต่อไปนี้

1. โครงสร้างระบบการให้บริการไฟล์วิดีโอ streaming
2. นโยบายแทนที่แคชโดยอ้างอิงค่าดัชนีความนิยมและการลดทอนตามระยะเวลา
3. แนวทางการออกแบบแคชและคำนวณหาขนาดความจุแคชที่เหมาะสมจากรูปแบบพฤติกรรมการใช้งานของระบบจากการกระจายตัวความค่าความนิยม ทั้งนี้ต้องมีข้อมูลเพียงพอต่อการพิจารณา

5.3 ข้อเสนอแนะ

ผู้วิจัยพบว่ามีรายละเอียดบางอย่างที่ต้องกล่าวถึงและเสนอแนะเพื่อให้เป็นแนวทางในการทำงานวิจัยเกี่ยวกับ HTTP streaming

5.3.1 ข้อเสนอแนะการเขียนโปรแกรมภาษา Lua เพื่อใช้งานกับ Nginx โดยใช้ไลบรารีของ OpenResty

การเขียนโปรแกรมภาษา Lua เพื่อควบคุมการทำงานของ HTTP server ดังที่ผู้วิจัยใช้กับ Nginx นั้น ๆ มีรายละเอียดมากมาย ซึ่งจำเป็นต้องทำความเข้าใจการทำงานของ Nginx โดยละเอียดพร้อมศึกษาคู่มือการใช้งาน OpenResty และไลบรารีของภาษา Lua ในแต่ละระดับการใช้งานโดยคำนึงถึงระดับโครงสร้างการทำงานของโปรแกรมด้วย

5.3.2 การวัดประสิทธิภาพระบบโดยใช้โปรแกรมจำลองสร้างภาระงานให้กับระบบ (Workload tester)

การทดลองในการทำวิจัยเกี่ยวกับระบบให้บริการไฟล์วิดีโอต้องใช้เวลาในการทดสอบนาน ไฟล์วิดีโอมักมีขนาดใหญ่และต้องเปิดไฟล์อย่างต่อเนื่อง ประกอบกับการทดสอบและวัดประสิทธิภาพเพื่อวัดผลความนิยม ซึ่งเป็นการวัดผลจากพฤติกรรมการใช้งานของผู้ใช้งานจำนวนมาก เพื่อวัดผลที่มีต่อระบบ ยิ่งต้องใช้ระยะเวลานานในการทดสอบ โดยในการทดสอบทุกครั้งผู้วิจัยได้ทดสอบโดยจำลองการเปิดใช้งานไฟล์วิดีโอด้วยโปรแกรม VLC และ Google Chrome ด้วยคอมพิวเตอร์ที่ทำหน้าที่เป็นไคลเอนต์ในระบบ ซึ่งทำให้การทดสอบได้ไม่คล่องตัวนัก ในโอกาสต่อไปควรใช้วิธีการอื่นร่วมในการทดสอบระบบเพิ่มเติม ซึ่งวิธีการหนึ่งคือการวัดประสิทธิภาพระบบโดยใช้โปรแกรมจำลองสร้างภาระงานในการทดสอบระบบซึ่งสามารถทำได้ดังนี้

1. ใช้โปรแกรมสร้าง workload จำลองเพื่อทดสอบ เช่น wrk [54] แต่จะได้ประสิทธิภาพของระบบในอีกมุมหนึ่งคือ จำนวนการร้องขอที่ระบบสามารถรับได้ต่อเวลา (request/second) และ ค่าเวลาแฝง (latency time)
2. ใช้โปรแกรมทดสอบระบบ JMeter [53] ได้ซึ่งเป็นโปรแกรมประยุกต์ที่เป็น open-source โปรแกรม JMeter สามารถนำมาประยุกต์ใช้กับการทดสอบระบบให้บริการไฟล์วิดีโอตามมาตรฐาน HLS ซึ่งให้การวัดประสิทธิภาพดี และได้มีผู้พัฒนาส่วนขยายเพิ่มเติม (Adds on) ใช้ในโปรแกรม JMeter เพื่อทดสอบวัดประสิทธิภาพของการให้บริการไฟล์วิดีโอตามมาตรฐาน MPEG-DASH แต่ส่วนขยายของ JMeter ดังกล่าวไม่เปิดให้ใช้งานในเชิงธุรกิจ ผู้ใช้หรือนักวิจัยต้องจ่ายเงินซื้อส่วนขยายที่เป็นลิขสิทธิ์ของผู้พัฒนาเพื่อนำมาใช้งาน

5.4 การนำไปใช้

มาตรฐานการให้บริการไฟล์วิดีโอ streaming ผ่านเครือข่ายอินเตอร์เน็ตด้วยโปรโตคอล HTTP ตามมาตรฐาน MPEG-DASH ถูกใช้งานอย่างแพร่หลายและมีแนวโน้มเพิ่มมากขึ้น ประกอบกับแนวโน้มการใช้งาน streaming ที่เพิ่มขึ้นตลอดเวลาทำให้ระบบให้บริการไฟล์วิดีโอเป็นที่นิยมของผู้พัฒนาระบบและผู้ให้บริการไฟล์วิดีโอในปัจจุบัน

ผู้วิจัยได้ทำการทดสอบระบบที่ออกแบบในงานวิจัยนี้ด้วยการทดสอบการใช้งานจริงผ่านระบบเครือข่ายภายใน และเปิดเล่นไฟล์วิดีโอด้วยไคลเอนต์ DASH คือ VLC และ Google Chrome ดังที่ได้กล่าวไปแล้ว ซึ่งสามารถเปิดเล่นไฟล์วิดีโอได้อย่างราบรื่น เพียงแต่ต้องใช้รุ่นและส่วนขยายที่ถูกต้อง ในงานวิจัยนี้ทุกการทดลองได้กระทำโดยการเปิดใช้งานโปรแกรมไคลเอนต์ดังกล่าวจริง ดังนั้นสามารถนำแนวทางการใช้งานนี้ไปใช้ต่อได้ทันที

ผู้สนใจสามารถนำโครงสร้างระบบให้บริการไฟล์วิดีโอที่ผู้วิจัยออกแบบไปใช้กับระบบให้บริการไฟล์วิดีโอผ่านระบบ HTTP ด้วยมาตรฐาน MPEG-DASH ได้ทันที โดยสามารถให้บริการกับอุปกรณ์ที่สามารถเชื่อมต่ออินเตอร์เน็ตผ่านทางโปรโตคอล HTTP และสามารถนำนโยบายการแทนที่แคชแบบต่าง ๆ มาปรับใช้กับระบบได้โดยการเขียนโปรแกรมเพิ่มเติม ในส่วนของนโยบายการแทนที่แคชที่ผู้วิจัยได้พัฒนาขึ้นสามารถทำงานได้เป็นอย่างดี และสามารถพัฒนาต่อยอดได้ โดยอาจต้องทดสอบการใช้งานกับระบบให้บริการไฟล์วิดีโอ streaming ที่ใหญ่ขึ้นกว่าเดิม และสามารถเพิ่มจำนวนแคชตามจำนวนระดับคุณภาพของไฟล์วิดีโอ (bitrate) ได้ตามต้องการ

5.5 แนวทางการพัฒนาต่อยอด

เนื่องจากระบบให้บริการไฟล์วิดีโอที่รองรับมาตรฐาน MPEG-DASH สามารถรองรับเป็นระบบที่สามารถเพิ่มความสามารถให้กับไคลเอนต์โดยเพิ่มและลดระดับคุณภาพของไฟล์วิดีโอ (bitrate) เมื่อใดก็ได้ตามสถานการณ์ของไคลเอนต์โดยพิจารณาจากค่าบัฟเฟอร์ (buffer) และประสิทธิภาพของอินเตอร์เน็ต (Throughput) ที่ไคลเอนต์รองรับได้ ซึ่งไคลเอนต์จะปรับเพิ่มหรือลด bitrate ขึ้นอยู่กับตัวแปรทั้งสอง

ในวิทยานิพนธ์นี้ผู้วิจัยได้นำเสนอนโยบายการแทนที่แคชโดยปรับใช้ค่าดัชนีความนิยมของการถูกใช้งานจากผู้ใช้ และนำเสนอแนวทางรองรับพฤติกรรมการเพิ่มและลดระดับคุณภาพวิดีโอของไคลเอนต์โดยการเพิ่มแยกแคชออกเป็นชุด ๆ เพื่อรองรับช่วงของ bitrate ของไฟล์วิดีโอ โดยทำงานอิสระจากกันด้วยนโยบายการแทนที่แคชที่เหมือนกัน โดยเน้นที่การนำออกจากแคช (eviction) แนวทางการพัฒนาต่อยอดคือการพัฒนาเพิ่มเติมในส่วนของการนำเข้า (insertion) ตามเงื่อนไขต่าง ๆ เช่น นำเข้าด้วยเงื่อนไขความนิยมของไฟล์นั้น ๆ หรือนำเข้าก่อนถึงเวลาที่จะใช้งาน segment นั้นจริง ๆ (prefetching) ด้วยเงื่อนไขต่าง ๆ หรือการเพิ่มเงื่อนไขการยอมให้ไคลเอนต์เพิ่มหรือลดระดับ bitrate ด้วยเงื่อนไขต่าง ๆ กัน (rate limit) หรือตามเงื่อนไขของภาระงานของระบบในขณะนั้น เป็นต้น ทั้งนี้ก็เพื่อเพิ่มประสิทธิภาพของระบบ และสามารถเป็นแนวทางใน

การศึกษาสร้างโมเดลการคำนวณขนาดแคชและคาดการณ์ภาระการทำงานของระบบได้ล่วงหน้าด้วยการศึกษาพฤติกรรมของผู้ใช้งานและปรับใช้กับนโยบายการแทนที่แคชที่เหมาะสม

บรรณานุกรม

[1] Cisco., “Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper.” [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html. [Accessed: 28-Apr-2019].

[2] S. Jiang and X. Zhang, “Making LRU friendly to weak locality workloads: a novel replacement algorithm to improve buffer cache performance,” IEEE Transaction on Computer, vol. 54, no. 8, pp. 939–952, Aug. 2005.

[3] A. M. Osman and N. I. Osman, “A Comparison of Cache Replacement Algorithms for Video Services,” International Journal of Computer Science and Information Technologies, vol. 10, no. 2, pp. 95–111, Apr. 2018.

[4] J. Summers, T. Brecht, D. Eager, and A. Gutarin, “Characterizing the workload of a netflix streaming video server,” in Proceedings of 2016 IEEE International Symposium on Workload Characterization (IISWC), 2016, pp. 1–12.

[5] A. Abhari and M. Soraya, “Workload generation for YouTube,” Multimedia Tools and Applications, vol. 46, no. 1, pp. 91–118, Jan. 2010.

[6] K. M. Agrawal, T. Venkatesh, and D. Medhi, “A dynamic popularity-based partial caching scheme for video on demand service in IPTV networks,” in Proceedings of the 6th International Conference on Communication Systems and Networks (COMSNETS), 2014, pp. 1–8.

[7] Hao Chen, Hai Jin, Jianhua Sun, Xiaofei Liao, and Dafu Deng, “A new proxy caching scheme for parallel video servers,” in Proceedings of International Conference on Computer Networks and Mobile Computing (ICCNMC 2003), 2003, pp. 438–441.

[8] Bitmovin, “Adaptive streaming - a simple explanation of how it works,” [Online]. Available: https://bitmovin.com/adaptive-streaming/. [Accessed: 1-May-2019].

[9] Cisco, “Next Generation Video Services Fundamentals,” Cisco Plus Canada, 2012.

[10] The WebM Project, “VP9 Video Codec Summary,” [Online]. Available: https://www.webmproject.org/vp9/. [Accessed: 29-Apr-2019].

[11] FFmpeg, “A complete, cross-platform solution to record, convert and stream audio and video,” [Online]. Available: https://ffmpeg.org/. [Accessed: 29-Apr-2019].

[12] Axiomatic Systems LLC, “Bento4 | Fast, Modern Tools and C++ Class Library for all your MP4 and DASH media format needs.” [Online]. Available: https://www.bento4.com/. [Accessed: 29-Apr-2019].

[13] GPAC, “MP4Box.” [Online]. Available: https://gpac.wp.imt.fr/mp4box/. [Accessed: 29-Apr-2019].

[14] Adobe, “HTTP Dynamic Streaming”. [Online].Available: https://www.encoding.com/http-dynamic-streaming-hds/. [Accessed: 29-Apr-2019].

[15] Bitmovin, “MPEG-DASH vs. Apple HLS vs. Microsoft Smooth Streaming vs. Adobe HDS.” [Online]. Available: https://bitmovin.com/mpeg-dash-vs-apple-hls-vs-microsoft-smooth-streaming-vs-adobe-hds/. [Accessed: 17-Apr-2019].

[16] M. Pantos, “IETF Draft: Apple HTTP Live Streaming (HLS),” Internet Engineering Task Force, Standard, 2013.

[17] I. Sodagar, “The MPEG-DASH Standard for Multimedia Streaming Over the Internet,” IEEE Multimedia, vol. 18, 2011, no. 4, pp. 62–67.

[18] DASH Industry Forum, “Catalyzing the adoption of MPEG-DASH.” [Online]. Available: https://dashif.org/. [Accessed: 29-Apr-2019].

[19]T. Stockhammer, “Dynamic Adaptive Streaming over HTTP: Standards and Design Principles,” in Proceedings of the 2nd Annual ACM Conference on Multimedia Systems, New York, NY, USA, 2011, pp. 133–144.

[20] Y. Lafon, R. Fielding, and J. Reschke, “Hypertext Transfer Protocol (HTTP/1.1): Range Requests.” [Online]. Available: https://tools.ietf.org/html/rfc7233. [Accessed: 29-Apr-2019].

[21] R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang, “QDASH: A QoE-aware DASH System,” in Proceedings of the 3rd Multimedia Systems Conference, New York, NY, USA, 2012, pp. 11–22.

[22]S. Lederer, C. Müller, and C. Timmerer, “Towards peer-assisted dynamic adaptive streaming over HTTP,” in Proceedings of the 19th International Packet Video Workshop (PV), 2012, pp. 161–166.

[23] Zakaria Ye, “Performance Analysis of HTTP Adaptive Video Streaming Services in Mobile Networks,” Networking and Internet Architecture, Université d’Avignon, 2017.

[24] W3school, “XML Tutorial.” [Online]. Available: https://www.w3schools.com/xml/. [Accessed: 16-Apr-2019].

[25] D. H. Lee, C. Dovrolis, and A. C. Begen, “Caching in HTTP Adaptive Streaming: Friend or Foe?,” in Proceedings of Network and Operating System Support on Digital Audio and Video Workshop, New York, NY, USA, 2014, pp. 31–36.

[26] N. C. Zakas, “How content delivery networks (CDNs) work.” [Online]. Available: https://humanwhocodes.com/blog/2011/11/29/how-content-delivery-networks-cdns-work/. [Accessed: 29-Apr-2019].

[27] M. Z. Shafiq, A. X. Liu, and A. R. Khakpour, “Revisiting Caching in Content Delivery Networks,” in Proceedings of ACM International Conference on Measurement and Modeling of Computer Systems, New York, NY, USA, 2014, pp. 567–568.

[28] Netflix, “Open Connect.” [Online]. Available: https://openconnect.netflix.com/. [Accessed: 29-Apr-2019].

[29] L. Ramaswamy, Ling Liu, and A. Iyengar, “Cache Clouds: Cooperative Caching of Dynamic Documents in Edge Networks,” in Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS’05), 2005, pp. 229–238.

[30] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and Sangheon Pack, “WAVE: Popularity-based and collaborative in-network caching for content-oriented networks,” in Proceedings of IEEE International Conference on Computer Communications Workshops, 2012, pp. 316–321.

[31] J. M. Almeida, D. Eager, and M. K. Vernon, “A Hybrid Caching Strategy for Streaming Media Files,” in Proceedings of SPIE - The International Society for Optical Engineering, vol. 4312, 2001.

[32] M.-C. Lee, F.-Y. Leu, and Y. Chen, “Pareto-based cache replacement for YouTube,” World Wide Web, vol. 18, no. 6, pp. 1523–1540, Nov. 2015.

[33] M. E. J. Newman, “Power laws, Pareto distributions and Zipf’s law,” Contemporary Physics, vol. 46, no. 5, pp. 323–351, Sep. 2005.

[34] B. Blasius and R. Tönjes, “Zipf’s Law in the Popularity Distribution of Chess Openings,” Physics Review Letters, vol. 103, 2009, no. 21, p. 218701.

[35] A. Abhari and M. Soraya, “Workload generation for YouTube,” Multimedia Tools and Applications, vol. 46, no. 1, pp. 91–118, Jan. 2010.

[36] X. Kang, H. Zhang, G. Jiang, H. Chen, X. Meng, and K. Yoshihira, “Measurement, Modeling, and Analysis of Internet Video Sharing Site Workload: A Case Study,” in Proceedings of IEEE International Conference on Web Services, 2008, pp. 278–285.

[37] A. Brodersen, S. Scellato, and M. Wattenhofer, “YouTube Around the World: Geographic Popularity of Videos,” in Proceedings of the 21st International Conference on World Wide Web, New York, NY, USA, 2012, pp. 241–250.

[38] H. Yu, L. Xie, and S. Sanner, “The Lifecyle of a Youtube Video: Phases, Content and Popularity,” in Proceedings of the 9th International AAAI Conference on Web and Social Media, 2015.

[39] NGINX, “High Performance Load Balancer, Web Server, & Reverse Proxy.” [Online]. Available: https://www.nginx.com/. [Accessed: 29-Apr-2019].

[40] OpenResty Inc., “OpenResty® - Official Site.” [Online]. Available: https://openresty.org/en/. [Accessed: 29-Apr-2019].

[41] Dormando, “Caching beyond RAM: the case for NVMe.” [Online]. Available: http://memcached.org/blog/nvm-caching/. [Accessed: 29-Apr-2019].

[42] E. Paraschiv, “Introduction to Caffeine,” Baeldung, 15-Oct-2017. [Online]. Available: https://www.baeldung.com/java-caching-caffeine. [Accessed: 17-Apr-2019].

[43] P-H. Kamp, “Varnish HTTP Cache.” [Online]. Available: https://varnish-cache.org/. [Accessed: 17-Apr-2019].

[44] Redislabs, “Redis.” [Online]. Available: https://redis.io/. [Accessed: 29-Apr-2019].

[45] Redis, “Using Redis as an LRU cache.” [Online]. Available: https://redis.io/topics/lru-cache. [Accessed: 29-Apr-2019].

[46] K. Spiteri, R. Sitaraman, and D. Sparacio, “From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player,” in Proceedings of the 9th ACM Multimedia Systems Conference, New York, NY, USA, 2018, pp. 123–137

[47] VideoLAN, “VLC media player.” [Online]. Available: https://www.videolan.org/. [Accessed: 1-Apr-2019].

[48] Cavar.net, “Native MPEG-Dash + HLS Playback,” Google Webstore. [Online]. Available: https://chrome.google.com/webstore/detail/native-mpeg-dash-%20-hls-pl/cjfbmleiaobegagekpmlhmaadepdeedn. [Accessed: 29-Apr-2019].

[49] R. V. Rasmussen and M. A. Trick, “Round robin scheduling – a survey,” European Journal of Operational Research, vol. 188, 2008, no. 3, pp. 617–636.

[50] M. Vora, L. Deek and E. Livengood, “Content Popularity for Open Connect,” Netflix TechBlog,2017.

[51] S. Lederer, C. Müller, and C. Timmerer, “Dynamic Adaptive Streaming over HTTP Dataset,” in Proceedings of the 3rd Multimedia Systems Conference, New York, NY, USA, 2012, pp. 89–94.

[52] C. Timmerer, “ITEC – Dynamic Adaptive Streaming over HTTP.” [Online]. Available: https://dash.itec.aau.at/. [Accessed: 29-Apr-2019].

[53] Apache JMeter™, “Apache JMeter.” [Online]. Available: https://jmeter.apache.org/. [Accessed: 29-Apr-2019].

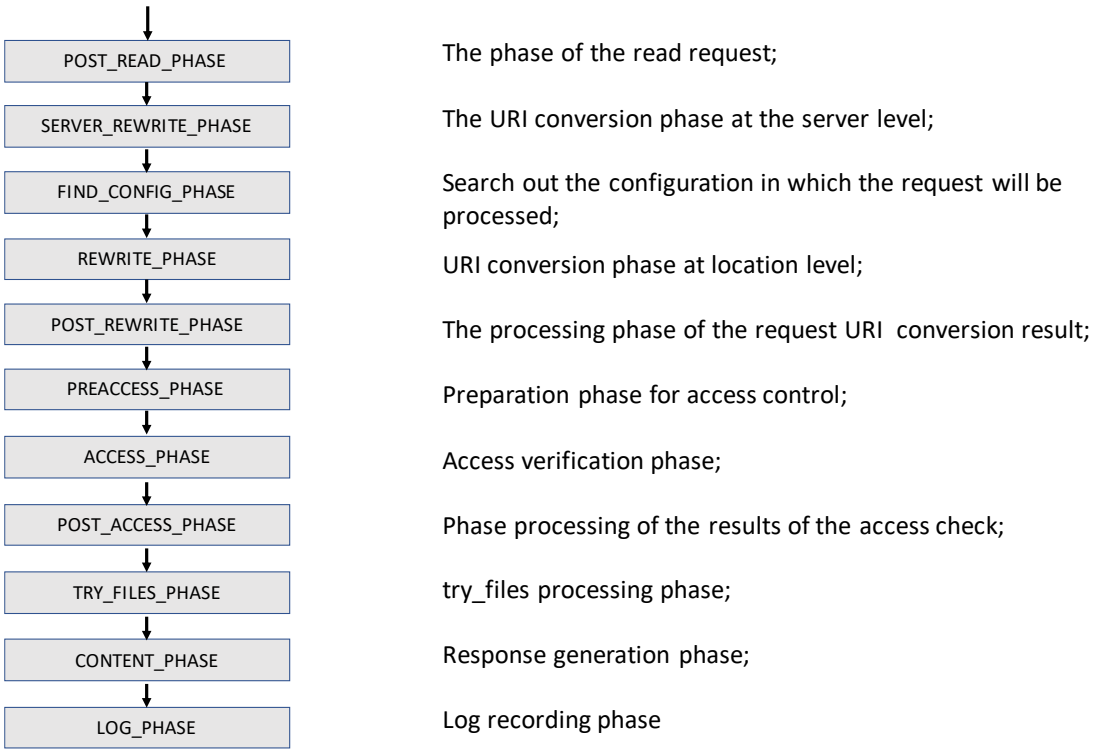
[54] DigitalOcean, “An Introduction to Load Testing.” [Online]. Available: https://www.digitalocean.com/community/tutorials/an-introduction-to-load-testing. [Accessed: 29-Apr-2019].

ภาคผนวก

ภาคผนวก ก

ตัวอย่างเฟส (phase) การทำงานของ Nginx

เฟสการทำงานของ Nginx ในการประมวลผลการร้องขอ (request) จากไคลเอนต์แสดงดังรูปที่ ก-1



รูปที่ ก-1 เฟสการทำงานของ Nginx ในการประมวลผลการร้องขอจากไคลเอนต์[40]

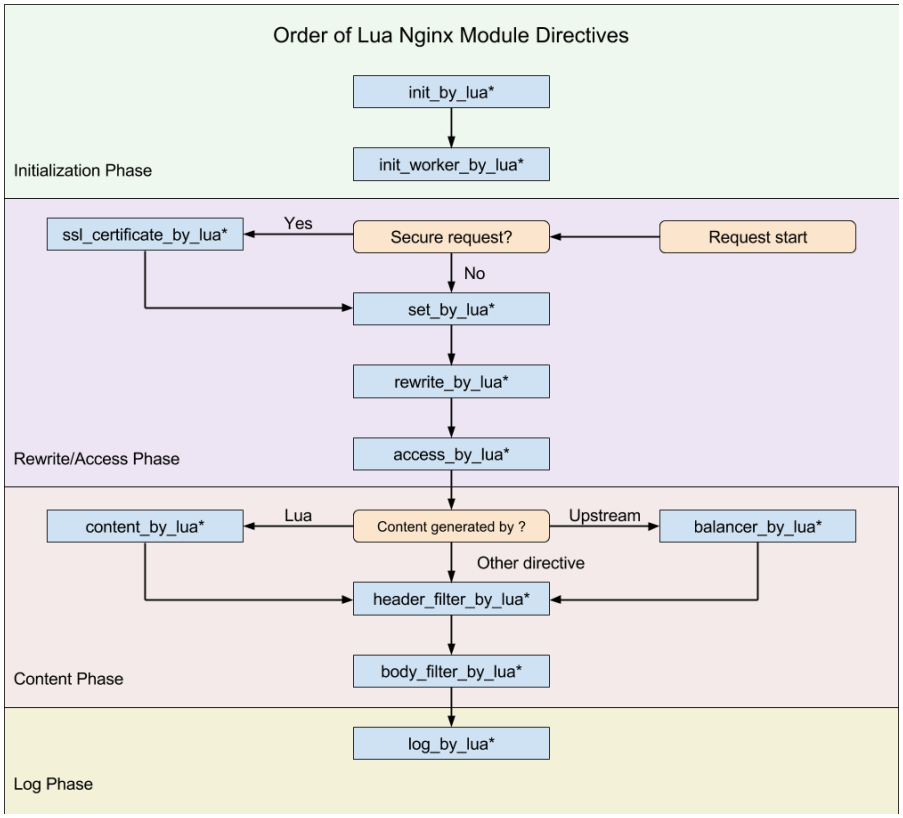
ภาคผนวก ข

การเขียนโปรแกรมด้วยภาษา Lua เพื่อควบคุมการทำงานของ Nginx โดยใช้ OpenResty

1. หากเขียนโปรแกรมภาษา Lua กับ Nginx จำเป็นต้องใช้คำสั่งและไลบรารีเฉพาะที่สามารถทำงานในระบบของ Nginx ซึ่งบางคำสั่งของภาษา Lua ไม่สามารถทำงานได้กับ Nginx หรือบางคำสั่งอาจทำให้เกิดข้อผิดพลาดขึ้นได้
2. ทุก ๆ phase ของ Nginx ทำงานในลักษณะเทรดย่อยทั้งหมด (light weight threads) ซึ่งทำงานอิสระจากกัน
3. การวางตำแหน่งส่วนของโปรแกรมให้เหมาะสมกับ phase ย่อยของ Nginx เป็นสิ่งสำคัญ
4. ระดับการเข้าถึงข้อมูลและตัวแปรต่าง ๆ ในระบบ เนื่องจาก Nginx ทำงานเป็น Master-workers และยังทำงานเป็นเทรดตามเหตุการณ์ (Event-based) ทำให้การแลกเปลี่ยนค่าตัวแปรระหว่าง master, worker และ threads นั้นเป็นเรื่องสำคัญมาก
5. การเขียนโปรแกรมภาษา Lua เพื่อควบคุมการทำงานร่วมกับ Nginx นั้นต้องติดตั้งโมดูลของ Nginx เพิ่มเติมดังในตารางโดยโมดูลทั้งหมดจะมาพร้อมกับการติดตั้ง OpenResty

ตารางที่ ข-1 โมดูลย่อยของ Nginx ที่ใช้ในการพัฒนาระบบ

โมดูล	รุ่น
ngx_devel_kit	0.3.0
echo-nginx-module	0.61
srcache-nginx-module	0.31
ngx_lua	0.10.13
ngx_lua_upstream	0.07
redis2-nginx-module	0.15
redis-nginx-module	0.3.7
ngx_stream_lua	0.5.0
lua_cjson	2.1.0
lua_resty_http	0.2



รูปที่ ข-1 เฟสของ Nginx ที่สามารถเพิ่มส่วนการทำงานได้ [40]

ภาคผนวก ค

การใช้งานโปรแกรมประยุกต์เพื่อเปิดเล่นไฟล์วิดีโอตามมาตรฐานการส่งผ่านข้อมูลผ่านโปรโตคอล HTTP ด้วยมาตรฐาน video streaming แบบ MPEG-DASH ในเครื่องไคลเอนต์

1. ใช้โปรแกรม VLC รุ่น 3.0.6 Vetinari ซึ่งผู้พัฒนาโปรแกรม VLC ได้นำโปรแกรม libdash รวมเข้ากับ VLC เรียบร้อยแล้ว
2. Google Chrome web browser ที่ติดตั้งส่วนขยายเพิ่ม (Extension) คือ Native MPEG-Dash + HLS Playback Google Chrome Extension [48]
3. วิธีการฝังโปรแกรมเข้าไปในหน้าเว็บโดยใช้โปรแกรมที่ชื่อว่า dash.js ที่หน่วยงาน DASH-IF [18] พัฒนาขึ้น

ภาคผนวก ง

ผลงานตีพิมพ์เผยแพร่จากวิทยานิพนธ์

The 19th International Symposium on Communications and Information Technologies (ISCIT 2019)

A Dynamic Popularity Caching Policy for Dynamic Adaptive Streaming over HTTP

Taofik Lamsub[1] and Pichaya Tandayya[2]
Department of Computer Engineering, Faculty of Engineering
Prince of Songkla University
Kohong, Hat Yai, Thailand, 90112
fik.lamsub@gmail.com[1], pichaya.t@coe.psu.ac.th[2]

***Abstract*— Nowadays, videos are increasingly stored, published and accessed across the Internet as the users can connect to the Internet and consume multimedia contents easily. Usually, video streaming data is massively large, and the user needs to play it continuously. Therefore, it requires a large scale of data processing and transportation between the source and destination. Caching techniques are applied to reduce the load of the storage server, but the performance depends on the caching policy and the users' behaviors. Dynamic Adaptive Streaming over HTTP (MPEG-DASH) is a new standard which has been increasingly used widely. The traditional caching policies were not adjusted to the streaming scenarios. Therefore, we propose a new caching policy applying the popularity and its attenuation to handle the streaming workload. Our proposed method outperformed the LRU and the LFU caching policies by increasing the hit rates when the cache percentage is in between 15%-35% under the sequential and Zipf-like workload scenarios.**

Keywords*— *popularity attenuation; cache eviction policy; MPEG-DASH; adaptive streaming video; video caching; dynamic popularity caching

I. INTRODUCTION

Currently, video streaming is the mainstream on the Internet and its volume is increasing every day. Generally, streaming data is broadcasted in video recording situations like important moments, sports matches and others. Video contents keep growing the Internet traffic and bandwidth up as shown in the Cisco dissertation [1]. Popular video applications include Netflix, YouTube, Daily Motion, Hulu, Amazon Prime, Spotify, live streaming for sports programs, videos in social media applications and videos for educational systems such as video on demands or video online courses. They typically use HTTP streaming to distribute video contents to the end users.

The streaming contents do not just include videos but also audios, and subtitles data. The streaming data in modern streaming can be divided into different types for benefitting different quality adjustment and the combination of source data according to the client usage and user requirements. There are many methods to stream videos for the users. The widely used one is HTTP streaming such as HTTP Live Streaming (HLS) [2], HTTP Dynamic Streaming (HDS) [3], Smooth Streaming and Dynamic Adaptive Streaming over HTTP (DASH) [4,5,7,10], also known as MPEG DASH as shown in Figure 1. [5] DASH is a new standard and is increasingly popular in the recent years, of which streaming contents are prepared for multi-encoding types and multiple qualities. DASH supports byte-range requests of the HTTP standard RFC7233 [6] of HTTP version 1.1 and the status code 206 partial content. The DASH client gets video data in a byte range by sending the specified byte range in the HTTP request header to the HTTP streaming server. The byte range of MPEG DASH video segment depends on segment duration [4] and the client will get a lot of small ranges along the entire video file.

A popular video can become viral on the Internet if the audience find it is significant or interesting. This increases the network traffic and server load. The video-on-demands like Netflix or TV broadcast reach their peak during the "prime time" hours. The video streaming will consume a lot of network resources and it will also urge the server load to the peak usage. User behaviors depend on the streaming application which means the demands can surge in any hour of the day. On the other hand, some video contents which were the most popular in the previous time may not be popular again at the present. The users desire a good Quality of Experience (QoE) [5] for smooth video watching which can relate to many factors such as client behavior, client buffer occupation, clients and server bandwidths, and server load. If the server cannot handle the load, it will lead to problems in video watching such as jitters and server response time. In-memory or cache are used in many HTTP servers to store data to solve this kind of problems.

Caching is a simple solution for reducing the server load. It is usually located between the backend storage and the HTTP server. Caching reduces the traffic between the server and backend storage. Thus, when the user requests a content which is already stored in the cache (cache hit), there will be no need to retrieve it from the backend storage. If the content is not in the cache (cache miss), then it is a penalty to fetch the content from the distant storage. Cache performance depends on the caching policy that decides which video range should be stored and which should be discarded. A caching policy is determined by its replacement policies: to make a decision whether to store a video range (insertion) or not and which video range is to be evicted from the cache (eviction).

There are many traditional caching policies being widely used such as Least-Recently-Used (LRU) [11,12] and Least Frequently Used (LFU) [11,12]. The insertion process inserts a new item to the cache and the eviction process discards the item. The LRU eviction policy is to discard the item which has the longest stay in the cache while the LFU eviction process discards the item which has the least accessed frequency. The LRU and LFU caching policies show good performances applying to traditional HTTP data usage such as documents or information contents, of which the sizes are small. The traditional cache policies are not suitable for video caching, especially video segment or byte range caching. The video streaming usage behavior is different to those of other applications. It depends on the user demands and the video popularity. The impact also is much higher. The LRU concerns age, while the LFU concerns frequency. However,

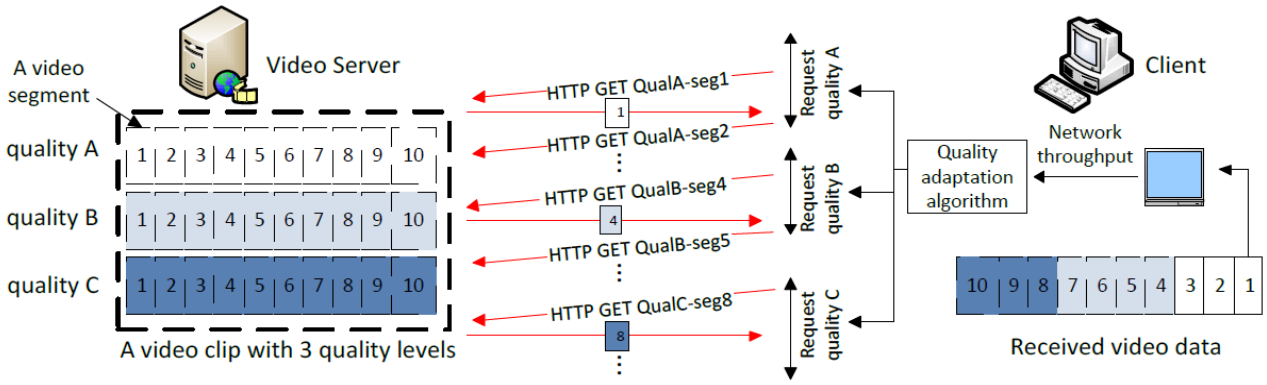


Figure 1 Dynamic Adaptive Streaming over HTTP Architecture [5]

in videos, the popularity and its attenuation play an important role. The popularity changes when the time passes by.

This paper proposes a new caching policy for MPEG-DASH applications in the context of video streaming for a video-on-demands system in order to reduce the video content server load, yield more popular items in cache, evict the less popular ones from the cache to increase the hit rate, and result in lower response times and reach a good QoE for multi-users.

II. Backgrounds and Related Works

In this section, we describe the details of MPEG-DASH, what it is, how it consists of and how it works. We also address the cache in the streaming context and traditional caching policies used to reduce the streaming server workloads, and why the traditional cache policies are not appropriate for streaming workloads. After that, we introduce tools and open source platforms used in our proposed method.

A. MPEG-DASH

The state-of-the-art streaming method is HTTP Adaptive bitrate Video Streaming (HDS) which is applied to solve the lack of video quality at the client-side caused by poor traffic. They can change the bitrate any time to sustain the video playing and to avoid waiting time in video downloading. They apply the pseudo streaming styles with byte range, downloading the small segments of the whole video. The video preparation process will split the entire video downloading into a series of segment downloading. MPEG-DASH [4,5] is a new standard for the HTTP streaming method as shown in Fig 1. It splits a video into data byte ranges according to a segment duration by video time splitting. The DASH client will request the video with HTTP byte range retrieval by sending the specified byte range in the HTTP request header to the HTTP streaming server. When the server receives the range request, it will read range data from the video file and send the range data back to the client. The byte range request of MPEG DASH depends on segment duration and it will get a series of small range segments along the entire video file.

B. Traditional Cache Eviction Policies

This subsection discusses the two famous traditional caching policies, LRU and LFU, as follows.

The Least Recently Used or LRU [11,12] is an age-based policy, which will store the new item at the top of the cache and if the requested item is already in the cache (cache hit), it will move the item to the top of the cache. It will reduce the response time when the recently requested item is requested again soon. When the cache is full and a new item is requested, then the LRU policy tries to evict an item which has not been requested for the longest time and the new item will replace it. The LRU policy is not the best for video streaming because the usage of video contents depends on the users' interests or the popularity of the videos. According to the video streaming, the popularity is an important factor. In addition, the item recency also is an important factor, but it does not consider to the popularity of the item. The lowest recently used item with the highest frequency can be replaced with a recent item which may be requested for only once.

The Least Frequency Used or LFU [11,12] is a frequency-based caching policy. It keeps the frequency values of all the items requested. The item with the highest frequency tends to have the longest stay in the cache. Then, the one with the lowest frequency will be replaced with the new item. The concept of LFU is that a high-frequency item can stay in the cache for a longer time as its frequency is accumulated even though it will not be requested anymore. The item will be evicted when its frequency is the lowest. It means that the new item must have a higher accumulated frequency than those of the older items in order to stay in the cache. Otherwise, the new item will just stay in the cache for a short time and will soon be evicted. It could be reinserted and soon re-evicted again and again until the frequency is high enough. The LFU policy gets a good performance if the frequency ranks of the popular items are high and they are already stored in the cache, and this scenario has not often been changed. The LFU can give a high performance if the users do not change the access patterns much. The LFU eviction policy does not consider the recency which also is usually important. The video which has the highest frequency from the past records may still be in the cache although it may not be popular anymore and currently has no longer been requested.

C. Caching for Video Streaming

A video server must handle complex workloads. Its cache should be large enough. Example caches in streaming services include Content Delivery Network (CDN) [8], Edge Server [8] or Open Connect Appliance (OCA) [9]. They provide large caches which employ distributed storage clusters or solid-state drives located near the clients geographically.

In-memory caching also is another option for the proxy server to reduce the retrieving time. The Redis [16] is an opensource, in-memory data structure storage. Redis has its own traditional cache policies, LRU, and LFU, but they can

be disable. The in-memory data of the Redis cache can also persist as it can be saved into the persistent storage which can be done in a background process. If the system fails, the data can be retrieved back once the system resumes.

D. Nginx HTTP Server

Nginx [14] is an open source HTTP web server that provides a multi-worker HTTP server and multi-thread event-based processes which handle a huge number of World Wide Web (WWW) clients' requests. The Nginx is widely used as an HTTP web server and a reverse proxy. It also provides load balancing. Nginx has its own cache solution with the traditional LRU policy that provides a cache hierarchy and zoning by pre-configuration. It is not flexible to modify the configuration during the run time, and its cache was also designed to support the WWW data, not video streaming. Therefore, we will bypass the Nginx cache and apply the Redis in-memory cache to handle our video streaming data caching.

E. OpenResty Lua Resty Platform

OpenResty [15] is a platform which helps embed the Lua programming language into an Nginx worker process. We apply Lua with OpenResty for our cache algorithm and an API gateway for handling streaming services.

III. Methodology

This section discusses our proposed method, caching for video streaming, dynamic popularity caching model and algorithm, system architecture, the video streaming data sets for experiments and how the popularity is important for video streaming.

A. Video Streaming Caching

The video streaming massively consumes data and creates more server workloads, especially the server load and traffic between the backend storage and the HTTP server. If the piece of video range that the user requires is stored in the HTTP buffer or cache, it will reduce the backend server load and traffic between the servers. From the DASH standard scenario, the video segment or range is chosen by the client who plays the video at the time frame. The most frequently used item obtains the highest popularity. In the context of video playing, the client will play the video segment until the end of the segment and will require the next segment for continuous playing. Therefore, the last video segment will no longer considered the highest popular as the time passes by. The recency should also change as the item, after some time, will not be considered recent anymore. Therefore, the video caching policy must consider both the popularity and recency. In this paper, we propose a combination of popularity-based and age-based caching policy. The item that is the most popular in the recent time will stay in the cache longer than the others but when the time passes by, the level of popularity will be decreased if there are not sufficient further requests to increase the popularity. Moreover, we have designed and implemented a workload-aware technique for DASH streaming with a bitrate selection cache node. Each cache node will store video segments of similar bitrates or quality.

B. Dynamic Popularity Caching Model

The popularity value is an important factor in the streaming video application that the popularity of user behavior patterns has a major effect on the system. The popularity indicates the demands of users and the priority. The priority changes as time passes by. However, in the video streaming scenarios, just considering either the frequency or the recency alone is not sufficient, because the clients play videos forward by requesting a sequence of small segments of the entire videos. Even though the video has a high popularity in the past, it may not be as popular at the present or in the future. When the time passes by, there will be attenuation in popularity, The LFU only considers frequency accumulation that does not attenuate with time but tends to be increased if the video segment is requested. We need to take into account both the popularity and its attenuation. If the segment obtains the same popularity level for a long time, and the user does not request it or requests it with a low frequency, then the popularity value must be decreased. The formula should consider how popular the item is and also for how long, and how recent the item is requested. The popularity attenuation depends on the time factor of the item in the cache defined by $\Delta time$ ($t_{present} - t_{last\_access}$). If the value of $\Delta time$ is large, the rate of the popularity attenuation is faster. On the other hand, if $\Delta time$ is small, the rate of the popularity attenuation is slower.

The popularity calculation for DASH streaming which is a segment-based streaming and range request support should concern the partial byte ranges or segmented-based popularity for investigating how popular the video segments are. We suppose that N is the total number of items already stored in the cache, $AccessCount_i$ is the access count of item i and $\Delta Time_i$ is the duration between the previous access time and the most recent time that the item i was requested by the user before the previous access time. We can calculate the popularity of item i at the present time (Pop_i) by applying Equation (1).

$$Pop_i = \frac{AccessCount_i}{\sum_{i=1}^{N} AccessCount_i} * \frac{1}{\Delta Time_i} \qquad (1)$$

When the cache is full and a new item has arrived our dynamic popularity caching policy will try to discard the segment that has a low popularity with a large $\Delta Time$ first. The Eq. (1) will be applied for the popularity calculation periodically and will also work as the attenuation factor. Figure.2 demonstrates how the time affects the popularity. By increasing the $\Delta Time$ of the segment, the popularity will be attenuated more quickly.

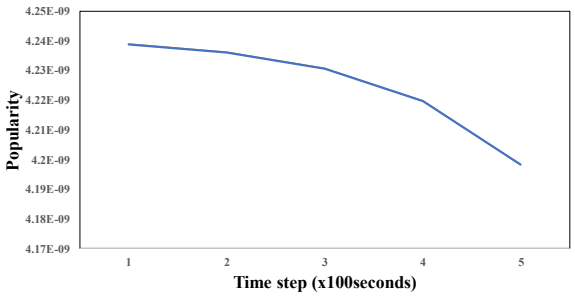


Figure 2 Attenuation of Popularity

Fig.2 shows that if the time passes by and the $\Delta Time$ increases with no further item requests after the last access of the segment, the popularity level will decrease as a function of time. In the graph, the $\Delta Time$ increases twice the time step. If the segment is requested again, the popularity attenuation will be reset, and the popularity will begin to attenuate again.

Figure 3 shows the transition of the popularity according to the time factor, and the attenuation of the popularity of items in the cache with the frequency rate and recency rate. It demonstrates that the importance of the popularity index previously ranked becomes less when the time passes by. For example, the item with a high popularity can fall into a medium popularity group, especially when there is no further request for the item, and later it will be ranked in a low popularity group which will be evicted out of the cache soon. When the item with a low popularity is requested again, the popularity will be set higher and the attenuation will be restarted as shown in the Eq.(1). The *ΔTime* plays a major role in popularity attenuation. The larger the *ΔTime*, the faster the popularity attenuation.

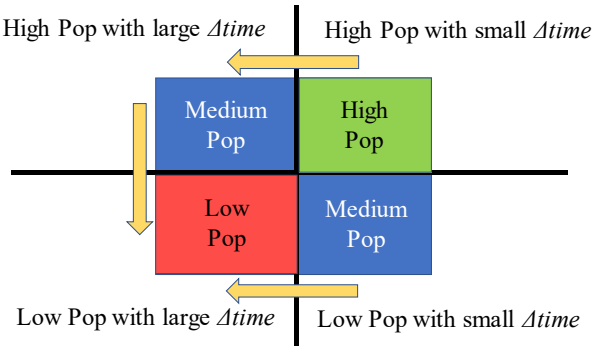


Figure 3 Popularity Model

C. Dynamic Popularity Caching Algorithm

Traditional cache policies tend to decide whether the segment has already been stored in the cache or not. If so, it will be read from the cache. If not, it will be retrieved from the distant storage which requires an additional time to get the data from the backend. Our proposed cache policy algorithm is shown in Algorithm 1. First, when the client requests for a video range or segment, the HTTP server will look for the video range in the cache. If it is already stored in the cache, it will send the segment to the client straightaway. If not, it needs to contact the distant server which, in this case, is the backend storage server in order to get the video segment which will also be stored into the cache. On every client request, the cache handler will update the popularity of the cache item. If the cache still has enough space, the cache handler will push the video segment into the cache. If not, the cache handler needs to evict some items to release the cache space for a new segment. These processes are the eviction policy and replacement policy.

The LRU eviction policy evicts the least recently used segment, while the LFU evicts the least frequency used segment. Both cache policies neither apply any co-efficient nor an aging factor of the popularity value. We propose a replacement strategy that while the cache is working, the cache handler will check the size of candidate eviction items list. The maximum size is 10% of the cache capacity. If the size of eviction list is less than 5% of its maximum size, the cache handler will spawn threads to calculate the popularity value of each item in the cache to determine the attenuated popularity and update the eviction list. The candidate evicted item is not just the least popular like in LFU, but also the least popular after being attenuated when the time passed by. In order to reduce the congestion that may occur due to multiple requests, we apply multiple queues and evicting threads to handle the cache eviction.

Algorithm 1

```
RequestFromClient()
1.  Pop = Popularity
2.  Cp = Cache capacity
3.  N = Number of segments in the cache
4.  evictList = Candidate evicted segment
5.  evictListSize  = Size of the candidate evicted segment = 10% of the
    cache capacity
6.  IF segmentIsInCache THEN
7.    // cache HIT
8.    Get the segment from cache()
9.    IF IsEvictList THEN
10.       Remove from Eviction Queue
11.   Update meta data
12. ELSE
13.   // cache MISS
14.   IF N reaches 98% of Cp OR evictListSize < 5% THEN
15.       Popularity calculation Eq.(1);
16.       Add evictList to Eviction Queue
17.   IF N = Cp THEN
18.       Evict all segments in the evictList Queue
19.       Update meta data
20.   ELSEIF N reach 99% of Cp THEN
21.       Evict the Least POP segment in Eviction Queue
22.       IF evict not success THEN
23.           Bypass cache
24.       Update meta data
25. ELSE
26.   Get video segment from the backend storage
27.   Store the new segment range in the cache
28. END
29. RETURN
```

Once the popularity calculation has been processed, the evicted lists will be assigned to eviction queues. When the new item has arrived and there is no more cache space, the cache handler will evict the item with the least attenuated popularity from the cache and push the new segment into the cache. In case that the request is hit, if the hit item is in the eviction queue, the cache handler will remove the item from the eviction queue and update the popularity which begins to start attenuating again. In case that the cache cannot handle the request, it will be bypass the cache and look for the item at the distant storage server.

D. System Environment

In our system, we used the in-memory [16] storage, Redis [16] for cache the MPEG-DASH video streaming data which are partial content from client byte range request. To evaluate the cache eviction model, we need to experiment with a constant bitrate in order to avoid client bitrate adaptation [4,5,10] following by bitrate oscillation which can occur anytime. The bitrate oscillation [17] is the reason for sudden rate changes for DASH clients. It leads to cache misses. It requires a mechanism to handle rate adaptation in the cache. First, we run the HTTP server with a proxy Redis cache policy [13] and configure the 'maxmemory-policy' to 'noevict' in order to disable the Redis eviction policy and then apply the proposed policy. The proxy server and backend are an Nginx web server with multiple workers (four workers). The connection among servers is 100 Mbps Ethernet and 1 Gbps between the client and the proxy server. The Redis cache runs on the same host of the HTTP proxy server, for acting as local streaming cache.

Our system architecture is shown in Figure 4. The system consists of the HTTP server with a byte range handler and a video quality proxy. The cache node consists of a cache handler which controls the cache and applies the cache policy

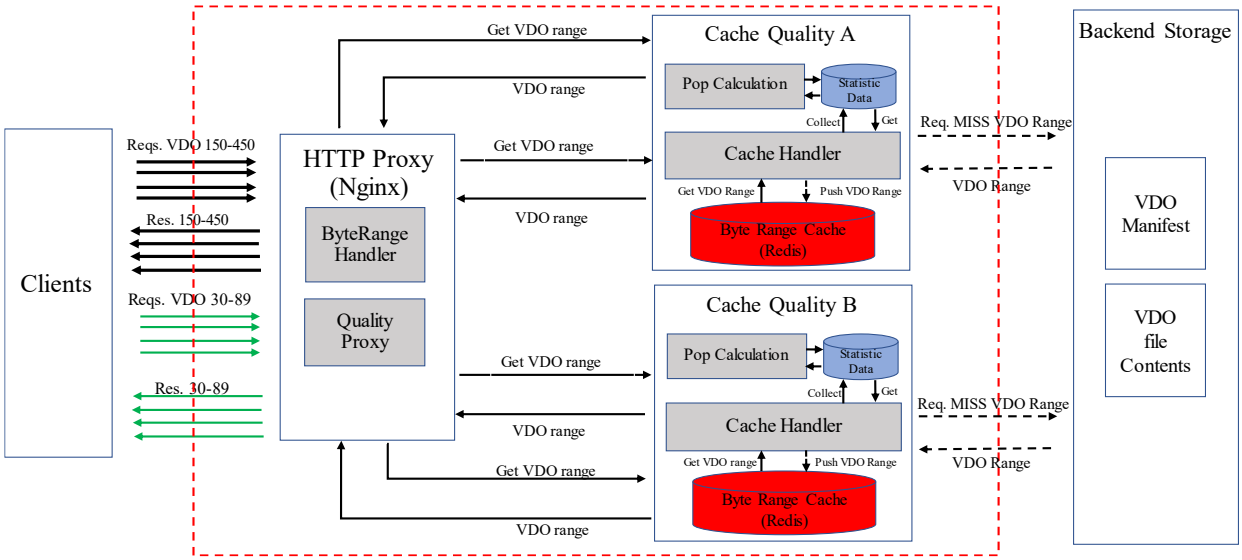


Figure 4 System architecture

for the Redis cache instance. The cache handler collects the usage data statistics of the data node for the popularity calculation process.

We have developed the HTTP streaming server with the Nginx, the HTTP server, and OpenResty platform [16] including the Lua library. We disabled the Nginx cache and used the Redis as our video byte-range cache. The video data was from MMsys 2014 [21,23] that provided the video file for MPEG-DASH open streaming video files with multiple bitrates and encoding types, for testing video streams, and for measuring the cache performance by using a 12- minute video length, and storing all video files and manifest files at the backend storage. Each video file consists of the 185 segments, 4-second segment duration and 1-second client mini buffer [10] which contains the MPEG-DASH manifest files [5,10] to be requested by the client before retrieving the entire video.

The video popularity distribution of the testing video streaming scenarios follows Power's Law distribution [18]. The items with the highest rank or highest frequency share a small group of videos and the rank of the rest but a majority video files are not as high. Therefore, the popular videos with the highest rank follows the same distribution theory as of the Zipf-like distribution [18,19,20,22], for the segment-based popularity distribution in each video file. However, in our case, the popularity distribution does not only follow the Zipf-like form, but its popularity can also attenuate when the time passes by. This affects the cache performance. Therefore, we set two testing scenarios: first, a sequential looping scenario with static popularity which videos are requested at different times, and second, a Zipf-like popularity distribution.

IV. Results and Discussions

We have evaluated the proposed cache policy with the open video streaming data by simulating the client scenarios and increasing the cache capacity. We compared our proposed cache policy with LRU and LFU under two scenarios of streaming workload at different cache sizes with multiple users. The cache size capacity varies as a proportion of the total video segment data in the system in percentage. First, we experimented with normal workload that the clients requested the video files with fair distribution but accessed them at different times. Second, we experimented with the workload following the popularity distribution [19,20]. The video workload distribution depends on the user behaviors. We applied a real workload following the Zipf-like distribution, a small group of video files, each had a high frequency request from the clients and the rest of video files had a lower frequency. The results are shown in Figure. 5. The hit rates of our proposed policy and LRU were similar because the clients generally requested videos starting from the first segments along to the last video segment. The requests were sequenced

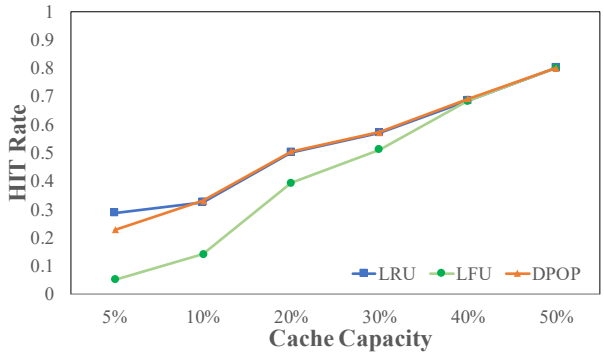


Figure 6 Hit rate comparison in the first scenario, a sequential request

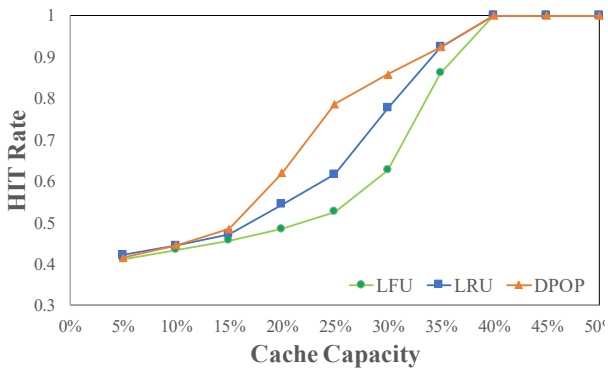


Figure 5 Hit rate comparison in a Zipf-like scenario

usually in a time order. The first segment is the first item to be fetched from the backend storage and to be pushed into the cache. It would stay in the cache for the longest time. Moreover, it has the most attenuated popularity, so that it will be the first candidate to be evicted in both our proposed policy and LRU according to the sequence of requests and access time. In LFU, the hit rate was lower because the accumulated frequency does not attenuate when the time passed by, and the items with a high frequency still keeps their frequency high, although the users do not request the items anymore.

In the other testing scenario, we have investigated whether the attenuated popularity affects the cache performance, applying a Zipf-like distribution workload and compared the results between our proposed policy and the LRU policy as shown in Figure. 6. The hit rates of both policies, our proposed policy and the LRU, started at the same level. When the cache capacity was increased to 15% of the data size, the performances began to differ. At the cache capacity of 20%, the hit rate of our proposed policy was higher than that of the LRU. Especially, at the 26% cache capacity, our proposed policy got the hit rate up to 0.8, while LRU got 0.62. The hit rate of our proposed policy was more than LRU. When applying more than 35% of cache capacity, the performance of both policies became equal again as with a large size of cache capacity, the cache can sufficiently hold data or video segments to fulfill the workload, no matter which policy is used.

According to MPEG-DASH, dynamic adaptive streaming, the clients will request the manifest [4] file from the server. The client can change the video quality any time, applying the client's adaption algorithm, quality adaption algorithm [10,24] , to seek for a suitable QoE. This leads to cache misses and bitrate oscillation. We can avoid the problems by enhancing the cache node with bitrate classification and storing the video segments of different qualities at different cache locations to provide the workload- aware and adaptive caching policy to handle the dynamically adaptive quality request workload.

V. Conclusions

In this paper, we have proposed a popularity-based caching policy along with our cache handler algorithm and popularity attenuation to be applied to the Dynamic Adaptive Streaming over HTTP which handles byte range video streaming requests. We found that the popularity-based caching policy and popularity attenuation can enhance the performance of the streaming cache. We have investigated the streaming workload by comparing our proposed cache policy with traditional policies, the classical LRU and LFU. For the sequential request, our caching policy performed like the powerful classical LRU policy and outperformed LFU policy. In the Zipf-like distribution workload, which emphasizes the popularity behaviors in the streaming scenario, our proposed method outperformed the LRU policy, by populating the popularity items and increasing the hit rate compared to the LRU policy when the cache percentage was in between 15% - 35%.

References

[1] "Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper," Cisco. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html. [Accessed: 28-Apr-2019].

[2] M. Pantos, "IETF Draft: Apple HTTP Live Streaming (HLS)," Internet Engineering Task Force, Standard, 2013.

[3] Adobe HTTP Dynamic Streaming (HDS). [Online]. Available:http://www.adobe.com/devnet/hds.html.[Accessed: 29-Apr-2019].

[4] I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming Over the Internet," IEEE Multimed., vol. 18, no. 4, pp. 62–67, Apr. 2011.

[5] R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang, "QDASH: A QoE-aware DASH System," in Proceedings of the 3rd Multimedia Systems Conference, New York, NY, USA, 2012, pp. 11–22.

[6] Y. Lafon, R. Fielding, and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Range Requests." [Online]. Available: https://tools.ietf.org/html/rfc7233. [Accessed: 29-Apr-2019].

[7] "DASH Industry Forum | Catalyzing the adoption of MPEG-DASH." [Online]. Available: https://dashif.org/. [Accessed: 29-Apr-2019].

[8] N. C. Zakas, "How content delivery networks (CDNs) work." [Online]. Available: https://humanwhocodes.com/blog/2011/11/29/how-content-delivery-networks-cdns-work/. [Accessed: 29-Apr-2019].

[9] "Netflix Open Connect | Open Connect." [Online]. Available: https://openconnect.netflix.com/. [Accessed: 29-Apr-2019].

[10] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP –: Standards and Design Principles," in Proceedings of the Second Annual ACM Conference on Multimedia Systems, New York, NY, USA, 2011, pp. 133–144.

[11] S. Jiang and X. Zhang, "Making LRU friendly to weak locality workloads: a novel replacement algorithm to improve buffer cache performance," IEEE Trans. Comput., vol. 54, no. 8, pp. 939–952, Aug. 2005.

[12] A. M. Osman and N. I. Osman, "A Comparison of Cache Replacement Algorithms for Video Services," Int. J. Comput. Sci. Inf. Technol., vol. 10, no. 2, pp. 95–111, Apr. 2018.

[13] "Using Redis as an LRU cache – Redis." [Online]. Available: https://redis.io/topics/lru-cache. [Accessed: 29-Apr-2019].

[14] "NGINX | High Performance Load Balancer, Web Server, & Reverse Proxy," NGINX. [Online]. Available: https://www.nginx.com/. [Accessed: 29-Apr-2019].

[15] "OpenResty® - Official Site." [Online]. Available: https://openresty.org/en/. [Accessed: 29-Apr-2019].

[16] "Redis." [Online]. Available: https://redis.io/. [Accessed: 29-Apr-2019].

[17] D. H. Lee, C. Dovrolis, and A. C. Begen, "Caching in HTTP Adaptive Streaming: Friend or Foe?," in Proceedings of Network and Operating System Support on Digital Audio and Video Workshop, New York, NY, USA, 2014, pp. 31:31–31:36

[18] M. E. J. Newman, "Power laws, Pareto distributions and Zipf's law," Contemp. Phys., vol. 46, no. 5, pp. 323–351, Sep. 2005.

[19] J. Summers, T. Brecht, D. Eager, and A. Gutarin, "Characterizing the workload of a netflix streaming video server," in 2016 IEEE International Symposium on Workload Characterization (IISWC), 2016, pp. 1–12.

[20] A. Abhari and M. Soraya, "Workload generation for YouTube," Multimed. Tools Appl., vol. 46, no. 1, pp. 91–118, Jan. 2010.

[21] S. Lederer, C. Müller, and C. Timmerer, "Dynamic Adaptive Streaming over HTTP Dataset," in Proceedings of the 3rd Multimedia Systems Conference, New York, NY, USA, 2012, pp. 89–94.

[22] X. Kang, H. Zhang, G. Jiang, H. Chen, X. Meng, and K. Yoshihira, "Measurement, Modeling, and Analysis of Internet Video Sharing Site Workload: A Case Study," in 2008 IEEE International Conference on Web Services, 2008, pp. 278–285.

[23] "ITEC – Dynamic Adaptive Streaming over HTTP." [Online]. Available: https://dash.itec.aau.at/. [Accessed: 29-Apr-2019].

[24] K. Spiteri, R. Sitaraman, and D. Sparacio, "From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player," in Proceedings of the 9th ACM Multimedia Systems Conference, New York, NY, USA, 2018, pp. 123–137.

ประวัติผู้เขียน

ชื่อ สกุล นายเตาฟีก หลำสุบ

รหัสประจำตัวนักศึกษา 5710120068

วุฒิการศึกษา

วุฒิ	ชื่อสถาบัน	ปีที่สำเร็จการศึกษา
วิศวกรรมศาสตรบัณฑิต เกียรตินิยมอันดับสอง (วิศวกรรมคอมพิวเตอร์)	มหาวิทยาลัยวลัยลักษณ์	2550

ทุนการศึกษา

ทุนการศึกษาบัณฑิตวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์

การตีพิมพ์เผยแพร่ผลงาน

Taofik Lamsub and Pichaya Tandayya, “A Dynamic Popularity Caching Policy for Dynamic Adaptive Streaming over HTTP,” in Proceedings of the 19th International Symposium on Communications and Information Technologies (ISCIT 2019), Ho Chi Minh City, Vietnam, 25-27 September 2019