การพัฒนาซอฟแวร์สำหรับวิเคราะห์แคช

Development of Cached Objects Analyzing Tool

โดย

นายวรวัฒน์ เชิญสวัสดิ์

รหัส 43067164

อาจารย์ที่ปรึกษา

อาจารย์ อัครินทร์ คุณกิตติ

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 1 ปีการศึกษา 2545

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

หัวข้อ	การพัฒนาซอฟแวร์สำหรับวิเคราะห์แคช
นักศึกษา	นายวรวัฒน์ เชิญสวัสดิ์
อาจารย์ที่ปรึกษา	อาจารย์ อัครินทร์ คุณกิตติ
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2545

บทคัดย่อ

ในปัจจุบันได้มีการใช้ Web Proxy/Cache Serverในการช่วยลดความคับคั่งของการสื่อสาร เนื่องจากการใช้งานอินเตอร์เน็ตที่เพื่อขึ้น ทำให้ช่องทางการสื่อสารที่มีอยู่ไม่สามารถรองรับความต้องการได้ และยังเป็นการช่วยลดภาระของ Web Server ได้อีกด้วย โดยการใช้งานจริงๆนั้นอาจมีการใช้ Cache มากกว่าหนึ่งตัวทำงานร่วมกันก็ได้ ทำให้เกิดความยากในการที่จะรู้ว่าใน Cache ตัวนั้นๆมีข้อมูลอะไรบ้าง และมีความซ้ำซ้อนของข้อมูลระหว่าง Cache แต่ละตัวมากน้อยแค่ไหน ดังนั้นโครงงานนี้จึงจะทำการพัฒนาซอฟแวร์ที่ใช้สำหรับวิเคราะห์ข้อมูลในCacheสำหรับโปรแกรม Squid โดยจะวิเคราะห์จาก log file ซึ่งได้แก่ Store.log การออกแบบและพัฒนาซอฟแวร์นี้จะแบ่งเป็นสองส่วนคือส่วนที่ใช้ในการรับอ่านค่าจาก store.log เพื่อนำมาแปรความหมายและจัดเก็บไว้ในฐานข้อมูล โดยที่ในโครงงานนี้ได้ใช้โปรแกรมMySQLทำหน้าที่เป็นฐานข้อมูล และส่วนที่ติดต่อกับผู้ใช้ ซึ่งจะพัฒนาในลักษณะของ Web Base โดยใช้เทคโนโลยีของ JAVA ผลที่ได้คือโปรแกรมที่สามารถใช้วิเคราะห์ความซ้ำซ้อนของ Object ระหว่าง Proxy/Cache Server ได้ ทำให้สามารถทำการวิเคราะห์ได้ง่ายและรวดเร็ว และผู้ใช้วิเคราะห์สามารถใช้งานได้ในทุกๆที่ เนื่องจากระบบส่วนของ User Interface เป็นแบบ Web base

Title	Development of Cached Objects Analyzing Tool
Student	Mr. Worawat Choensawat
Advisor	Mr. Akharin Khunkitti
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	2002

ABSTRACT

Nowadays, there are many usages of Web Proxy/ Cache Server to reduce the traffic congestion of information communication and also help relieving the burden of Web Server as well. In practical, more than one cache may be used. And this shall lead to some difficulty in tracking of which data is contained in each cache and also the any repetition of data in each one. Therefore, this project will be developing software to analyze the information in the cache from Squid's program by analyzing the log file (store.log).Design and developing this software will consist of two parts, first part is to receive the data from store.log, analyze the data and then managing it into database. In this project will be using MYSQL as database program. The Second part is the user interface, which will be developing as a web bases that will especially base on JAVA Technology. The result of this development will archive a program that is easy to use for analyze the duplication of object in each of Proxy/Cache Server .Due to the user part of this program were built as web base ,the user can then use this program in any places and any time .

กิตติกรรมประกาศ

โครงงานพัฒนาระบบงานฉบับนี้สำเร็จลุล่วงลงได้เพราะความอนุเคราะห์อย่างสูงจากหลายท่าน ผมจึงขอขอบพระคุณผู้ที่มีส่วนร่วมและช่วยเหลือทุกๆท่าน โดยเฉพาะบุคคลดังต่อไปนี้

ขอขอบพระคุณ

บิดามารดา	ผู้มีพระคุณเหลือล้น และเป็นทุกสิ่งทุกอย่าง
อาจารย์ อัครินทร์ คุณกิตติ	อาจารย์ที่ปรึกษาโครงงาน
นายพรชัย ยิ่งเจริญธนา	ที่คอยให้คำปรึกษาในทุกๆด้าน
นายสุทธิพงศ์ สุวนิช	สำหรับคำปรึกษาด้านโปรแกรม Java และการติดตั้งเครื่องมือต่างๆ
นายชินพงศ์ สมสืบ	เอื้อเฟื้ออุปกรณ์ในการทดลอง
คณาจารย์คณะสารสนเทศทุกท่าน	ที่ประสิทธิประสาทวิชา

และเพื่อนๆ คณะสารสนเทศ สจล. ที่ช่วยเหลือและให้กำลังใจตลอดมา

นายวรวัฒน์ เชิญสวัสดิ์

09/10/2545

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

หน้า

ตารางที่

สารบัญภาพ

หน้า

ภาพที่

สารบัญภาพ (ต่อ)

หน้า

ภาพที่

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

องค์กรหรือหน่วยงานต่างๆในปัจจุบัน ได้มีการใช้อินเตอร์เน็ตกันอย่างแพร่หลาย ไม่ว่าจะเพื่อค้นหาข้อมูลข่าวสาร เพื่อความบันเทิง หรือการทำธุระกรรม ซึ่งมีความจำเป็นจะต้องรองรับการใช้อินเตอร์เน็ตพร้อมๆกัน ทำให้เกิดความคับคั่งของช่องสัญญาณเครือข่ายมากขึ้น และเป็นผลให้เกิดความล่าช้าของข้อมูล และในความเป็นจริงแล้วบ่อยครั้งที่ผู้ใช้แต่ละคนอาจร้องขอ Web page หรือข้อมูลที่เหมือนๆกัน หรือมีการร้องขอข้อมูลที่เคยได้ขอไปก่อนหน้านี้แล้วที่ตนเอง หรือคนอื่นๆในเครือข่ายเดียวกันเป็นผู้ขอก็ได้ จึงได้มีการนำเอาระบบของ Web Proxy/Cache Server มาใช้งาน ในปัจจุบัน Software ที่เป็นที่นิยมและใช้กันอย่างกว้างขวางตัวหนึ่งก็คือโปรแกรม Squid โดยหลักการของ Proxy/Cache ก็คือการนำเอาข้อมูลที่ได้รับจากการร้องขอมาเก็บไว้ใน Cache ซึ่งจะเก็บเป็นลักษณะของ object เช่น ไฟล์ HTML รูปภาพ และอื่นๆ เพื่อรองรับการร้องขอที่เกิดขึ้นอีก เป็นการลดปริมาณการติดต่อกับเครือข่ายภายนอกที่มีความเร็วที่ต่ำกว่ามาก โดยอาจมีการใช้ Web Proxy/Cache Server มากกว่าหนึ่งตัวทำงานร่วมกัน เพื่อแบ่งปันข้อมูลและทรัพยากรซึ่งกันและกัน ช่วยลดเวลาในการเข้าถึงข้อมูล สามารถลด Bandwidth จำนวนมากได้ และยังเป็นการเพิ่มประสิทธิภาพของระบบอีกด้วย เนื่องจากการที่มี Proxy Server เครื่องเดียวจะเกิดปัญหาต่างๆในลักษณะต่อไปนี้

- คอขวดเนื่องจากเครื่องลูกข่ายทุกเครื่องที่อยู่ในเครื่องข่ายภายในจะส่งคำร้องมายังเครื่อง Server ที่ให้บริการอยู่เพียงเครื่องเดียว
- Single point of failure เนื่องจากการที่มีเครื่อง Server ให้บริการเพียงเครื่องเดียว หากเกิดปัญหาที่ Server แล้วระบบ Proxy/Cache จะไม่สามารถที่จะให้บริการได้

การใช้ Web Proxy/Cache Server มากกว่าหนึ่งตัวทำงานร่วมกันยังเป็นการลดภาระของ Server อีกด้วย แต่การใช้ Server มากกว่าหนึ่งตัวก็ทำให้เกิดความยากในการที่จะรู้ว่าใน Cache ตัวนั้นๆมีข้อมูลอะไรบ้าง และมีความซ้ำซ้อนของข้อมูลระหว่าง Cache แต่ละตัวมากน้อยแค่ไหน โปรแกรมที่ทำหน้าที่เป็น Proxy/Cache Serverที่มีใช้อยู่ในปัจจุบันมีมากมายหลายโปรแกรม ซึ่งแต่ละตัวก็จะมีรูปแบบการเก็บobject และ log file ที่ไม่เหมือนกัน โปรแกรมที่ได้รับความนิยมมากตัวหนึ่งได้แก่โปรแกรม Squid เนื่องจาก

Squid เป็นซอฟแวร์ที่ฟรี(Freeware) และมีเสถียรภาพ โดยในระหว่างการทำงานโปรแกรมSquid นั้นจะทำการบันทึกเหตุการณ์ที่เกี่ยวข้องกับObjectที่เกิดขึ้นไว้ใน Log file ข้อมูลที่ถูกเก็บไว้นี้สามารถนำมาวิเคราะห์เพื่อทราบถึงความซ้ำซ้อนของ Object ในCache ได้

1.2 ขอบเขตของโครงการพัฒนาระบบงาน

ในโครงการพัฒนาระบบงานได้นำการวิเคราะห์ log fileมาใช้เพื่อให้ทราบถึงความซ้ำซ้อนของObject ในCache โดยจะทำการศึกษาและพัฒนา software ที่ใช้กับโปรแกรม Squid โดยโครงการนี้จะมุ่งเน้นในเรื่องของลักษณะการเก็บ object ใน Squid โดยจะใช้การวิเคราะห์ log file ที่เกิดขึ้นซึ่งได้แก่ Store.log

การพัฒนาโปรแกรมจะแบ่งออกเป็นสองส่วนด้วยกัน ส่วนแรกเป็นโปรแกรมที่ใช้ในการรับค่า Store.log จากโปรแกรม Squid เพื่อนำมาแปรความหมายและจัดเก็บไว้ในฐานข้อมูล โดยที่ในโครงงานนี้ได้ใช้ภาษา Java ในการพัฒนาตัวโปรแกรมและใช้โปรแกรม MySQL ทำหน้าที่เป็นฐานข้อมูล และในส่วนที่สองเป็นส่วนติดต่อกับผู้ใช้ซึ่งจะพัฒนาในลักษณะของ Web Base โดยใช้เทคโนโลยีของ JAVA โดยจะใช้ Tomcat เป็น JSP/Servlets Container การเข้าถึงข้อมูลที่ถูกเก็บไว้ใน MySQL จะผ่านทาง JDBC เพื่อแสดงผลให้กับผู้ใช้ทราบถึงผลสรุปและรายละเอียดของ Objects ใน Cacheแต่ละตัว

1.3 ขั้นตอนการดำเนินงาน

ในโครงการนี้ได้แบ่งขั้นตอนการศึกษาและการพัฒนาโปรแกรมไว้ดังต่อไปนี้

ขั้นตอนที่ 1 ศึกษาการทำงานของระบบ Proxy/Cache โดยเฉพาะโปรแกรม Squid ทั้งลักษณะการเก็บ Objects และlog file ที่ เกิดขึ้น เพื่อที่จะใช้ในการออกแบบโปรแกรมต่อไป

ขั้นตอนที่ 2 ศึกษาเครื่องมือต่างๆที่เกี่ยวข้องกับการพัฒนาโปรแกรม

ขั้นตอนที่ 3 วิเคราะห์และออกแบบระบบ การทำงานของโปรแกรม และฐานข้อมูล ภายใต้ขอบเขตที่ได้กำหนดไว้

ขั้นตอนที่ 4 ทำการพัฒนาตัวโปรแกรม และทดสอบโปรแกรมแต่ละส่วน ให้ได้ตามที่ออกแบบไว้

ขั้นตอนที่ 5 ทำการทดสอบและปรับปรุงโดยรวม ให้เหมาะสมกับสภาพแวดล้อมที่จะเกิดขึ้น

บทที่ 2
ระบบ Proxy/Cache และทฤษฎีที่เกี่ยวข้อง

2.1 ความหมายของ Proxy/Cache

ปัจจุบันในเครือข่ายอินเตอร์เน็ตมีคอมพิวเตอร์ที่เข้ามาต่อเชื่อมอย่างมากมาย และยังมีแนวโน้มที่จะเพิ่มขึ้นอย่างรวดเร็ว ทำให้ปริมาณข้อมูลที่ผ่านเข้าออกเครือข่ายมีจำนวนสูงมาก ส่งผลให้เกิดความล่าช้าและสูญหายของข้อมูล จึงได้มีความพยามยามมากมายที่จะแก้ปัญหาดังกล่าวทั้งด้านของการพัฒนาเทคโนโลยีของสายนำสัญญาณ กล่าวคือสามารถที่จะส่งข้อมูลได้ด้วยความเร็วที่สูงขึ้น แต่ก็ยังไม่เพียงพอกับการใช้อยู่ดี

แนวทางหนึ่งก็คือแนวคิดที่จะนำข้อมูลที่มีการใช้บ่อยๆมาเก็บไว้ให้ใกล้กับผู้ใช้มากที่สุด เพื่อลดปริมาณการติดต่อกับเครือข่ายภายนอก ทำให้เครือข่ายภายนอกมีความคล่องตัวสูงขึ้น ความคิดนี้ได้นำมาสู่หลักการของProxy/Cache หรือ Web caching ซึ่งหมายถึง การเก็บสำเนาข้อมูลของ website ที่เคยเข้าไปเยี่ยมชมมาแล้วในลักษณะของ object เช่น ไฟล์HTML รูปภาพ และอื่นๆ เอาไปเก็บไว้ใน memory หรือ hard disk และเมื่อจะใช้อีกครั้ง ก็ไม่ต้องไปดึง จาก website ต้นฉบับ ทำให้ลดปริมาณ ข้อมูลที่วิ่งอยู่ใน network และ ผู้ใช้ได้ข้อมูลเร็วขึ้น แต่ก็จะใช้ไม่ได้กับข้อมูลลักษณะแบบ Dynamic เช่น พวกผลลัพธ์ ของการค้นหาข้อมูล เป็นต้น การทำงานเช่นนี้เหมาะสำหรับการติดต่อที่ไม่ต้องการการโต้ตอบแบบทันทีทันใด เช่น HTTP, FTP หรือ GOPHER การทำงานในลักษณะเช่นนี้จำเป็นต้องอาศัยตัวกลางระหว่าง Client กับ Host Server ซึ่งเรียกว่า Proxy

การทำงานก็คือ เมื่อClient ต้องการดึงข้อมูลจากเครือข่ายภายนอก จะทำการติดต่อผ่านไปยัง Proxy/Cache ก่อน เพื่อจะดูว่ามี Object ที่ต้องการในCacheหรือไม่ ถ้ามีก็จะทำการตรวจสอบความล้าสมัยของObjectดังกล่าวซึ่งจะกล่าวในรายละเอียดในหัวข้อต่อๆไป ถ้ามีและไม่ล้าสมัยก็จะส่งข้อมูลนั้นกลับไปยัง Client ที่ติดต่อมา แต่ถ้าไม่ก็จะติดต่อไปยัง Web Site ต้นฉบับ แล้วดึงเอาข้อมูลนั้นมาทำสำเนาไว้ เพื่อประโยชน์ในการร้องขอข้อมูลในครั้งต่อไป แล้วก็ส่งต่อไปยัง Client ที่ติดต่อมาอีกทีหนึ่ง

องค์กรบางแห่งมีผู้ใช้งานอินเตอร์เน็ตเป็นจำนวนมาก จึงอาจจะต้องมีการติดตั้ง Proxy/Cache Server มากกว่าหนึ่งเครื่องทำงานร่วมกันเพื่อรองรับความต้องการที่เกิดขึ้น แบ่งปันข้อมูลและทรัพยากรซึ่งกันและกัน ช่วยลดเวลาในการเข้าถึงข้อมูล สามารถลด Bandwidth จำนวนมากได้ และยังเป็นการเพิ่มประสิทธิภาพของระบบอีกด้วย โดยจะกล่าวในรายละเอียดต่อไป

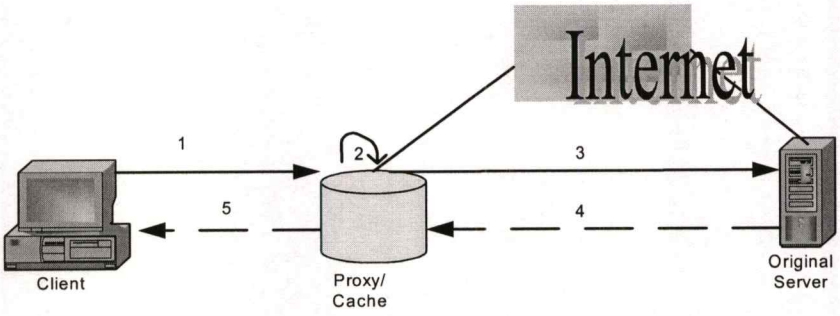
2.2 ที่มาของโปรแกรม SQUID

Squid คือ Internet proxy-caching program ซึ่งมีจุดเริ่มต้นมาจากการพัฒนาโปรแกรมของ Harvest project ซึ่งไม่ได้มีจุดมุ่งหมายในการพัฒนา Cache โดยตรง เป็นโครงการที่ได้รับเงินทุนสนับสนุนจาก National Laboratory of Network Research (NLANR) โปรแกรม Squid เป็นโปรแกรมที่แจกฟรี และ เปิดเผย Source-code จึงสามารถนำ Squid ไปพัฒนาเพิ่มเติมความสามารถที่ต้องการ และ แก้ไขข้อผิดพลาดที่เกิดขึ้นได้ ทำให้ Squid เป็นที่นิยมนำไปใช้เพราะผู้ที่มีความรู้ด้านการเขียนโปรแกรมสามารถปรับปรุงประสิทธิภาพและความสามารถของ Squid ได้ตามที่ตนเองต้องการ โปรแกรม Squid เป็นโปรแกรมสำหรับทำ Web Caching ที่แรกๆนิยมในการใช้งานบนระบบ UNIX เช่น AIX, Linux, HP-UX, FreeBSD, NetBSD และฯลฯ แต่ในปัจจุบันนั้นได้ถูกพัฒนาให้ทำงานบน OS จากค่าย Microsoft อีกด้วย

ความสามารถของ Squid ประกอบด้วยกลไกในการ proxy และ caching HTTP, FTP และโปรโตคอลอื่นที่สามารถใช้การอ้างถึงแบบ URL เช่น SSL ทั้งยังสามารถจะทำงานร่วมกันได้ในรูปแบบ Cache Hierarchies โดยใช้โปรโตคอล ICP, HTCP, CARP, Cache Digest ในการสื่อสารระหว่างกัน การที่ Cache จะสามารถทำงานได้อย่างมีประสิทธิภาพได้นั้น จะต้องประกอบด้วยประสิทธิภาพที่ดีของระบบโดยรวม ต่อไปนี้เป็นองค์ประกอบที่สำคัญต่อประสิทธิภาพของ Cache คือเวลาที่ใช้ในการเข้าถึงข้อมูลใน disk, หน่วยความจำทั้งหมดของระบบ, Disk throughput และความสามารถของ CPU

2.3 หลักการทำงานของโปรแกรม Squid Proxy/Cache

โปรแกรม Squid Proxy/Cache จะทำหน้าที่ในการเป็นตัวกลางในการติดต่อ การทำงานนั้นได้แสดงในรูปที่ 2.1 กล่าวคือเมื่อ client ทำการร้องขอจะกระทำผ่านโปรแกรม Squid โดยโปรแกรมนี้จะทำการตรวจสอบว่ามีข้อมูลดังกล่าว (object) ใน Cache หรือไม่ ถ้ามีก็จะทำการตรวจสอบเรื่องความล้าสมัย(Consistency) ของข้อมูลจาก TTL (Time To Live) และถ้า object นั้นหมดอายุก็จะทำการร้องขอแบบ IMS GET (If Modify Since GET) ส่วนในกรณีที่ไม่พบใน Cache ก็จะทำการร้องขอแบบ GET ธรรมดา ลักษณะการเก็บข้อมูลหรือ object ในรูปแบบของไฟล์ ซึ่งในไฟล์นั้นจะเก็บทั้งตัว object และส่วนของ Header โดยที่ชื่อไฟล์ มีการเข้ารหัส Hash Function เก็บไว้ใน Path สองลำดับชั้น ซึ่งสามารถปรับเปลี่ยนขนาดได้ เมื่อเนื้อที่เก็บข้อมูลเต็มก็จะทำการลบข้อมูลเก่าๆที่มีการใช้น้อยเพื่อให้มีเนื้อที่เหลือพอสำหรับเก็บข้อมูลใหม่ต่อไป



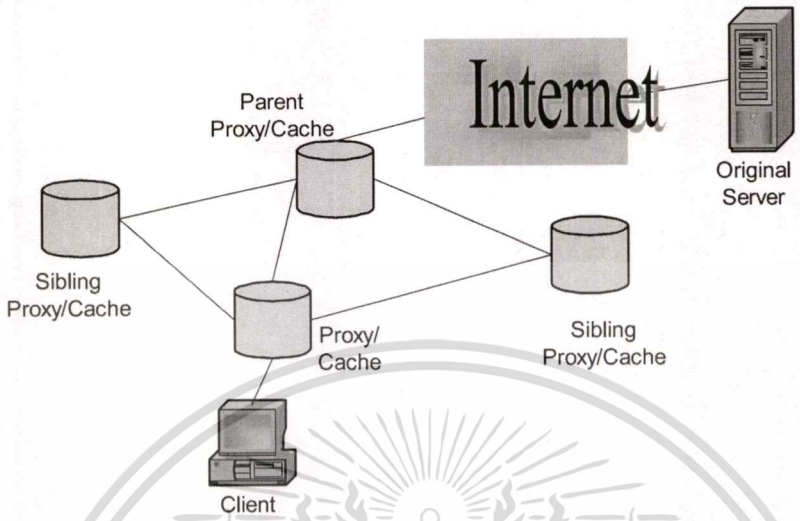
รูปที่ 2.1 การทำงานของ Proxy/Cache

ในโครงงานนี้จะไม่ได้มุ่งที่ลักษณะการเก็บ object ใน Squid มากนัก แต่จะเน้นที่การวิเคราะห์ log file ต่างๆ เพื่อจะนำมาใช้ในการพัฒนาโปรแกรมสำหรับวิเคราะห์ความซ้ำซ้อนของ Cache ต่อไป ซึ่งจะกล่าวในรายละเอียดต่อไป แต่ก่อนอื่นจะกล่าวถึงการทำงานร่วมกันของ Proxy/Cache มากกว่าหนึ่งตัวของโปรแกรม Squid

2.3.1 การทำงานร่วมกันของ Squid

โปรแกรม Squid สามารถทำให้ Proxy/Cache Server มากกว่าหนึ่งตัวทำงานร่วมกัน เพื่อให้มีประสิทธิภาพมากยิ่งขึ้น โดยทั่วไปโครงสร้างการทำงานจะเป็นแบบระดับชั้น (Hierarchy) แบ่งออกเป็น Parent Proxy / Cache และ Sibling Proxy / Cache โดยที่ Parent มักจะเป็นเครื่องที่อยู่ใกล้กับข้อมูลมากกว่า Sibling ดังรูปที่ 2.2 ซึ่งได้แสดงระดับชั้นของ Parent และSibling Proxy/Cache หลักการก็คือ เริ่มแรกผู้ใช้จะทำการส่งการร้องขอ object ไปที่ Proxy / Cache จากนั้น Proxy / Cache ก็จะทำการตรวจสอบว่ามี Object นั้นหรือไม่ ถ้าหากไม่มีก็จะทำการตรวจสอบไปยัง Cache ตัวอื่นๆที่อยู่ข้างเคียง โดยใช้โปรโตคอล ICP (Internet Caching Protocol) ในการตรวจสอบ ซึ่ง ICPเป็นโปรโตคอลที่ใช้สื่อสารกันระหว่าง Squid Proxy/Caches โดยที่Cache ข้างเคียงจะตอบกลับด้วย ICP (ICP replies) ว่าเป็น “HIT” หรือ “MISS” (HIT คือมี, MISS คือไม่มี object นั้นๆ) เมื่อ Squid ได้รับ ICP Replies จากCache ข้างเคียงแล้วก็จะเลือกเครื่องที่ตอบกลับเป็น “HIT”และตอบสนองเร็วที่สุด ในกรณีที่ไม่มีเครื่องใดตอบกลับเป็น”HIT” เลย ก็จะเลือกเครื่อง Parent ที่ตอบสนองที่เร็วที่สุด (ในกรณีที่มี Parent มากกว่าหนึ่งเครื่อง) เพราะ Parent จะเป็นเครื่องที่อยู่ใกล้เหล่งข้อมูลมากที่สุด ส่วนในกรณีที่ไม่ได้รับ ICP replies เลย คือเกิด timeout โดย default ใน Squid จะเท่ากับ 2 วินาที(สามารถเปลี่ยนค่าได้) ก็จะทำการเอามาเอง ฉะนั้น Parent Proxy / Cache มักมีภาระในการทำงานสูงจากการให้บริการ Client มากกว่า ดังนั้นส่วนใหญ่ Proxy/Cache จึงทำการ

สำเนาข้อมูลที่ได้จาก Parent เก็บไว้ด้วย แต่อย่างไรก็ตามสามารถกำหนดให้ไม่มีการเก็บข้อมูลก็ได้



รูปที่ 2.2 Parent และ Sibling Proxy/Cache

2.3.2 Log File ในโปรแกรม Squid

Log file ใน Squid จะเป็นตัวบันทึกการทำงาน และบอกถึงประสิทธิภาพของการทำงานของโปรแกรม โดย Squid จะทำการบันทึกเหตุการณ์และสถานะ การทำงานไว้ใน log file ตลอดเวลา ซึ่งSquid จะมี log file อยู่หลายตัวด้วยกัน ตัวอย่างเช่น access.log(ข้อมูลการใช้งาน),cache.log (ข้อมูลการทำงานทั่วไป) store.log(ข้อมูลobjectที่เก็บไว้), useragent.log และฯ ในที่นี้จะกล่าวเฉพาะ log file ที่จะนำมาใช้ต่อไป ซึ่งได้แก่ store.log

2.3.3store.log

ไฟล์นี้บอกถึงสถานะของ objects ว่าถูกนำเก็บลงหรือนำออกจากcache หรือดิสก์ รูปแบบของ log file นี้จะประกอบด้วยสิบเอ็ด field ซึ่งแต่ละ field จะแบ่งคั่นด้วยช่องว่าง ดังที่แสดงในรูปที่ 2.3โดยแต่ละ field จะมีความหมายดังต่อไปนี้

Time	Action	File No.	Status	Datehdr	Lastmod	Expires	Type	Sizes	Method	Key

รูปที่ 2.3 ตัวอย่าง Formatของ store.log

- Time

  Timestamp ที่ทำการบันทึกบรรทัดนี้ เป็นค่า 32-bitของ UNIX Time เป็นวินาที โดยมีความจะมีละเอียดถึง msec

- Action

  แสดงถึงว่า object ถูกดำเนินการอย่างไร มีสี่รูปแบบ

  - CREATE (ปัจจุบันถูกเลิกใช้แล้ว)
  - RELEASE (object ถูกนำออกจาก Cache)

    CacheในSquid จะเก็บ object 2 ลักษณะ คืออยู่ใน ดิสก์กับอยู่ใน Memory การที่ค่าในfieldนี้เป็น release เป็นการแสดงว่าได้ทำการปล่อย object นั้นออกไป การที่จะดูว่าเป็นการปล่อยออกจากดิสก์ หรือ Memory นั้นให้ดูที่ file number ซึ่งรายละเอียดจะกล่าวต่อไป
  - SWAPOUT (object ถูกบันทึกลงดิสก์)

    ค่าเป็น Swapout หมายถึงการนำ object ที่ได้รับเข้ามาบันทึกลง cache ในส่วนของดิสก์ เพื่อรองรับการร้องขอครั้งต่อๆไป
  - SWAPIN (object ที่อยู่บนดิสก์ถูกอ่านเข้า memory)

    ในกรณีที่มีการร้องขอ object และobjectได้อยู่ใน cache ส่วนของดิสก์แล้ว ก็จะทำการสำเนา objectไปไว้ในส่วนของ Memory ด้วย (จะถือว่า object นี้ถูกร้องขอบ่อย) เพราะการเข้าถึงข้อมูลในส่วน Memory นั้นเร็วกว่าในดิสก์ จึงเป็นการรองรับการใช้ครั้งต่อๆไป โดยสถานะของ object นี้จะอยู่ในทั้งส่วนของ Memory และดิสก์

- File Number

  เป็นเลขไฟล์สำหรับไฟล์ที่เก็บ object ซึ่งตัวเลขนี้จะถูกคำนวณขึ้นจากตามที่เราได้ config ไว้ใน cache_dir ในกรณีที่เลขไฟล์เป็น *FFFFFFFF* ก็แสดงว่าเป็น object ที่อยู่ในส่วนของ memory เท่านั้น เช่น ถ้ารหัสเป็น RELEASE แล้วเลขไฟล์เป็น FFFFFFFF ก็แสดงว่า object นั้นอยู่ใน memory เท่านั้น และได้ถูกปล่อยจาก memory

- Status

  หมายเลขแสดงสถานะของHTTP (HTTP status code) สามารถดูได้จาก RFC 2626 แต่มีบางหมายเลขที่เพิ่มเติม และไม่ใช้จะไม่ขอกล่าวในรายละเอียด

- Datehdr

  วันที่ใน HTTP reply header

- Lastmod

  ค่าของ"Last-Modified" ในHTTP reply header

- Expires

  ค่าของ"Expires" ในHTTP reply header

- Type

  แสดงชนิดของ content ของ HTTP และถ้าไม่สามารถจำแนกได้ก็จะเป็นค่า "unknown"

- Sizes (field นี้จะประกอบด้วยกันสองส่วนคั้นด้วย /)

  1. ขนาดของ advertised content จากreply header ของ HTTP "Content-Length:"
  2. ขนาดที่อ่านจริงๆ

     ถ้าค่าขนาดของ advertisedหายไป ก็จะค่าเท่ากับศูนย์ และในกรณีที่ไม่เท่ากับศูนย์และไม่เท่ากับขนาดจริง objectจะถูกปล่อย(RELEASE)จากCache

- Method

  แสดงประเภทวิธีการร้องขอ objet เช่น GET

- Key

  ชื่อที่จะชี้ไปยัง object ปกติแล้วจะเป็น URL แต่บางกรณีที่ object นั้น ไม่เป็น public ก็จะมีการใส่ตัวเลขที่ไม่ซ้ำกันและวิธีการการร้องขอไว้กับ URLด้วย

จากการวิเคราะห์ field ต่างๆใน store.log ทำให้เราทราบว่า object ไหนอยู่ใน cacheและ object ไหนถูกปล่อยไปแล้วบ้าง โดย object ที่ส่งมานั้นใน field ของ "action" เป็นค่า SWAPOUT ก็แสดงว่า object นั้นๆถูกเก็บเข้า cache แต่ถ้า เป็นค่า RELEASE แสดงว่า object นั้นๆไม่ได้ถูกเก็บเข้าไปใน cache

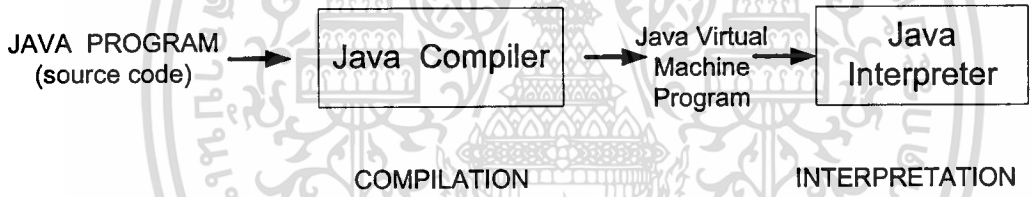
ในกรณีที่ object มีอยู่ใน cacheส่วนของดิสก์ และได้ถูกปล่อยออกจาก cacheส่วนของดิสก์ (เช่น การแทนที่เมื่อcache เต็ม) ใน field ของ "action" ก็จะมีค่าเป็น RELEASE เหมือนกัน การที่จะดูว่า object นั้นไม่ได้ถูกนำเข้า cache หรือ เป็นการเอาออกจาก cache ก็ให้ดูที่ field ถัดไปคือ field ของ "file number" ถ้า objectไม่ถูกนำมาเก็บใน cache ค่าของ file numberจะมีค่าเป็น FFFFFFFF (-1) ส่วนในกรณีที่ object ได้ถูกเอาออกจาก cache ค่าของ file numberจะเป็นค่าอื่น

2.4 เครื่องมือที่ใช้ในการพัฒนาระบบ

ในหัวข้อนี้จะกล่าวถึงเครื่องมือที่ใช้ในการพัฒนาระบบ ก่อนที่จะเริ่มต้นการพัฒนาโปรแกรมจะต้องทำการติดตั้งเครื่องมือให้เรียบร้อยก่อน ไม่ว่าจะเป็น Java compiler , Java interpreter ,Tomcat , MySQL และอื่นๆ โดยจะมีรายละเอียดดังนี้

2.4.1 โปรแกรมภาษา JAVA

ภาษาJavaนั้นเป็นภาษาแบบ OOP (Object-Oriented Programming) ซึ่งจะช่วยให้สามารถสร้างโปรแกรมที่มีความยืดหยุ่นมาก โค้ดที่สร้างแล้วจะเก็บเป็นจุด class (เป็นลักษณะ object) สามารถนำกลับมาใช้ได้อีก ภาษาJava นั้นเป็นอิสระต่อแพล็ตฟอร์ม (Platform-Independent) กล่าวคือใช้ source code ที่เขียนบน Operating System(OS) หนึ่งไปใช้อีก OSหนึ่งได้โดยไม่ต้องเขียนโค้ดใหม่ เพื่อให้โปรแกรมไม่ขึ้นกับระบบ จึงนำแนวคิด Virtual Machine มาใช้ โดย Java นั้นมีคอมไพเลอร์ทำการแปลภาษา (source code) ให้เป็นโปรแกรมของ Java Virtual Machine (JVM) แล้วนำโปรแกรมนั้นมาทำการใน JVM นั้นโดย Java Interpreter ดังรูปที่ 2.4



รูปที่ 2.4 หลักการการทำงานของโปรแกรม JAVA

ในการสร้างโปรแกรมภาษา Java ก่อนอื่นจะต้องติดตั้ง Java Developer Kit (JDK) โดยปัจจุบันเป็น version 1.4.x แล้ว ซึ่งสามารถ Download ได้แบบฟรีๆ จาก http://java.sun.com การติดตั้งก็เพียงแค่แตกไฟล์ที่ Download มา จากนั้นก็ชี้ path ไปที่ <default directory>\bin เพื่อที่จะเรียกใช้โปรแกรมใน JDK เช่น javac.exe คือ Java Compiler และ java.exe คือ Java Interpreter เป็นต้น วิธีการชี้ path นั้นจะขึ้นกับระบบปฏิบัติการที่ใช้ สามารถหาดูได้ในเอกสารของ JDKที่ http://java.sun.com จากนั้นจะต้องทำการกำหนด CLASSPATH ให้ไปให้ Java Compiler หรือ Java Interpreter ค้นหาคลาสนั้นเจอ โดยควรให้ Java ทำการค้นหาดังนี้

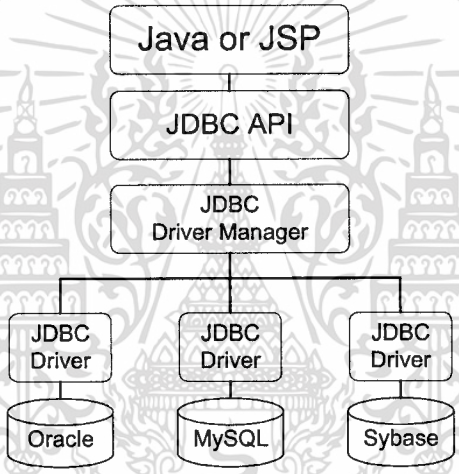
- เริ่มหาที่ Standard Classes ที่อยู่ใน rt.jar และ i18n.jar (rt.jar เป็นไฟล์ที่เก็บคลาสที่เป็นมาตรฐาน ส่วน i18n.jar เก็บคลาสที่เกี่ยวกับ internationalization)
- หากไม่พบแล้วจึงเริ่มหาคลาสที่จัดเป็นประเภท extension classes (อยู่ใน package ที่ขึ้นต้นด้วย javax)

- หากยังไม่พบจึงไปหาคลาสประเภท User defined classes ในไดเรกทอรีปัจจุบันที่ทำงาน

ที่กำหนดไว้เช่นนี้ เพียงพอต่อการทำงานทั่วไป

2.4.2 JAVA DATABASE CONTIVITY (JDBC)

JDBC คือตัวเชื่อมต่อระหว่างโปรแกรมกับฐานข้อมูล โครงสร้างของ JDBC เป็นกลุ่มของ interface ของ Java ที่เรียกว่า ไดเวอร์(Driver) ฐานข้อมูลแต่ละประเภทก็จะมีไดเวอร์ต่างกันออกไป ดังรูปที่ 2.5



รูปที่ 2.5 การติดต่อระหว่างโปรแกรม JAVA กับฐานข้อมูลผ่าน JDBC

จากรูปจะเห็นว่าเมื่อโปรแกรมต้องการติดต่อกับฐานข้อมูล ก็จะทำผ่านทาง JDBC API และ JDBC Driver Manager ส่วนไดเวอร์อาจ MySQL, Oracle หรืออื่นๆ ขึ้นกับฐานข้อมูลที่ใช้ ในที่นี้จะกล่าวถึงการติดตั้งไดเวอร์ของฐานข้อมูล MySQL เท่านั้น วิธีการก็คือ

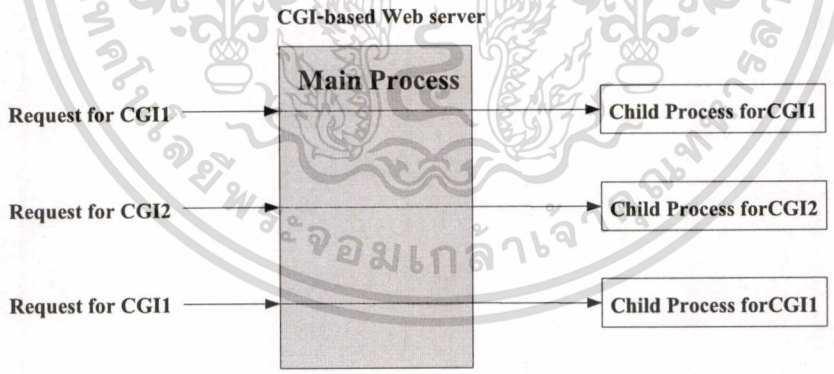
1. ให้ download MySQL Driver ในที่นี้จะเป็น MySQL Driver สามารถ download ได้ที่ http://sourceforge.net โดยไฟล์ที่ได้จะอยู่ในรูปแบบของคลาสหรือแพ็กเกจ (.jar)
2. ทำการกำหนดให้ classpath ชี้ไปยังไฟล์ .jar ที่ downloadมาได้ หรือจะนำไฟล์ .jar ที่ได้ไปไว้ที่ $JAVA_HOME/jre/lib/ext ก็ได้

เพียงเท่านี้โปรแกรม Java หรือ JSP ก็จะสามารถติดต่อกับฐานข้อมูลได้

2.4.3 SERVLET & JAVA SERVER PAGES (JSP)

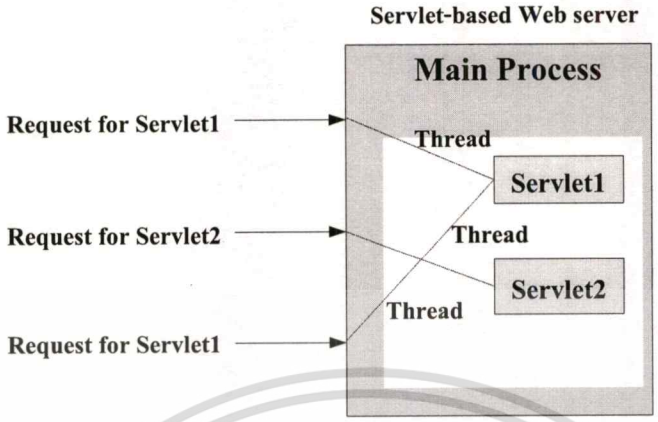
โครงงานนี้ได้พัฒนาส่วนที่ใช้ติดต่อกับผู้ใช้เป็นในลักษณะของ Web Base โดยใช้เทคโนโลยีของ JAVA (JSP) การทำ User Interface ในลักษณะของ Web Base ก็เพื่อให้ผู้ใช้ใช้งานได้สะดวกยิ่งขึ้น สามารถเรียกใช้งานได้จากทุกที่ทุกเวลา โดยผ่าน Web Browser

Servlet & JSP คือภาษาJava ที่เขียนแอพพลิเคชันบนฝั่งserver เพื่อเพิ่มประสิทธิภาพให้หน้าเว็บเพจมีความยืดหยุ่นมากยิ่งขึ้น ซึ่งอ้างอิงคอนเซ็ปมาจาก CGI ข้อดีของ Servlet ที่อยู่เหนือ CGI อย่างแรกก็คือตัวภาษาที่ใช้เขียนซึ่งก็คือ java นั่นเอง Javaนั้นเป็นภาษาแบบ OOP (Object-Oriented Programming) ซึ่งจะช่วยให้สามารถสร้างโปรแกรมที่มีความยืดหยุ่นมาก โค้ดที่สร้างแล้วจะเก็บเป็นจุด class (เป็นลักษณะ object) สามารถนำกลับมาใช้ได้อีก ภาษาJava นั้นเป็นอิสระต่อแพล็ตฟอร์ม (Platform-Independent) กล่าวคือใช้ source code ที่เขียนบน Operating System(OS) หนึ่งไปใช้อีก OSหนึ่งได้โดยไม่ต้องเขียนโค้ดใหม่ นอกจากนี้ Servlet ยังมีความเร็วที่สูงกว่า CGIอีกด้วย เนื่องจากการทำงานของ CGI นั้นเมื่อserver ได้รับการร้องขอ (request) ในการใช้งาน CGIโปรแกรมมันจะสร้างการทำงาน(new process) ขึ้นใหม่เพื่อที่จะรันโปรแกรม CGI การสร้างหนึ่งprocessต่อหนึ่งrequestนั้นเสียเวลาและสิ้นเปลืองทรัพยากร



รูปที่ 2.6 วงรอบชีวิตของ CGI (CGI life cycle)

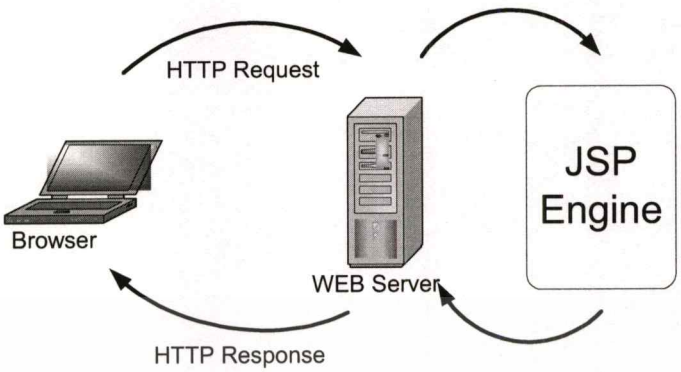
ในขณะที่ Servlet ใช้หลักการของหน่วยการทำงานย่อย(thread) โดยจะทำการสร้าง 1 thread ต่อหนึ่งrequest ที่มาจาก client ดังรูปที่ 2.7 ซึ่งแสดงวงรอบชีวิตของ Servlet



รูปที่2.7 วงรอบชีวิตของ Servlet

ถึงแม้ว่า Servlet จะอ้างอิงหลักการของ CGI อย่างไรก็ตามในการที่จะทำการรัน Servlet แล้ว ตัว web server จะไม่สามารถส่งข้อมูลไปให้ Servlet ได้โดยตรงเหมือนกับหลักการของ CGI แต่ตัว web server จะต้องเพิ่มอีกส่วนหนึ่งซึ่งเป็นส่วนที่ใช้เป็นเสมือนตัวห่อหุ้ม Servlet ต่าง ๆ ไว้ โดยส่วนที่เพิ่มขึ้นมานี้เราเรียกว่า Servlet Engine หรือ Servlet Container

การทำงานของทั้ง JSP และ Servlet ก็คล้ายกันทุกอย่าง แต่ JSP จะมีขั้นตอนที่เพิ่มขึ้นมาคือ การแปลงเอกสาร JSP ให้เป็น Servlet ก่อน การที่ JSP ได้รับความนิยมและนำมาใช้แทนที่ Servlet ก็เนื่องจากการเขียน Servlet ให้แสดงผลค่อนข้างยุ่งยากเพราะไม่สามารถใส่แท็ก (TAG) HTML แทรกเข้าไปได้ ต้องพิมพ์แท็ก HTML ออกมาเอง โดยใช้คำสั่ง out.print() แต่ถ้าเป็น JSP แล้วก็จะสามารถนำแท็ก HTML มารวมกับแท็ก JSP ได้เลย โครงสร้างของ JSP นั้นเป็นลักษณะของแท็กชนิดพิเศษที่แทรกเข้าไปในเอกสาร HTML และเปลี่ยนนามสกุลเป็นของเอกสารเป็น .JSP แทนที่จะเป็น .HTML โดยแท็กเหล่านี้ Web Browser ไม่สามารถตีความหมายได้ จะต้องนำไปประมวลผลที่ฝั่ง Server เท่านั้น (เป็นการทำงานแบบ Server Side) แล้วนำผลลัพธ์ทั้งหมดส่งกลับมายัง Web Browser ในลักษณะของเอกสาร HTML ดังรูปที่ 2.8



รูปที่ 2.8 การทำงานของ JSP

จากรูปเป็นการอธิบายการทำงานโดยรวมของ JSP การทำงานทั้งหมดจะเริ่มจาก Browser (ฝั่งผู้ใช้) ร้องขอ (HTTP REQUEST) เอกสาร JSP ไปยัง Web Server จากนั้นWeb Server การจะนำเอกสารดังกล่าวส่งต่อให้ JSP Engine (JSP Engine ทำงานอยู่บน Web Server หน้าที่หลักคือการแปลความหมายและประมวลผลเอกสาร JSP) จากนั้น JSP Engine ก็จะประมวลผลและส่งกลับมายัง Web Server เพื่อจะส่งไปให้กลับยัง Browser (HTTP RESPONSE) อีกที ในลักษณะของเอกสาร HTML

ในการที่จะทำการติดต่อสื่อสารกับ Client ตัว JSP Engine นี้จะต้องทำงานร่วมกัน Web Server ซึ่งเปรียบเสมือนฉากหน้าที่ติดต่อกับ Client อีกทีหนึ่ง JSP Engine ที่ใช้กันอยู่ในปัจจุบันก็อาจจะเป็น Apache Jserv, Apache Tomcat, Allaire Jrun, IBM Websphere, BEA Weblogic, Servlet Exec เป็นต้น ในโครงงานนี้ได้ใช้ Apache Tomcat ทำหน้าที่เป็น JSP Engine ซึ่งจะกล่าวในหัวข้อถัดไป

2.4.4 APACHE TOMCAT

Apache Tomcat เป็นที่ทำหน้าที่เป็น SERVLET & JSP Engine ที่สามารถทำหน้าที่เป็น Web Server ได้ด้วย หรือจะให้ทำการงานร่วมกับ Web Server ตัวอื่นๆก็ได้ เช่น IIS เป็นต้น การที่เลือกใช้ Apache Tomcat ในโครงงานนี้ก็เนื่องจากใช้งานง่าย กินทรัพยากรน้อย และยังไม่เสียค่าใช้จ่ายอีกด้วย การที่จะใช้งาน JSP จะต้องติดตั้งส่วนประกอบ 2 ส่วนด้วยกัน คือ

1. ติดตั้ง JDK เพื่อใช้ในการสร้างคอมโพเน็นบีน และใช้งานร่วมกับ Tomcat
2. ติดตั้ง Apache Tomcat ซึ่งในโครงงานนี้ใช้ Version 4.0.1

ในส่วนที่ 1. ได้ทำการอธิบายไว้แล้วในหัวข้อ 4.4.1 ต่อไปจะคือการติดตั้งให้ Tomcat ทำงานร่วมกับ IIS บน WIN2000 และ NT SERVICE โดยจะกล่าวในภาคผนวก ก.

2.4.5 ระบบจัดการฐานข้อมูล MySQL

ระบบจัดการฐานข้อมูล MySQL เป็นโปรแกรมที่สามารถนำมาใช้งานได้ฟรี สามารถทำงานได้ทั้งในระบบปฏิบัติการวินโดวส์ และ ยูนิกซ์ใช้ในการจัดการฐานข้อมูลเชิงสัมพันธ์ (Relation Database) ที่ทำงานในรูปแบบไคลเอนต์/เซิร์ฟเวอร์ ในการนำมาใช้งานจะต้องทำการติดตั้งส่วนของโปรแกรมที่เป็นเซิร์ฟเวอร์เพื่อให้ทำงานในเบื้องหลังของระบบปฏิบัติการหรือ เดมอนโปรเซสเพื่อดูแลจัดการฐานข้อมูล หลังจากนั้นจึงทำการสร้างฐานข้อมูล, ตาราง และ กำหนดรูปแบบของข้อมูลที่ต้องการจัดเก็บในตารางรวมถึงคีย์หลักตามที่ได้ออกแบบไว้ เพื่อใช้ในการค้นหา เพิ่ม และแก้ไขข้อมูล ในกรณีที่ระบบปฏิบัติการยังไม่ได้ถูกติดตั้งระบบจัดการฐานข้อมูลสามารถทำได้โดยการดาวน์โหลดโปรแกรมที่ตรงกับระบบปฏิบัติการที่ต้องการได้ฟรีจากเว็บไซต์ http://www.mysql.com เพื่อนำมาติดตั้งตามขั้นตอนในภาคผนวก ก. และกำหนดค่าปฏิบัติการตามที่ได้ออกแบบไว้

2.5 สรุปบท

Proxy/Cache หรือ Web caching คือการเก็บสำเนาข้อมูลของ website ที่เคยเข้าไปเยี่ยมชมมาแล้วในลักษณะของ object เช่น ไฟล์HTML รูปภาพ และอื่นๆ เอาไปเก็บไว้ใน memory หรือ hard disk และเมื่อจะใช้อีกครั้ง ก็ไม่ต้องไปดึง จาก website ต้นฉบับ ทำให้ลดปริมาณ ข้อมูลที่วิ่งอยู่ใน network และ ผู้ใช้ได้ข้อมูลเร็วขึ้น ลดความคับคั่งในเครือข่ายอินเตอร์เน็ตได้เป็นอย่างดี แต่มีข้อเสียคือข้อมูลอาจเกิดความล้าสมัยเนื่องจาก ข้อมูลต้นฉบับอาจมีการเปลี่ยนแปลงหลังจากที่ได้ทำสำเนาเก็บไว้แล้ว จึงต้องมีการตรวจสอบความถูกต้องของข้อมูลด้วย

Squid โปรแกรมที่ทำหน้าที่เป็นProxy/Cache Server ที่มีการใช้กันอย่างแพร่หลาย เพื่อให้มีประสิทธิภาพมากยิ่งขึ้น สามารถใช้ Proxy/Cache Server มากกว่าหนึ่งตัวทำงานร่วมกัน โดยทั่วไปโครงสร้างการทำงานจะเป็นแบบระดับชั้น (Hierarchy) แบ่งออกเป็น Parent Proxy / Cache และ Sibling Proxy / Cacheโดยsquid จะใช้โปรโตคอล ICP (Internet Caching Protocol) ในการตรวจสอบ ซึ่ง ICPเป็นโปรโตคอลที่ใช้สื่อสารกันระหว่าง squid caches

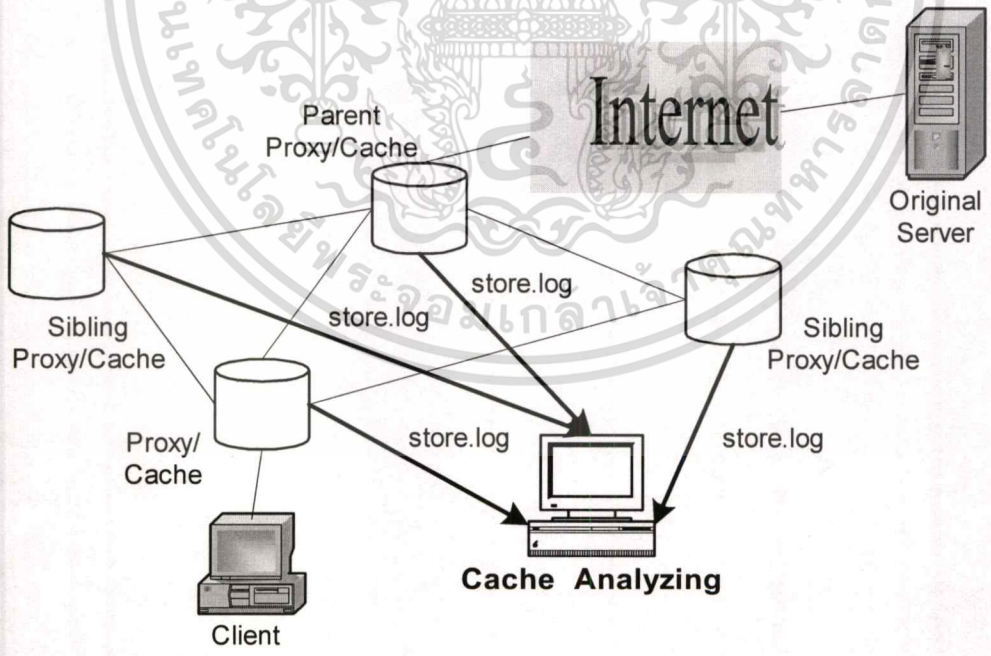
Squid จะเป็นตัวบันทึกการทำงาน และบอกถึงประสิทธิภาพของการทำงานของโปรแกรม เหตุการณ์และสถานะ การทำงานไว้ใน log file ตลอดเวลา ซึ่งSquid จะมี log file อยู่หลายตัวด้วยกัน ตัวอย่างเช่น access.log(ข้อมูลการใช้งาน),cache.log (ข้อมูลการทำงานทั่วไป) store.log (ข้อมูลobjectที่เก็บไว้), useragent.log โดยที่ Store.log เป็น log file บอกถึงสถานะของ objects ว่าถูกนำเก็บลงหรือนำออกจากcache หรือดิสก์

บทที่ 3
วิเคราะห์ ออกแบบระบบงาน

ในบทนี้จะกล่าวถึงรายละเอียดของการวิธีการที่ใช้ในการวิเคราะห์ความซ้ำซ้อนของ object การออกแบบโปรแกรม และขั้นตอนการพัฒนาระบบ รวมถึงเครื่องมือที่ใช้ในการพัฒนาโปรแกรม

3.1 การวิเคราะห์และออกแบบ

เพื่อที่จะสามารถทำการวิเคราะห์ Log File จำเป็นที่จะต้องมีเครื่องที่เป็นศูนย์กลางที่ใช้สำหรับวิเคราะห์ การออกแบบจะเป็นในลักษะของการใช้เครื่องหนึ่งเครื่องเป็นศูนย์กลางเป็นตัวที่จะทำการวิเคราะห์ทั้งหมด รวมถึงเป็นส่วนที่ติดต่อกับผู้ใช้งานด้วย โดยที่เครื่องศูนย์กลางนี้จะทำการติดต่อกับ Proxy / Cache Server ทุกๆตัวในเครือข่ายเพื่อนำ store.log มาวิเคราะห์และดำเนินการต่อไป ดังรูปที่ 3.1



รูปที่ 3.1 การเชื่อมต่อและร้องขอ store.log

ในการออกแบบนั้นจะต้องทำความเข้าใจสถานะต่างๆ ของ object ก่อนว่า ณ. เวลานั้น object จะถูกเก็บอยู่ในสถานะใด ซึ่งจะกล่าวถึงสถานะ/วงจร (State Diagram) ของ object สำหรับการเก็บ object เพื่อให้มองเห็นภาพที่ชัดเจน

3.2 การวิเคราะห์สถานะของ object

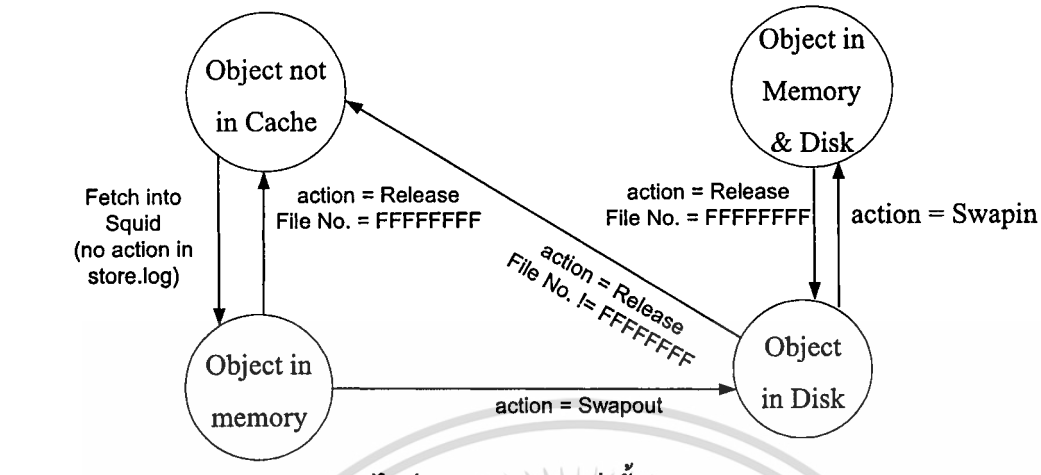
จากที่ได้กล่าวถึงการทำงานของ Squid ไว้ตอนต้นแล้ว ทำให้ทราบถึงกระบวนการในการส่งobject นั้นๆให้กับผู้ที่ร้องขอมา ซึ่งตอนนี้จะกล่าวถึงสถานะของ object ตั้งแต่รับ object มาจากแหล่งข้อมูลจริงๆ การเก็บไว้ใน cache จนกระทั่งปล่อย objectนั้นออกจาก cache ดังที่แสดงในรูปที่ 3.2 โดยเริ่มแรก Squid จะทำร้องขอ object ไป ซึ่งในสถานะแรกนั้น object ยังไม่ได้อยู่ใน cache จากนั้นเมื่อได้รับ object มาก็จะเก็บไว้ในส่วนของ memory ซึ่งพอถึงตอนนี้แล้ว สถานะของ object จะแบ่งออกเป็น 2 กรณีใหญ่ๆ

1.กรณี object ไม่ได้ถูกเก็บลงใน Cache

ในกรณีนี้เมื่อ squid ได้รับ object แล้ว และไม่ทำการเก็บลงในดิสก์ object นี้ก็จะอยู่ในส่วนของ memory เท่านั้น จากนั้นก็จะปล่อยออกไป ดังเช่นในรูปที่ 3.2 การปล่อย Object ออกจาก memoryนั้นสามารถทราบได้โดยการพิจารณาจาก Store. log ซึ่งถ้า field ของ action และ file no. เป็น Release และ FFFFFFFF ตามลำดับ ก็แสดงว่าเป็นการปล่อย object ออกจากส่วนของ memory

2.กรณี Object นั้นได้ถูกเก็บลงใน Cache

ในกรณีนี้เมื่อ Squid ได้รับ Object แล้วก็จะทำการเก็บลงในดิสก์ ดังที่แสดงในรูปที่ 3.2 การที่จะทราบว่า Objectไหนได้ถูกบันทึกลงในส่วนของดิสก์นั้น ก็สามารถทราบได้โดยการพิจารณาจาก store. log ถ้า field action เป็น Swapout แสดงว่า object ได้เปลี่ยนสถานะจากที่อยู่ใน memory ไปอยู่ในส่วนของดิสก์ (ให้ดูรูปที่ 3.2 ประกอบ) จากนั้นถ้ามีการถูกร้องขอ objectนี้อีก โปรแกรม squid จะทำการสำเนา object นี้ไปยังส่วนของ memory กล่าวคือ object ก็จะอยู่ในทั้งในส่วนของ memory และดิสก์ ซึ่งในสถานะนี้ field action ใน store. log จะเป็น Swapin จากนั้นเมื่อobjectนี้ไม่ได้ถูกเรียกใช้เป็นระยะเวลาหนึ่ง object นี้จะถูกปล่อยออกจาก memory และจะอยู่ในส่วนของดิสก์อย่างเดียว โดยที่ใน store. log ที่บอกถึงสถานะนี้ field actionและ file no. จะเป็น Release และ FFFFFFFF ตามลำดับ และถ้าในสถานะที่ object อยู่ในดิสก์ และobject ไม่ได้ถูกเรียกใช้เป็นช่วงระยะเวลาหนึ่ง objectก็จะถูกปล่อยออกไปเลย (เช่นในการแทนที่ของ object เมื่อ cache เต็ม) โดยใน store. log ที่บอกถึงสถานะนี้ field actionจะเป็น release และ field ของ file no. จะค่าที่ไม่เท่ากับ FFFFFFFF



Object in memory: Object อยู่ในส่วนของ Memory เท่านั้น

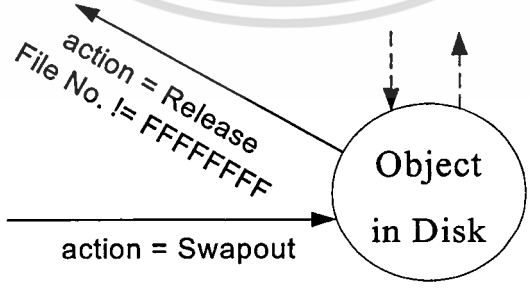
Object in disk: Object อยู่ในส่วนของ disk เท่านั้น

Object not in cache: Object ไม่อยู่ทั้งในส่วนของ Memory และดิสก์

Object in memory & disk: Objectอยู่ทั้งในส่วนของ Memory และดิสก์

รูปที่ 3.2 สถานะของ Object (Object's State Diagram)

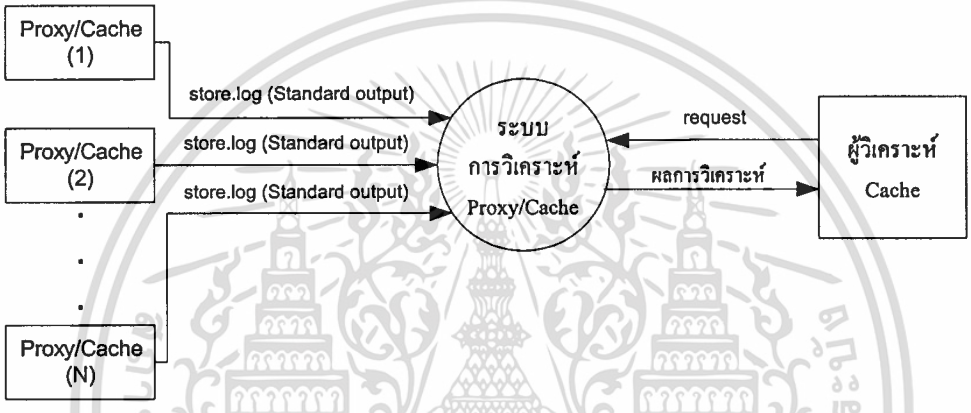
จากที่ได้กล่าวมาทำให้เห็นภาพของสถานะของ Object ที่นำมาเปรียบเทียบกับ field ของ store.log ชัดเจนขึ้น การที่จะดูว่า ใน proxy/cache นั้นมี objectไหนอยู่บ้าง ในที่นี้จะพิจารณาในส่วนของ object ที่เก็บในดิสก์เท่านั้น ซึ่งfield ของ store.log ที่เกี่ยวข้องก็คือ field action เป็น Swapout แสดงว่า object ได้เก็บลงดิสก์(ดังในรูปที่ 3.3) กับ field actionเป็น release และ field ของ file no. จะค่าที่ไม่เท่ากับ FFFFFFFF แสดงว่าปล่อยออกจากดิสก์ ในส่วนของเส้นประในรูป 3.3 หมายถึงส่วนที่ทำสำเนาเข้าและปล่อยออกจากส่วนของ memory แต่ก็ยังคงมีอยู่ในส่วนของดิสก์



รูปที่ 3.3 Field ใน store.log ที่เกี่ยวข้องกับการเก็บ object ในดิสก์

3.3 กระบวนการทำงานของระบบ

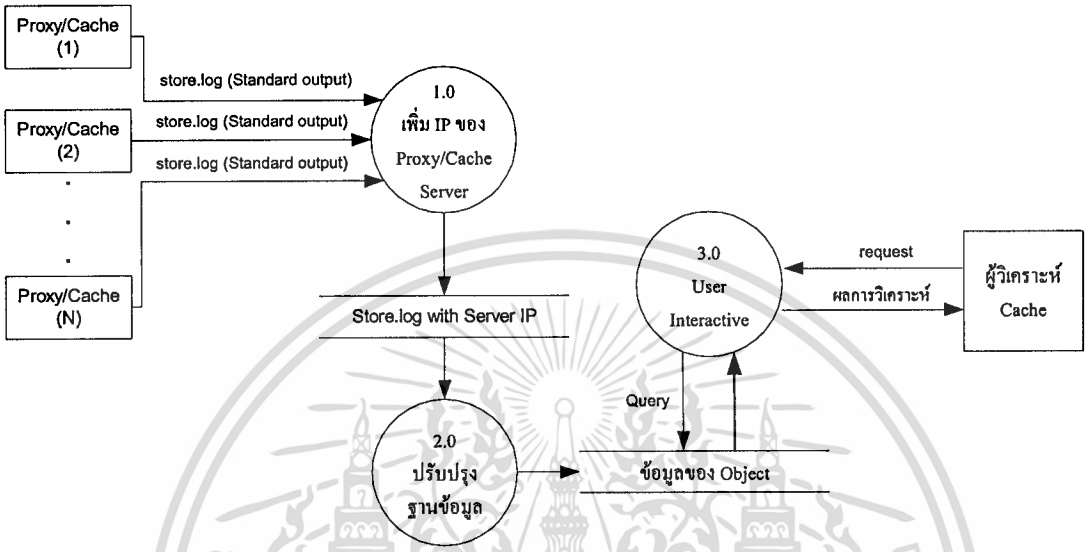
จากที่ได้ออกแบบไว้เบื้องต้น คือมีเครื่องที่ศูนย์กลางใช้สำหรับวิเคราะห์ โดยที่เครื่องศูนย์กลางนี้จะทำการติดต่อกับ Proxy/Cache Server ทุกๆตัวในเครือข่ายเพื่อนำ store.log จากมาวิเคราะห์ สามารถนำมาเขียนเป็น Context Diagram ดังรูปที่ 3.4 โดย Context Diagram นี้ได้อธิบายให้เห็นถึงภาพโดยรวม ซึ่งมีการเชื่อมต่อกับสิ่งภายนอกคือ Proxy/Cache และผู้วิเคราะห์ Cache



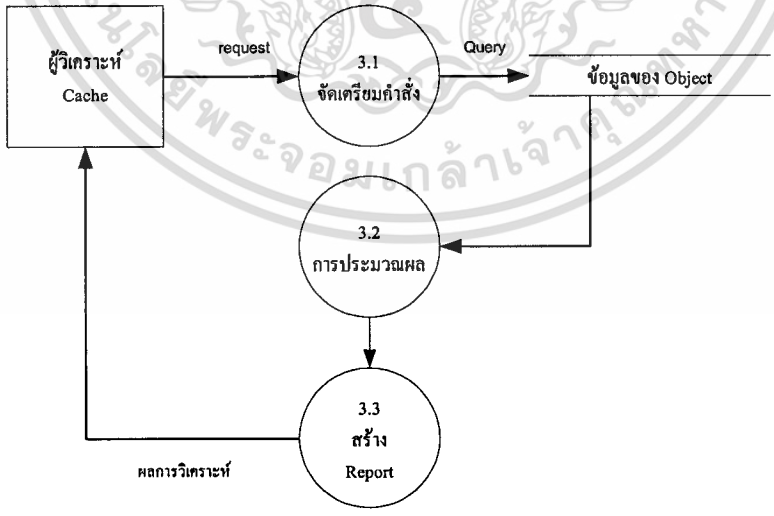
รูปที่ 3.4 แผนภาพ Context Diagram ของระบบ

เพื่อได้เห็นภาพของระบบมากขึ้น จะทำการแตกContext Diagram ข้างต้น ออกเป็นกระบวนการย่อยในแผนภาพการไหลของข้อมูลระดับที่ 1 (Data Flow Diagram Level 1) ดังรูปที่ 3.5 โดยแผนภาพการไหลของข้อมูลระดับที่ 1 ระบบนี้จะสามารถแบ่งการทำงานออกเป็น 3 ส่วน ได้แก่ กระบวนการที่ 1 เป็นการเพิ่ม IP ของเครื่อง Proxy/Cache Server แต่ละเครื่อง ให้กับ store.log ที่รับเข้ามา เนื่องจากใน store.log นั้นไม่มี field ใดเลยที่ระบุ ว่า store.log นั้นเป็นของ Proxy / Cache Server ของตัวเอง จึงจะต้องทำการระบุว่าได้ store.log มาจากตัวใด กระบวนการที่ 2 เป็นการปรับปรุงแก้ไขฐานข้อมูล คือนำ store.log ที่ได้จากกระบวนการแรกที่ได้เพิ่ม IP ของเครื่องแล้ว นำมาวิเคราะห์ถึงสถานะของ object ณ Time Stamp ขณะนั้น และทำการเก็บลงในฐานข้อมูลเพื่อรองรับในกระบวนการที่ 3 ที่จะมาเรียกใช้อีกทีหนึ่ง และเมื่อจบในกระบวนการที่2 แล้วก็จะได้ฐานข้อมูลเกี่ยวกับ object เช่น ในขณะนั้น object ได้ถูกเก็บนั้นอยู่ที่เครื่องใด โดยจะไว้ค่อยกล่าวถึงการออกแบบฐานข้อมูลในรายละเอียดต่อไป และในกระบวนการที่ 3 เป็นส่วนที่ติดต่อกับผู้ที่ทำการวิเคราะห์ (ผู้ใช้งานระบบ) ซึ่งส่วนนี้คือส่วนที่รับคำสั่งจากผู้ใช้ ว่าต้องการวิเคราะห์ในลักษณะ

ใด ซึ่งในกระบวนการนี้สามารถแตกออกเป็นกระบวนการย่อยๆในแผนภาพการไหลของข้อมูล ระดับที่ 2 (DFD Level 2) ในรูปที่ 3.6



รูปที่ 3.5 แผนภาพการไหลของข้อมูลระดับที่ 1 (DFD 1)



รูปที่ 3.6 แผนภาพการไหลของข้อมูลระดับที่ 2 (DFD 2) ของกระบวนการที่ 3.

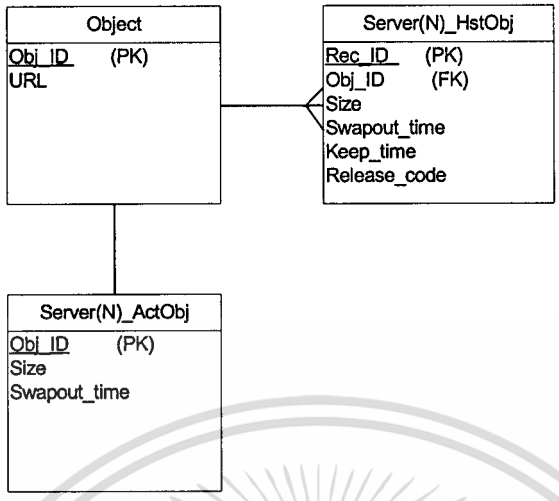
จากแผนภาพการไหลของข้อมูลระดับที่ 2 ระบบนี้สามารถแบ่งการทำงานออกเป็น 3 ส่วน ได้แก่ กระบวนการที่ 3.1 คือการจัดเตรียมคำสั่ง โดยจะเริ่มจากการมีการสั่งงานจากผู้วิเคราะห์ระบบ ส่งการร้องขอไปยังกระบวนการที่ 3.1 เพื่อสั่งงานตามที่ผู้ใช้ต้องการ และทำการ query ไปยังฐานข้อมูลของ object สำหรับข้อมูลที่จะต้องใช้ในการประมวลผลคำสั่งนั้นๆ กระบวนการที่ 3.2 เป็นส่วนของการประมวลผล(การคำนวณ)ข้อมูลที่ได้จากการ Query ในกระบวนการที่ 3.1 จากนั้นจะส่งต่อไปยัง กระบวนการที่ 3.3 ซึ่งในกระบวนการนี้ เป็นการนำผลที่ได้จากการประมวลผลในกระบวนการที่ 3.2 มาจัดทำเป็น report ในการแสดงผลให้กับผู้ที่ทำการวิเคราะห์ Cache

3.4 การออกแบบฐานข้อมูล

จากที่ได้กล่าวถึง store.log ข้างต้น จะเห็นว่ามี field อยู่หลายตัวที่จะนำมาใช้ในการออกแบบ ซึ่งได้แก่

1. Time Stamp เพื่อที่จะได้รู้ว่าเหตุใดเกิดขึ้นก่อนหลัง และการวิเคราะห์ต่างๆในช่วงเวลาต่างๆ ได้
2. Action เพื่อที่จะได้รู้ว่า object ถูกนำเข้า หรือออกจาก Cache
3. File Number เพื่อใช้ในการจำแนกว่า object นั้นไม่ถูกนำเข้า Cache หรือได้ถูกปล่อยออกจาก Cache
4. Size เพื่อให้รู้ขนาด(Byte)
5. Key เพื่อที่จะรู้ว่า object มาจากไหน

ลักษณะของฐานข้อมูลของ Object จะประกอบด้วยกันสามส่วนหลัก ดังในรูปที่ 3.7 ซึ่งได้แสดง ER - Diagram ของการออกแบบฐานข้อมูล



รูปที่ 3.7 แสดง Entity Relationship Diagram (ERD)

ERD ในรูปที่ 3.7 จะประกอบด้วยกันอยู่ 3 ส่วน โดยส่วนแรกได้แก่ Object Entity ซึ่งจะประกอบด้วยกัน 2 field ได้แก่ Object ID (Obj_ID) และ URL (กำหนด ID ให้กับ Object) ส่วนที่สองคือ Server(N)_ActObj (Server(N)_Active Object) ซึ่งจะการเก็บว่า Object ตัวใดบางที่ได้เก็บอยู่ใน Cacheส่วนของดิสก์ใน Proxy/Cache แต่ละตัว โดยจำนวนของตารางนี้ในฐานข้อมูลก็จะขึ้นกับจำนวนของ Proxy/Cache เช่น ในเครือข่ายมี Proxy/Cache มีจำนวนสามตัว ในฐานข้อมูลก็จะมีตาราง Active Object อยู่สามตาราง โดยตารางนี้จะประกอบด้วยกัน 3 field ได้แก่ Object_ID Size(ขนาดของ Object) และ Swapout_time (เป็นเวลาที่บันทึก Object นั้นลงในดิสก์) และส่วนสุดท้ายคือ Server(N)_HisObj (Server(N)_History Object) ซึ่งจะการเก็บประวัติของ Object (Object's History) ที่เอาออกจะ Cache แล้วของ Proxy/Cache แต่ละตัว จำนวนของตารางนี้ในฐานข้อมูลก็จะเท่ากับจำนวนของ Proxy/Cache โดยตารางนี้จะประกอบด้วยกัน 6 field ได้แก่ Record ID Object_ID Size Swapout_time Keep_time (เป็นระยะเวลาที่ objectนั้นอยู่ใน Cache ซึ่งจะเท่ากับ Release time stamp – Swapout time stamp) Release_code (เป็น Code ที่บอกว่า object นั้นถูก Release ในลักษณะใดเช่น Release เพราะจากการแทนที่เมื่อดิสก์เต็ม หรือ object expireเป็นต้น) ส่วนของ history ทำให้สามารถติดตามพฤติกรรมของ object ได้เช่น ความถี่ที่ object เข้าออก cache ระยะเวลาที่ object อยู่ใน cache เป็นต้น

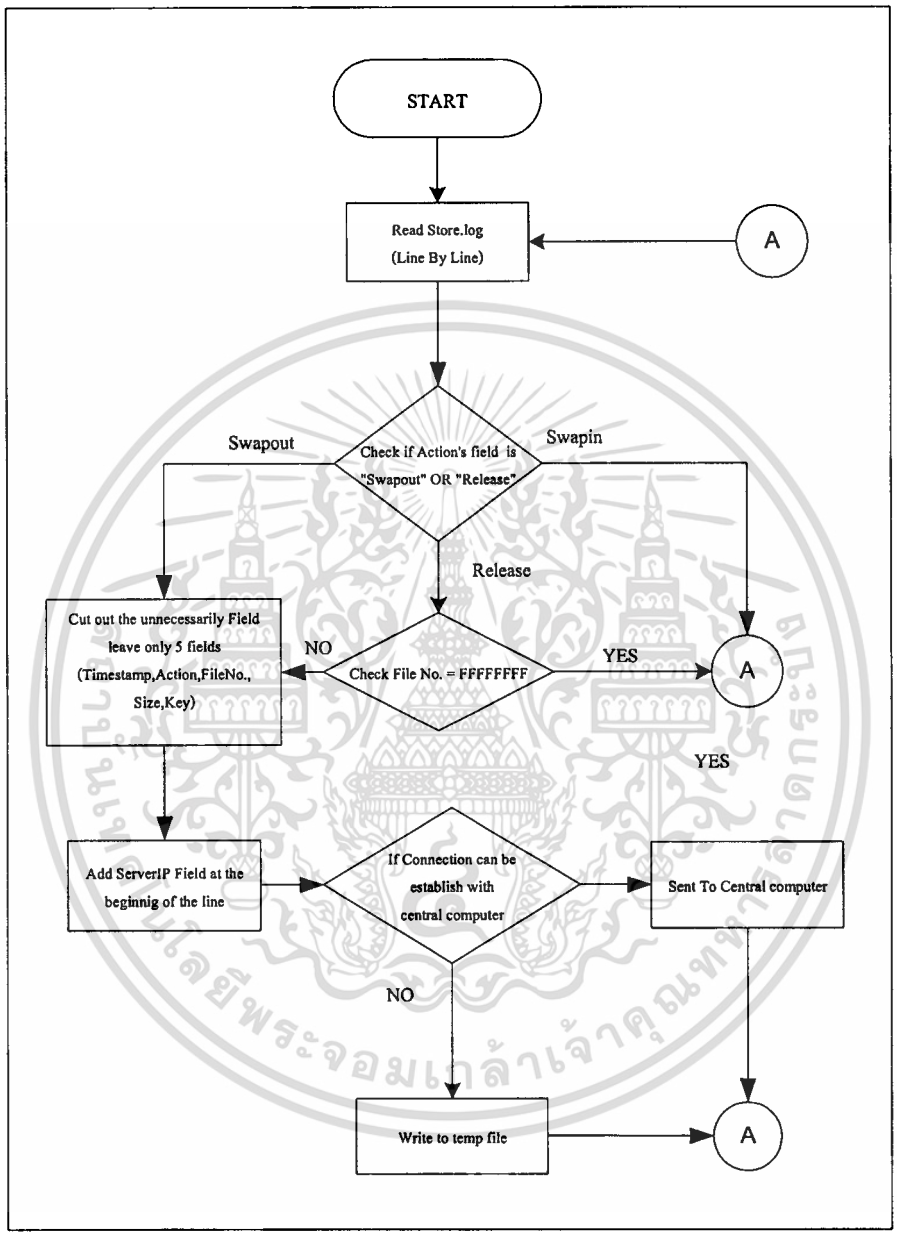
การที่ออกแบบฐานข้อมูลในลักษณะนี้ทำการวิเคราะห์ได้หลายๆอย่าง ได้แก่

- จำนวน object ของแต่ละ Proxy/Cache ในช่วงเวลาหนึ่งๆ
- ปริมาณของ object ที่ซ้ำซ้อนกัน
- ระยะเวลาโดยเฉลี่ยที่ object นั้นอยู่ใน server แต่ละตัว
- Proxy/Cache ตัวไหนที่มี object ที่ซ้ำซ้อนมาที่สุด และน้อยที่สุด

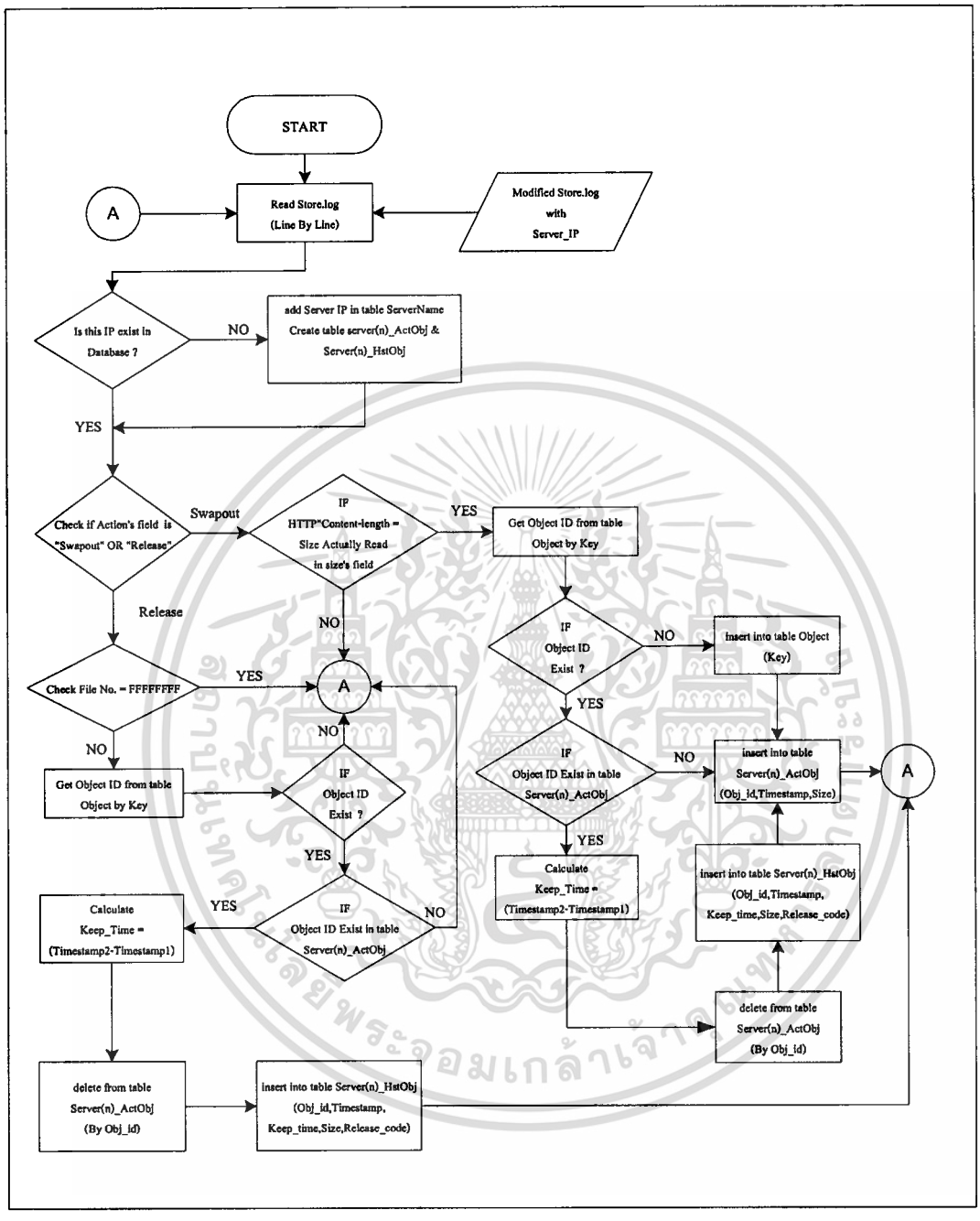
3.5 อัลกอริทึมการทำงานของระบบ

อัลกอริทึมในส่วนของการจัดเตรียมข้อมูล และใส่ Server IP (Process ที่1ในรูปที่ 3.5) ได้แสดงด้วย Flow Chart ในรูปที่ 3.8 โดยเริ่มต้นคืออ่าน Store.log มาทีละบรรทัด จากนั้นก็ตรวจสอบดูว่าใน Action's field มีค่าเป็นอะไร ถ้าเป็น Swapin ก็จะไม่ดำเนินการใดๆกับบรรทัดนั้น และทำการอ่านบรรทัดใหม่ต่อไป แต่ถ้าเป็น Release ก็ต้องตรวจสอบว่า File No's field เป็น FFFFFFFF หรือไม่ ถ้าเป็นก็จะไม่ดำเนินการใดๆกับบรรทัดนั้น และทำการอ่านบรรทัดใหม่ต่อไป ในกรณีที่ Action's field เป็น Swapout หรือเป็น Release ที่มี File No's field ไม่เป็น FFFFFFFF ก็จะทำการจัดรูปแบบให้พร้อมในการที่จะส่งต่อให้กับส่วนการประมวลผล ซึ่งจะใช้ 5 field ด้วยกันได้แก่ Timestamp, Action, File Number, Size และ Key นอกเหนือจากนี้ทำการตัดทิ้ง จากนั้นก็ใส่ Server IP ข้างหน้าของบรรทัดนั้น เพื่อให้ทราบว่าส่งมาจาก Serverตัวใด จากนั้นก็จะทำการส่งให้ส่วนของการประมวลผลซึ่งจะเป็นอีกเครื่องหนึ่ง โดยถ้าไม่สามารถส่งได้ก็จะทำการเขียนเป็น Temp file และเมื่อสามารถส่งได้จึงจะทำการส่งไป

อัลกอริทึมในส่วนของการนำเอา Store.log ได้รับมาจาก Process ที่1 มาประมวลและจัดเก็บลงฐานข้อมูลที่ได้กล่าวไว้แล้วในหัวข้อที่3.4 (Process ที่2ในรูปที่ 3.5) ได้แสดงด้วย Flow Chart ในรูปที่ 3.9 เริ่มโดยคืออ่าน Store.log ที่รับมาจาก Process ที่1 ทีละบรรทัด ทำการตรวจสอบดู Store. log ถูกส่งมาจาก Server ตัวใด โดยการดูที่ Server IP แล้วตรวจสอบว่า Server ตัวนี้มีข้อมูลอยู่ฐานข้อมูลหรือยัง ถ้ายังก็จะทำการสร้างตารางที่ต้องใช้ขึ้นมา จากนั้นตรวจว่าใน Action's field มีค่าเป็นอะไร ถ้าเป็น Swapout ก็หมายถึงการที่ Squid ได้นำ Object นั้นใน Cache ถ้าเป็น Release ก็แสดงว่าได้ปล่อย Object นั้นออกจาก Cache ขั้นตอนการทำการทั้งหมดได้สรุปดังรูปที่ 3.9 โดยการหา Keep Time ก็คือ Timestamp2 – Timestamp1 ซึ่ง Timestamp2 คือ Timestamp ที่ของการ Relasee ส่วน Timestamp1 คือ Timestamp ที่นำ Object เข้า Cache



รูปที่ 3.8 Flow Chart แสดงอัลกอริทึมการทำงานของระบบใน Process ที่ 1



รูปที่ 3.9 Flow Chart แสดงอัลกอริทึมการทำงานของระบบใน Process ที่ 2

3.6 สรุปบท

จากการศึกษาสถานะของ Object ที่นำมาเปรียบเทียบกับ field ของ store.log ทำให้เห็นภาพที่ชัดเจนขึ้นในการนำเอา Store.log มาวิเคราะห์ความซ้ำซ้อนของ Object ระหว่าง Proxy Server แต่ละเครื่องได้ การที่จะดูว่า ใน proxy/cache นั้นมี objectไหนอยู่บ้าง ในที่นี้จะพิจารณาในส่วนของ object ที่เก็บในดิสก์เท่านั้น ซึ่งfield ของ store.log ที่เกี่ยวข้องก็คือ field action เป็น Swapout แสดงว่า object ได้เก็บลงดิสก์(ดังในรูปที่ 3.3) กับ field actionเป็น release ที่ค่าใน field ของ file no. มีค่าไม่เท่ากับ FFFFFFFF

ได้ออกแบบระบบในลักษณะของการที่มีเครื่องหนึ่งเครื่องเป็นศูนย์กลางที่คอยรับ Store.log จาก Proxy / Cache Server ทุกๆตัวในเครือข่ายเพื่อนำ Store.log มาวิเคราะห์ ประมวลผลและจัดเก็บลงฐานข้อมูล การออกแบบระบบได้ออกแบบให้มีส่วนประกอบหลักๆอยู่สามส่วน คือ ส่วนที่ทำงานบนฝั่ง Squid Proxy Server ทำหน้าที่รอรับ Store.log ใหม่ที่เกิดขึ้น จัดการให้ Store.log อยู่ใน Format ที่สามารถนำไปประมวลผลต่อได้ และส่งไปให้เครื่องที่ใช้ทำการประมวลผล ส่วนที่สองคือส่วนที่ทำงานบนฝั่งเครื่องที่ใช้วิเคราะห์ทำหน้าที่รับ Store.log จาก Proxy / Cache Server ทุกๆตัวในเครือข่ายเพื่อนำ Store.log มาวิเคราะห์ ประมวลผล และจัดเก็บลงฐานข้อมูล และส่วนสุดท้ายเป็นส่วนของ User Interface ทำหน้ามี่จัดเตรียมคำสั่ง ทำการ query ไปยังฐานข้อมูล นำผลที่ได้จากการประมวลผล และส่งผลให้กับผู้ใช้ในลักษณะที่เข้าใจง่าย

บทที่ 4
ขั้นตอนการพัฒนาระบบงาน

ในบทที่แล้วได้กล่าวถึงรายละเอียดของการออกแบบโปรแกรม โดยในบทนี้จะกล่าวถึงขั้นตอนการพัฒนา รวมถึงเครื่องมือที่ใช้ในการพัฒนาโปรแกรมดังกล่าว โดยการพัฒนาระบบในที่นี้ได้แบ่งออกเป็นสองส่วนใหญ่ๆ ก็คือส่วนที่ใช้ในการรับอ่านค่าจาก Store.log เพื่อนำมาแปรความหมายและจัดเก็บไว้ในฐานข้อมูล และส่วนที่ติดต่อกับผู้ใช้ ซึ่งจะพัฒนาขึ้นในลักษณะของ Web Base โดยใช้เทคโนโลยีของ JAVA (JSP)

4.1 การพัฒนาระบบส่วนประมวลผลและจัดเก็บลงฐานข้อมูล

การทำงานของโปรแกรมส่วนที่1.นั้นจะต้องทำหน้าที่อ่านข้อมูลที่ต้องการจากโปรแกรม Squid ดังนั้นจึงออกแบบให้มีโปรแกรมส่วนหนึ่งอยู่บนฝั่ง Proxy Server ที่โดยส่วนใหญ่จะเป็นระบบ Unix ซึ่งจะทำหน้าที่รับข้อมูลเข้ามาจากทาง Standard Input ของ UNIX แล้วทำการส่งมายังเครื่องที่เป็นศูนย์กลาง และอีกส่วนหนึ่งอยู่ที่เครื่องศูนย์กลางทำหน้าที่รับค่า Store.log จากเครื่อง Proxy Server ต่างๆ มาทำการประมวลผลและเก็บลงฐานข้อมูลดังที่ออกแบบไว้ในบทที่แล้ว โดยภาษาที่ใช้ในการพัฒนาคือภาษา JAVA เนื่องจากภาษาJAVAนั้นเป็นภาษาแบบ OOP (Object-Oriented Programming) ซึ่งจะช่วยให้สามารถสร้างโปรแกรมที่มีความยืดหยุ่นมาก โค้ดที่สร้างแล้วจะเก็บเป็นจุด class (เป็นลักษณะ object) สามารถนำกลับมาใช้ได้อีก ภาษาJava นั้นยังเป็นอิสระต่อแพล็ตฟอร์ม (Platform-Independent) กล่าวคือใช้ source code ที่เขียนบน Operating System(OS) หนึ่งไปใช้อีก OSหนึ่งได้โดยไม่ต้องเขียนโค้ดใหม่ และในส่วนของฐานข้อมูลนั้นในโครงงานนี้จะใช้โปรแกรม MySQL ทำหน้าที่เป็นฐานข้อมูล

4.1.1 โปรแกรมส่วนที่ทำงานบน Proxy Server

โปรแกรมส่วนที่ทำงานบน Proxy Server จะรับ Store.log ที่เพิ่มขึ้นผ่านทาง Standard Input จากนั้นก็จะทำการตรวจสอบว่าขณะนั้นได้มี Connection อยู่กับเครื่องที่เป็นศูนย์กลางอยู่หรือไม่ ถ้ามีก็จะทำการส่งผ่านไปให้ แต่พบว่าขณะนั้นไม่มี Connection อยู่ก็จะไม่สามารถทำการส่งไปให้ได้จึงจำเป็นที่จะทำการจัดเก็บลงไฟล์ และเมื่อไหร่ที่เครื่องศูนย์กลาง Connect เข้ามาจึงค่อยส่งไปให้ และเพื่อไม่ให้เป็นการส่งข้อมูลมากเกินไป ตัวโปรแกรมนี้จะทำการ

จัดเก็บลงไฟล์ และเมื่อไหร่ที่เครื่องศูนย์กลาง Connect เข้ามาจึงค่อยส่งไปให้ และเพื่อไม่ให้เป็นการส่งข้อมูลมากเกินไป ตัวโปรแกรมนี้จะทำการคัดบรรทัดของStore.log ที่เป็นField ของ action เป็น Swapin หรือ Release ที่มี fieldของ File Number เป็น FFFFFFFF ออก และโปรแกรมยังจะทำการตัด field ที่ไม่จำเป็นของ Store. log ออก คือจาก 11 Field ในStore.log เหลือเพียง 5 Field และเพิ่มส่วนที่เลข IP ของ Proxy Server อีกหนึ่ง field ได้แก่ ServerIP,Time Stamp, Action ,File Number ,Size และ Key ดังรูปที่ 4.1

ServerIP	TimeStamp	Action	File Number	Size	Key

รูปที่ 4.1 Format ของ Store.log ที่ไว้ปรับแล้ว

4.1.2 โปรแกรมส่วนที่ทำงานบน เครื่องศูนย์กลาง

ส่วนของโปรแกรมที่อยู่บนเครื่องศูนย์กลางจะทำหน้าที่รับค่า Store.log จากเครื่อง Proxy Server ต่างๆ มาทำการประมวลผลและเก็บลงฐานข้อมูล โปรแกรมส่วนนี้ได้ใช้ภาษาJAVAในการพัฒนา โดยการทำงานของโปรแกรมนั้นจะเริ่มโดยการสร้าง Connection ไปยังโปรแกรมที่อยู่บนฝั่ง Proxy Server ทุกๆตัว เมื่อสร้างConnection สำเร็จโปรแกรมที่อยู่บนฝั่ง Proxy Server ก็จะตรวจสอบว่ามี Log File ที่ยังไม่ได้ส่งไปหรือไม่ ถ้ามีก็จะทำการส่งไป เมื่อได้รับค่าเรียบร้อยแล้วก็จะทำการปิด Connection ดังกล่าว แล้วนำค่าที่รับเข้ามาทำการวิเคราะห์เพื่อจัดเก็บลงฐานข้อมูลโดยใช้ JDBC (JAVA DATABASE CONNECTIVITY) ติดต่อผ่านทาง MySQL ODBC หรือ MySQL Driver ซึ่งรายละเอียดในการที่จะให้ภาษา JAVA ติดต่อกับฐานข้อมูลซึ่งในที่นี้คือ MySQL จะกล่าวในหัวข้อต่อๆไป หลังจากที่จัดเก็บค่าต่างๆลงในฐานข้อมูลเรียบร้อยแล้ว ก็จะเปิด Port 3456 เพื่อให้โปรแกรมทาง ฝั่ง Proxy Server สามารถส่งค่ามาให้เรื่อยๆเมื่อมีการเปลี่ยนแปลงของ Store. log

4.2 การพัฒนาระบบส่วนติดต่อกับผู้ใช้

ส่วนนี้จะเป็นส่วนที่ติดต่อกับผู้ใช้(User Interface) ซึ่งจะพัฒนาขึ้นในลักษณะของ Web Base โดยใช้เทคโนโลยีของ JAVA (JSP) การทำ User Interface ในลักษณะของ Web Base ก็เพื่อให้ผู้ใช้ใช้งานได้สะดวกยิ่งขึ้น สามารถเรียกใช้งานได้จากทุกที่ทุกเวลา โดยผ่าน Web Browser ซึ่งในการทำงานของส่วนนี้ เริ่มจากการรับการสั่งงานจากผู้ใช้ในรูปแบบของแบบ form ของ HTML ส่งการร้องขอไป Web Server ในที่นี้คือ Tomcat ทำหน้าที่เป็น Web Server เนื่องจาก Tomcat เป็น JSP & Servlets Container ตัวหนึ่ง ซึ่งจะ

กล่าวในรายละเอียดของ JSP & Servlets Container ต่อไป จากนั้นJSP จะทำการติดต่อกับฐานข้อมูลเพื่อจะทำการ query ข้อมูลที่จะต้องใช้ในการประมวลผลคำสั่งนั้นๆ ผ่าน JDBC การประมวลผลจะกระทำของฝั่งของ Web Server จากนั้นจะส่งต่อไปยังผู้ใช้ในรูปแบบHTML

4.3 รูปแบบที่ใช้ในฐานข้อมูล

ในบทที่ 3 ได้กล่าวเกี่ยวกับการออกแบบฐานข้อมูลไว้แล้ว ในหัวข้อนี้จะพูดถึงรายละเอียดของรูปแบบในการจัดเก็บฐานข้อมูล อย่างที่กล่าวไว้แล้ว ฐานข้อมูลของระบบจะประกอบด้วยกัน 4 ตาราง คือ

1. Object ซึ่งจะประกอบด้วยกัน 2 field ได้แก่ Object ID (Obj_ID) และ URL (กำหนด ID ให้กับ Object)
2. Server(N)_ActObj โดยตารางนี้จะประกอบด้วยกัน 3 field ได้แก่ Obj_ID, Sizeและ Swapout_time
3. Server(N)_HisObj จะประกอบด้วยกัน 6 field ได้แก่ Rec_ID, Obj_ID, Size, Swapout_time, Keep_timeและ Release_code
4. Servername ประกอบด้วยกัน 2 field ได้แก่ ServerID กับ ServerIP

สามตารางแรกได้กล่าวไว้แล้วในหัวข้อ 3.4 รูปที่ 3.7 ส่วนตารางที่ 4 ServerName ออกแบบมาเพื่อที่จะกำหนดชื่อให้แก่ Proxy Server แทนที่จะเป็น IP ของ Server นั้นๆ ตารางต่างๆที่ได้กล่าวมาจะมีชื่อของข้อมูล รูปแบบการเก็บข้อมูล และความหมาย ดังที่ได้แสดงในตารางที่ 4-1, 4-2,4-3 และ 4-4

ตารางที่ 4-1 รูปแบบการเก็บข้อมูลของตาราง OBJECT

OBJECT	รูปแบบของข้อมูล	คุณสมบัติพิเศษ	ความหมาย
Obj_ID	mediumint(9) UNSIGNED	auto_increment	เก็บค่า ID ของ Object
URL	tinytext	-	เก็บค่าที่มาจริงของ Object (URL) เป็น Key Field ใน Store.log

ตารางที่ 4-2 รูปแบบการเก็บข้อมูลของตาราง SERVER(N)_ActObj

SERVER(N)_ActObj	รูปแบบของข้อมูล	คุณสมบัติพิเศษ	ความหมาย
Obj_ID	mediumint(9) UNSIGNED	-	เก็บค่า ID ของ Object
Size	midiumint(9)	-	เก็บค่าขนาดของObject (Byte)
Swapout_time	decimal(12,3)	-	เก็บค่าเวลาที่บันทึก Object นั้นลง Cache ในส่วนของดิสก์เป็น Unix time ที่มีความละเอียดถึง millisecond (Second)

ตารางที่ 4-3 รูปแบบการเก็บข้อมูลของตาราง SERVER(N)_HstObj

SERVER(N)_HstObj	รูปแบบของข้อมูล	คุณสมบัติพิเศษ	ความหมาย
Rec_ID	mediumint(9) UNSIGNED	auto_increment	เก็บค่า ID ของ RECORD นั้นๆ
Obj_ID	midiumint(9) UNSIGNED	-	เก็บค่า ID ของ Object
Size	mediumint(9)	-	เก็บค่าขนาดของObject (Byte)
Swapout_time	decimal(12,3)	-	เก็บค่าเวลาที่บันทึก Object นั้นลง Cache ในส่วนของดิสก์เป็น Unix time ที่มีความละเอียดถึง millisecond (Second)
Keep_time	decimal(12,3)	-	เก็บเวลาที่ objectนั้นอยู่ใน Cache ซึ่งจะเท่ากับ Release time stamp – Swapout time
Release_code	Tinyint	-	เก็บสถานะของการ release 1 Releaseเมื่อดิสก์เต็ม 2. Release เมื่อ Object ล้าสมัย

ตารางที่ 4-4 รูปแบบการเก็บข้อมูลของตาราง SERVERNAME

SERVERNAME	รูปแบบของข้อมูล	คุณสมบัติพิเศษ	ความหมาย
ServerID	smallint(5) UNSIGNED	auto_increment	กำหนดค่า ID ของให้กับ Server
ServrIP	varchar(25)	-	เก็บ IP ของ Proxy Server

ประเภทของข้อมูลที่ใช้ในโครงงานนี้มีความหมายดังนี้

- MEDIUMINT [UNSIGNED]

  เป็นชนิดตัวเลขแบบคิดเครื่องหมายจะได้ค่าตั้งแต่ -8388608 ถึง 8388607

  แต่ถ้าไม่คิดเครื่องหมายจะได้ค่าตั้งแต่ 0 ถึง 16777215

- TINYINT [UNSIGNED]

  เป็นชนิดตัวเลขแบบคิดเครื่องหมายจะได้ค่าตั้งแต่ -127 ถึง 127

  แต่ถ้าไม่คิดเครื่องหมายจะได้ค่าตั้งแต่ 0 ถึง 255

- DECIMAL(12,3)

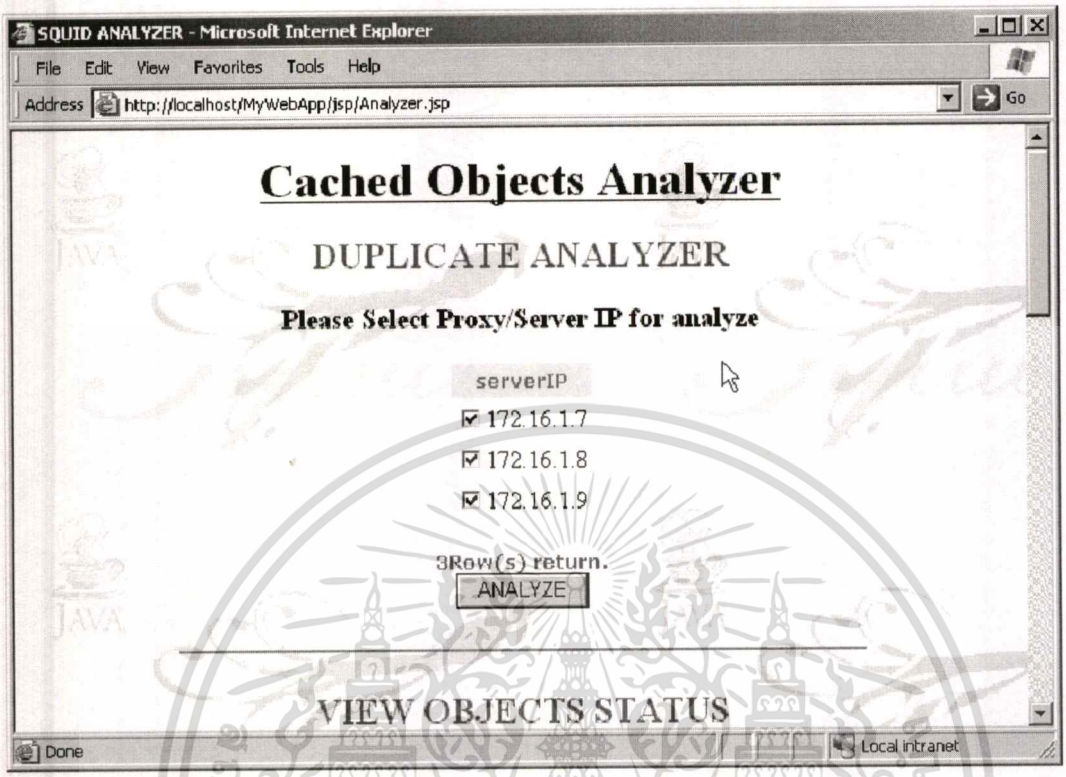
  เป็นชนิดตัวเลขสามารถเก็บข้อมูลได้ 12 หลัก เป็นทศนิยม 3 หลัก

- TINYTEXT

  ข้อมูลประเภทนี้สามารถใช้ความกว้างข้อมูลได้สูงสุด 255 ตัวอักษร และใช้เนื้อที่เก็บข้อมูลเท่ากับจำนวนข้อมูลจริง + 1 ไบต์

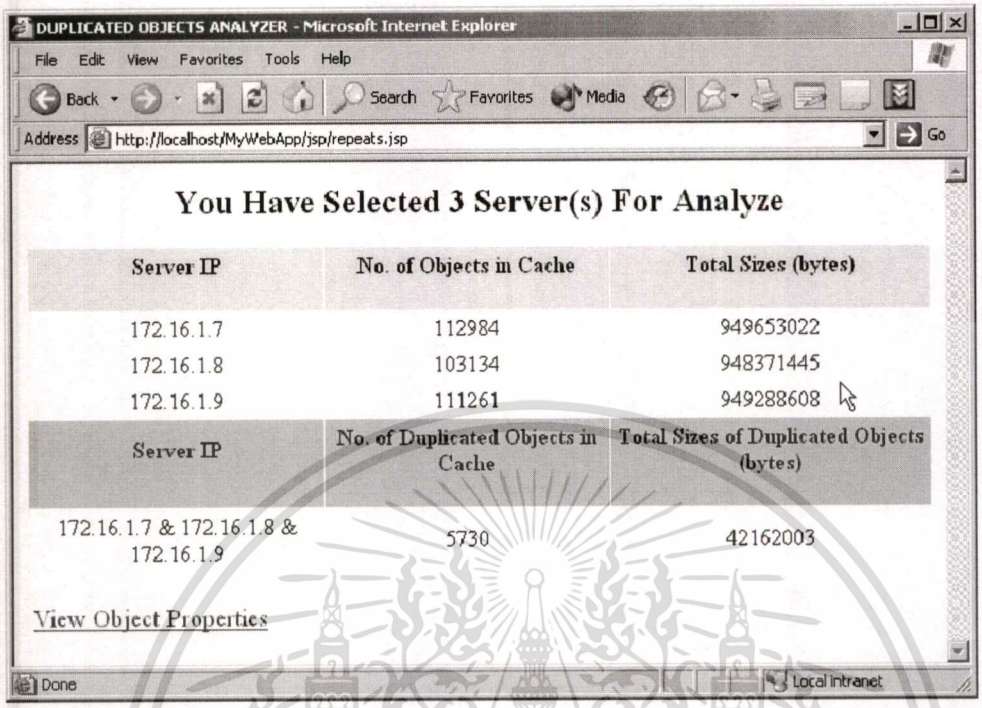
4.4 ระบบงานและ การใช้งาน

ในส่วนของ User Interface นั้นจากทำเตรียมคำสั่งต่างๆที่ใช้สำหรับการวิเคราะห์ ในรูปแบบของ form ใน HTML ให้ง่ายต่อการใช้งาน ที่พัฒนาด้วย Java Server Page (JSP) การทำงานคือ ผู้ใช้จะทำการสั่งคำสั่งผ่าน form ที่จัดเตรียมไว้ให้ จากนั้นทางด้าน JSP จะทำการดึงข้อมูลที่ต้องการจากฐานข้อมูลผ่านทาง JDBC (Jva Database Connectivity) เพื่อใช้ในการประมวลผลคำสั่งนั้นๆ และส่งผลกับให้กับผู้ใช้ในลักษณะของตารางใน HTML ดังรูปที่ 4.2 ซึ่งเป็นคำสั่งที่ใช้วิเคราะห์ความซ้ำซ้อนระหว่างProxy Server เป็นคำสั่งที่ใช้วิเคราะห์ว่า ณ ปัจจุบันระบบเชื่อมต่อกับ Proxy อะไรบ้าง ผู้ใช้สามารถเลือก Proxy ที่ต้องการจะวิเคราะห์โดยให้เลือก IP ของ Proxy ที่ต้องการจากนั้นให้คลิกที่ปุ่ม ANALYZE เพื่อทำการวิเคราะห์

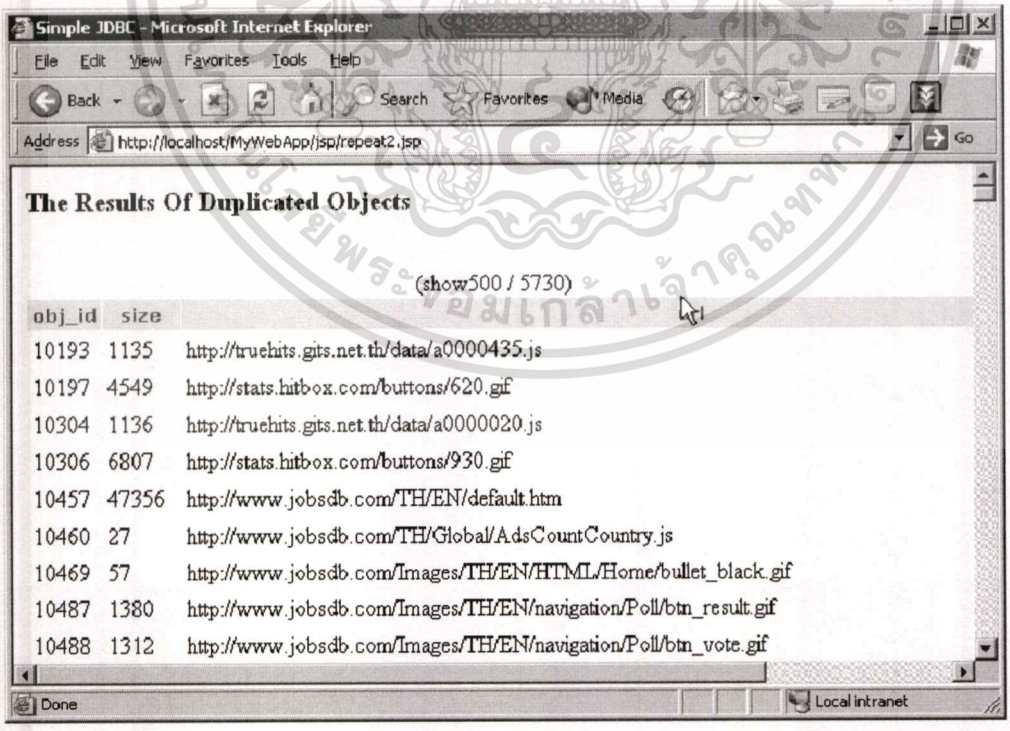


รูปที่ 4.2 หน้าจอส่วนของ User Interface ที่ใช้สำหรับวิเคราะห์

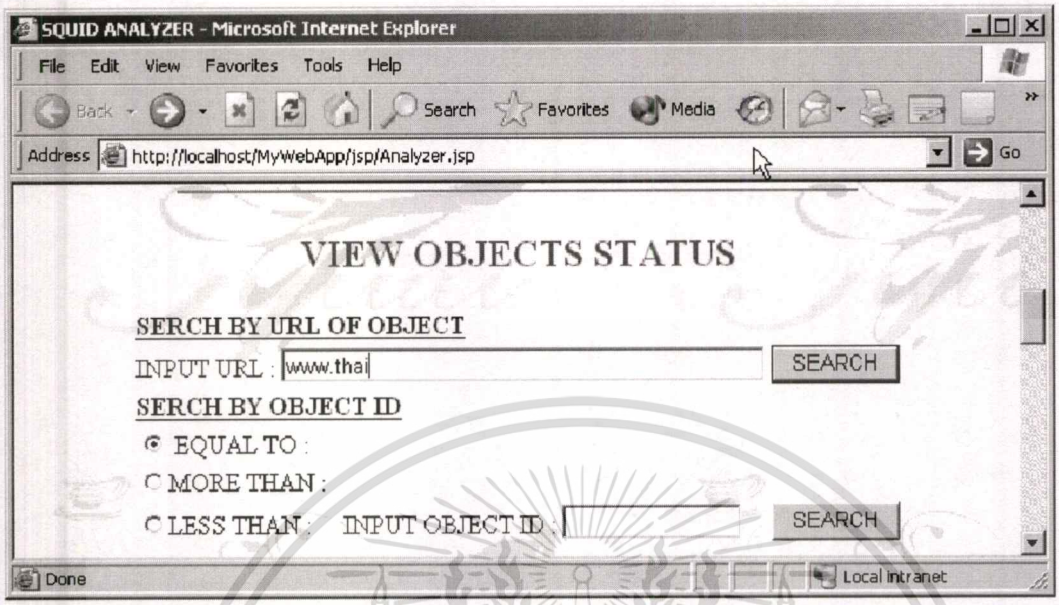
ผลที่ได้ของคำสั่งนี้จะอยู่ในรูปของตาราง ในแบบ HTML ดังเช่นในรูป 4.3 โดยในรูปที่ 4.3 นั้น จะประกอบด้วยข้อมูลสองส่วน ส่วนแรกจะเป็นข้อมูลของ Object ที่อยู่ใน Proxy ที่ได้เลือกไว้ตอนต้น ประกอบด้วยจำนวน Objects ในปัจจุบันที่อยู่ใน Proxy นั้นๆ และผลรวมของขนาดของ Objects ทั้งหมดที่อยู่ใน Proxy ขณะนั้น และส่วนที่สองคือส่วนที่เกิดความซ้ำซ้อนของ Objects ระหว่าง Proxy Server ที่ได้ทำการเลือกตอนต้น ซึ่งจะประกอบด้วย จำนวน Objects ที่ซ้ำซ้อน และผลรวมของขนาดของ Objects ทั้งหมดที่เกิดการซ้ำซ้อนระหว่าง Proxy ณ.ขณะนั้น และถ้าผู้ที่ทำการวิเคราะห์ต้องการดูข้อมูลของ Objects ที่ซ้ำซ้อนก็สามารถทำได้โดยกดที่ View Objects Properties ดังที่แสดงในรูปที่ 4.4 ผลการได้รับก็จะเป็นตารางข้อมูลของ Objects ดังกล่าวดังที่แสดงในรูปที่4.5



รูปที่ 4.3 การแสดงผลของการวิเคราะห์ความซ้ำซ้อนของ Objects

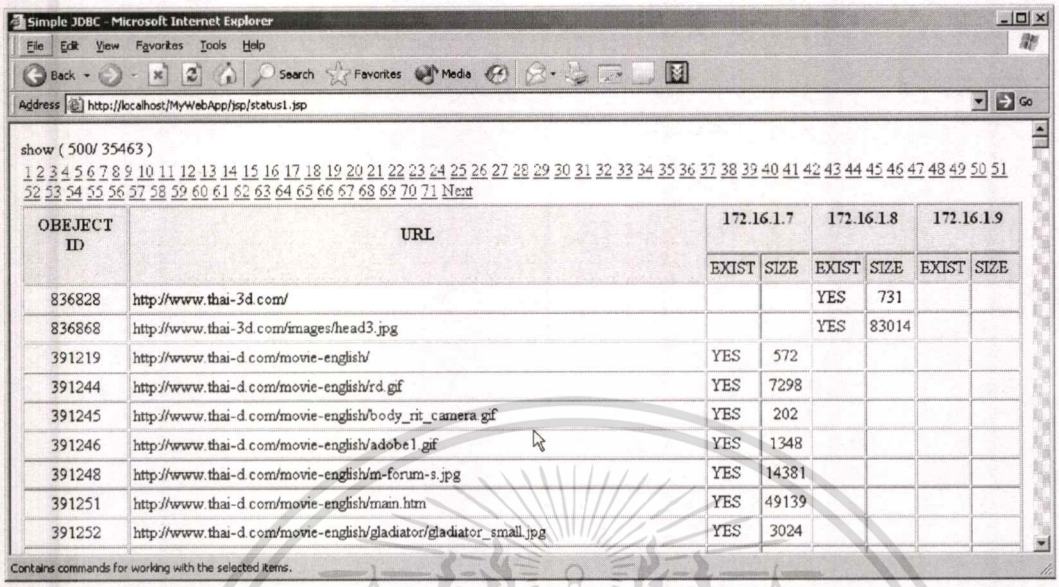


รูปที่ 4.4 ข้อมูลของ Objects ที่มีการซ้ำซ้อน

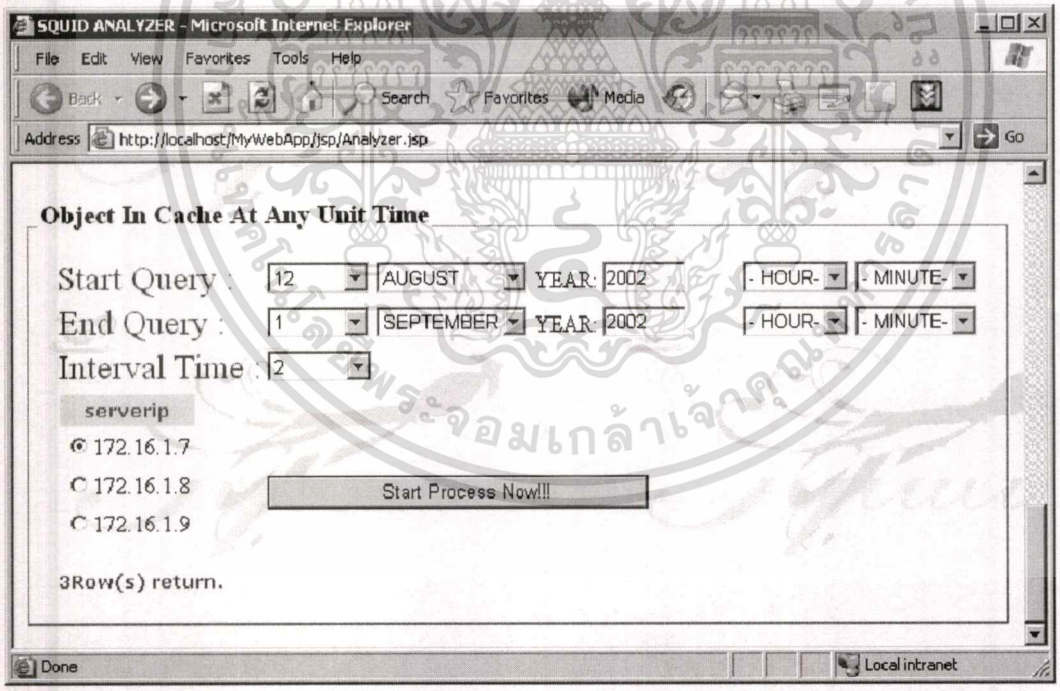


รูปที่ 4.5 หน้าจอส่วนของ User Interface ที่ใช้สำหรับดูสถานะของ Object

รูปที่ 4.5 เป็นส่วนของการที่จะดูสถานะของ Object ว่า ณ ปัจจุบัน Object นั้นๆอยู่ใน Proxy อะไรบ้าง ผู้ใช้สามารถเลือกว่าจะใส่เป็น URLของObject หรือจะใส่เป็นObject ID ก็ได้ ในการเลือกใส่ URLของObjectผู้ใช้สามารถใส่เพียงส่วนหน้าของ URL ก็ได้ อย่างเช่นในรูปที่ 4.5 ในกรณีที่ผู้ใช้ใส่เป็น Object ID ก็จะมีสาม functions ให้เลือกใช้คือ 1. เลือก Object ที่มี Object Id เท่าค่าที่ใส่ไป 2. เลือก Object ทั้งหมดที่มีค่ามากกว่าค่าที่ใส่ไป 3. เลือก Object ทั้งหมดที่มีค่าน้อยกว่าค่าที่ใส่ไป ผลที่ได้ของคำสั่งนี้จะอยู่ในรูปของตาราง ในแบบ HTML ดังเช่นในรูป 4.6 ซึ่งจะประกอบด้วยข้อมูลต่างๆ ของ Object ดังนี้ Object ID, URL และสถานะว่าขณะนั้น Object ได้อยู่ใน Proxy Server ตัวใด ในกรณีที่ Object นั้นอยู่ใน Proxy Server ตัวใดตัวหนึ่งก็จะแสดงขนาดของ Object ด้วย

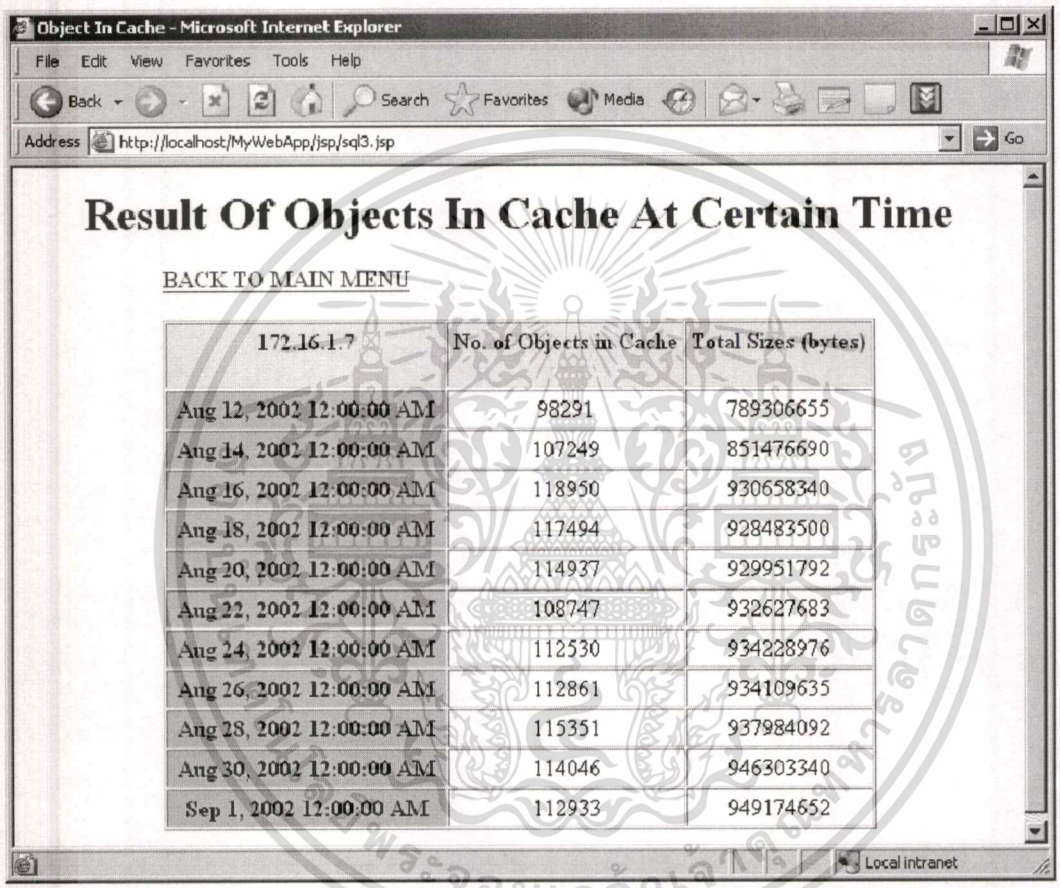


รูปที่ 4.6 แสดงข้อมูลสถานะของ Objects



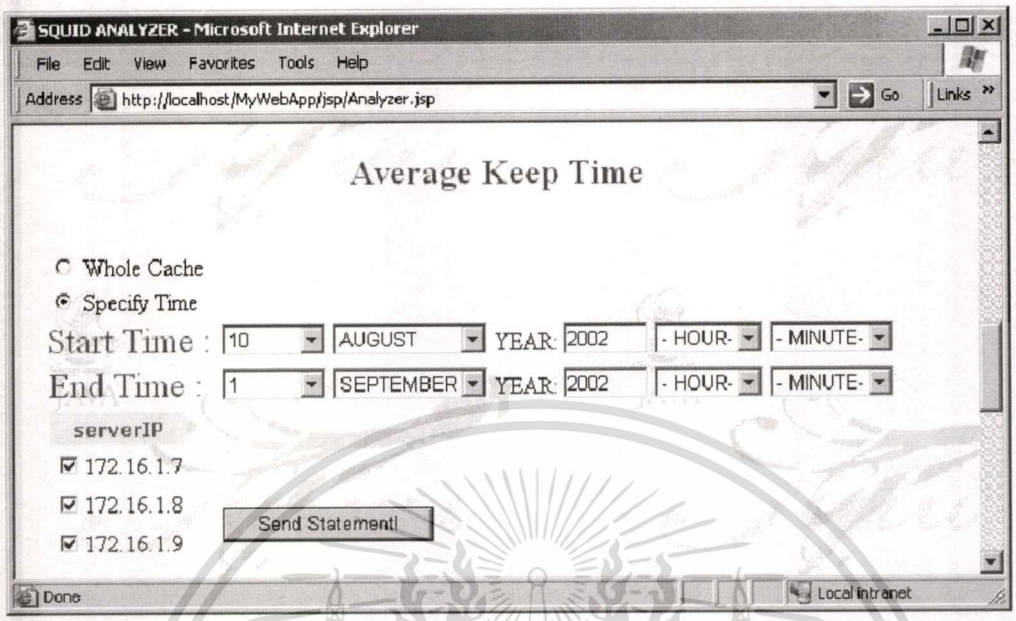
รูปที่ 4.7 หน้าจอส่วนของ User Interface หาจำนวน Object ณ เวลาต่างๆ

รูปที่ 4.7 เป็นส่วนของการที่จะหาจำนวนของ Object ในขณะเวลาหนึ่งๆ ผู้ใช้สามารถเลือกได้ว่าจะทำการวิเคราะห์ Proxy Server ตัวไหน โดยผู้ใช้สามารถเลือกได้ว่าให้เริ่มวิเคราะห์เวลาใด สิ้นสุดเวลาใด และระยะเวลาระหว่างการวิเคราะห์เป็นเวลาเท่าใด รูปที่ 4.8 แสดงผลของการวิเคราะห์ โดยผลจะประกอบด้วย จำนวน Object และขนาดทั้งหมดของ Object ใน Proxy Server ณ เวลาใดๆ



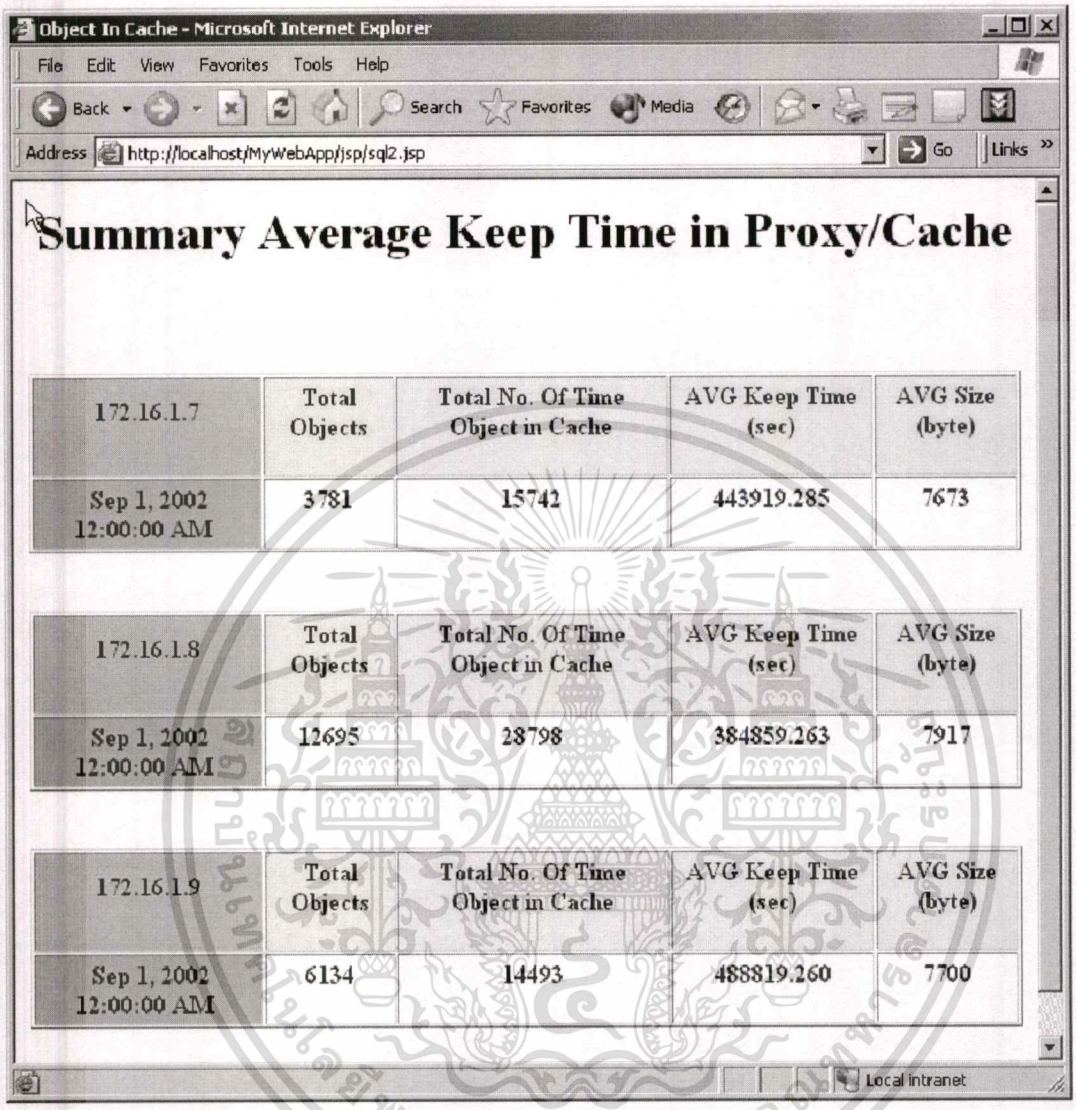
172.16.1.7	No. of Objects in Cache	Total Sizes (bytes)
Aug 12, 2002 12:00:00 AM	98291	789306655
Aug 14, 2002 12:00:00 AM	107249	851476690
Aug 16, 2002 12:00:00 AM	118950	930658340
Aug 18, 2002 12:00:00 AM	117494	928483500
Aug 20, 2002 12:00:00 AM	114937	929951792
Aug 22, 2002 12:00:00 AM	108747	932627683
Aug 24, 2002 12:00:00 AM	112530	934228976
Aug 26, 2002 12:00:00 AM	112861	934109635
Aug 28, 2002 12:00:00 AM	115351	937984092
Aug 30, 2002 12:00:00 AM	114046	946303340
Sep 1, 2002 12:00:00 AM	112933	949174652

รูปที่ 4.8ผลของการหาจำนวนของ Objects ณ เวลาหนึ่งๆ



รูปที่ 4.9 หน้าจอส่วนของการหาค่าเฉลี่ยของ Keep Time ในแต่ละ Server

รูปที่ 4.9 เป็นส่วนของการหาค่าเฉลี่ยของ Keep Time ของแต่ละ Serverในขณะเวลาหนึ่งๆ ผู้ใช้สามารถเลือกได้ว่าจะทำการวิเคราะห์ Proxy Server ตัวไหน โดยผู้ใช้สามารถเลือกได้ว่าให้เริ่มวิเคราะห์เวลาใด สิ้นสุดเวลาใด รูปที่ 4.10 แสดงผลของการวิเคราะห์ โดยผลจะประกอบด้วย 1. จำนวน Object ที่ใช้วิเคราะห์ 2. จำนวนครั้งของ Objects ทั้งหมดเข้าออก Cache 3. ค่าเฉลี่ยของ Keep Time และ 4.ขนาดเฉลี่ยของ Objects



รูปที่ 4.10 ผลของการหาค่าเฉลี่ยของ Keep Time ในแต่ละ Server

4.5 การทดสอบการใช้งานกับระบบจริง

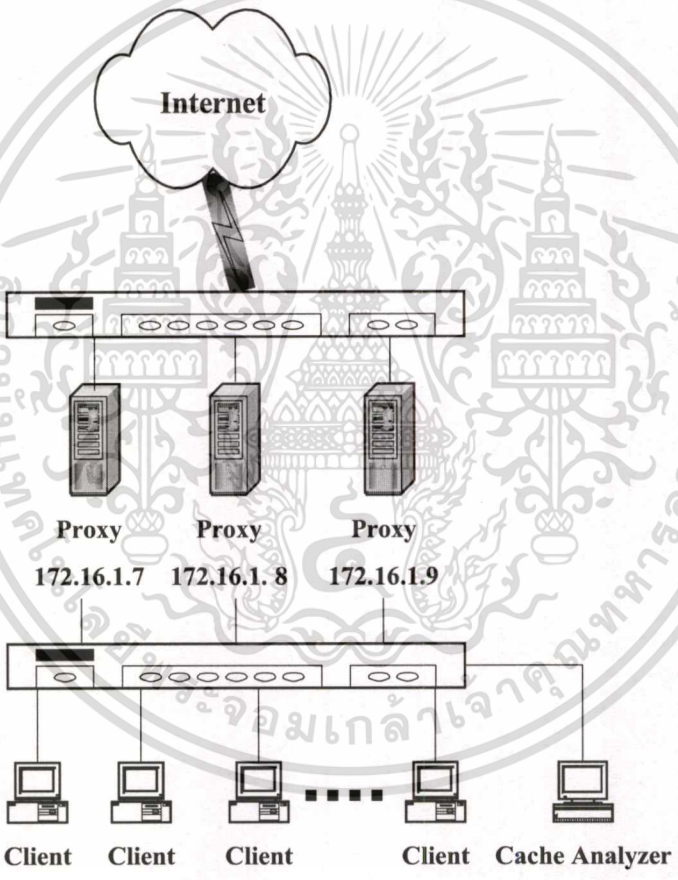
ได้ทำการทดสอบการทำงานกับระบบ Proxy/Cache ของกรมประมง ซึ่งใช้ Squid Proxy/Cache จำนวน 3 เครื่องทำงานร่วมกัน ดังรูปที่ 4.9 ในการทดสอบได้ทำการทดสอบแบบ Batch เนื่องจากต้องใช้เวลานานในการประมวลผล store.log ใน Proxy Server ทั้ง3เครื่อง เพราะ store.log มีขนาดค่อนข้างใหญ่ คือประมาณ 450Mbyte ต่อ Proxy Server หนึ่งเครื่อง การทดสอบครั้งนี้ใช้เวลาประมวลผลทั้งหมดในการจัดเก็บลงฐานข้อมูลประมวล 16ชม โดยเครื่องที่ใช้ในการประมวลผลมีคุณสมบัติดังนี้

คุณสมบัติด้าน Hardware

- Notebook CPU : Pentium 4 , 1.6GHz
- RAM : 256 MB
- Hard disk : 20 GB

ระบบปฏิบัติการเป็น Windows XP Professional

ผลที่ได้คือข้อมูลการทำงานของ Proxy Server ทั้งสามเครื่องตั้งแต่วันที่ 11/08/2545 ถึง 1/09/2545 และสามารถทราบถึงความซ้ำซ้อนของ Proxy Server แต่ละตัวได้



รูปที่ 4.11 ระบบ Proxy/Cache Server ของกรมประมง

4.6 ทดสอบความถูกต้องของโปรแกรม

การพัฒนาระบบครั้งนี้ ได้ทำการทดสอบความถูกต้องด้วยใช้ SNMPWALK ของตัว Squid เอง การ Config สำหรับการทดสอบจะอยู่ในภาคผนวก ค. ในการทดสอบนี้กระทำโดย Set เครื่องหนึ่งเครื่องเป็น Proxy Server แล้วทดลองใช้โปรแกรมที่พัฒนาขึ้นทำการวิเคราะห์ log file และทำการเก็บลงฐานข้อมูล จากนั้นก็นำผลลัพธ์ที่ได้เปรียบเทียบกับผลลัพธ์ที่ได้จาก SNMPWALK ซึ่งได้ผลดังต่อไปนี้

- ผลจาก SNMPWALK
  - Total Number Of Objects = 736
  - Total Cache Size = 3985 Kb
- ค่าที่ได้จากโปรแกรม
  - Total Number Of Objects = 736
  - Total Cache Size = 3720 Kb

จากผลที่ได้จะเห็นว่าจำนวน objects ที่ได้จากโปรแกรมและ SNMPWALK ได้ค่าที่เท่ากัน แต่ขนาดรวมของ Objects ที่ได้จาก SNMPWALK มีค่ามากกว่าที่ได้จากโปรแกรมก็เนื่องจากโปรแกรมจะเก็บค่าขนาดของ Object จริง (field Size ใน store.log) แต่ส่วน SNMPWALK จะเป็นขนาดของ file ของ Object ที่เก็บจริงใน Cache ซึ่งจะประกอบด้วยสองส่วนคือ ทั้ง Object จริง และ Header จึงทำให้ผลที่ได้จาก SNMPWALK มีค่ามากกว่าผลที่ได้จากโปรแกรม

บทที่ 5
สรุปผล

5.1 ผลจากการพัฒนาระบบงาน

จากการพัฒนาระบบงานครั้งนี้ สามารถสรุปผลที่ได้รับดังนี้

5.1.1 การศึกษาการทำงานของ Proxy/Cache โปรแกรม Squid

ผลการศึกษาพบว่า การใช้ Proxy/Cache มีประโยชน์อย่างมาก สามารถช่วยลดปริมาณการติดต่อกับเครือข่ายภายนอกที่มีความเร็วที่ต่ำกว่ามาก โดยอาจมีการใช้ Web Proxy/Cache Server มากกว่าหนึ่งตัวทำงานร่วมกัน เพื่อแบ่งปันข้อมูลและทรัพยากรซึ่งกันและกัน ช่วยลดเวลาในการเข้าถึงข้อมูล สามารถลด Bandwidth จำนวนมากได้ และยังเป็นการเพิ่มประสิทธิภาพของระบบ

5.1.2 การวิเคราะห์และออกแบบ

การวิเคราะห์และออกแบบนั้น ได้ออกแบบระบบที่เป็นศูนย์กลางที่คอยรับ Store.log จาก Proxy / Cache Server ทุกๆตัวในเครือข่ายเพื่อนำ Store.log มาวิเคราะห์ ประมวลผล และจัดเก็บลงฐานข้อมูล การออกแบบระบบได้ออกแบบให้มีส่วนประกอบหลักๆอยู่สามส่วน คือส่วนที่ทำงานบนฝั่ง Squid Proxy Server ทำหน้าที่รอรับ Store.log ใหม่ที่เกิดขึ้น จัดการให้ Store.log อยู่ใน Format ที่สามารถนำไปประมวลผลต่อได้ และส่งไปให้เครื่องที่ใช้ทำการประมวลผล ส่วนที่สองคือส่วนที่ทำงานบนฝั่งเครื่องที่ใช้วิเคราะห์ทำหน้าที่รับ Store.log จาก Proxy / Cache Server ทุกๆตัวในเครือข่ายเพื่อนำ Store.log มาวิเคราะห์ ประมวลผล และจัดเก็บลงฐานข้อมูล และส่วนสุดท้ายเป็นส่วนของ User Interface ทำหน้ามี่จัดเตรียมคำสั่ง ทำการ query ไปยังฐานข้อมูล นำผลที่ได้จากการประมวลผล และส่งผลให้กับผู้ใช้ในลักษณะที่เข้าใจง่าย

5.1.3 การพัฒนาระบบ

ได้พัฒนาโปรแกรมทั้งในส่วนที่ทำงานบนฝั่ง Proxy Server และในส่วนของเครื่องศูนย์กลาง ที่สามารถนำ Store. log มาวิเคราะห์และจัดเก็บลงฐานข้อมูล ที่ออกแบบมาเพื่อให้สามารถวิเคราะห์ความซ้ำซ้อนของ Object ระหว่าง Proxy Server ที่ทำงานร่วนกันได้ และได้พัฒนาส่วนของ User Interface ในลักษณะของ Web base ที่สามารถประมวลผลและทำการวิเคราะห์ข้อมูลจากฐานข้อมูลได้ และได้ทำการทดสอบความถูกต้องของโปรแกรมด้วย SNMPWALK ของ Squid เอง โดยผลที่ได้รับพบว่าโปรแกรมนี้สามารถทำงานได้ถูกต้อง

5.2 ประโยชน์ที่ได้รับ

ได้โปรแกรมที่สามารถใช้วิเคราะห์ความซ้ำซ้อนของ Object ระหว่าง Proxy/Cache Server ได้ ทำให้สามารถทำการวิเคราะห์ได้ง่ายและรวดเร็ว และผู้ใช้วิเคราะห์สามารถใช้งานได้ในทุกๆที่ เนื่องจากระบบส่วนของ User Interface เป็นแบบ Web base และนำผลที่ได้มาปรับปรุงประสิทธิภาพการทำงานของ Proxy/Cache ในหน่วยงานได้ โปรแกรมนี้จะเป็นประโยชน์มากกับหน่วยงานที่ใช้ Proxy/Cache หลายเครื่องทำงานร่วมกัน

5.3 อุปสรรคในการพัฒนาระบบ

5.3.1 การพัฒนาระบบนี้จะต้องใช้ store.log จาก Proxy/Cache Server มากกว่าหนึ่งตัว ทำให้ไม่สามารถทดสอบโปรแกรมเองได้ เพราะการทดสอบจริงๆนั้นจะต้องมีเครื่องที่ทำหน้าที่เป็น Proxy Server อย่างน้อยสองเครื่อง และเครื่องที่ใช้ในการวิเคราะห์อีกหนึ่งเครื่อง

5.3.2 การทดสอบกับระบบงานจริงจะต้องติดตั้งโปรแกรมส่วนหนึ่งใน Proxy Server ของหน่วยงานที่จะทำการทดสอบ ซึ่งทางหน่วยงานนั้นๆอาจไม่ให้สิทธิได้

5.4 ข้อเสนอแนะ

5.4.1 สำหรับหน่วยงานที่ต้องให้บริการผู้ใช้จำนวนมาก จะทำให้ store.log มีขนาดที่ค่อนข้างใหญ่ในเวลาไม่นาน จึงควร connect เครื่องที่ทำหน้าที่ประมวลผลเข้ากับ Proxy Server บ่อยครั้งที่สุดเท่าที่จะทำได้ เพราะจะทำให้การประมวลผลไม่นานจนเกินไป

5.4.2 การทำงานของระบบนี้จะใช้ Timestamp ที่บันทึกโดย Squid ในการประมวลผล ซึ่งจะเป็นเวลาในเครื่อง server นั้น จึงควรทำการให้เวลาแต่ละเครื่องตรงกัน เพื่อให้ผลการวิเคราะห์นั้นถูกต้อง

บรรณานุกรม

สงกรานต์ ทองสว่าง. 2544. **MySQL ระบบฐานข้อมูลสำหรับอินเทอร์เน็ต. พิมพ์ครั้งที่ 1.** กรุงเทพฯ: เอช.เอ็น. กรุ๊ป.

Duane Wessels.2000. **SQUID Frequently Asked Question**. [Online].Available : http://www.squid-cache.org/Doc/FAQ/FAQ.html

H.M.Deitel, P.J.Deitel. 2001. **Java How To Program 4th Edition**, Prentice Hall

Hughes et al.1999. **Java Network Programming 2nd Edition**, Greenwich : Manning

Jason Hunter with William Crawford.1998. **Java Servlet Programming**, Sebastopal: O'Reilly & Associates.

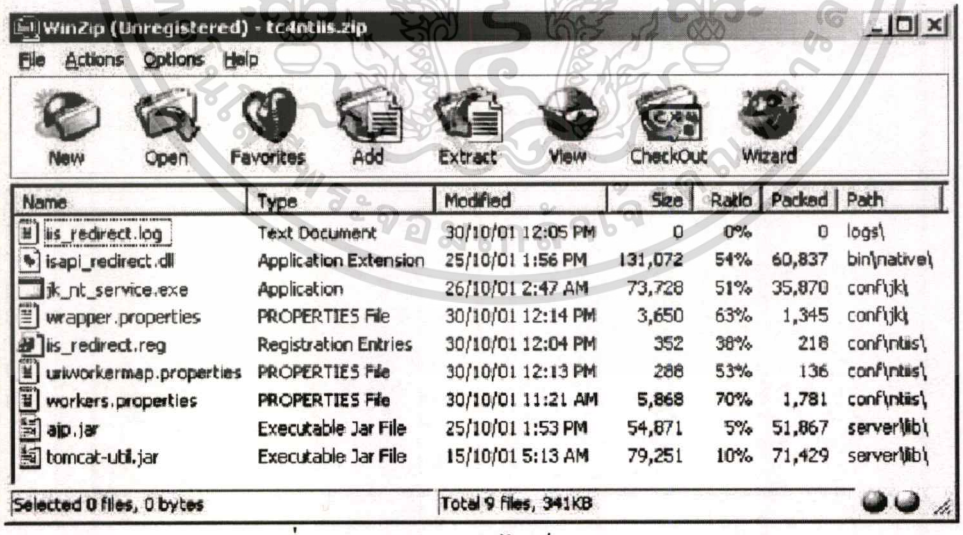
RFC2186 - Internet Cache Protocol (ICP), version 2 http://www.ietf.org/rfc/rfc2186.txt

ภาคผนวก

ภาคผนวก ก.
ตัวอย่างการติดตั้งเครื่องมือที่ใช้พัฒนาระบบ

1. การติดตั้ง Tomcat ทำงานร่วมกับ IIS

- ทำการ Download Tomcat binary ได้ที่ http://jakarta.apache.org และทำการติดตั้ง
  - ทำการ execute ไฟล์ โดยติดตั้ง ที่ C:\Tomcat4
  - กำหนดตัวแปลสิ่งแวดล้อม (Environment Variable) CATALINA_HOME ชี้ไปที่ PATH ที่ได้ทำการติดตั้ง Tomcat มีขั้นตอนดังนี้
    - ไปที่ Control Panel -> System -> Advanced Tab -> Environment Variables
    - สร้าง System Variable ใหม่ชื่อ CATALINA_HOME โดยมีค่า C:\Tomcat4
- ทำการDownload NT service และ ส่วนเพิ่มเติมของIIS เพื่อให้สามารถเปิด(Start) Tomcat ได้อัตโนมัติ และสามารถทำงานร่วมกับ IIS ได้ โดยทำการ Download ไฟล์ tc4ntiis.zip ได้ที่ http://members.ozemail.com.au/~lampante/howto/tomcat/iisnt/tc4ntiis.zip แต่ในที่นี้จะกล่าวถึงวิธีใช้งาน Tomcat แบบ stand alone ไม่ได้ทำงานร่วมกับ IIS เมื่อทำการDownload เรียบร้อยแล้วก็ทำการแตกไฟล์ เข้าไปใน %CATALINA_HOME% เช่น C:\Tomcat4\ ดังรูปที่ ก-1



รูปที่ ก-1 แสดงการแตกไฟล์ tc4ntiis.zip

- ทำ Tomcat ให้เป็น Service ใน NT

มี 2 ไฟล์ที่ต้องใช้ซึ่งได้ทำการ Download มาแล้วจากขั้นตอนที่แล้ว ได้แก่

1. %CATALINA_HOME%\conf\jk\jk_nt_service.exe
2. %CATALINA_HOME%\conf\jk\wrapper.properties

- ทำการแก้ไขไฟล์ wrapper.properties เพื่อให้เหมาะสมกับสภาพแวดล้อมของเครื่อง

wrapper.tomcat_home=c:\tomcat4

wrapper.java_home=c:\jdk1.3.1_01

โดยที่ tomcat_home คือdirectory ที่ติดตั้ง Tomcat และ java_home สำหรับ JDK

- เปิด command prompt แล้วพิมพ์คำสั่งดังนี้

C:\>cd %CATALINA_HOME%

C:\tomcat4>cd conf\jk

C:\tomcat4\conf\jk>jk_nt_service -i tomcat -a wrapper.properties

Asked (and given) winsock 1.1

The service named tomcat was created. Now adding registry entries

Registry values were added

If you have already updated wrapper.properties you may start the tomcat service

by executing "jk_nt_service -s tomcat" from the command prompt

- ตอนนี้ก็สามารถใช้งาน Tomcat โดยใช้ NT Service ด้วยการใช้คำสั่งดังนี้ใน command prompt แต่ต้องแน่ใจว่า Tomcat ไม่ได้ทำการอยู่

C:\tomcat4\conf\jk>jk_nt_service -s tomcat

Asked (and given) winsock 1.1

Starting tomcat.

tomcat started.

ตอนนี้ก็สามารถใช้งาน Tomcat ได้โดยเรียกผ่านพอร์ต 8080

2. การติดตั้ง MySQL API

การติดตั้ง MySQL สำหรับ Windows Platfrom มีขั้นตอนดังนี้

- ทำการดาวน์โหลด Binary Installer จาก www.mysql.com
- Unpack ไฟล์มาไว้ในไดเรกทอรีใดๆ (ปกติจะเป็น Temporary directory เช่น c:\temp)
- เข้าไปที่ไดเรกทอรีนั้น แล้วรัน Setup.exe
- เมื่อให้ระบุไดเรกทอรีที่ต้องการติดตั้งก็ให้ใช้ค่าดีฟอลต์คือ C:\mysql
- Copy ไฟล์ mysql\my-xxxxx.conf ไปไว้ที่ c:\ และทำการเปลี่ยนชื่อเป็น my.cnf
- ปรับแต่งค่าต่างๆในในไฟล์ my.cnf ให้เหมาะสม

2.1 การ Start/Stop MySQL Server ใน Win9X

ที่ MS-DOS Prompt

Start: c:\mysql\bin\mysqld

Stop: c:\mysql\bin\mysqladmin –u root shutdown

(หรือใช้ User อื่นก็ได้ ในตัวอย่างเป็นการใช้ User root ในการทำงาน)

2.2 การ Start/Stop MySQL Server ใน Windows NT, Windows 2000

- ติดตั้งเพื่อให้ MySQL Server ทำงานเป็นลักษณะ Service

  c:\myqsl\binmysqld-nt –install

- Start/Stop MySQL Server Service

  NET START mysql

  NET STOP mysql

หรือใช้ mysqladmin ในการ Shutdown ก็ได้

ภาคผนวก ข.
การติดตั้งระบบ

1. การติดตั้งระบบงาน

การทำงานทั้งหมด 3 ส่วน คือ ฝั่ง Squid Proxy Server , ฝั่งเครื่องที่ใช้วิเคราะห์ และตัว User Interface จึงทำให้การติดตั้งระบบมีความยากอยู่บ้าง เนื่องจากระบบงานครั้งนี้ถูกพัฒนาด้วยภาษา Java เพราะฉะนั้น จึงต้องทำการติดตั้ง JDK ก่อนที่จะสั่งให้โปรแกรมทำงานได้ ซึ่งได้กล่าวไว้แล้วในหัวข้อ 4.4.1 ขั้นตอนในการติดตั้งระบบมีดังนี้

1.1 การติดตั้งโปรแกรมบนฝั่ง Squid Proxy Server

การทำงานบนฝั่ง Squid Proxy Server จะมีอยู่สองโปรแกรมย่อย คือ Client.java และ Server.java ขั้นตอนใช้งานโปรแกรมนั้นจะเป็นดังนี้

1. รันโปรแกรมตัวที่สองก่อน คือ Server.java ซึ่งจะต้อง compile ให้เป็นจุด class ก่อนด้วยคำสั่ง javac Server. java(จะใช้ครั้งแรกที่ทำการรันโปรแกรมเท่านั้น กล่าวคือมีแต่จุด java แต่ไม่มีจุด class การรันโปรแกรมในครั้งต่อๆไปให้ข้าม) จากนั้นให้ใช้คำสั่ง java Server & การใส่ & ไว้ตอนท้ายจะทำให้รันโปรแกรมเป็น Background
2. รันโปรแกรมย่อยตัวแรก คือ Client.java จากนั้นcompile ให้เป็นจุด class ด้วย javac Client.java การ compile (จะใช้ครั้งแรกที่ทำการรันโปรแกรมเท่านั้น) จากนั้นใช้คำสั่ง tail – f /[squid home directory]/Store. log | java Client (คำสั่ง tail –f เป็นการเรียกดูท้ายของไฟล์ และ –f จะติดตามการเปลี่ยนแปลงของไฟล์นั้น คือถ้ามีการเพิ่มขึ้นของ Store.log ก็จะทำการส่งบรรทัดที่เพิ่มเข้าไปออกมาแสดง) เพื่อทำการส่ง Store. log ในส่วนที่เพิ่มข้น ผ่าน Standard Input ให้กับโปรแกรม Client

1.2 การติดตั้งโปรแกรมบนฝั่งเครื่องที่ใช้วิเคราะห์

การทำงานบนฝั่งนี้สิ่งที่จะต้องติดตั้งก่อนที่รันโปรแกรมก็คือ ส่วนของ JDK และโปรแกรมฐานข้อมูล MySQL ซึ่งได้เคยกล่าวถึงไปแล้ว โดยจะกล่าวเพิ่มเติมในส่วนของการสร้างฐานข้อมูลใน MySQL การสร้างฐานข้อมูลเพื่อรองรับการทำงานของโปรแกรมนี้ จะต้องสร้างตารางเปล่าๆ โดยไม่ต้องมีข้อมูลใดๆของ SERVERNAME และ OBJECT ขึ้นมาก่อน ตาราง SERVERNAME

เป็นตารางที่ใช้เก็บ Server ID กับ Server IP เพื่อให้โปรแกรมสามารถเปลี่ยนจาก Server IP เป็นชื่อของ Server ได้ โดยตารางที่เหลือนั้นโปรแกรมจะสร้างขึ้นมาเองภายหลัง การสั่งงานโปรแกรมนี้ก่อนอื่นจะต้องเข้าไปแก้ไฟล์ Project.conf ว่าเครื่องนี้ได้เชื่อมต่อกับ Proxy Server ตัวไหนบ้างโดยให้ใส่ IP ของ Server ดังกล่าวลงไป ในกรณีที่เชื่อมกับ Server มากกว่าหนึ่งตัว ให้ใส่ให้ครบทุกตัวด้วยการเว้นที่ว่างระหว่าง Server IP แต่ละตัว เพื่อให้โปรแกรมสามารถรู้ได้ว่าจะต้องทำ Connection กับ Server ตัวไหนบ้าง จากนั้นให้รันโปรแกรม ด้วยคำสั่ง Java Client2 (ในการใช้งานครั้งแรกให้ Compile จุด java เป็นจุด class ก่อน โดยคำสั่ง javac Client2.java) ใน Command Prompt โดยโปรแกรมส่วนจะสร้าง Connection กับ Proxy Server แต่ละตัวเพื่อนำเอาจากเครื่อง Proxy Server ต่างๆ มาทำการประมวลผลและเก็บลงฐานข้อมูล

1.3 การติดตั้งและการทำงานส่วนของ User Interface

ในส่วนของ User Interface จะเป็นลักษณะ Web Base ที่พัฒนาด้วย JSP ดังนั้นก่อนอื่นจะต้องติดตั้ง โปรแกรม Web Server ก่อนในที่นี้คือ Tomcat วิธีติดตั้ง Tomcat นั้นได้กล่าวไว้ในหัวข้อ 4.4.4 เมื่อติดตั้ง Tomcat เรียบร้อยแล้ว ให้นำเอาไฟล์ .jsp และไฟล์ HTML ทั้งหมดไปไว้ที่ /[Tomcat Default Web Page]/JSP/ และ ส่วนที่พัฒนาเป็น Java Bean หรือ Java's Packet ให้นำไปไว้ที่ /[Tomcat Default Web Page]/WEB-INF/classes/ ดังรูป

รูปที่ ข-1 โครงสร้างของ Folder สำหรับการใช้งาน JSP ใน Tomcat

ภาคผนวก ค.

การ Config SNMPWALK

ขั้นตอนการติดตั้ง SNMPWALK

1. Enable SNMP protocal ที่เครื่อง Proxy Server
2. แก้ snmpd.conf สำหรับ access control

   2.1 map community name "public" เข้าไปใน "security name" โดยเพิ่ม 2 บรรทัดนี้

   ```
   com2sec     local     localhost       public
   com2sec     outside   default         public
   ```

   2.2 map security name เข้า group name โดยเพิ่ม 2 บรรทัดนี้

   ```
   group       MyRWGroup           v1      local
   group       MyROGroup           v1      outside
   ```

   2.3 สร้าง View

   ```
   view        all       included   system
   ```

   2.4 กำหนดสิทธิให้กับ group

   ```
   access MyROGroup "" any noauth exact all none       none
   access MyRWGroup ""  any noauth exact all none      none
   ```

3. แก้ squid.conf

   3.1 แก้ค่าต่างๆดังนี้สำหรับ Squid2.2

   ```
   acl snmppublic snmp_community public
   acl snmpjoebloggs snmp_community joebloggs
   snmp_access allow snmppublic localhost
   snmp_access deny all
   snmp_incoming_address 0.0.0.0
   snmp_outgoing_address 0.0.0.0
   ```

   3.2 แก้ค่าต่างๆดังนี้สำหรับ Squid2.1

   ```
   snmp_port 3401
   snmp_mib_path /local/squid/etc/mib.txt
   snmp_agent_conf view all .1.3.6 included
   snmp_agent_conf view squid .1.3.6 included
   snmp_agent_conf user squid - all all public
   snmp_agent_conf user all all all all  squid
   snmp_agent_conf community public squid squid
   snmp_agent_conf community readwrite all all
   ```

4.run คำสั่ง snmp –m /etc/squid/mib.txt –p3401 –IR localhost public squid

ภาคผนวก ง.

Source Code

โปรแกรมที่1 : Server.java ส่วนที่ทำงานบนฝั่ง Squid's Proxy Server

```
// Server.java
// Run in Squid Proxy Server Side
import java.io.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;

class Squid {
  public static String cutString(String s){
                    StringTokenizer st = new StringTokenizer(s);
                    int i = 0;
                    String a[] = new String [12];
            //      a[0] = "serverA";
                    while (st.hasMoreTokens()){
                          a[i] =    st.nextToken();
                          i++ ;
                    }
                    if   (a[1].equalsIgnoreCase("swapin")   ||   (a[1].equalsIgnoreCase("RELEASE")   &&
a[2].equalsIgnoreCase("FFFFFFFF"))){
                    s = "skip";}
                    else {s = a[0] + " " +a[1] + " " +a[2] + " " +a[8]+" "+a[10];}
  //                  else { s= "skip";}
  //                  System.out.println(s);
                    return(s);
  } //cutString
  } // End Class Squid
public class Server extends JFrame {
  static ServerSocket server;
  static Socket connection;
  private ObjectOutputStream output;
  private ObjectInputStream input;
  private String message="";
  private int counter =1 ;
```

```
static Vector vec = new Vector();

public Server(){
        super("Server");
}

public void runServer()
{
        try {
                server = new ServerSocket(3456,100);
                while(true){
                        String address = waitForConnection();
                        System.out.println(address);
                        if(address.equalsIgnoreCase("localhost")){
                        getStream();
                        processConnection();
                        closeConnection();
                        ++counter;
                        Client3 application2 ;
                        application2 = new Client3("192.168.0.1");
                        application2.connectToServer();

                 }
                        else{
                                getStream2();
                                processConnection2();
                                closeConnection();
                                ++counter;
                        }
                }
}
        catch (EOFException eofException) {
                        System.out.println("Client terminated connection");
        try {
                        closeConnection();
                        Server application = new Server();
                        application.runServer();
                }
        catch (Exception e) {}
        }  // End Catch
```

```
            catch (IOException ioException){
                        ioException.printStackTrace();
            }
    }

    private String waitForConnection() throws IOException{
            System.out.println("Waiting for connection\n");

            connection = server.accept();
            System.out.println("Connection " + counter + "recived from: " +
                        connection.getInetAddress().getHostName().toString());
            return connection.getInetAddress().getHostName().toString();
    }

    private void getStream() throws IOException {
            input = new ObjectInputStream(connection.getInputStream());
            System.out.println("\nGot I/O Streams\n");
    }
    private void processConnection() {
    try{
            while((message = (String)input.readObject()) != "null"){
                        message = Squid.cutString(message);
            if (!message.equalsIgnoreCase("skip")){
                        vec.addElement(new String(message));
            }
    }
                        for (int i= 0; i < vec.size() ; i++ )
                                    {
                                    message = (vec.elementAt(i).toString());
                                    System.out.println(message);
                                    }
    }
            catch (EOFException e){System.out.println("EFO");}
            catch (Exception ex){System.out.println(ex);}
}

    private void closeConnection() throws IOException {
            System.out.println("\nUserTerminatedConnection");
            input.close();
            connection.close();  }
    private void getStream2() throws IOException {
```

```
            output = new ObjectOutputStream(connection.getOutputStream());
            output.flush();
            input = new ObjectInputStream(connection.getInputStream());
            System.out.println("\nGot I/O Stream\n");
}

private void processConnection2(){

            try{

//          FileInputStream fin = new FileInputStream("d:/read/sam/tmp.txt");
            FileInputStream fin = new FileInputStream("/home/bagglo/tmp.txt");
            BufferedInputStream bin = new BufferedInputStream(fin);
            DataInputStream din = new DataInputStream(bin) ;
            while((message= din.readLine()) != null){
//          String message = din.readLine();
//          message = din.readLine();
//          if (!message.equalsIgnoreCase("skip")){
            output.writeObject(message);
            output.flush();
//          }
            System.out.println(message);
            }
            File f = new File("/home/baggio/tmp.txt");
            f.delete();
            }
            catch(Exception network) {}
}

public static void main (String args[])
{
            Server application = new Server();
            application.runServer();
}

}

/* ********************** Client3 Class ***************************/

class Client3 extends JFrame {//At Squid Proxy Side to Send Data
private ObjectOutputStream output;
```

```
        private Socket connection;
        private int counter =1 ;
        private ObjectInputStream input;
        private String message="SAM";
        private String chatServer;
        private Socket client3;

public Client3(String host){
        super("Client3");
        chatServer = host;

        }

public void runClient3()
        {
                try {
                        connectToServer();
                        getStreams();
                        processConnection();
                        closeConnection();

                        }

                catch (EOFException eofException) {
                        System.out.println("Server terminated connection");
                }

                catch (IOException ioException){
                        ioException.printStackTrace();
                        System.out.println("??????");
                }
        }

        public void getStreams() throws IOException {
                output = new ObjectOutputStream(client3.getOutputStream());
                System.out.println("\nGot I/O Stream\n");
                processConnection();
        }

        public void connectToServer()
        {
```

```
            try{
            System.out.println("Attempting connection\n");
            client3 = new Socket(InetAddress.getByName(chatServer),3456);
            System.out.println("Connected To : "+
                        client3.getInetAddress().getHostName());
                        getStreams();
            }
            catch (Exception e){
                        try {
                        RandomAccessFile raf = new RandomAccessFile("/home/baggio/tmp.txt","rw");
                        for(int i=0 ; i < Server.vec.size() ; i++){
                        raf.seek(raf.length());
                        raf.writeBytes(Server.vec.elementAt(i).toString()+'\n');
                                    }
                                    Server.vec.clear();
                        }
                                    catch(Exception IO){System.out.println(IO);}
}
}

public void processConnection(){
            try{
            System.out.println(Server.vec.size());
            for(int i=0 ; i < Server.vec.size() ; i++){
                        String message = Server.vec.elementAt(i).toString();
                        System.out.println(message);
                        output.writeObject(message);
                        output.flush();
            }
                        Server.vec.clear();
                        closeConnection();
            }
            catch (Exception ProCon){System.out.println(ProCon);}
}

public void closeConnection() throws IOException {
            System.out.println("\nUserTerminatedConnection");
            output.close();
            client3.close();
}
}
```

โปรแกรมที่2 : Client.java ส่วนที่ทำงานบนฝั่ง Squid's Proxy Server รับค่าจาก Store.log เพื่อส่งให้โปรแกรม Server.java

```
// Client.java
// Run in Squid Proxy Server Side

import java.io.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;

public class Client extends JFrame {//At Squid Proxy Side to Send Data
  private ObjectOutputStream output;
  private Socket connection;
  private int counter =1 ;
  private ObjectInputStream input;
  private String message="SAM";
  private String chatServer;
  private Socket client;
  static Vector v = new Vector();

  public Client(String host){
        super("Client");
        chatServer = host;
  }
  public void runClient()
  {
        try {
              connectToServer();
              getStreams();
              processConnection();
              closeConnection();
              }
              catch (EOFException eofException) {
                    System.out.println("Server terminated connection");
              }

              catch (IOException ioException){
                    ioException.printStackTrace();
              }
  }


  private void getStreams() throws IOException {
        output = new ObjectOutputStream(client.getOutputStream());
        System.out.println("\nGot I/O Stream\n");

//              input = new ObjectInputStream(client.getInputStream());
//              System.out.println("\nGot I/O Streams\n");
  }

  private void connectToServer() throws IOException
  {
        System.out.println("Attempting connection\n");

        client = new Socket(InetAddress.getByName(chatServer),3456);
        System.out.println("Connected To : "+
              client.getInetAddress().getHostName());
  }
```

```
private void processConnection()throws IOException {
        InputStreamReader rin = new InputStreamReader(System.in);
        BufferedReader bin = new BufferedReader(rin);
        String message;
                while (  (message = bin.readLine())!= null){
                        v.addElement(message);
                        }
                for (int i= 0; i < v.size() ; i++ ){
                System.out.println(v.elementAt(i).toString());
                output.writeObject(v.elementAt(i).toString());
                        output.flush();
                }
}

private void closeConnection() throws IOException {
        System.out.println("\nUserTerminatedConnection");
        output.close();
        client.close();
        System.exit(0);
}
public static void main (String args[])
{
        Client application ;
        if(args.length == 0)
        //      application = new Client("192.168.0.2");
                application = new Client("127.0.0.1");
        else
                application = new Client(args[0]);
        application.runClient();
}
}               // End Program
```

โปรแกรมที่3 : Client2.java ส่วนที่ทำงานบนฝั่งเครื่องที่เป็นศูนย์กลาง

```
import java.io.*;
import java.net.*;
import javax.swing.*;
import java.util.*;
import java.sql.*;
  class Connect{
            static String serverName = "none";
            static String serverName2 = "none";
            static Connection  conn = null;
            static      int idVal = 0;
            static  int id = 0;
            static      int swapID = 0;

  public static void cutVector(String vec){
            StringTokenizer st = new StringTokenizer(vec);
            int i = 0;
            String a[] = new String [11];
            while (st.hasMoreTokens()){
                        a[i] =      st.nextToken();
                        i++ ;
            }
//          System.out.println(a[5]);
  /* a[0] = server_IP , a[1] = timeStamp , a[2]= action,
  a[3] = File No. , a[4] = size/size , a[5] = URL */
            id = getObjectId(a[5], "object" , "URL","obj_ID");
            if (id == 0){
                        insertToDatabase("object","URL",a[5]); //("URL" is name of column)
                        }
  //        System.out.println(id);
            if (a[0].equalsIgnoreCase(serverName2)){
  //                    int size = getSize(a[4]);
                        }
            else {
                        id = getObjectId(a[0], "ServerName" , "ServerIP","ServerID");
                        if (id == 0){
                        insertToDatabase("ServerName","ServerIP",a[0]);
                         //("URL" is name of column)
                        id = getObjectId(a[0], "ServerName" , "ServerIP","ServerID");
                                                }
                        if (showTable("Server"+id)==1){ }
                                                //Check to if that Serverip exist or not
                        else {
                                    createTable("Server"+id, "_actobj");
                                    createTable("Server"+id, "_hisobj");
  //                                serverName=a[0];
                                    } End else
                        serverName="Server"+id;     //ServerName in database
                        serverName2=a[0];           //Server IP
                                    } End If
            int size = Math.abs(getSize(a[4]));
            if (size > 0){
                        if (a[2].equalsIgnoreCase("SWAPOUT")){
                        id = getObjectId(a[5] , "object" , "URL","obj_ID");
                        swapID = getObjectId(id+"", serverName+"_actobj", "Obj_ID","obj_ID");
                        if (swapID == 0){
                                    insertToDatabase(serverName+"_actobj","Obj_ID,Size,SwapTime",
                                    id+",'"+size+"','"+a[1]);}
                                    else {
                                                double    k_time=Double.parseDouble(a[1]);
                                                double    s_time = keepTime(swapID+"", serverName+"_actobj"
, "Obj_ID");
                                                k_time = k_time - s_time;

                        deleteFromDatabase(serverName+"_actobj",swapID);
                        insertToDatabase(serverName+"_hisobj","Obj_ID,Size,Swapout_Time,
```

```
Keep_Time,Release_code",
            swapID+"','"+size+"','"+a[1]+"','"+k_time+"','2");
            insertToDatabase(serverName+"_actobj","Obj_ID,Size,SwapTime",
                                                id+"','"+size+"','"+a[1]);
  //                                            System.out.println(swapID);
                                                }
                                    } else {
if (a[2].equalsIgnoreCase("RELEASE") /*&& !a[3].equalsIgnoreCase("FFFFFFFF")*/){
                                                id = getObjectId(a[5] , "object" , "URL","obj_ID");
                                                swapID = getObjectId(id+"", serverName+"_actobj",
"Obj_ID","obj_ID");
  //                                            System.out.println(id+a[3]+swapID);
            if (swapID!=0){
            //          System.out.println(swapID);
                        double k_time=Double.parseDouble(a[1]); // release time
                        double s_time = keepTime(swapID+"", serverName+"_actobj" , "Obj_ID");
                        k_time = k_time - s_time;
                        deleteFromDatabase(serverName+"_actobj",id);
  insertToDatabase(serverName+"_hisobj","Obj_ID,Size,Swapout_Time,Keep_Time,
Release_code",swapID+"','"+size+"','"+a[1]+"','"+k_time+"','1");
            //System.out.println(swapID+"A"+size+"A"+a[1]+"A"+k_time+"1");
            //System.out.println(size);
            }
            else System.out.println("Release before Swapout"+ swapID);
                        }
}
  //return(vec);
  }
}
  public static double keepTime(String url , String table ,String column){
            double time = 0;
            try {
                        Statement s = conn.createStatement ();
                        s.executeQuery ("SELECT * FROM "+ table +" where " +column+" ="+"'"+ url+"'");
                        ResultSet rs = s.getResultSet ();
  //                    System.out.println("YES");
                        int count = 0;
                        if (rs.next ())
                                    {
                                                time = rs.getDouble ("swaptime");
  //                                            System.out.println (time);
                                    }
                        else time = 0;
                                    rs.close ();
                                    s.close ();
                        //          System.out.println (count + " rows were retrieved");
            }
                        catch (Exception e) {}
                        return (time);
  } //keepTime
  public static int getSize(String s){
            try {
            int no = s.length();
            no = no/2+1;
            s = s.substring(no);
            no = Integer.parseInt(s);
            return(no);
  } catch (Exception e){
            return(0);}
  }
  public static void makeConnect(){
  //        Connection conn = null;
            try
            {
                        String userName = "root";
                        String password = "sambaggio";
                        String url = "jdbc:mysql://localhost/squid";
                        Class.forName ("org.gjt.mm.mysql.Driver").newInstance ();
```

```
                    conn = DriverManager.getConnection (url, userName, password);
                    System.out.println ("Database connection established");

            }
            catch (Exception e)
                    {
                    System.err.println ("Cannot connect to database server");
                    System.out.println(e);
                    }
    } // End method makeConnect.
    public  static void closeConnect(){
            try
            {
            if (conn != null)
            {
                    conn.close ();
                    System.out.println ("Database connection terminated");
            }
    }
    catch (Exception e) {System.out.print("ERROR ON closeConnect");}
            }

    public static int getObjectId(String url , String table ,String column,String getCol){
    try {
    Statement s = conn.createStatement ();
    s.executeQuery ("SELECT * FROM "+ table +" where " +column+" ="+"'"+ url+"'");
    ResultSet rs = s.getResultSet ();
    int count = 0;
            if (rs.next ())
                    {
                    idVal = rs.getInt (getCol);
    //              System.out.println ("id = " + idVal);
                            }
                    else idVal = 0;
            rs.close ();
            s.close ();
    } // End try
            catch (Exception e) {System.out.print("ERROR ON getOgjectID" + e);}
            return(idVal);
            }
    public static int showTable(String table){
    //      System.out.println(s);
            try {
                    Statement s = conn.createStatement ();
                    s.executeQuery ("show  columns from " + ""+table+"_actobj");
                    ResultSet rs = s.getResultSet ();
                    int count = 0;
                    if (rs.next ())
                            {
                                    idVal = 1;
                            }
                    else idVal = 0;
                            rs.close ();
                            s.close ();
            }
            catch (Exception e) {System.out.print("Table does'nt Exist");
                    idVal = 0;
                    return (idVal);
            }
                    System.out.println(idVal);
                    return (idVal);
    } //showTable

  public static void createTable(String table ,String type){
    if(type.equalsIgnoreCase("_actobj")){
    try {
            Statement s = conn.createStatement ();
            s.executeUpdate (
```

```
                    "CREATE TABLE "+table+type+" ("
                    + "Obj_ID MEDIUMINT(9) UNSIGNED NOT NULL DEFAULT'0',"
                    + "PRIMARY KEY (Obj_ID),"
                    + "Size MEDIUMINT(9),"
                    + "SwapTime DECIMAL(12,3))");
                s.close ();
                    //          System.out.println (count + " rows were retrieved");
                }
                catch (Exception e) {System.out.print("ERROR ON CreateTable (_actobj)" + e);}
                    }
                if(type.equalsIgnoreCase("_hisobj")){
        try {
                Statement s = conn.createStatement ();
                s.executeUpdate (
                "CREATE TABLE "+table+type+" ("
                + "Rec_ID MEDIUMINT(8) UNSIGNED NOT NULL AUTO_INCREMENT DEFAULT'0',"
                + "Obj_ID MEDIUMINT(9) UNSIGNED NOT NULL DEFAULT'0',"
                + "PRIMARY KEY (Rec_ID),"
                + "Size MEDIUMINT(9),"
                + "Swapout_Time DECIMAL(12,3),"
                + "Keep_Time Decimal(12,3),"
                + "Release_code SMALLINT(6))");
        s.close ();
                }
        catch (Exception e) {System.out.print("ERROR ON CreateTable (_hisobj)" + e);}
        }
}
public static void insertToDatabase(String table ,String column,String url){
        try {
        Statement s = conn.createStatement ();
        int count;
        count = s.executeUpdate (
          "INSERT INTO "+table+" ("+column+")"
          + " VALUES"
          + "('" + url + "')");
        s.close ();
//        System.out.println (count + " rows were inserted");
          }
          catch (Exception e) {System.out.print("ERROR ON INSERT "+e);}
          }         //End insertToDatabase method.

public static void deleteFromDatabase(String table ,int column){
        try {
          Statement s = conn.createStatement ();
          int count;
          count = s.executeUpdate (
          "DELETE FROM "+table+" where obj_ID = '"+ column +"'");
          s.close ();
          //                          System.out.println (column);
          //                          System.out.println (count + " rows were inserted");
        }
          catch (Exception e) {System.out.print("ERROR ON DELETE");}
          }         //End insertToDatabase method.

        } //End Class Connect

public class Client2 extends JFrame {
          private ObjectOutputStream output;
          private ObjectInputStream input;
          private String message="";
          private String chatServer;
          private Socket client;
          private int counter =1 ;
          static Vector v = new Vector();

public Client2(String host){
          super("Client2");
          chatServer = host;
```

```
        }

public void runClient(){
        try {
        connectToServer();
        getStreams();
        processConnection();
        closeConnection();
                }
        catch (EOFException eofException) {
                System.out.println("Server terminated connection");
                }
        catch (IOException ioException){
                ioException.printStackTrace();
                }
        }

private void getStreams() throws IOException {
       output = new ObjectOutputStream(client.getOutputStream());
       output.flush();
       input = new ObjectInputStream(client.getInputStream());
       System.out.println("\nGot I/O Streams\n");
        }

private void connectToServer() throws IOException
        {
        System.out.println("Attempting connection\n");
        client = new Socket(InetAddress.getByName(chatServer),3456);
        System.out.println("Connected To : "
        +client.getInetAddress().getHostName());
        }

private void processConnection()throws IOException {
//      Vector v = new Vector();
        try{
                while(input.available() >=0){
//      message =(String)input.readObject();
        message = (String)(client.getInetAddress().getHostAddress()+" " + input.readObject());
        v.addElement(new String(message));
        }
        //      System.out.println(v.size());
        }
                catch (Exception e){}
        }

private void closeConnection() throws IOException {
        System.out.println("\nUserTerminatedConnection");
        output.close();
        input.close();
        client.close();
        }

public static void main (String args[]) throws IOException
        {
         FileInputStream fin = new FileInputStream("d:/read/project.conf");
        BufferedInputStream bin = new BufferedInputStream(fin);
        DataInputStream din = new DataInputStream(bin) ;
        String ipConf;
        while(( ipConf= din.readLine()) != null){
                if(ipConf.startsWith("//") || ipConf.equalsIgnoreCase("\n")){}
                else {
                        StringToVector.cutString(ipConf);
                 }
        }
        for (int a = 0; a < StringToVector.v.size() ; a++)
        {
                Client2 application ;
```

```
                if(args.length == 0)
                        application = new Client2(StringToVector.v.elementAt(a).toString());
                else
                        application = new Client2(args[0]);

                application.runClient();                //        receive Vector;
                Connect.makeConnect();      //        Start connect to database.
                for (int i= 0; i < v.size() ; i++ )
                        {
                        Connect.cutVector(v.elementAt(i).toString());
                        }
                System.out.println(v.size());
                Connect.closeConnect();
                        }

                while(true){
                        Server application2 = new Server();
                        application2.runServer();
        //              Connect.makeConnect();
                        System.out.println(Server.v.size());
                        Connect.makeConnect();
                        for (int i= 0; i < Server.v.size() ; i++ )
                        {
                                Connect.cutVector(Server.v.elementAt(i).toString());
                        }
                        Server.v.clear();
                 Server.server.close();
                        Connect.closeConnect();
                }
          }

          }

        /*****************************************************************************/

class Server extends JFrame {           //Act as Server in Client Side
        static ServerSocket server;
        static Socket connection;
        private ObjectOutputStream output;
        private ObjectInputStream input;
        private String message="";
        private int counter =1 ;
        static Vector v = new Vector();

public Server(){
        super("Server");
          }
public void runServer()
          {
          try {
                server = new ServerSocket(3456,100);
                String address = waitForConnection();
                System.out.println(address);
                getStream();
                processConnection();
                closeConnection();
                }
          catch (EOFException eofException) {
                System.out.println("Client terminated connection");
          try {
                closeConnection();
                Server application = new Server();
                application.runServer();
                }
          catch (Exception e) {}
                }
                catch (IOException ioException){
                ioException.printStackTrace();
```

```
                }
        }

private String waitForConnection() throws IOException{
        System.out.println("Waiting for connection\n");
        connection = server.accept();
        System.out.println("Connection " + counter + "recived from: " +
        connection.getInetAddress().getHostName().toString());
        return connection.getInetAddress().getHostName().toString();
        }

private void getStream() throws IOException {
        input = new ObjectInputStream(connection.getInputStream());
        System.out.println("\nGot I/O Streams\n");
        }

private void processConnection()throws IOException {
        //      Vector v = new Vector();
        try{
                while(input.available() >=0){
        //      message =(String)input.readObject();
                message = (String)(connection.getInetAddress().getHostAddress()+" " + input.readObject());
                v.addElement(new String(message));
                }
        }
        catch (Exception e){System.out.println(e);}
        }

private void closeConnection() throws IOException {
        System.out.println("\nUserTerminatedConnection");
        input.close();
        connection.close();
        }

       }

class StringToVector {
      static Vector v = new Vector();

public static void cutString(String s){
        StringTokenizer st = new StringTokenizer(s);
        while (st.hasMoreTokens())
                {
                v.addElement(st.nextToken());
                }
        } //cutString
} // End Class Squid
```

ประวัติผู้เขียน

ชื่อ-สกุล : นายวรวัฒน์ เชิญสวัสดิ์

วัน-เดือน-ปีเกิด : 20 กันยายน 2519

ที่อยู่ปัจจุบัน : 97/122 ซอย สวนฝรั่ง ถ.ริมคลองประปา เขต บางซื่อ แขวง บางซื่อ กทม. 10800

E-MAIL : sambaggio@yahoo.com

ประวัติการศึกษา

- ปีการศึกษา 2531 ชั้นประถมศึกษา โรงเรียนอัสสัมชัญ ธนบุรี
- ปีการศึกษา 2534 ชั้นมัธยมศึกษาตอนต้น โรงเรียน Canley Vale High School (Australia)
- ปีการศึกษา 2537 ชั้นมัธยมศึกษาตอนปลาย โรงเรียนอัสสัมชัญ ธนบุรี
- ปีการศึกษา 2542 ปริญญาตรี วิศวกรรมศาสตร์ สาขาก่อสร้าง สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง