วิทยานิพนธ์

เรื่อง

การออกแบบและพัฒนาระบบเสิร์จเอ็นจินแบบกระจาย

A Design and Implementation of Distributed Search Engine

โดย

นาย สุทธิพล วรางฤทธิ์

พ.ศ. 2544

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์)

วิศวกรรมคอมพิวเตอร์ วิศวกรรมคอมพิวเตอร์

การออกแบบและพัฒนาระบบเสิร์จเอ็นจินแบบกระจาย

A Design and Implementation of Distributed Search Engine

นายสุทธิพล วรางฤทธิ์

รองศาสตราจารย์สุรศักดิ์ สงวนพงษ์, วศ.ม.

ผู้ช่วยศาสตราจารย์สมนึก คีรีโต, Ph.D.

ผู้ช่วยศาสตราจารย์ภุชงค์ อุทโยภาศ, Ph.D.

อาจารย์ประดนเดช นีละคุปต์, M.S.

ศาสตราจารย์ทัศนีย์ อัตตะนันทน์, D.Agr

วิทยานิพนธ์

เรื่อง

การออกแบบและพัฒนาระบบเสิร์จเอ็นจินแบบกระจาย

A Design and Implementation of Distributed Search Engine

โดย

นายสุทธิพล วรางฤทธิ์

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์
เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์)
พ.ศ. 2545

ISBN 974-160-174-3

สุทธิพล วรางฤทธิ์ 2545: การออกแบบและพัฒนาระบบเสิร์จเอ็นจินแบบกระจาย ปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์) สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ ประธานกรรมการที่ปรึกษา: รองศาสตราจารย์สุรศักดิ์ สงวนพงษ์, M.Eng. 54 หน้า
ISBN 974-160-174-3

การใช้เทคโนโลยีคอมพิวเตอร์คลัสเตอร์เป็นเทคนิคประการหนึ่งที่ใช้ประยุกต์เพื่อพัฒนาเสิร์จเอ็นจินแบบกระจาย โครงสร้างของเสิร์จเอ็นจินที่พัฒนามีองค์ประกอบหลัก 3 ส่วน คือ ส่วนรวบรวมเอกสาร ส่วนสร้างดัชนีฐานข้อมูล และส่วนบริการสืบค้น คอมพิวเตอร์แต่ละเครื่องในคลัสเตอร์จะประกอบด้วยส่วนประกอบหลักทั้งสามส่วนและสามารถทำงานได้อย่างเป็นอิสระจากกัน โดยใช้การแฮชของยูอาร์แอล ผลลัพธ์การสืบค้นจากคลัสเตอร์จะได้รับการจัดรวมโดยอาศัยหลักการแบบเมตาเสิร์ชที่พัฒนาขึ้นเพื่อนำไปจัดลำดับการแสดงผลต่อไป ในวิทยานิพนธ์นี้ยังได้เสนอวิธีการจัดลำดับผลการค้นหาโดยอาศัยจำนวนลิงค์ชี้เข้าของเว็บเพจ

จากผลการทดลองโดยใช้คลัสเตอร์ที่ประกอบด้วยคอมพิวเตอร์จำนวน 4 เครื่อง ระบบสามารถอ่านเอกสารได้เร็วขึ้น 3.87 เท่า สามารถสร้างดัชนีค้นได้เร็วขึ้น 4.63 เท่า และสามารถรองรับการค้นหาได้เร็วขึ้น 4.79 เมื่อเทียบกับการใช้คอมพิวเตอร์เพียงเครื่องเดียว

Suttiphol Warangrit 2002: A Design and Implementation of Distributed Search Engine. Master of Engineering (Computer Engineering), Major Field Computer Engineering, Department of Computer Engineering. Thesis Advisor: Associate Professor Surasak Sanguanpong, M.Eng. 54 pages.
ISBN 974-160-174-3

Cluster Computing is one of the most powerful techniques to be applied for a development of distributed search engines. The search engine is composed of three main components: a web pages collector, an indexing, and a query service. Each machine insides the cluster has the three mentioned components and can independently work using URL hashing. Search results are combined with metasearch techniques before they have been ranked. In this thesis, a simple ranking method, which utilizes the page link connectivity, has been proposed.

The experimental results with four nodes of cluster show that the system increases collection speed 3.87 times. An indexing speed increases 4.63 times and query performance increase 4.93 times.

คำนิยม

งานวิจัยฉบับนี้สามารถเสร็จสมบูรณ์ได้โดยได้รับความอนุเคราะห์จากบุคคลหลายท่าน ผู้เขียนขอกราบขอบพระคุณ รองศาสตราจารย์สุรศักดิ์ สงวนพงษ์ อาจารย์ที่ปรึกษาของผู้เขียน ที่ให้คำปรึกษา แนะนำและจัดหาอุปกรณ์ต่าง ๆในการทำวิจัย ขอกราบขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.สมนึก คีรีโต ผู้ช่วยศาสตราจารย์ ดร.ภุชงค์ อุตโยภาศ และผู้ช่วยศาสตราจารย์ ดร.ชุลีรัตน์ จรัสกุลชัย และอาจารย์ทุก ๆท่านในภาควิชาที่กรุณาให้คำแนะนำในการสอบและตรวจแก้ไขวิทยานิพนธ์จนเสร็จสมบูรณ์

ขอขอบคุณ สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ(สวทช) สถาบันวิจัยมหาวิทยาลัยเกษตรศาสตร์ และบริษัท ซี.เอส. คอมมิวนิเคชั่น ที่ให้ทุนในการดำเนินการวิจัย

ขอขอบคุณ นายกษม โคตรอาษา นายนาถพงษ์ อัมพรอร่ามเวทย์ และเพื่อน ๆ ในห้องปฏิบัติการเครือข่ายประยุกต์ ที่ให้คำแนะนำในการออกแบบและเขียนโปรแกรม สุดท้ายผู้เขียนกราบขอบพระคุณ นายสมพงษ์ แซ่ตั้ง และนางปัทมา วรางฤทธิ์ บิดาและมารดาของผู้เขียนที่ได้ให้กำลังใจมาโดยตลอด จนทำให้งานวิจัยชิ้นนี้สำเร็จได้โดยสมบูรณ์

สุทธิพล วรางฤทธิ์
20 มิถุนายน 2544

ประวัติการศึกษาและการทำงาน

ชื่อ นายสุทธิพล วรางฤทธิ์
เกิดวันที่ 9 เดือนกรกฎาคม พ.ศ. 2520
สถานที่เกิด จังหวัด กรุงเทพมหานคร
ประวัติการศึกษา วศ.บ. (คอมพิวเตอร์) มหาวิทยาลัยเกษตรศาสตร์ (พ.ศ. 2541)
สถานที่ทำงานปัจจุบัน บริษัท เคเอสซี คอมเมอร์เชียล อินเทอร์เน็ต จำกัด
ทุนการศึกษาที่ได้รับ ได้รับทุนบัณฑิตศึกษาภายในประเทศ สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ

สารบาญ

หน้า

สารบาญตาราง

ตารางที่ **หน้า**

ตารางผนวกที่

สารบาญภาพ

ภาพที่ **หน้า**

การออกแบบและพัฒนาระบบเสิร์จเอ็นจินแบบกระจาย

A Design and Implementation of Distributed Search Engine

คำนำ

ในปัจจุบันความนิยมในการใช้งานเสิร์จเอ็นจินได้เพิ่มขึ้นอย่างรวดเร็ว เสิร์จเอ็นจินได้เข้ามามีบทบาทสำคัญในฐานะเครื่องมือที่ใช้สำหรับค้นคว้าหาข้อมูลในอินเทอร์เน็ต การพัฒนาระบบเสิร์จเอ็นจินที่สามารถค้นหาข้อมูลได้ถูกต้องตรงตามความต้องการและสามารถรับปริมาณเอกสารจำนวนมากได้ จึงเป็นงานมีผู้วิจัยและพัฒนาจำนวนมาก หากแต่ความก้าวหน้าในการวิจัยและพัฒนามักไม่เป็นที่เผยแพร่เนื่องจากผลประโยชน์ทางธุรกิจ

ปัญหาที่สำคัญประการหนึ่งของเสิร์จเอ็นจินก็คือ จำนวนเอกสารที่ต้องเก็บรวบรวมเพื่อทำดัชนีในปัจจุบันมีจำนวนมากมายมหาศาลเกินกว่าที่จะใช้คอมพิวเตอร์เพียงหนึ่งเครื่องในการรวบรวมเอกสารได้ จึงจำเป็นที่จะต้องใช้ระบบที่มีสมรรถนะสูง

คอมพิวเตอร์ที่มีสมรรถนะสูงอย่างซูเปอร์คอมพิวเตอร์ มีราคาสูงเกินกว่าที่จะจัดหาได้ และยังมีขีดจำกัดในด้านการขยายขีดความสามารถ ในปัจจุบันได้มีเทคโนลียีที่เข้ามาเป็นทางเลือกอีกทางเลือกหนึ่ง แทนซูเปอร์คอมพิวเตอร์ ซึ่งนั้นก็คือ ระบบคลัสเตอร์(Clustering system)

ระบบคลัสเตอร์เป็นระบบที่นำเอาคอมพิวเตอร์หลายเครื่องมาเชื่อมต่อกันผ่านเครื่อข่ายความเร็วสูง คอมพิวเตอร์แต่ละเครื่องในระบบคลัสเตอร์จะช่วยกันทำงานซึ่งกันและกัน ข้อดีของระบบคลัสเตอร์ก็คือ มีสรรถนะสูงในราคาถูก อีกทั้งยังสามารถเพิ่มประสิทธิภาพของระบบได้ง่าย

<u>วัตถุประสงค์</u>

วิทยานิพนธ์นี้มีวัตถุประสงค์เพื่อศึกษาและออกแบบระบบเสิร์จเอ็นจินแบบกระจายสามารถทำงานได้บนระบบคลัสเตอร์ ทั้งในส่วนของการรวบรวมเอกสาร การสร้างดัชนีฐานข้อมูลและการค้นหาเอกสาร โดยที่การออกแบบจะเน้นให้คอมพิวเตอร์แต่ละเครื่องมีการทำงานที่เป็นอิสระไม่ขึ้นต่อกัน ทั้งนี้เพื่อเพิ่มความรวดเร็วในการทำงานและความยืดหยุ่นของระบบ

นอกจากนี้ยังได้เสนอวิธีการจัดลำดับผลลัพธ์ของการค้นหา โดยนำเอาจำนวนลิงค์ชี้เข้าของเว็บเพจแต่ละเว็บเพจ มาเป็นตัวช่วยในการจัดเรียงลำดับให้ดียิ่งขึ้น

การตรวจเอกสาร

ประเภทของบริการค้นหาเว็บ

บริการค้นหาเว็บเพจที่มีอยู่ในปัจจุบันเช่น Google, AltaVista, Lycos, Yahoo, HotBot มักเรียกโดยรวมว่าเป็นเสิร์จเอ็นจินนั้นแท้ที่จริงแล้วสามารถแบ่งประเภทของเครื่องมือที่ช่วยในการค้นหาเว็บเพจได้เป็น 3 ประเภท (Sullivan, 1996) ดังนี้

เสิร์จเอ็นจิน

เสิร์จเอ็นจิน เป็นระบบซอฟต์แวร์ที่มีโปรแกรม โรบอต (Robot) ซึ่งบางครั้งเรียกว่า สไปเดอร์ (Spider) หรือ ครอว์เลอร์ (Crawler) ทำหน้าที่เดินทางไปยังเว็บไซต์ต่าง ๆในอินเทอร์เน็ต และอ่านเว็บเพจจากไซต์เหล่านั้นเพื่อนำมาสร้างดัชนีรายการของเว็บเพจโดยอัตโนมัติ การทำดัชนีรายการเว็บเพจด้วยเสิร์จเอ็นจินจะสามารถสร้างดัชนีของเว็บเพจได้เป็นจำนวนที่มากและรวดเร็ว ถ้าหากเว็บเพจที่ถูกทำดัชนีแล้วเกิดมีการเปลี่ยนแปลง เสิร์จเอ็นจินนำเว็บเพจที่มีการเปลี่ยนแปลงนั้นมาสร้างดัชนีใหม่ ซึ่งการสร้างดัชนีใหม่นี้อาจส่งผลต่อการจัดลำดับของเว็บเพจนั้น ตัวอย่างของเสิร์จเอ็นจินที่มีอยู่ในปัจจุบันเช่น HotBot, AltaVista เป็นต้น

ไดเรกทอรี (Directory)

ไดเรกทอรี แตกต่างจากเสิร์จเอ็นจินตรงที่ไดเรกทอรีถูกจัดการโดยบุคคล เว็บไซต์ใดที่ต้องการมีรายชื่อในไดเรกทอรีต้องติดต่อผู้ดูแลไดเรกทอรีเพื่อให้ผู้ดูแลไดเรกทอรีจำแนกประเภทของเว็บเพจนั้น ๆให้อยู่ในประเภทที่เหมาะสม เนื่องจากไดเรกทอรีถูกจัดเป็นหมวดหมู่ การค้นหาเว็บเพจด้วยไดเรกทอรีจึงให้ผลลัพธ์ที่ตรงกับความต้องการมากกว่าเสิร์จเอ็นจิน แต่ปริมาณเอกสารของไดเรกทอรีด้วยส่วนใหญ่จะมีจำนวนน้อยกว่าของเสิร์จเอ็นจิน ทั้งนี้เพราะดัชนีของไดเรกทอรีถูกจัดการโดยบุคคล ทำให้การปรับปรุงข้อมูลช้ากว่าการใช้ระบบอัตโนมัติ หากมีการเปลี่ยนแปลงเกิดขึ้นกับเว็บเพจที่ได้รับการจัดลำดับแล้ว การเปลี่ยนแปลงนั้นจะไม่ส่งผลกระทบต่อลำดับเดิมที่ได้รับการจัดไว้ ตัวอย่างไดเรกทอรีที่มีอยู่ในปัจจุบันเช่น Yahoo

ไฮบริดเสิร์จเอ็นจิน (Hybrid Search Engine)

ไฮบริดเสิร์จเอ็นจิน เป็นเสิร์จเอ็นจินที่การผสมผสานการทำงานของเสิร์จเอ็นจินและไดเรกทอรีเข้าด้วยกัน มีข้อดีของทั้งเสิร์จเอ็นจินและไดเรกทอรีรวมอยู่ด้วยกัน

ความเป็นมาของเสิร์จเอ็นจิน

เสิร์จเอ็นจิน หมายถึง ระบบซอฟต์แวร์ที่ให้บริการค้นหาเอกสาร ในปัจจุบันมีเสิร์จเอ็นจินที่เป็นที่นิยมอยู่มากมาย เช่น Google, AltaVista, Lycos, Hotbot เป็นต้น แม้ว่าเสิร์จเอ็นจินนั้นจะมีอยู่มากมายหลายแห่ง แต่ต้นแบบของเสิร์จเอ็นจินนั้นเริ่มต้นมาจากโปรแกรมเดียวกัน คือ โปรแกรมที่มีชื่อว่า “Archie” (Sonnenreich และ Macinta, 1998)

ปี ค.ศ. 1990 นาย Alan Emtage นักศึกษาของมหาวิทยาลัย McGill ในสหรัฐอเมริกาได้พัฒนา Archie ซึ่งพัฒนาโดย โปรแกรม Archie มีความสามารถให้บริการค้นหาไฟล์ต่าง ๆ ที่อยู่บนเอฟทีพีเซิร์ฟเวอร์ (FTP Server) การทำงานของโปรแกรม Archie จะมีโปรแกรมเล็ก ๆซึ่งทำหน้าที่รวบรวมรายชื่อไฟล์จากเอฟทีพีเซิร์ฟเวอร์ต่าง ๆมาสร้างฐานข้อมูลเก็บไว้ และจะมีอีกโปรแกรมทำหน้าที่รับชื่อไฟล์ที่ผู้ใช้ต้องการมาค้นหาในฐานข้อมูล

ต่อมาในปี ค.ศ.1993 มหาวิทยาลัย Nevada ได้พัฒนาโปรแกรมที่มีชื่อว่า Veronica (Very Easy Rodent-Oriented Netwide Index to Computerized Archives) Veronica มีการทำงานคล้ายกับโปรแกรม Archie แต่แตกต่างกันตรงที่ โปรแกรม Archie นั้นนำชื่อไฟล์มาจากเอฟทีพีเซิร์ฟเวอร์ แต่โปรแกรม Veronica นำชื่อไฟล์ที่เป็นเอกสารมาจากโกเฟอร์เซิร์ฟเวอร์ (Gopher Server)

ในปี ค.ศ. 1993 นาย Matthew Gray ได้พัฒนาโปรแกรม Wanderer ขึ้นมาเพื่อใช้สำหรับรวบรวม URL มาเก็บไว้ในฐานข้อมูลที่มีชื่อว่า Wandex ข้อมูล URL ที่รวบรวมมาได้จะถูกนำไปวิเคราะห์หาอัตราการเติบโตของเวิลด์ไวด์เว็บ (World Wide Web) โปรแกรม Wanderer นี่เองถือเป็นเว็บโรบอต (Web Robot) ตัวแรกของโลก

ในเดือนตุลาคม ปี ค.ศ. 1993 นาย Martijn Koster ได้เปิดตัวโปรแกรม ALIWEB ซึ่งมีการทำงานของโปรแกรมคล้ายกับโปรแกรม Archie แต่ข้อมูลที่ใช้แทนที่จะเป็นชื่อไฟล์จากเอฟทีพีจะเป็นข้อมูลของเว็บแทน แต่เป็นที่น่าเสียดายที่โปรแกรม ALIWEB นั้นไม่มีเว็บโรบอตอยู่ในตัวเอง แต่ให้เจ้าของเว็บไซต์มากรอกรายละเอียดของเว็บไซต์แทน ด้วยเหตุนี้เองทำให้ฐานข้อมูลของโปรแกรม ALIWEB มีขนาดเล็กและไม่เป็นที่นิยม ซึ่งต่อมาได้มีเสิร์จเอ็นจินเกิดขึ้นอีกหลายแห่งซึ่งมีเว็บโรบอตทำหน้าที่รวบรวมเว็บเพจ เช่น JumpStation, WWW Worm และ Excite แต่เสิร์จเอ็นจินเหล่านี้ก็ยังไม่สามารถค้นหาคำทั้งหมดในเว็บเพจ ค้นหาได้แต่เฉพาะคำที่อยู่ในส่วนของหัวเรื่อง(Title) หรือ 100 ตัวอักษรแรกเท่านั้น จนกระทั่งในปี 1994 ได้เกิดเสิร์จเอ็นจิน Webcrawler ซึ่งสามารถที่จะค้นหาคำทั้งหมดในเว็บเพจ(Full Text Search)ได้เป็นแห่งแรก

หลังจากที่ Webcrawler ได้เปิดตัวเป็นเสิร์จเอ็นจินที่สามารถค้นหาคำทั้งหมดในเว็บเพจได้เป็นแห่งแรก ต่อมาก็ได้มีการพัฒนาเสิร์จเอ็นจินอีกมากมายหลายแห่ง เช่น Lycos (Mauldin, 1997) ซึ่งพัฒนาโดยมหาวิทยาลัย Carnegie Mellon แม้ว่า Webcrawler จะสามารถค้นหาคำทั้งหมดได้เป็นโปรแกรมแรก แต่ webcrawler ก็ยังไม่เป็นที่นิยมมากนั้น เสิร์จเอ็นจินที่เป็นที่นิยมและมีความสามารถพิเศษโดดเด่นในขณะนั้นมีเพียง AltaVista

AltaVista เป็นเสิร์จเอ็นจินที่พัฒนาโดย Digital Equipment Corporation (DEC) ในปี ค.ศ. 1995 โดยมีความสามารถพิเศษนอกจากที่จะค้นหาคำศัพท์แบบ Full Text Search ได้แล้ว ยังมีการค้นหาแบบพิเศษคือ สามารถค้นหาแบบมีเครื่องหมายบูลีน (Boolean Operator), สามารถค้นหาเว็บไซต์ที่มีไฮเปอร์ลิงค์(hyperlink) ชี้มายังเว็บเพจใด ๆได้ นอกจากนี้ยังสามารถให้บริการการค้นหาได้มากกว่า 1 ล้านครั้งต่อวัน ซึ่งในขณะนั้นถือว่าเป็นเสิร์จเอ็นจินที่มีดัชนีเว็บเพจมากที่สุด คือมีถึง 40 ล้านเว็บเพจ

การจัดเรียงลำดับผลลัพธ์ของเสิร์จเอ็นจิน เสิร์จเอ็นจินโดยทั่วไปจะใช้เนื้อหาและคำศัพท์ในเอกสารมาช่วยในการจัดเรียงเพียงอย่างเดียว ต่อมาได้มีผู้เสนอการนำข้อมูลด้านอื่น ๆมาช่วยในการจัดเรียงลำดับ เช่น ความสัมพันธ์ของไฮเปอร์ลิงค์ระหว่างเว็บเพจ เป็นต้น

ในปี ค.ศ. 1997 (Carriere และ Kazman, 1997) ได้มีการเสนอวิธีการจัดเรียงลำดับผลลัพธ์ของการค้นหาโดยอาศัย จำนวนลิงค์ชี้เข้าและจำนวนลิงค์ชี้ออก ซึ่งวิธีนี้ยังมีข้อเสียอยู่คือ หากมีเว็บเพจ 2 เว็บเพจที่มีจำนวนของลิงค์ชี้เข้าและออกเท่ากัน จะไม่สามารถบอกได้ว่าเว็บเพจในควรอยู่ลำดับที่ดีกว่ากัน แม้เว็บเพจรอบ ๆจะไม่เหมือนกัน นอกจากนั้นยังมีปัญหากับเว็บเพจที่มีจำนวนลิงค์ชี้เข้ามาก แต่ว่าไม่ได้มีเนื้อหาตรงกับเรื่องที่ผู้ค้นหาต้องการ เช่น เว็บเพจโฆษณา เป็นต้น

ต่อมาในปี ค.ศ. 1998 ได้มีงานวิจัย (Kleinberg, 1998) ที่เสนอวิธีการจัดเรียงลำดับผลลัพธ์ของเอกสารโดยอาศัยลิงค์ งานวิจัยนี้ได้พยายามชี้ให้เห็นปัญหาของการจัดเรียงผลลัพธ์ของการใช้จำนวนลิงค์ชี้เข้าเพียงอย่างเดียว และได้เสนอแนวทางแก้ไขโดยได้เสนอวิธีการคำนวนหาค่า Hub และ Authority ค่าทั้งสองค่านี้สามารถคำนวณได้จากสมมุติฐานที่ว่า เว็บเพจที่มีเนื้อหาที่ตรงกับเรื่องที่ค้นหาควรที่จะถูกชี้โดยเว็บเพจที่เป็นแหล่งรวมของเรื่องที่ค้นหาหลาย ๆเว็บเพจ และในทางกลับกัน เว็บเพจที่เป็นแหล่งรวมของเรื่องที่ค้นหาควรที่จะชี้ไปยังเว็บเพจที่มีเนื้อหาตรงกับเรื่องที่ค้นหาหลาย ๆเว็บเพจ จากสมมุติฐานดังกล่าวจะสามารถคำนวณหาค่า hub และ authority ได้ ซึ่งค่าที่ได้จะสามารถบอกได้ว่าเว็บเพจได้ตรงกับเรื่องที่ค้นหามากน้องเพียงใด

ในปีเดียวกัน ได้มีงานวิจัย (Page และคณะ, 1998) ที่เสนอแนวทางในการจัดลำดับผลลัพธ์ของการค้นหาโดยอาศัยค่า Page Rank ซึ่งค่า Page Rank สามารถคำนวณได้จากการนำเอาความสัมพันธ์ของลิงค์ซึ่งคล้ายกับการนับจำนวนลิงค์ชี้เข้า แต่จะดีกว่าตรงที่ค่า Page Rank ของเว็บใด ๆนั้นขึ้นอยู่กับ ค่า Page Rank ของเว็บเพจที่ชี้มาหามัน ซึ่งหากเว็บเพจที่ชี้มาหามันมีค่า Page Rank มากก็จะทำให้มันมีค่า Page Rank มากแต่หากเว็บเพจที่ชี้มาหามันมีค่า Page Rank น้อยก็จะทำให้มันมีค่า Page Rank น้อย ซึ่งวิธีนี้ดีกว่าการนับจำนวนลิงค์ชี้เข้าธรรมดา วิธีการคำนวณ Page Rank นี้เองได้ถูกนำมาใช้ในเสิร์จเอ็นจินที่มีชื่อว่า Google (Page และ Brin, 1998)

แม้ว่าเสิร์จเอ็นจินในปัจจุบันจะมีอยู่หลายแห่งก็ตาม แต่เสิร์จเอ็นจินที่มีขนาดใหญ่และเป็นที่รู้จักนั้นมีอยู่เพียงไม่กี่แห่ง ซึ่งแต่ละแห่งก็จะมีขนาด วิธีการจัดเรียงผลลัพธ์ และคุณสมบัติด้านต่าง ๆแตกต่างกันไป (Randy, 2002)(Greg, 2002) ดังสามารถสรุปได้ดังตารางที่ 1

ตารางที่ 1 เปรียบเทียบขนาดและคุณสมบัติของเสิร์จเอ็นจินที่เป็นที่นิยม

เสิร์จเอ็นจิน	ขนาด (ล้าน URLs)	การค้นหาแบบบูลีน	เคสเซ็นสิทีฟ	ค้นหาแบบวลี	ค้นหาแบบไม่เฉพาะเจาะจง
Google	1,300	and(default), or, not	ไม่ได้	ได้	ไม่ได้
AllTheWeb	575	and(default), or, not	ไม่ได้	ได้	ไม่ได้
Altavista	550	and, or(default), not	ได้	ได้	ได้ (ใช้เครื่องหมาย *)
NorthenLight	310	and(default), or, not	ไม่ได้	ได้	ได้ (ใช้เครื่องหมาย %)
Hotbot	500	and(default), or, not	ได้	ได้	ได้ (ใช้เครื่องหมาย *)
Excite	250	and, or(default), not	ไม่ได้	ได้	ไม่ได้

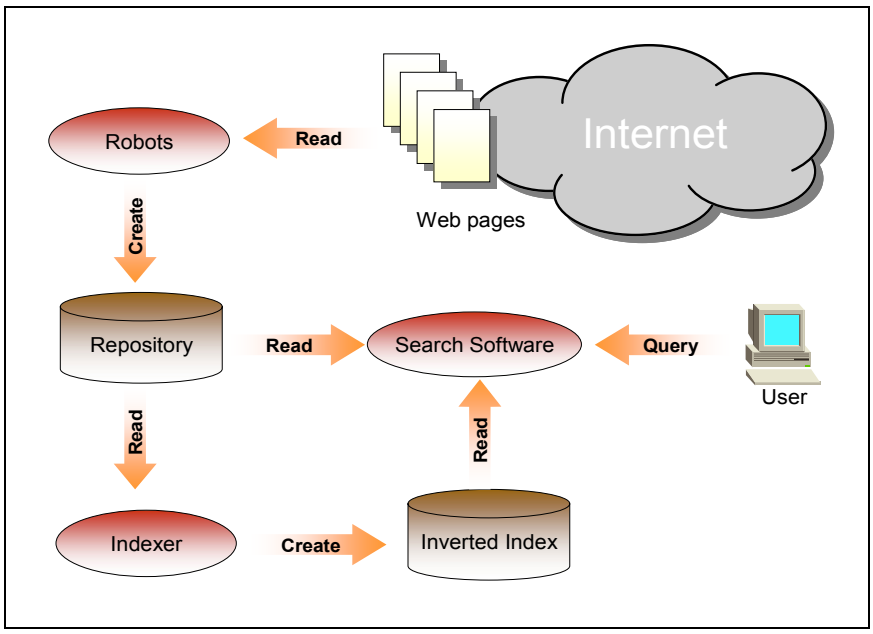
การทำงานของเสิร์จเอ็นจิน

ส่วนประกอบของเสิร์จเอ็นจิน

เราสามารถแบ่งองค์ประกอบต่าง ๆ (Sanguanpong, 1998 A) ของเสิร์จเอ็นจินตามหน้าที่การทำงานได้ 3 ส่วนดังภาพที่ 1 โดยที่แต่ละส่วนมีการทำงานดังต่อไปนี้

โรบอต หรือ สไปเดอร์ หรือ ครอว์เลอร์ (Koster, 1994) ทำหน้าที่เดินทางไปยังไซต์ต่าง ๆ เพื่อสะสมไฟล์ เฮชทีเอ็มแอล (HTML) ของเว็บเพจ แล้วติดตามลิงค์จากเว็บเพจนั้นไปยังเว็บเพจอื่น ๆ ภายหลังจากที่โรบอตได้อ่านเว็บเพจใด ๆแล้ว หลังจากนั้นโรบอตจะกลับไปยังไซต์ที่เคยสำรวจแล้วเพื่อตรวจสอบการเปลี่ยนแปลงตามจังหวะเวลาที่กำหนด

อินเด็กเซอร์ (Indexer, Catalog) เป็นส่วนที่ทำหน้าที่สร้างดัชนีค้นหาจากไฟล์เฮชทีเอ็มแอลที่โรบอตหามา เว็บเพจใด ๆจะสามารถสืบค้นได้จากเสิร์จเอ็นจินก็ต่อเมื่อเว็บเพจนั้นผ่านการทำดัชนีมาแล้วเท่านั้น ถ้าหากมีการเปลี่ยนแปลงกับเว็บเพจจะต้องแก้ไขข้อมูลดัชนีใหม่



ภาพที่ 1 การทำงานของส่วนประกอบแต่ละส่วนในระบบเสิร์จเอ็นจิน

โปรแกรมค้นหา(เสิร์จเอ็นจินซอฟต์แวร์) เป็นโปรแกรมทำหน้าที่รับคำศัพท์ที่ต้องการค้นหาผ่านทางซีจีไอ (CGI, Common Gateway Interface) เพื่อหาเว็บเพจที่ตรงกับความต้องการของผู้ค้นหาในฐานข้อมูล เสิร์จเอ็นจินแต่ละตัวจะมีวิธีจัดเรียงลำดับผลลัพธ์แตกต่างกันออกไป

การทำงานของโรบอต

การทำงานของโรบอตโดยทั่วไปจะมีลักษณะใกล้เคียงกันคือ โรบอตจะอ่านเว็บเพจเริ่มต้นที่กำหนดไว้ให้ หลังจากได้เว็บเพจเริ่มต้นมาแล้วโรบอตจะไปอ่านเว็บเพจเหล่านั้นมาแล้วตรวจสอบดูว่าภายในเว็บเพจนั้นมีไฮเปอร์ลิงค์ชี้ไปยัง URL อะไรบ้าง และตรวจสอบดูว่าโรบอตเคยพบ URL เหล่านั้นมาก่อนหรือไม่ หากเคยพบ URL มาก่อนก็จะทิ้ง URL นั้นไป แต่หากยังไม่เคยพบมาก่อน ก็จะมาวิเคราะห์ดูว่าสมควรที่จะอ่านเว็บเพจนั้นหรือไม่ กลไกนี้แสดงได้ดังภาพที่ 2

โรบอตทั่วไปจะสามารถกำหนดกฎได้ โดยกฎที่ตั้งไว้จะใช้สำหรับกำหนดคุณสมบัติของ URL เช่น จะอ่านเว็บไซต์ใดและไม่อ่านเว็บไซต์ใดบ้างดังภาพที่ 3

จากกฎที่ตั้งไว้หากโรบอตพบเว็บเพจที่ผ่านกฎที่ตั้งไว้ มันก็จะเก็บ URL นั้นไว้ในคิวเพื่อรอที่จะอ่านเว็บเพจนั้นต่อไป แต่หากเว็บเพจใดไม่ผ่านกฎโรบอตก็ข้าม URL นั้นไปโดยไม่สนใจ โปรแกรมโรบอตจะวนซ้ำอ่านเว็บเพจเช่นนี้ไปเรื่อย ๆจนกว่า URL ที่อยู่ในคิวจะหมดหรือจนกว่าจะได้เว็บเพจครบตามจำนวนที่ต้องการ

```
pseudo code of robots
begin
    initial $mainURLQueue from configuration file and
    initial $URLHash from $mainURLQueue
    initial $ConfigurationRule  from configuration file
    loop until ($mainURLQueue is empty)
    begin
        $URL=dequeue($mainURLQueue)
        download($URL)
        $URLQueue=parseURLLink($URL)
        loop until $URLQeue is empty
        begin
            $URL=dequeue($URLQueue)
            if (CheckConfigurationRule($URL)==PASS)
            && (have never seen this $URL before)
          enqueue($mainQueue,$URL)
         end loop
    end loop
end
```

ภาพที่ 2 Psudo code แสดงการทำงานของโปรแกรมโรบอต

```
################################################
#AN EXAMPLE OF ROBOT CONFIGURATION FILE
################################################
ALLOW *.th
DENY *.ku.ac.th
DENY *nectec.or.th
DENY *pantip.inet.co.th
#GET ALL WEBPAGE IN .th DOMAIN BUT NOT IN .ku.ac.th, nectec.or.th AND
pantip.inet.co.th
```

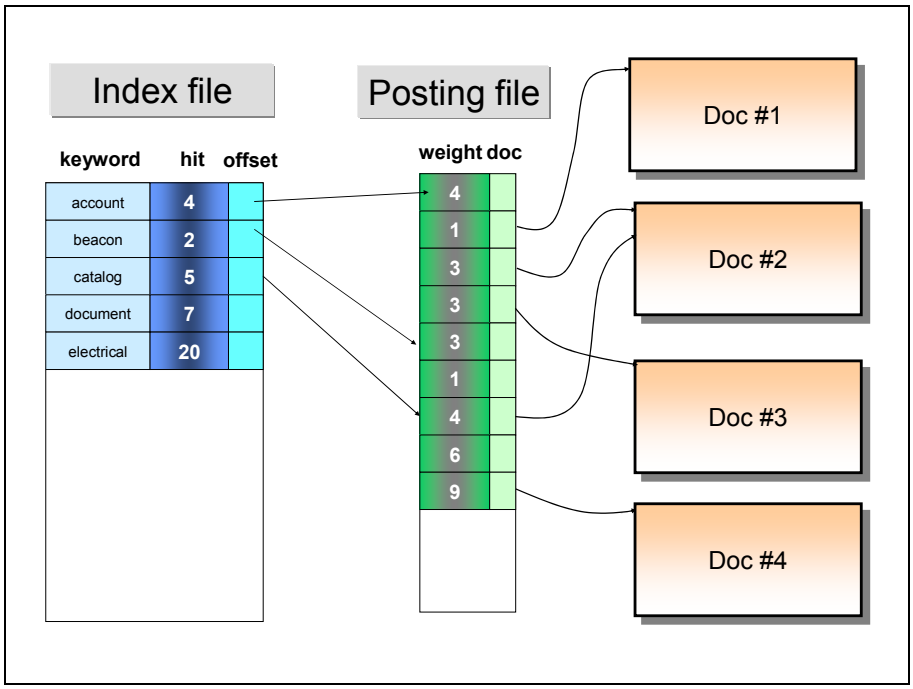
ภาพที่ 3 ตัวอย่าง Configuration ของโปรแกรมโรบอต

การทำงานของโปรแกรมอินเด็กเซอร์

การทำงานของโปรแกรมอินเด็กเซอร์นั้นมีวิธีในการสร้างดัชนีค้นหาอยู่หลายวิธี เช่น แบบอินเวิร์ตอินเด็กซ์ (Fox และLee, 1991) ซิกเนเจอร์ไฟล์ (Faloutsos และ Christodoulakis, 1984) เวกเตอร์สเปซโมเดล (vector space model) (Salton และคณะ, 1974) เป็นต้น ในวิทยานิพนธ์จะใช้เทคนิคการสร้างดัชนีค้นหาแบบอินเวิร์ตอินเด็กซ์ ก่อนที่จะอธิบายถึงวิธีการจัดเก็บข้อมูลของอินเวิร์ตอินเด็กซ์ จะขอยกตัวอย่างเพื่ออธิบายหลักการทำงาน หากเรามีหนังสือเล่มหนึ่งและต้องการค้นหาคำว่า “network” ในหนังสือเล่มนั้น เราย่อมไม่ใช้วิธีเปิดทีละหน้าอ่านทีละบรรทัดเพื่อค้นว่ามีคำว่า ″network″ อยู่ที่ใด หากแต่เรามักใช้วิธีเปิดไปที่ท้ายเล่มเพื่อดูดัชนีคำศัพท์ของหนังสือเล่มนั้นว่ามีคำว่า ″network″ อยู่ที่หน้าใด

รูปแบบของดัชนีค้นหาของอินเวิร์ตอินเด็กซ์ก็อาศัยหลักการเดียวกันกับดัชนีท้ายเล่มของหนังสือ อินเวิร์ตอินเด็กซ์ก็คือตารางของคำศัพท์ที่ถูกสร้างขึ้นมาไว้สำหรับค้นหา โดยที่ภายในของอินเวิร์ตอินเด็กซ์นั้นจัดเก็บคำศัพท์และตำแหน่งที่อยู่ของคำศัพท์เหล่านั้นว่าปรากฏอยู่ที่ใด เมื่อต้องการค้นหาก็ไม่จำเป็นต้องค้นหาคำศัพท์จากเอกสารทีละเอกสาร แต่สามารถตรวจดูที่อินเวิร์ตอินเด็กซ์ได้ทันทีว่าคำศัพท์คำที่ต้องการค้นหาอยู่ที่ใด

โปรแกรมอินเด็กเซอร์มีหน้าที่สร้างตารางอินเวิร์ตอินเด็กซ์ ข้อมูลในอินเวิร์ตอินเด็กซ์ นั้นนอกจากจะเก็บตำแหน่งที่อยู่ของคำศัพท์แล้วจะมีค่าสำคัญอื่น ๆอีกหลายค่าด้วยกัน ซึ่งค่าหลายนี้มีไว้สำหรับช่วยให้โปรแกรมเสิร์จเอ็นจินสามารถจัดเรียงลำดับของผลลัพธ์ได้ดียิ่งขึ้น อาทิ ค่าน้ำหนักที่ใช้ในการจัดเรียงลำดับ เป็นต้น



ภาพที่ 4 ตัวอย่างอินเวิร์ตอินเด็กซ์

จากตัวอย่างอินเวิร์ตอินเด็กซ์ในภาพที่ 4 อธิบายได้ว่า คำศัพท์ “account” นั้นมีปรากฏอยู่ในเอกสารทั้งหมดจำนวน 4 เอกสาร โดยที่เอกสารทั้ง 4 นี้มีค่าน้ำหนักที่ใช้ในการจัดเรียงลำดับ 4, 1, 3 และ 3 ตามลำดับ ค่าน้ำหนักที่ใช้ในการจัดเรียงลำดับนี้ เสิร์จเอ็นจินแต่ละแห่งจะมีวิธีการในการคำนวณที่แตกต่างกันออกไป แต่ส่วนใหญ่จะคำนวณมาจากค่าพารามิเตอร์ เช่น ค่าความถี่ของคำศัพท์ในเอกสาร ค่าความถี่ของคำศัพท์ในชุดของเอกสารทั้งหมด จำนวนของคำศัพท์ในเอกสาร เป็นต้น ตัวอย่างสูตรที่ใช้คำนวณเช่น

$$similarity_{ij} = \frac{\log_2(freq_{ij} + 1) * IDF_i}{\log_2 length_j} \qquad \text{(Harman,1986)}$$

$$IDF_i = \log 2 \frac{N - n_i}{n_i} \qquad \text{(Croft, W.B และ Ruggles, 1979)}$$

$similarity_{ij}$	ค่า similarity ของคำศัพท์ที่ i สำหรับเอกสาร j
$freq_{ij}$	ความถี่ของคำศัพท์ที่ i ในเอกสารที่ j
$length_j$	จำนวนคำศัพท์ที่ไม่ซ้ำกันในเอกสาร j
N	จำนวนเอกสารทั้งหมดในดัชนีค้นหา

n_i จำนวนครั้งของการปรากฏ ของคำศัพท์ที่ i

IDF_i ค่า IDF ของคำศัพท์ที่ i

การทำงานของโปรแกรมค้นหา

โปรแกรมค้นหามีหน้าที่รับคำศัพท์ที่ต้องการค้นหาจากผู้ใช้ มาค้นหาในดัชนีค้นหา และจัดเรียงลำดับผลลัพธ์ให้ได้ตรงกับความต้องการของผู้ใช้ให้มากที่สุด โดยอาศัยข้อมูลที่ได้เตรียมไว้ก่อนแล้วจากโปรแกรมอินเด็กเซอร์

การค้นหาแต่ละครั้งนั้น โปรแกรมค้นหาจะนำคำแต่ละคำในชุดวลีค้นหา (query string) ที่ผู้ใช้ป้อนเข้ามา เพื่อเปรียบเทียบกับคำศัพท์ในตารางอินเวิร์ตอินเด็กซ์ เพื่อหาเอกสารที่มีคำศัพท์ตรงกัน หลังจากที่ได้รายการเอกสารที่ตรงกันกับคำศัพท์แต่ละคำแล้ว โปรแกรมค้นหาจะนำรายการเอกสารมาจัดเรียงลำดับของเอกสาร โดยเรียงลำดับจากเอกสารที่มีค่าน้ำหนักในการค้นหามากที่สุดไปน้อยที่สุด หลังจากนั้นโปรแกรมค้นหาจะส่งคืนรายการเอกสารที่ได้มีการจัดเรียงลำดับเรียบร้อยกลับแล้วกลับไปแสดงผล

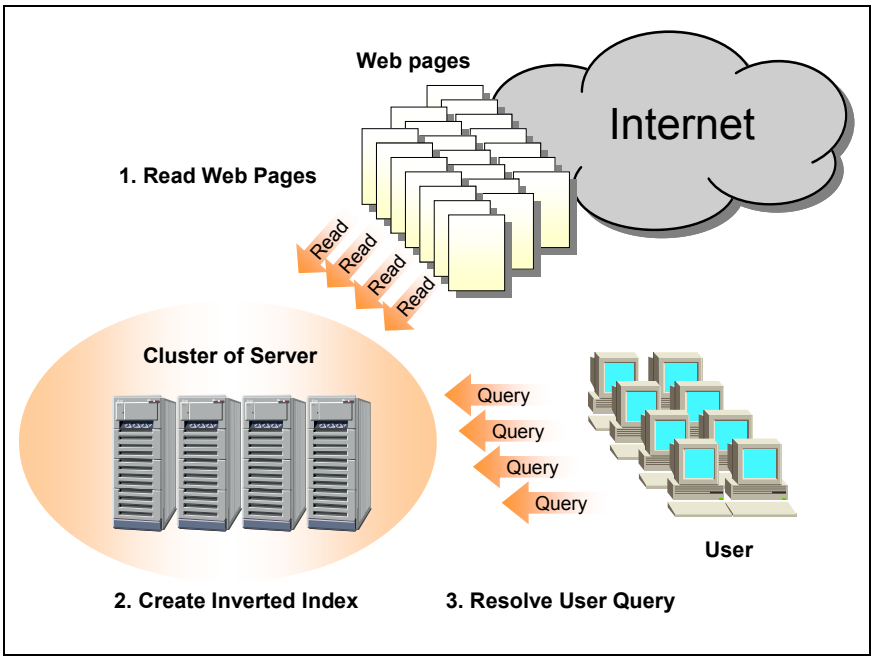
อุปกรณ์และวิธีการ

อุปกรณ์

1. คอมพิวเตอร์พีซี Pentium III 800 เมกะเฮิร์ซ , หน่วยความจำหลัก 512 เมกะไบต์, ฮาร์ดดิสก์ SCSI ขนาด 10 GB จำนวน 4 เครื่อง
2. อีเทอร์เน็ตสวิทซ์
3. ระบบปฏิบัติการ Linux (Redhat 7.1) และโปรแกรมสนับสนุนระบบปฏิบัติการ
4. คอมไพเลอร์ภาษาซีพลัซพลัซ (gnu c++) ที่ทำงานบนระบบปฏิบัติการ Linux

วิธีการ

การออกแบบเสิร์จเอ็นจินแบบกระจาย (Distributed Search Engine)



ภาพที่ 5 แสดงการทำงานของเสิร์จเอ็นจินแบบกระจาย

เสิร์จเอ็นจินที่ออกแบบมีการทำงานแบบกระจาย เพื่อที่จะให้ระบบสามารถรองรับเอกสารจำนวนมากและสามารถค้นหาได้อย่างรวดเร็ว การออกแบบนั้นจะเน้นให้เครื่องแต่ละเครื่องมีการ

ทำงานที่เหมือนกัน แบ่งภาระงานกันในปริมาณที่สมดุลกัน โดยผู้ใช้ไม่จำเป็นต้องทราบถึงโครงสร้างภายใน กล่าวคือสามารถเรียกใช้บริการได้เสมือนว่าเป็นเสิร์จเอ็นจินที่เป็นเครื่อง ๆเดียว ดังภาพที่ 5

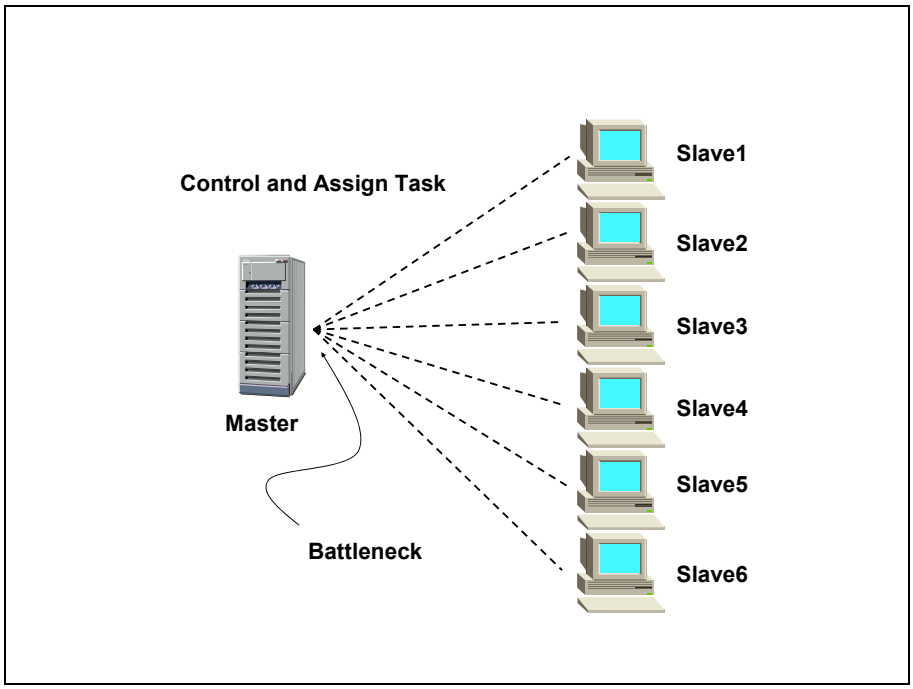
<u>การออกแบบโรบอตแบบกระจาย (Distributed Robots)</u>

การออกแบบโรบอตแบบกระจายที่สามารถทำงานได้บนคอมพิวเตอร์หลายเครื่องพร้อมกันจะมีหลักการทำงานเช่นเดียวกับโรบอตที่ทำงานบนคอมพิวเตอร์เครื่องเดียว แต่มีจุดที่ต้องออกแบบเพิ่มเติมคือ การแบ่งงานให้กับคอมพิวเตอร์แต่ละเครื่อง นอกจากนี้ยังต้องหาวิธีจัดการไม่ให้คอมพิวเตอร์แต่ละเครื่องอ่านเว็บเพจซ้ำซ้อนกัน

ในขั้นแรกของการออกแบบ ได้ออกแบบโครงสร้างการทำงานของระบบเป็นแบบมาสเตอร์สเลฟ (Master−Slave) คือ มีคอมพิวเตอร์เครื่องหนึ่งเป็นเครื่องมาสเตอร์ ทำหน้าที่คอยเก็บรวบรวม URL ทั้งหมดที่คอมพิวเตอร์ทุกตัวในระบบเคยพบ พร้อมทั้งเก็บสถานะของ URL แต่ละอันว่า URL ใดได้อ่านมาแล้ว นอกจากนี้เครื่องมาสเตอร์ยังทำหน้าที่แจกจ่าย URL ที่ยังไม่ได้อ่านไปยังคอมพิวเตอร์ที่เป็นเครื่องสเลฟเพื่อให้เครื่องที่เป็นสเลฟเก็บรวบรวมเว็บเพจจาก URL มาส่งคืนให้เครื่องมาสเตอร์

การทำงานของคอมพิวเตอร์ที่เป็นสเลฟ จะเริ่มต้นการทำงานโดยร้องขอ URL จากเครื่องมาสเตอร์ เมื่อได้รับ URL มาเรียบร้อยแล้วสเลฟจะอ่าน URL นั้นจากเว็บเซิร์ฟเวอร์มาบันทึกไว้ในฮาร์ดดิสก์ หลังจากนั้นจะตรวจสอบ URL ที่อ่านมาได้ว่ามีลิงค์ชี้ไปยัง URL อื่นใดบ้าง หากพบว่ามีลิงค์ชี้ไปยัง URL ใหม่ซึ่งยังไม่เคยพบมาก่อน เครื่องสเลฟจะส่ง URL ใหม่นั้นไปให้เครื่องมาสเตอร์ตรวจสอบอีกครั้ง

ปัญหาหลักที่พบในโรบอตที่มีการทำงานแบบมาสเตอร์สเลฟคือการกระจายภาระงาน ทั้งนี้เนื่องจากการทำงานของโรบอตส่วนใหญ่จะกระจุกตัวอยู่ที่เครื่องมาสเตอร์มากเกินไป หากพิจารณาไปถึงขั้นตอนการทำดัชนีค้นหาและขั้นตอนการค้นหาด้วยแล้วจะเห็นว่า การทำงานทุกอย่างจะต้องมีการสอบถามเครื่องมาสเตอร์ว่า URL ใด ๆนั้นอยู่ที่เครื่องสเลฟเครื่องไหนอยู่ตลอดเวลา ดังนั้นหากมีการเพิ่มเครื่องสเลฟเข้าไปในระบบมากขึ้นจะเป็นภาระทำให้เครื่องมาสเตอร์ทำงานหนักเพิ่มขึ้น และจะไม่สามารถทำงานได้ทันในที่สุด เป็นผลทำให้ประสิทธิภาพโดยรวมลดลง ดังภาพที่ 6



ภาพที่ 6 ปัญหาที่เกิดจากการออกแบบโรบอตแบบ master-slave

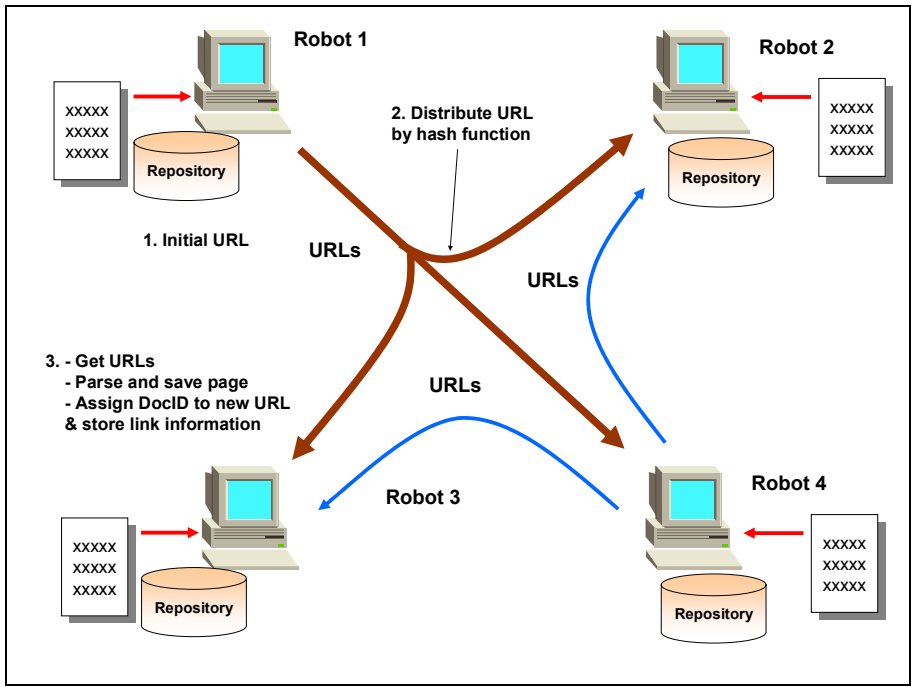
จากปัญหาที่ได้กล่าวมาข้างต้น จึงได้มีการออกแบบการทำงานของโรบอตแบบกระจายขึ้นใหม่ โดยพยายามจะหาวิธีการกระจายงานและตรวจสอบ URL ที่ทำให้เครื่องแต่ละเครื่องสามารถทำงานเป็นอิสระต่อกัน ไม่มีเครื่องหนึ่งเครื่องใดรับภาระมากเป็นพิเศษ เทคนิคที่เลือกใช้ในระบบนี้คือการนำแฮชชิ่งฟังก์ชัน (Hashing function) มาใช้กระจายภาระงาน

วิธีการกระจายงานโดยอาศัยแฮชชิ่งฟังก์ชันอาศัยข้อกำหนดที่ว่าคอมพิวเตอร์แต่ละเครื่องในระบบจะเก็บข้อมูลและรวบรวมเฉพาะ URL ที่อยู่ในความดูแลของตนเองเท่านั้น โดยเครื่องแต่ละเครื่องมีวิธีการตรวจสอบว่า URL นั้นอยู่ในความรับผิดชอบของคอมพิวเตอร์เครื่องใดโดยอาศัยแฮชชิ่งฟังก์ชัน

การคำนวนหาค่าแฮชทำได้โดยนำ URL ที่ต้องการตรวจสอบมาเข้าแฮชชิ่งฟังก์ชันแล้วหารเอาเศษด้วยจำนวนเครื่องทั้งหมดในระบบ ผลลัพธ์คือเศษของตัวเลขซึ่งจะมีค่า 0 ถึง n-1 เมื่อ n คือจำนวนเครื่องในคลัสเตอร์

ตัวอย่างเช่น หากมีคอมพิวเตอร์ในระบบทั้งหมด 4 เครื่อง ดังภาพที่ 7 คือมีเครื่อง Robot1, Robot2, Robot3 และ Robot4 เริ่มต้นเครื่อง Robot1 จะอ่าน URL เริ่มต้นจากไฟล์

แล้วจะแจกจ่าย URL เหล่านั้นไปยังเครื่องอื่นๆ โดยนำ URL แต่ละอันมาเข้าแฮชชิ่งฟังก์ชัน แล้วหารด้วย 4 ซึ่งเป็นจำนวนเครื่องทั้งหมด หากหารแล้วได้เศษ 1 แสดงว่า URL นั้นเป็นของเครื่อง Robot1 หากได้เศษ 2 แสดงว่า URL นั้นเป็นของเครื่อง Robot2 หากได้เศษ 3 แสดงว่า URL เป็นของเครื่อง Robot3และหากไม่มีเศษแสดงว่า URL นั้นเป็นของเครื่อง Robot4 เป็นต้น



ภาพที่ 7 การทำงานของโรบอตแบบกระจาย

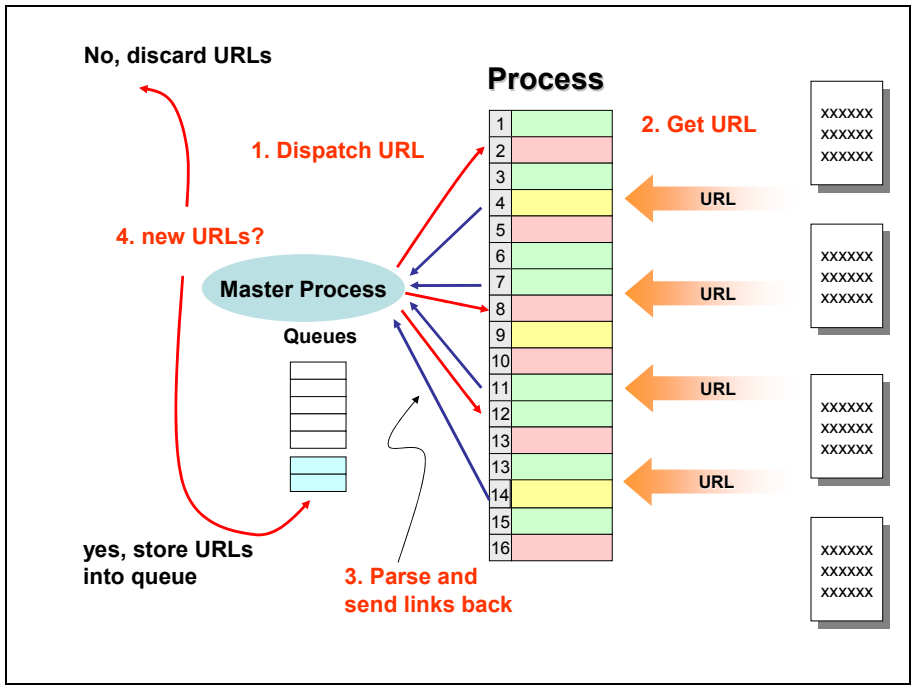
การทำงานของโรบอตที่ใช้วิธีการกระจายงานโดยอาศัยแฮชชิ่งฟังก์ชัน จะมีการทำงานเหมือนกับโรบอตธรรมดาแต่จะแตกต่างกันตรงที่ หากโรบอตพบ URL ใหม่ซึ่งเมื่อนำมาเข้าแฮชชิ่งฟังก์ชันแล้ว ปรากฏว่า URL นั้นไม่ได้เป็น URL ที่อยู่ในความรับผิดชอบของมัน โรบอตจะส่ง URL นั้นไปยังคอมพิวเตอร์ที่รับผิดชอบเพื่อให้คอมพิวเตอร์เครื่องนั้นตรวจสอบและอ่าน URL นั้นมา แต่หากเข้าแฮชชิ่งฟังก์ชันแล้วพบว่า URL นั้นอยู่ในความรับผิดชอบของตัวมันเอง โรบอตจะเก็บ URL นั้นไว้ในคิวรอไว้สำหรับอ่านต่อไป

จะสังเกตได้ว่าวิธีการกระจายงานโดยอาศัยแฮชชิ่งฟังก์ชัน จะไม่มีคอมพิวเตอร์กลางซึ่งทำหน้าที่ดูแลคอยตรวจสอบว่า URL ใดถูกเก็บอยู่ที่คอมพิวเตอร์เครื่องใด โรบอตแต่ละเครื่องจะสามารถตัดสินใจได้ด้วยตัวเองว่า URL ใดๆอยู่ในความรับผิดชอบของตัวเองหรือไม่ ด้วยวิธีการนี้เองเราสามารถเพิ่มจำนวนของคอมพิวเตอร์ในระบบได้มากขึ้น โดยไม่มีคอมพิวเตอร์เครื่องใดได้

รับภาระงานหนักเป็นพิเศษและไม่เกิดปัญหาคอขวด ถ้าหากเราสามารถหาแฮชชิ่งฟังก์ชันที่สามารถกระจาย URL ให้กับโรบอตแต่ละตัวได้เท่า ๆกัน

จากการทำงานจะเห็นได้ว่าข้อมูลที่ถูกส่งผ่านในระบบเครือข่ายประกอบด้วย ข้อมูลไฟล์เว็บเพจที่โรบอตอ่านเข้ามาและข้อมูล URL ซึ่งถูกส่งไปมาระหว่างโรบอตเพื่อตรวจสอบ ข้อมูลที่ส่งไปมาเพื่อตรวจสอบเหล่านี้มีปริมาณมาก เราสามารถที่จะลดปริมาณข้อมูลเหล่านี้ได้โดยเปลี่ยนวิธีการจัดเก็บ URL ในตารางแฮช จากเดิมที่โรบอตจะเก็บเฉพาะ URL ที่อยู่ในความรับผิดชอบของมันเองเท่านั้น เปลี่ยนเป็น ให้โรบอตจัดเก็บ URL ทั้งหมดที่พบไม่ว่า URL นั้นจะอยู่ในความรับผิดชอบของมันหรือไม่

ในขั้นตอนการตรวจสอบไฮเปอร์ลิงค์ แทนที่โรบอตจะส่ง URL ที่ไม่ได้อยู่ในความรับผิดชอบไปให้เครื่องที่รับผิดชอบเลย โรบอตจะตรวจสอบ URL นั้นกับตารางแฮชของตัวเองเสียก่อนเพื่อดูว่าเคยพบมาก่อนหรือไม่ หากโรบอตพบว่าเคยพบ URL นี้มาก่อนแสดงว่าเคยส่ง URL นี้ไปยังเครื่องที่รับผิดชอบแล้ว จึงไม่จำเป็นต้องส่งซ้ำอีก แต่หากไม่เคยพบมาก่อนจึงจะส่ง URL ไปให้เครื่องที่รับผิดชอบ วิธีนี้จะช่วยลดไม่ให้มีการส่ง URL ที่เคยเจอมาแล้วซ้ำหลาย ๆครั้ง



ภาพที่ 8 การติดต่อระหว่างมาสเตอร์และสเลฟโปรเซส

หลังจากที่ได้ออกแบบวิธีการแบ่งการทำงานของโรบอตในแต่ละเครื่องแล้ว ต่อไปจะพิจารณาถึงการทำงานภายในของโรบอตที่แต่ละเครื่อง เวลาส่วนใหญ่ที่โปรแกรมโรบอตใช้ไปนั้นจะอยู่ในขั้นตอนของการอ่านเว็บเพจจากเว็บเซิร์ฟเวอร์ หากเว็บเซิร์ฟเวอร์นั้นอยู่ห่างไกลและปริมาณข้อมูลในเครือข่ายมีความหนาแน่นสูงการทำงานในส่วนนี้จะยิ่งเสียเวลานานมากยิ่งขึ้น

เพื่อเพิ่มสมรรถนะในการอ่านเว็บเพจจึงออกแบบการทำงานของโรบอตเป็นแบบมัลติโปรเซส (Multi-process) โดยเพิ่มจำนวนโปรเซสที่ใช้อ่านเว็บเพจให้มากขึ้น เพื่อที่จะสามารถอ่านเว็บเพจจากเว็บเซิร์ฟเวอร์หลายเครื่องได้พร้อม ๆกัน ส่งผลให้สามารถใช้เครือข่ายได้อย่างมีประสิทธิภาพมากยิ่งขึ้น และลดเวลาเฉลี่ยในการอ่านเว็บเพจแต่ละเว็บเพจลง

การทำงานของมัลติโปรเซสโรบอตจะมีการทำงานแบบมาสเตอร์-สเลฟ โดยระบบจะประกอบด้วยโปรเซส 2 ชนิดคือ สเก็ตดูเลอร์ และ คอลเล็คเตอร์

หน้าที่ของสเก็ตดูเลอร์

1. ควบคุมและแจกจ่ายงานให้คอลเล็คเตอร์
2. ตรวจสอบ URL ที่ถูกส่งต่อมาจากคอลเล็คเตอร์โปรเซสว่า เป็น URL ใหม่หรือไม่
3. ตรวจสอบ URL ใหม่ว่า URL นั้นอยู่ในความรับผิดชอบของ คอมพิวเตอร์เครื่องใด แล้วส่ง URL นั้นไปยังคอมพิวเตอร์ที่รับผิดชอบ

หน้าที่ของคอลเล็คเตอร์

1. รับ URL จากสเก็ตดูเลอร์โปรเซสแล้วอ่าน URL นั้นจากเว็บเซิร์ฟเวอร์มา
2. หาไฮเปอร์ลิงค์ใน URL ที่อ่านมาได้ และส่งลิงค์ที่ได้กลับไปยังสเก็ตดูเลอร์โปรเซส

```
Scheduler ()
begin
    init $URLlist  //Initial URLs
    init $HashTable //URLs has table
    loop until empty ($URLlist) and (every collector idle)
    begin
        //Dispatch a URL to a collector
        loop until empty ($URLlist)
        begin
            $URL = getlist ($URLlist);
            $Collector = Find ($ProcessTable, IDLE);
            sendMessage ($Collector, $URL);
        end loop
        // Receive URLlist from collectors
        loop until no message arrival
        begin
            $mesg=receiveMessage ($Collector);
            if ($mesg == IDLE)
```

```
                set ($ProcessTable, Collector, BUSY)
            else
                if ($mesg == HYPERLINK)
               begin
                    $URL = $mesg
                    if not ($URL in $HashTable)
                    begin
                        $resMachine=hashFucntion($URL)%TotalMachine
                        if ( $resMachine == MachineID)
                        begin
                            putlist ($URLlist, $URL)
                            add ($HashTable, $URL)
                        else
                            sendMessage($resMachine,$URL)
                        end if
                    end if
               end if
            end if
        end loop
    end loop
end
```

ภาพที่ 9 รหัสคำสั่งของสเก็ตดูเลอร์

```
Collector ()
Begin
    sendMessage (Scheduler, IDLE)
    loop until receive terminated signal
    Begin
        $URL = receiveMessage (Scheduler)
        HTTPget ($URL)
        $URLQueue = filter(parser(URLLink ($URL))
        loop until empty ($URLQueue)
        begin
            getlist ($URL)
            SendMessage (Scheduler, $URL)
        End loop
        sendMessage (scheduler, IDLE)
    end loop
End
```

ภาพที่ 10 รหัสคำสั่งของคอลเล็คเตอร์

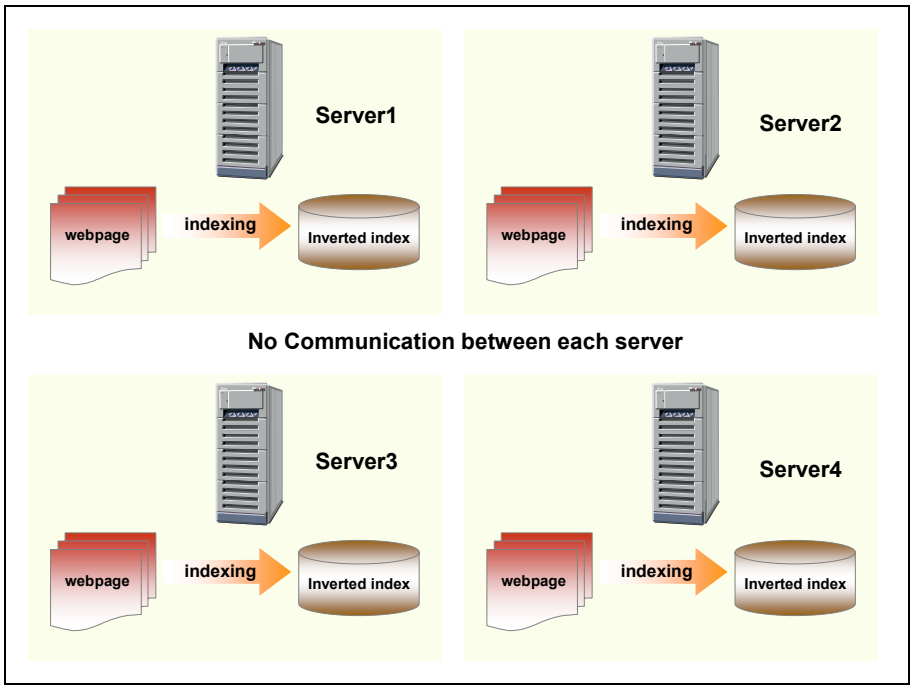
แม้ว่าการออกแบบโรบอตให้เป็นแบบมัลติโปรเซสจะมีข้อดีคือ ช่วยให้อ่านเว็บเพจได้ปริมาณมากในเวลาที่รวดเร็ว แต่มีข้อควรระวังคือ จะต้องป้องกันไม่ให้โรบอตหลาย ๆโปรเซสพยายามอ่านเว็บเพจจากเว็บไซต์เดียวกันมากเกินไป หากมีปริมาณการร้องขอเว็บเพจมากอาจเป็นผลทำให้เว็บไซต์ได้รับภาระงานหนักจนส่งผลกระทบต่อประสิทธิภาพของการให้บริการ

เว็บเพจบนโดยส่วนใหญ่จะมีคุณสมบัติคล้ายกัน (Sanguanpong และคณะ ,2000 B) คือ มีลิงค์ชี้ไปหาเว็บเพจที่อยู่บนเว็บไซด์เดียวกัน ด้วยคุณสมบัตดังกล่าวเป็นผลทำให้ลำดับของ URL ในคิวมี URL ที่เป็นของเว็บไซด์เดียวกันอยู่ติดกัน และเป็นไปได้ที่โรบอตจะอ่าน URL จากคิว แล้วร้องขอไปยังเว็บไซตเดียวกันเป็นจำนวนมาก

เราสามารถแก้ไขปัญหาไม่ให้โรบอตหลาย ๆโปรเซสติดต่อเว็บเพจจากเว็บไซด์เดียวในอัตราที่มากเกินไปได้โดยเปลี่ยนวิธีการเพิ่ม URL เข้าไปในคิวใหม่ จากเดิมที่เมื่อพบ URL ใหม่ จะเพิ่ม URL ใหม่นั้นเข้าไปอยู่ที่ท้ายคิวทั้นที เปลี่ยนเป็นเพิ่ม URL เข้าไปแบบไม่เรียงลำดับ กล่าวคือทุกครั้งที่มีการเพิ่ม URL เข้าไปในคิว จะมีการสุ่มตัวเลขขึ้นมาหนึ่งค่า และใช้ตัวเลขนั้นแทนตำแหน่งที่จะเพิ่ม URL เข้าไปในคิว การทำเช่นนี้จะทำให้ URL ที่พบจากเว็บเพจเดียวกันอยู่ในคิวกระจายกันออกไปทั่วทั้งคิวมากขึ้น

<u>การออกโปรแกรมสร้างดัชนีค้นหาแบบกระจาย</u>

โปรแกรมสร้างดัชนีค้นหาแบบกระจายมีหน้าที่สร้างดัชนีค้นหาของเว็บเพจที่โปรแกรมโรบอตรวบรวมมา ดัชนีค้นหาเหล่านี้จะถูกเก็บไว้ใช้สำหรับโปรแกรมค้นหาแบบกระจายต่อไป



<u>ภาพที่ 11</u> การสร้างดัชนีแบบกระจาย

การทำงานของโปรแกรมสร้างดัชนีค้นหาแบบกระจาย ถูกออกแบบให้มีการทำงานที่สอดคล้องกับการทำงานของโปรแกรมโรบอตแบบกระจาย โปรแกรมสร้างดัชนีค้นหาแบบกระจายจะสร้างดัชนีค้นหาจากเว็บเพจที่ถูกจัดเก็บอยู่บนเครื่องที่มันทำงานอยู่เท่านั้น กล่าวคือ โปรแกรมสร้างดัชนีค้นหาทำงานแยกกันบนแต่ละเครื่อง ทำหน้าที่สร้างดัชนีค้นหาบนเครื่องมันเอง ไม่ยุ่งเกี่ยวกัน

ตัวอย่างเช่น เว็บเพจ ก. ถูกบันทึกไว้ที่คอมพิวเตอร์ A. โปรแกรมสร้างดัชนีค้นหาที่ทำงานอยู่ที่เครื่อง A จะทำหน้าที่สร้าง ดัชนีค้นหาของเว็บเพจ ก และจะบันทึกดัชนีค้นหานี้ไว้ที่คอมพิวเตอร์ A.

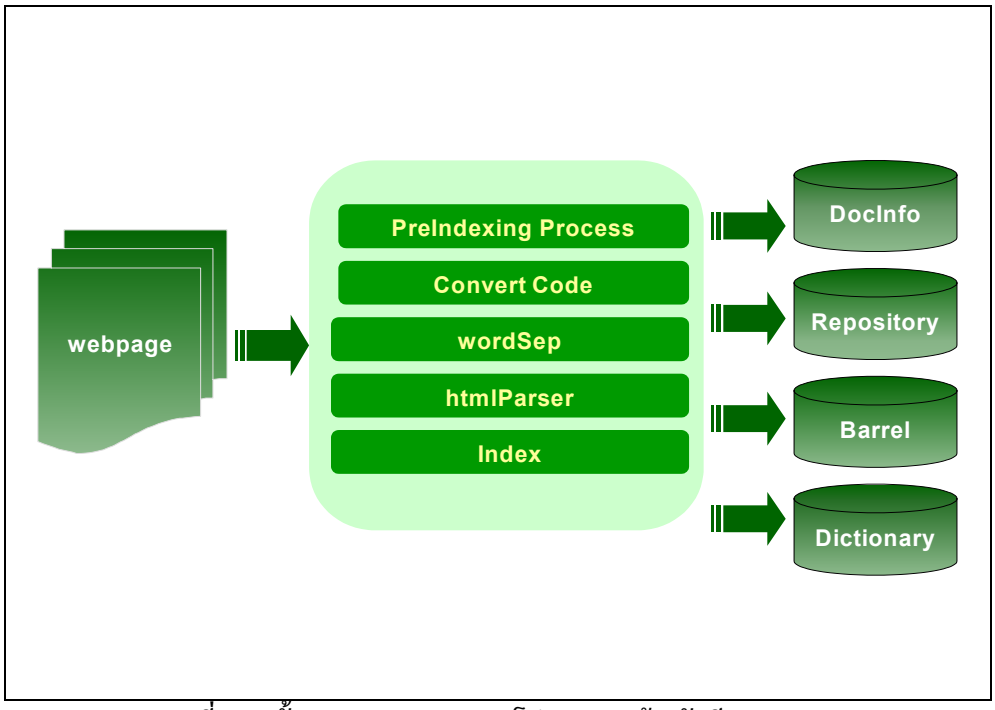
สาเหตุที่ออกแบบให้จัดเก็บดัชนีค้นหาบนเครื่องเครื่องเดียวกับที่เว็บเพจถูกเก็บไว้นั้น เพราะเราจะสามารถลดการติดต่อขอข้อมูลหว่างเครื่องได้ ด้วยเหตุนี้เองในขั้นตอนการทำงานของโปรแกรมสร้างดัชนีแบบกระจายจึงไม่มีการติดต่อสื่อสารกันระหว่างคอมพิวเตอร์ในระบบเลย ดังภาพที่ 11

การทำงานของโปรแกรมสร้างดัชนี สามารถแบ่งเป็นส่วนต่าง ๆได้ดังต่อไปนี้

PreIndexing มีหน้าที่จัดเก็บ URL และเว็บเพจที่ได้มาจากโปรแกรมโรบอต เพิ่มเข้าไปในฐานข้อมูล DocInfo และ Repository (จะกล่าวถึงรายละเอียดของ DocInfo และ Repository ในส่วนต่อไป)

RemoveCode มีหน้าที่เปลี่ยนอักขระพิเศษที่ถูกเข้ารหัสในแบบ International Code Character ไปเป็นอักขระแบบแอสกี (ASCII) เช่น ¡ จะถูกเปลี่ยนเป็นตัวอักษร ก เป็นต้น

WordSep มีหน้าที่ตัดคำในประโยคภาษาไทยออกเป็นคำ เช่น “ฉันไปโรงเรียน” จะถูกตัดเป็น “ฉัน ไป โรงเรียน” เป็นต้น โมดูล WordSep ใช้โปรแกรมที่ชื่อว่า swath (Smart Word Analysis for Thai) (Sornlertlamvanich, 1993) ซึ่งเป็นโปรแกรมสำหรับการตัดคำภาษาไทยที่ได้รับการพัฒนาโดยศูนย์อิเล็คทรอนิคและคอมพิวเตอร์แห่งชาติ (NECTEC)



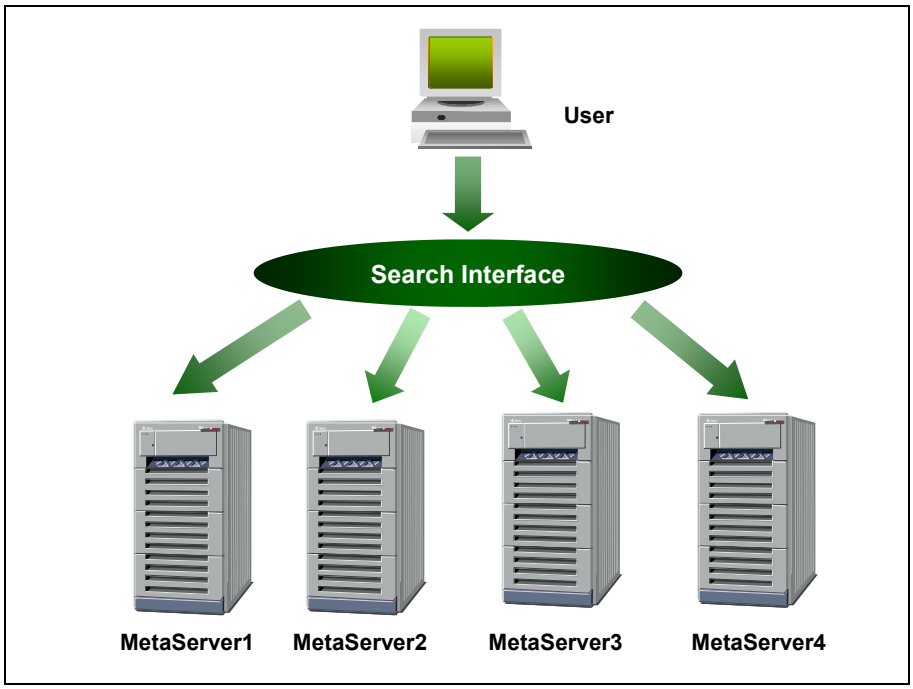
ภาพที่ 12 ขั้นตอนการทำงานของโปรแกรมสร้างดัชนี

HTMLParser มีหน้าที่ตรวจสอบดูว่าคำต่าง ๆในไฟล์ HTML คำไหนอยู่ในแท็ก(Tag) HTMLใด เพื่อที่จะได้กำหนดน้ำหนัก ของการให้คะแนนในขั้นตอนของ Index ต่อไป ตัวอย่างของ HTML ที่มีน้ำหนักพิเศษเช่น <TITLE>, <H1>, <H2>, <H3>, <A>, <B> เป็นต้น

Index มีหน้าที่นำคำศัพท์และน้ำหนักการให้คะแนนของแต่ละคำที่ได้รับจาก HTMLParser มาเพิ่มเข้าไปในฐานข้อมูล Barrel ซึ่งเป็นฐานข้อมูลที่เก็บอินเวิร์ดอินเด็กซ์ซึ่งจะกล่าวถึงโครงสร้างและรายละเอียดในส่วนต่อไป

การออกแบบโปรแกรมค้นหาแบบกระจาย

โปรแกรมค้นหาแบบกระจาย มีหน้าที่ค้นหาเว็บเพจให้กับผู้ใช้โดยอาศัยดัชนีค้นหาที่โปรแกรมสร้างดัชนีแบบกระจายได้สร้างเตรียมไว้ให้ การทำงานจะแบ่งออกเป็นสองส่วนคือ metaServer และ metaSearch



ภาพที่ 13 การทำงานของโปรแกรม metaSearch

metaSearch มีหน้าที่รับคำศัพท์จากผู้ใช้ แล้วส่งคำศัพท์เหล่านั้นไปยัง metaServer ที่ทำงานกระจายเป็นดีมอน (daemon) อยู่บนเครื่องแต่ละเครื่อง หลังจากนั้นจะรวบรวมและจัดเรียงลำดับของผลลัพธ์ที่ได้รับกลับมาจาก metaServer เพื่อหาเว็บเพจที่ตรงกับความต้องการของผู้ใช้มากที่สุด

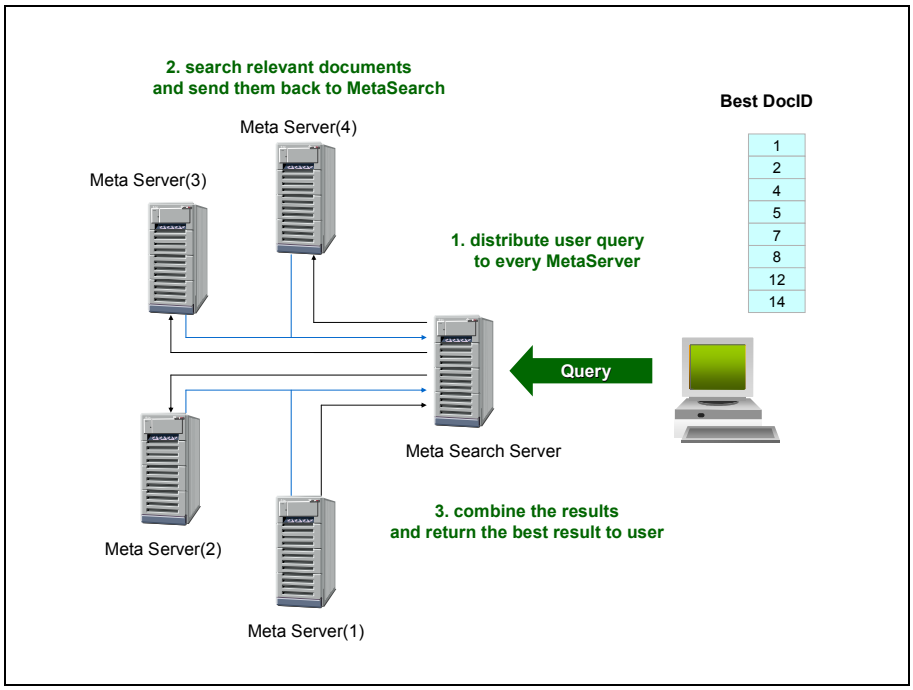
metaServer มีหน้าที่ค้นหาเว็บเพจที่มีคำศัพท์ตามคำสั่งที่ metaServer ส่งมาให้ การค้นหาจะค้นหาจากดัชนีค้นหาที่โปรแกรมสร้างดัชนีค้นหาสร้างเตรียมไว้ก่อนแล้ว เมื่อ metaServer ค้นหาได้ชุดของเอกสารผลลัพธ์แล้ว metaServer จะส่งผลลัพธ์เหล่านั้นพร้อมกับน้ำหนักในการจัดเรียงลำดับกลับไปให้ metaSearch ดังภาพที่ 13

ในการค้นหาแต่ละครั้งผู้ใช้จะพิมพ์คำที่ต้องการค้นหาส่งมายังโปรแกรม MetaSearch จากนั้นโปรแกรม MetaSearch จะกระจายคำสั่งโดยส่งชุดของคำศัพท์ค้นหาไปยังโปรแกรม MetaServer ที่ทำงานเป็นดีมอนอยู่บนเครื่องแต่ละเครื่องในระบบ หลังจากนั้นโปรแกรม MetaServer จะค้นหาหมายเลขเอกสาร (DocID) และค่าน้ำหนักในการค้นหา (RankingWeight) ของเอกสารที่มีคำศัพท์ตรงตามที่ต้องการ แล้วส่งค่าหมายเลขเอกสารและน้ำหนักในการค้นหา

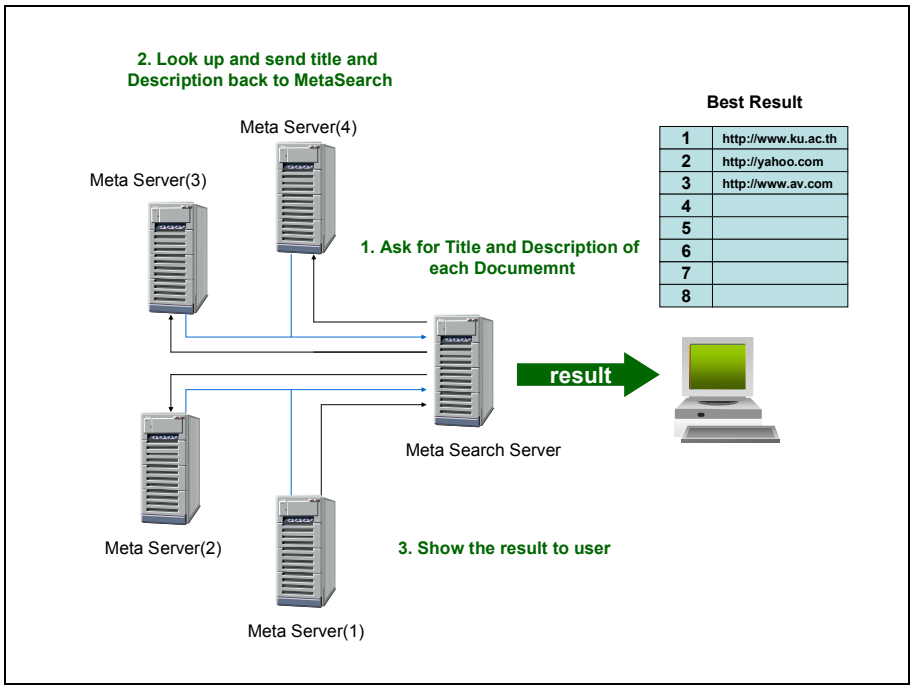
กลับไปยัง โปรแกรม MetaServer เพื่อให้โปรแกรม MetaServer รวบรวมผลลัพธ์จากทุก ๆเครื่องเข้าด้วยกัน ดังภาพที่ 14

เมื่อโปรแกรม MetaSearch ได้รับผลลัพธ์จากทุกเครื่องเรียบร้อยแล้ว โปรแกรม MetaSearch จะจัดเรียงลำดับเอกสารผลลัพท์จากเอกสารที่มีค่าน้ำหนักมากที่สุดไปหาเอกสารที่มีค่าน้ำหนักน้อยที่สุด เมื่อเรียงลำดับเรียบร้อยแล้ว โปรแกรม MetaSearch จะคัดเฉพาะเอกสาร 20 อันดับแรก ออกมาแล้วหา URL และ คำบรรยายประจำเอกสารเหล่านั้น

การหา URL และคำบรรยายประจำเอกสารสามารถหาได้โดย ติดต่อสอบถามไปยัง MetaServer ที่เป็นที่เก็บของเอกสารนั้นอีกครั้ง ดังภาพที่ 15



ภาพที่ 14 ตัวอย่างขั้นตอนการทำงานของโปรแกรมค้นหาแบบกระจาย



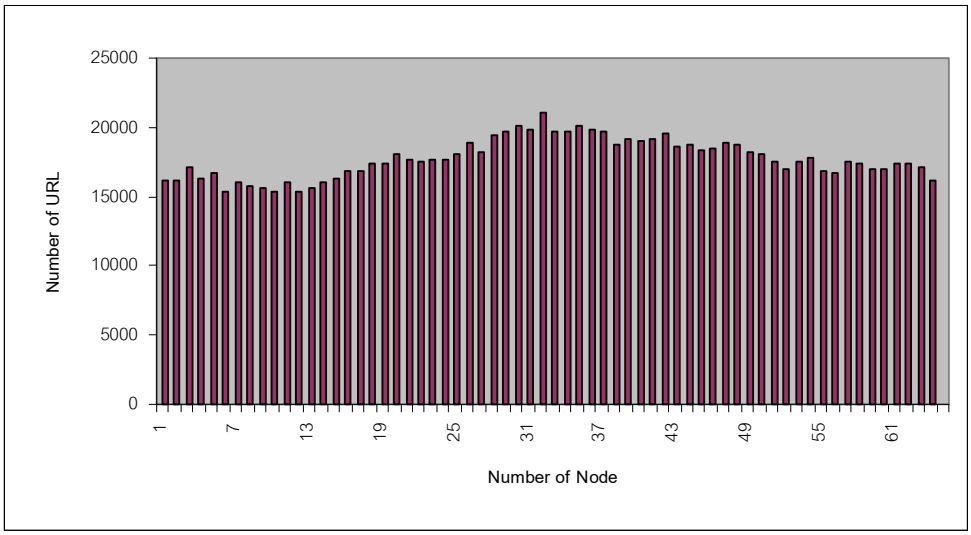
ภาพที่ 15 ตัวอย่างขั้นตอนการทำงานของโปรแกรมค้นหาแบบกระจาย (2)

โมดูลย่อยและโครงสร้างการจัดเก็บข้อมูล

แฮชชิ่งฟังก์ชันสำหรับโรบอต

แฮชชิ่งฟังก์ชันที่จะใช้สำหรับกำหนดว่า URL ใด ๆจะอยู่ในความรับผิดชอบของคอมพิวเตอร์เครื่องไหนนั้นควรมีคุณสมบัติสามารถกระจาย URL ได้ดีโดยที่ผลลัพธ์ของการคำนวณจะต้องไม่กระจุกอยู่ที่คอมพิวเตอร์เครื่องใดมากผิดปกติ นอกจากนี้แฮชชิ่งฟังก์ชันที่ดีควรที่จะคำนวณได้รวดเร็ว

จากการทดลองโดยนำ URL จำนวน 1 ล้าน URL มาทดสอบกับแฮชชิ่งฟังก์ชันต่าง ๆโดยกำหนดให้มีคอมพิวเตอร์ตั้งแต่ 2 ถึง 64 เครื่องพบว่า แฮชชิ่งฟังก์ชันส่วนใหญ่สามารถกระจาย URL ได้ดีเหมือนกัน เราจึงเลือกแฮชชิ่งฟังก์ชันที่สามารถคำนวณได้เร็วที่สุดซึ่ง ซึ่งนั้นก็คือผลบวกของรหัสแอสกีของอักขระแต่ละตัวใน URL ซึ่งจะได้ผลลัพธ์ดังรูปภาพที่ 16



ภาพที่ 16 แสดงการกระจายตัวของ URL เมื่อใช้ผลบวกของรหัสแอสกีเป็นแฮชชิ่งฟังก์ชัน

$$\text{Hash Function(URL)} = \sum_{i=1}^{strlen\ (URL\ )} ascii\ \ (C_i)$$

ตัวอย่าง

Hash Function (www.ku.ac.th) = 119 + 119 + 119 + 46 + 107 + 117 + 46
+ 97 + 99 + 46 +116 + 104 + 47 + 105
+ 110 + 100 + 101 + 120 + 46 + 104
+ 116 + 109 + 108
= 2201

โมดูล DocInfo

โมดูล DocInfo มีหน้าที่จัดเก็บข้อมูลและกำหนดหมายเลขประจำเอกสารให้กับเว็บเพจแต่ละเว็บเพจ ตามที่โรบอตรวบรวมมาได้ การเก็บจะเก็บข้อมูลในตาราง DocIndex_tb ในฐานข้อมูลชื่อ NSE ในระบบฐานข้อมูล PostgresSQL โมดูล DocInfo จะถูกเรียกเพื่อเพิ่มข้อมูลใน

ตารางในขั้นตอนการสร้างดัชนีฐานข้อมูล และจะถูกเรียกใช้เพื่อสอบถาม URL ของเอกสารในขั้นตอนการค้นหา

ตารางที่ 2 ฟิลด์ในตาราง DocIndex_tb

ชื่อฟิลด์	ชนิดข้อมูล	คำอธิบาย
DocID	int	หมายเลขประจำเอกสาร เว็บเพจแต่ละเว็บเพจที่โรบอตรวบรวมมาจะมีหมายเลข ประจำเอกสารที่ซ้ำกัน
Server	char(64)	ชื่อเครื่อง (Hostname)ของเว็บเซิร์ฟเวอร์ของเว็บเพจ
Path	char(128)	ตำแหน่งที่อยู่ของเว็บเพจในเซิร์ฟเวอร์นั้น
LastUpdate	datetime	วันเวลาที่โรบอตอ่านเว็บเพจมาล่าสุด
Info	int	ข้อมูลอื่น ๆ เช่น ขนาดของเว็บเพจ

ตารางที่ 3 ตัวอย่างข้อมูลในตาราง DocIndex_tb

DocID	Server	Path	LastUpdate	Info(byte)
1	www.ku.ac.th	/index.html	12:00 01/12/2001	2034
2	www.ku.ac.th	/intro.html	14:37 31/01/2001	34
3	std.cpe.ku.ac.th	/student/index.html	23:34 15/03/2000	234
4	www.yahoo.com	/test.html	01:36 23/06/2000	346

โมดูล Repository

Repository มีหน้าที่จัดเก็บเว็บเพจในฮาร์ดดิสก์ การบันทึกเว็บเพจลงในดิสก์จะบันทึก 1 เว็บเพจต่อ 1 ไฟล์ การบันทึกนั้นสามารถที่จะเลือกได้ว่าจะบีบอัดไฟล์ด้วยหรือไม่ สาเหตุที่เราเลือกที่จะบันทึกข้อมูล 1 เว็บเพจต่อ 1 ไฟล์ แม้ว่าการบันทึกเว็บเพจหลาย ๆเว็บเพจลงในไฟล์เดียวกันจะสามารถเพิ่มประสิทธิภาพในการบีบอัดข้อมูลให้ดีขึ้น ทั้งนี้เพราะว่าในขั้นตอนของการสร้างดัชนี การที่จะนำเว็บเพจออกจากไฟล์ที่บีบอัดแบบหลายเว็บเพจจะทำได้ช้ากว่าแบบไฟล์เดียว นอกจากนี้หากในอนาคตระบบได้รับการพัฒนาเพิ่มเติมให้สามารถแก้ไขดัชนีค้นหาของเว็บเพจที่มีการเปลี่ยนแปลงได้โดยไม่ต้องสร้างดัชนีใหม่ทั้งหมด

ไฟล์เว็บเพจที่อยู่ในฮาร์ดดิสต์จะถูกเก็บกระจายแยกกันในหลายไดเรกทอรี โดยจะเก็บไดเรกทอรีละ 1000 ไฟล์ดังตารางที่ 4 สาเหตุที่ต้องแยกการจัดเก็บไว้หลายไดเรกทอรีก็เพราะหากมีไฟล์เป็นจำนวนมากอยู่ในไดเรกทอรีเดียว ทำให้การเรียกใช้ไฟล์ในไดเรกทอรีนั้นทำได้ช้า

ตารางที่ 4 วิธีการจัดเก็บไฟล์ใน Repository

ไดเรกทอรี	จำนวนไฟล์	เก็บเอกสารหมายเลข
$(REPDIR)/D1/	1,000	1-1000
$(REPDIR)/D2/	1,000	1001-2000
$(REPDIR)/D3/	1,000	2001-3000

หมายเหตุ $REPDIR แทนไดเรกทอรีที่เก็บข้อมูลของ Repository

โมดูล Dictionary

Dictionary ใช้สำหรับเก็บคำศัพท์ทั้งหมดที่พบในระหว่างขั้นตอนการสร้างดัชนี คำศัพท์ในที่นี้เป็นทั้ง คำศัพท์ที่เป็นภาษาไทย ภาษาอังกฤษและตัวเลข โดยคำศัพท์แต่ละคำจะมีหมายเลขประจำคำศัพท์อยู่หนึ่งหมายเลขที่ไม่ซ้ำกันเรียกว่า WordID หมายเลขนี้จะมีการอ้างอิงถึงในข้อมูลในส่วนของ Barrel ซึ่งจะกล่าวถึงต่อไป

ในขั้นตอนของการทำดัชนีค้นหาจะต้องมีการเรียกใช้ Dictionary อยู่บ่อยครั้งเพื่อถามหา WordID ของคำศัพท์ จึงได้มีการออกแบบ Dictionary ให้มีการจัดเก็บข้อมูลเป็นแบบ Btree โดยใช้ไลบรารี dbopen เพื่อให้สามารถค้นหาคำศัพท์ได้อย่างรวดเร็ว

เนื่องจากตัวเลขจัดเป็นคำศัพท์ประเภทหนึ่ง ในเว็บเพจต่าง ๆจะพบว่ามีตัวเลขอยู่เป็นจำนวนมากเป็นจำนวนมาก เราสามารถลดเวลาในการหา WordID ของตัวเลขลงได้โดย กำหนด WordID ของตัวเลขตายตัวไว้แต่แรก โดยกำหนด WordID ตั้งแต่หมายเลข 1-100,000,000 ให้เป็นWordID ของตัวเลข 1-100,000,000 ตามลำดับ แต่หากตัวเลขที่มีค่าเกิน 100,000,000 หรือ ตัวเลขที่มีทศนิยม ให้มีวิธีการกำหนดค่า WordID เป็นเหมือนคำศัพท์ทั่วไป

ดังนั้นข้อมูลของ Dictionary จะมี WordID ตั้งแต่หมายเลข 1-100,000,000 ตั้งแต่ยังไม่ได้สร้างดัชนีค้นหาของเว็บเพจใดเลย หลังจากในขั้นตอนการสร้างดัชนีค้นหาหากโปรแกรมพบคำศัพท์คำใหม่ที่ไม่ได้อยู่ใน Dictionary โปรแกรมจะเพิ่มคำศัพท์นั้นเข้าไปใน Dictionary โดยกำหนดหมายเลข WordID หมายเลขใหม่เพิ่มเข้าไป ดังตารางที่ 5

ตารางที่ 5 ตัวอย่างข้อมูลใน Dictionary

หมายเลข WordID	คำศัพท์	จำนวนครั้งที่พบ
1	1	0
2	2	0
...	...	0
99,999,999	99,999,999	0
100,000,000	100,000,000	0
100,000,002	Search	1
100,000,003	Engine	1
100,000,004	302,323,342	1

โมดูล Barrel

Barrel มีหน้าที่จัดเก็บดัชนีค้นหาซึ่งเป็นแบบอินเวิร์ตอินเด็กซ์ลงดิสก์เพื่อเป็นข้อมูลเตรียมไว้สำหรับนำไปใช้ในโปรแกรมค้นหาต่อไป ใน Barrel จะจัดเก็บข้อมูลคำศัพท์ทั้งหมดว่ามีคำศัพท์อะไรบ้าง แต่ละคำนั้นปรากฏในเว็บเพจใดและที่ตำแหน่งที่เท่าไหร่ของเว็บเพจ

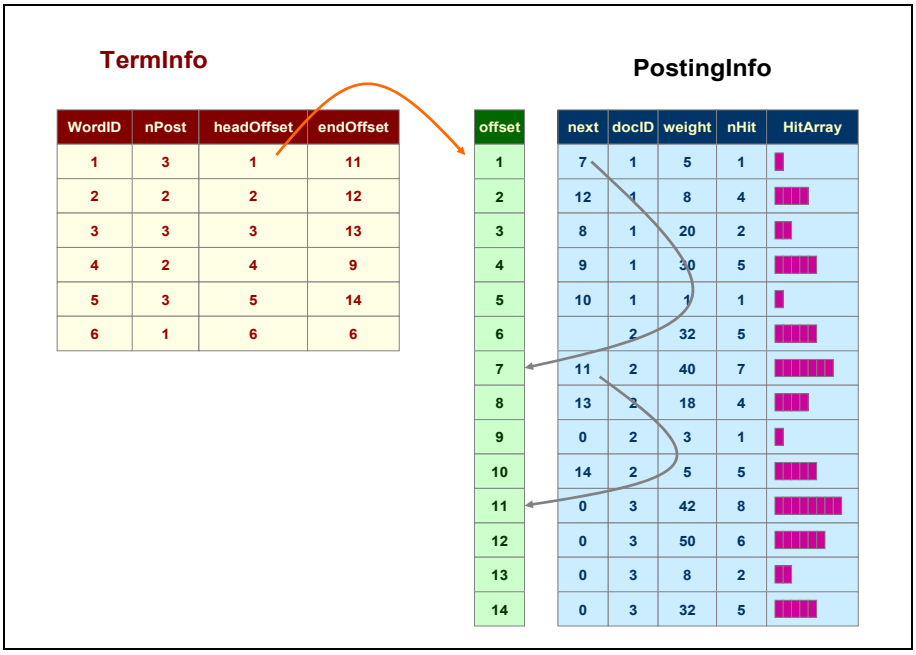
การออกแบบวิธีการจัดเก็บข้อมูลใน Barrel นั้น จะต้องออกแบบให้การค้นหาหมายเลขของเอกสารที่มีคำศัพท์ใด ๆปรากฏสามารถทำได้อย่างรวดเร็ว เราจึงเก็บข้อมูลเหล่านี้ในรูป Btree โดยใช้หมายเลขคำศัพท์ (wordID) เป็นคีย์ (key) ข้อมูลใน Barrel นี้สามารถแบ่งได้เป็น 2 ส่วน คือ TermInfo และ Posting Info ดังภาพที่ 17

ตารางที่ 6 ฟิลด์ต่าง ๆใน TermInfo

ชื่อฟิลด์	ชนิดข้อมูล	ความหมาย
WordID	int	หมายเลขประจำคำศัพท์
Npost	int	จำนวนเอกสารทั้งหมดที่มีคำศัพท์คำนี้ปรากฏ
HeadOffset	int	ตำแหน่งที่อยู่ของเรคอร์ดแรกในลิงค์ลิสต์ ของ PostingInfo
EndOffset	int	ตำแหน่งที่อยู่ของเรคอร์ดสุดท้ายในลิงค์ลิสต์ ของ PostingInfo

ตารางที่ 7 ฟิลด์ต่าง ๆใน PostingInfo

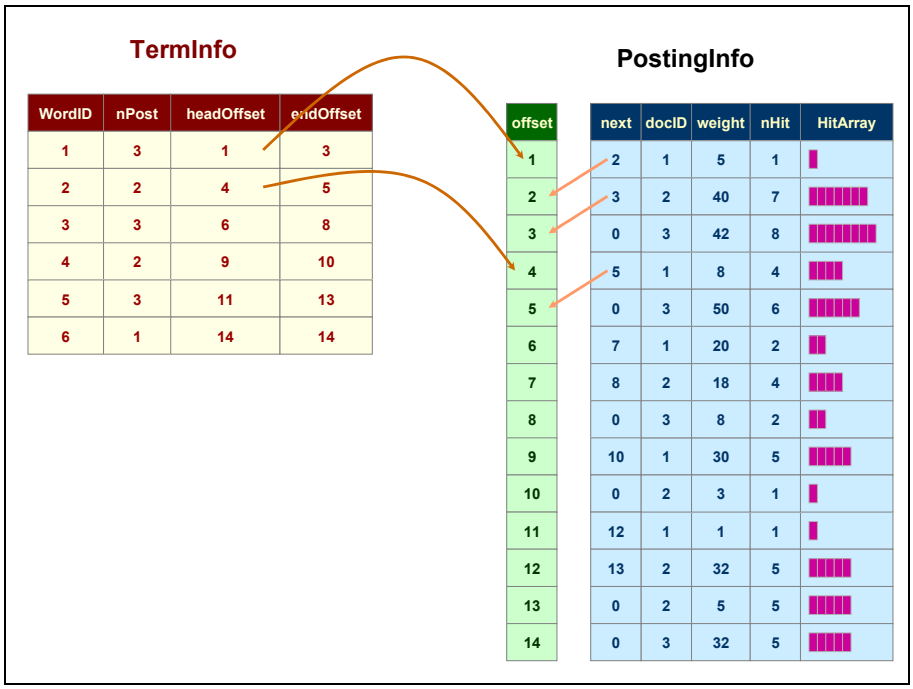
ชื่อฟิลด์	ชนิดข้อมูล	คำอธิบาย
Next	int	ตำแหน่งที่อยู่ของเรคอร์ดอันดับถัดในลิงค์ลิสต์
DocID	int	หมายเลขประจำเอกสาร ที่มีคำศัพท์ปรากฏ
Weight[at]	int	ค่าน้ำหนักในการจัดเรียงพิจารณาตามคุณสมบัติของคำศัพท์ ซึ่งจะกล่าวถึงวิธีคำนวนในหัวข้อต่อไป
Nhit	int	จำนวนครั้งของการปรากฏของคำศัพท์ ในเอกสาร
HitArray	int	ชุดของข้อมูลที่เก็บตำแหน่งการปรากฏของคำศัพท์ในแต่ละเอกสาร



ภาพที่ 17 ตัวอย่างการจัดข้อมูลใน Barrel ก่อนการเรียงลำดับ

ข้อมูลในส่วนของ TermInfo จะเก็บรายละเอียดของคำศัพท์ ว่าคำศัพท์แต่ละคำปรากฏอยู่ในเอกสารทั้งหมดกี่เอกสาร นอกจากนี้ยังเก็บตำแหน่ง Offset ซึ่งเป็นพอยต์เตอร์ชี้ไปยังตำแหน่งหัวและท้ายของลิงค์ลิสต์ของ PostingInfo ดังตารางที่ 6

ข้อมูลในส่วนของ PostingInfo จะเก็บข้อมูลการปรากฏของคำศัพท์ในแต่ละเอกสารว่าคำศัพท์แต่ละคำนั้นปรากฏอยู่ในเอกสารใดบ้างและที่ตำแหน่งใดของเอกสาร โดยเราสามารถแบ่งข้อมูลออกเป็นฟิลด์ได้ตารางที่ 7



ภาพที่ 18 ตัวอย่างการจัดเก็บข้อมูลใน Barrel หลังการเรียงลำดับ

จากตัวอย่างในภาพที่ 17 ข้อมูลเรคอร์ดแรกของ TermInfo หมายถึง คำศัพท์ที่มีหมายเลข WordID เท่ากับ 1 ปรากฏอยู่ในเอกสารทั้งหมด 3 เอกสาร คือเอกสาร ที่มีหมายเลข docID เท่ากับ 1,2 และ 3 ตามลำดับ โดยที่เอกสารที่ 1 มีคำศัพท์คำนี้ปรากฏอยู่ทั้งหมด 1 ครั้ง และมีค่าน้ำหนักในการจัดเรียงลำดับเท่ากับ 5 ในขณะที่ เอกสารที่ 2 มี คำศัพท์คำนี้ปรากฏทั้งหมด 7 ครั้งมีค่าน้ำหนักในการจัดเรียงลำดับเท่ากับ 40 และที่เอกสารที่ 3 มีคำศัพท์คำนี้ปรากฏทั้งหมด 8 ครั้งและมีค่าน้ำหนักในการจัดเรียงลำดับเท่ากับ 42

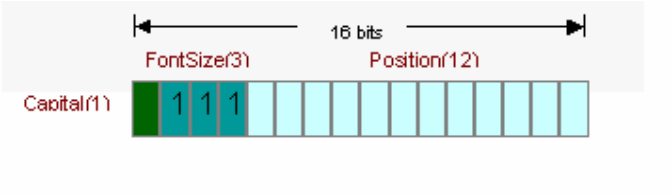
จากภาพที่ 17 จะสังเกตได้ว่าการจัดเก็บข้อมูลในส่วนของ PostingInfo จะมีการจัดเก็บเป็นแบบลิงค์ลิสต์ เนื่องจากไม่สามารถนับจำนวนคำศัพท์และคำนวนค่าล่วงหน้าได้

หลังจากการสร้างดัชนีค้นหาของเอกสารทุกเอกสารเรียบร้อยแล้ว จะจัดเรียงเรคอร์ดใน PostingInfo ใหม่ โดยจัดเรียงให้เรคอร์ดที่มีคำศัพท์เดียวกันให้อยู่ติดกัน เพื่อเพิ่มประสิทธภาพในการสืบค้น ดังภาพที่ 18

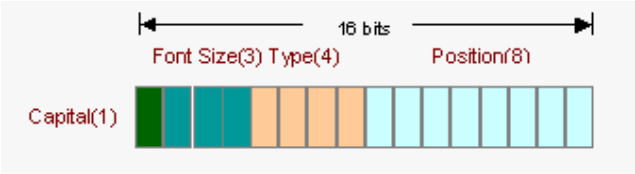
จากภาพจะพบว่าเมื่อจัดเรียงลำดับใน PostingInfo ใหม่แล้ว เรคอร์ดของคำศัพท์คำเดียวกันจะอยู่ติดกัน เมื่อหากมีการค้นหาข้อมูลก็จะสามารถทำได้รวดเร็วขึ้น เพราะหัวอ่านฮาร์ดดิสก์สามารถอ่านข้อมูลหลายเรคอร์ดอย่างต่อเนื่องกัน

โมดูล HIT

Hit เป็นรูปแบบข้อมูลชนิดหนึ่งที่ใช้สำหรับจัดเก็บข้อมูลการปรากฏของคำศัพท์ การปรากฏของคำศัพท์หนึ่งครั้งจะแทนด้วย Hit หนึ่ง Hit ข้อมูล Hit ของคำศัพท์แต่ละคำในแต่ละเอกสารจะอยู่ในฟิลด์ HitArray ในข้อมูลของ PostingInfo ดังภาพที่ 18



ภาพที่ 19 รูปแบบการจัดเก็บข้อมูลของ Plain Hit



ภาพที่ 20 รูปแบบการจัดเก็บข้อมูลของ Fancy Hit

Hit สามารถแบ่งออกเป็น 2 ประเภทคือ Plain Hit และ Fancy Hit โดยที่ Plain Hit ใช้สำหรับเก็บข้อมูลการปรากฏของคำศัพท์ที่เป็นตัวอักษรธรรมดา ในขณะที่ Fancy Hit ใช้สำหรับ

เก็บข้อมูลการปรากฏของคำศัพท์ที่เป็นตัวอักษรแบบพิเศษ ตัวอักษรแบบพิเศษในที่นี้หมายถึงตัวอักษรที่อยู่ใน HTML Tag พิเศษ เช่น Title, Meta, Header เป็นต้น

เนื้อที่ที่ใช้สำหรับเก็บ Plain Hit และ Fancy Hit จะใช้เนื้อที่ 2 ไบต์เท่ากัน แตกแต่งกันตรงที่เฮดเดอร์ใน Hit แต่ละชนิดจะไม่เหมือนกัน โดยใน Plain Hit จะมีเฮดเดอร์ขนาด 4 bits เหลือเนื้อที่ในการเก็บตำแหน่ง 12 bits ดังภาพที่ 19 ในขณะที่ Fancy Hit จะมีเฮดเดอร์ขนาด 8 bits เหลือเนื้อที่ในการเก็บตำแหน่ง 8 bits ดังภาพที่ 20

วิธีแยกความแตกต่างระหว่าง Fancy Hit และ Plain Hit จะสามารถแยกได้โดยดูที่ฟิลด์ Font Size หากมีค่าเป็น 7 (111) จะเป็น Plain Hit แต่หากมีค่าเป็นค่าอื่นที่ไม่ใช่ 7 จะเป็น Fancy Hit

ตารางที่ 8 ความหมายของฟิลด์ต่าง ๆใน Hit

ชื่อฟิลด์	ขนาด	ความหมาย
Capital	1	บอกลักษณะตัวอักษรว่าเป็นตัวอักษรภาษาอังกฤษตัวใหญ่หรือไม่
Font Size	3	เก็บขนาดของตัวหนังสือ
Type	4	เก็บชนิดของ HTML TAG ของคำศัพท์ (ดูรายละเอียดได้ที่ตารางที่ 9)
Position	8	เก็บตำแหน่งการปรากฏของคำศัพท์

ตารางที่ 9 ความหมายของบิตต่าง ๆในฟิลด์ Type ของ Fancy Hit

ตำแหน่งของ bit	รูปแบบ	ความหมาย
1	1000	ปรากฏอยู่ในแท็ก title
2	0100	ปรากฏอยู่ในแท็ก A
3	0010	ปรากฏอยู่ในแท็ก header
4	0001	ปรากฏอยู่ในแท็ก B,I,U

การค้นหาแบบ Boolean

การทำงานของ MetaSearch และ MetaServer สามารถค้นหาคำได้แบบบูลีน (Boolean Search) ได้ การค้นหาแบบบูลีนเป็นการค้นหาโดยมีโอเปอร์เรเตอร์ที่ใช้ได้ดังตารางที่ 10

ก่อนที่โปรแกรม MetaSearch จะส่งคำศัพท์ที่ผู้ใช้ค้นหาไปยังโปรแกรม MetaServer โปรแกรม MetaSearch จะตรวจวลีค้นหา หากตรวจพบว่าเป็นการค้นหาแบบบูลีน โปรแกรม MetaSearch จะจัดเรียงลำดับของคำที่ผู้ใช้ค้นหาใหม่โดยนำคำที่มีเครื่องหมาย & มาไว้ข้างหน้าสุด ตามมาด้วยคำที่มีเครื่องหมาย + และเครื่องหมาย – อยู่ท้ายสุด ทั้งนี้เพื่อความถูกต้องในการค้นหา เช่น "–เชียงใหม่ &กรุงเทพ +ประเทศไทย &สงขลา" จะถูกจัดเรียงใหม่เป็น "&กรุงเทพ &สงขลา +ประเทศไทย –เชียงใหม่"

ตารางที่ 10 เครื่องหมายต่าง ๆในการค้นหาแบบบูลีน

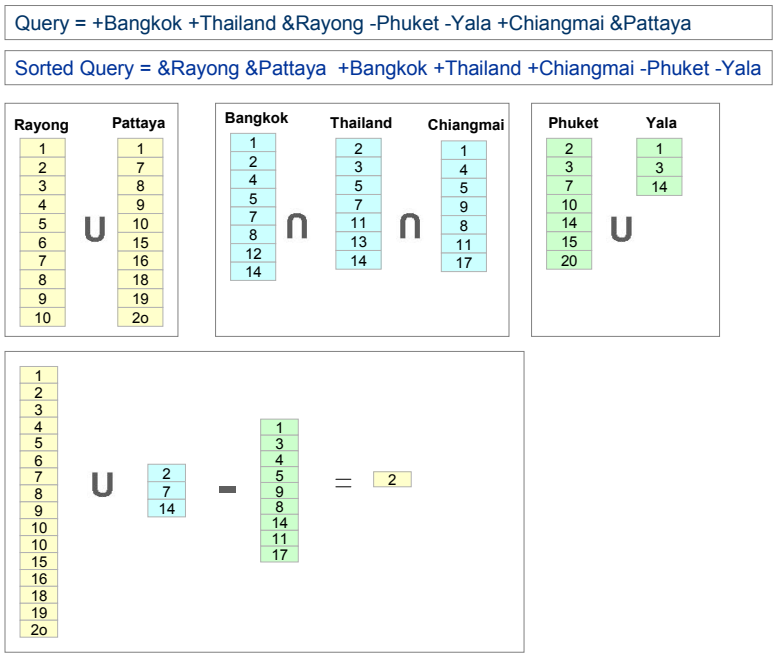
เครื่องหมาย	หน้าที่	ตัวอย่าง
+	ใช้สำหรับระบุว่าเอกสารที่ต้องการนั้นจะต้องมีคำ ๆ นี้ปรากฏอยู่ในเอกสาร	กรุงเทพ +ประเทศไทย ให้ค้นหาเอกสารที่มีทั้งคำว่า กรุงเทพ และประเทศไทย
&	ใช้สำหรับระบุว่าเอกสารที่ต้องการนั้นจะมีหรือไม่มีคำ ๆนี้ก็ได้	กรุงเทพ &ประเทศไทย ให้ค้นหาเอกสารที่มีคำว่า กรุงเทพ หรือประเทศไทย เพียงคำใดคำหนึ่ง หรือทั้งสองคำก็ได้
–	ใช้สำหรับระบุว่าเอกสารที่ต้องการจะต้องไม่มีคำ ๆ นั้นปรากฏอยู่ในเอกสาร	กรุงเทพ –ประเทศไทย ให้ค้นหาเอกสารที่มีคำว่า กรุงเทพ แต่ในเอกสารนั้นต้องไม่มีคำว่าประเทศไทย

หมายเหตุ หากไม่มีการใส่เครื่องหมายใด ๆ หน้าคำศัพท์ โปรแกรมจะใส่เครื่องหมาย & นำหน้าให้โดยอัตโนมัติ

สาเหตุที่จะต้องจัดเรียงลำดับใหม่เนื่องจาก ลำดับของการค้นหามีผลต่อผลลัพธ์ การค้นหาชุดของคำศัพท์ที่มีคำศัพท์และเครื่องหมายเหมือนกัน แต่ลำดับไม่เหมือนกันอาจให้ผลที่ไม่เหมือนกัน เช่น "+กรุงเทพ –เชียงใหม่ &สงขลา" อาจให้ผลไม่เหมือนกับ "&สงขลา +กรุงเทพ –เชียงใหม่"

$$D_{output} = (o_1 \cup o_2 \cup ... o_i) \cup (i_1 \cap i_2 \cap .. i_j) - (m_1 \cup m_2 \cup ... m_k)$$

$D_{outpout}$	เซ็ตของเอกสารผลลัพธ์
o	เซ็ตของเอกสารที่มีคำศัพท์ที่มีเครื่องหมาย”&”นำหน้า
i	เซ็ตของเอกสารที่มีคำศัพท์ที่มีเครื่องหมาย”+”นำหน้า
m	เซ็ตของเอกสารที่มีคำศัพท์ที่มีเครื่องหมาย”–“นำหน้า



ภาพที่ 21 ตัวอย่างการหาเซ็ตของเอกสารผลลัพธ์จากการค้นหาแบบบูลีน

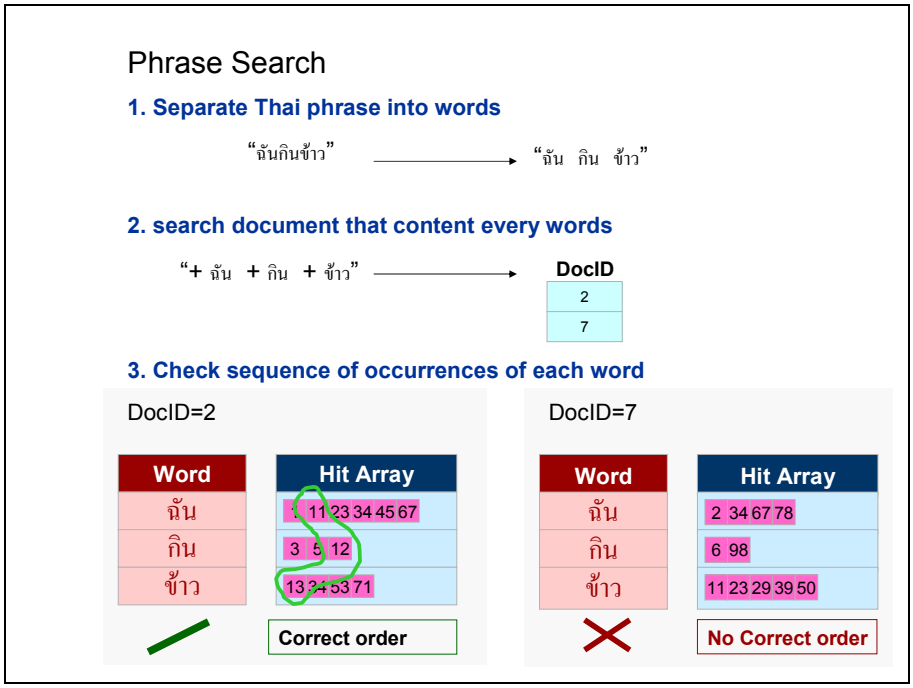
การรวมผลลัพธ์ของการค้นหาแบบบูลีนของคำแต่ละคำจะกระทำในโปรแกรม MetaServer โดยจะรวมผลลัพธ์เข้าด้วยกันทีละคำจากซ้ายไปขวา โดยรวมเซ็ตของเอกสารที่ได้จากคำศัพท์ที่มีเครื่องหมาย “&” มาอินเตอร์เซ็ก (intersect) กับเซ็ตของคำศัพท์ที่มีเครื่องหมาย “+” หลังจากนั้นจึงมาลบ ออกด้วยเซ็ตของคำศัพท์ที่มีเครื่องหมายดังตัวอย่างในภาพที่ 21 การรวมผลลัพธ์จะรวมโดยใช้สูตรต่อไปนี้

เราสามารถเพิ่มความเร็วในการหาเซ็ตของเอกสารที่มีเครื่องหมาย“+”ได้ โดยก่อนการอินเตอร์เซ็ก ให้จัดเรียงลำดับของการอินเตอร์เซ็กเสียใหม่ โดยเริ่มต้นอินเตอร์เซ็กเซ็ตที่มีจำนวนสมาชิกน้อยที่สุดไปหาเซ็ตที่มีขนาดใหญ่ที่สุด สาเหตุที่มีการทำงานเร็วขึ้นเพราะว่าในการอินเตอร์เซ็กนั้น จำนวนสมาชิกของเซ็ตผลลัพธ์จะมีค่าน้อยกว่าหรือเท่ากับจำนวนสมาชิกของเซ็ตที่มีจำนวนสมาชิกน้อยที่สุด เพราะฉะนั้นเราสามารถลดการตรวจสอบที่ไม่จำเป็นลงได้

การค้นหาแบบวลี

การที่โปรแกรมสามารถค้นหาคำศัพท์แบบเป็นวลีได้ ข้อมูลในดัชนีค้นหานอกจากจะต้องเก็บว่าคำศัพท์ใดปรากฏอยู่ในเอกสารไหนแล้ว ยังจะต้องจัดเก็บว่าคำศัพท์ที่ปรากฏนั้นปรากฏอยู่ ณ ตำแหน่งใดของเอกสารอีกด้วย ซึ่งข้อมูลส่วนนี้จะถูกจัดเก็บในฟิลด์ Hit ของข้อมูลในส่วนของ Barrel

โปรแกรมค้นหาแบบกระจายจะมีการทำงานคล้ายกับการค้นหาแบบบูลีนคือ เมื่อโปรแกรม MetaServer ได้รับชุดของวลีที่จะค้นหาจากโปรแกรม MetaSearch แล้ว โปรแกรม MetaServer จะนำวลีแต่ละวลีมาตรวจสอบดูว่า เป็นภาษาไทยหรือภาษาอังกฤษ หากวลีนั้นเป็นภาษาไทย จะตัดคำในวลีให้แยกออกจากกันเป็นคำ ๆเสียก่อน หลังจากนั้นจะนำคำแต่ละคำที่ได้มาค้นหาแบบบูลีนโดยใช้เครื่องหมาย “+” สาเหตุที่ต้องทำเช่นนี้เพราะว่า เราต้องการลดจำนวนของเอกสารที่จะนำมาตรวจสอบว่ามีวลีค้นหาหรือไม่ โดยจะเลือกเอาเฉพาะเอกสารที่มีคำศัพท์ทุกคำในวลีมาค้นหาเท่านั้น หลังจากได้ชุดของเอกสารที่มีคำทุกคำในวลีค้นหาเรียบร้อยแล้ว จึงนำเอกสารเหล่านี้ไปตรวจสอบดูว่ามีวลีที่ค้นหาหรือไม่



ภาพที่ 22 ตัวอย่างการทำงานของการค้นหาแบบวลี

วิธีการตรวจสอบทำได้โดยนำเอา HitArray ของคำศัพท์แต่ละคำในแต่ละเอกสาร มาดูว่าตำแหน่งที่ปรากฏของคำแต่ละคำนั้นอยู่ติดกันหรือไม่ หากคำทุกคำในวลีอยู่ในตำแหน่งติดกันในลำดับที่ถูกต้องแสดงว่า เอกสารนั้นมีวลีค้นหา หากไม่อยู่ติดกันหรือไม่อยู่ในลำดับที่ถูกต้องแสดงว่าเอกสารนั้นไม่มีวลีที่ค้นหา ดังตัวอย่างในภาพที่ 22

ในเอกสารที่มีจำนวนคำมาก คำที่อยู่ท้ายเอกสารอาจไม่สามารถค้นหาแบบวลีได้ ทั้งนี้เนื่องจากฟิลด์ที่ใช้จัดเก็บตำแหน่งการปรากฏของคำแต่ละครั้งใน Hit นั้นมีขนาดเล็ก คือ ขนาด 8 บิตสำหรับ Hit แบบ Fancy และ ขนาด 12 บิต สำหรับ Hit แบบ Plain ซึ่งข้อมูลขนาด 8 บิตและ 12 บิตเก็บตำแหน่งได้มากที่สุดคือ 255 และ 4095 ตามลำดับ เพราะฉะนั้นคำที่ปรากฏอยู่ท้ายของเอกสารคือเกิน 255 คำแรกสำหรับ Hit แบบ Fancy และ 4095 สำหรับ Hit แบบ Plain จะไม่สามารถค้นหาแบบวลีได้

วิธีการจัดลำดับผลลัพธ์

โปรแกรมค้นหาแบบกระจาย มีวิธีการจัดเรียงลำดับของผลลัพธ์โดยมีหลักเกณฑ์ในการพิจารณาอยู่ 3 ข้อคือ คุณลักษณะของคำศัพท์ในเอกสาร (Term Attribute) ความเหมือนของชุดคำศัพท์ที่ค้นหากับคำศัพท์ในเอกสาร (Similarity) และจำนวนลิงค์ชี้เข้าของเอกสาร (In-Link)

ตารางที่ 11 อัตราส่วนการให้คะแนนของแท็กต่าง ๆในไฟล์ HTML

ลำดับที่	HTML Tag	W^{at}
1	Title	100
2	Meta	80
3	A	20
4	H1,H2,H3,B,I,U	20
5	ตัวอักษรธรรมดา,FONT	1

หมายเหตุ สามารถดูรายละเอียดของอัตราส่วนการให้คะแนนของ HTML Tag ได้ที่ตารางที่ 11

วิธีการคำนวณคะแนนสำหรับคุณลักษณะของคำศัพท์ ($Weight^{at}$) ของแต่ละเอกสาร จะอาศัยตำแหน่งการปรากฏของคำศัพท์ในเอกสารว่าคำศัพท์ที่ค้นหาอยู่ใน HTML Tag ใด ซึ่งค่า $Weight^{at}$ นั้นถูกคำนวณและจัดเก็บไว้ในข้อมูลของ Barrel เรียบร้อยแล้ว

ค่า W^{at} ของคำศัพท์แต่ละคำในแต่ละเอกสารสามารถคำนวณได้จากค่า Weight ของแท็กที่คำศัพท์คำนั้นปรากฏที่มีค่ามากที่สุด ตามสูตร

$$w_{ki}^{at} = \max(W^{at})$$

วิธีการคำนวณคะแนนสำหรับวัดค่าความเหมือนของวลีค้นหากับคำศัพท์ในเอกสาร จะคำนวณโดยอาศัยคุณสมบัติต่าง ๆของเอกสารและวลีค้นหา เช่น จำนวนของคำศัพท์ที่ปรากฏในชุดของคำศัพท์ค้นหา จำนวนคำศัพท์ในเอกสาร เป็นต้น ซึ่งสูตรที่ใช้คือ

$$w_{ki}^{tf} = (0.5 + 0.5\frac{tf_{ki}}{\max_{j}(tf_{kj})}) \quad \text{(Salton, G. และ C. Buckley. 1988)}$$

$$w_{ki}^{ds} = \frac{1}{\sqrt{\sum_{i}(w_{ki}^{at} w_{ki}^{tf})^2}}$$

$$w_{ki} = w_{ki}^{tf}.w_{ki}^{ds}.w_{ki}^{at} \quad \text{(Weiss, 1996)}$$

$$Similarity(Q,D) = \sum_{k} w_{ik}.w_{jk}$$

w_{ki}^{tf} Contribution to weight from frequency tf_{ki}

w_{ki}^{ds} Contribution to weight from size of d_k

w_{ki}^{at} Contribution to weight from term attribute

จำนวนลิงค์ชี้เข้าของแต่ละเอกสารสามารถหาได้จากข้อมูลของเว็บเพจที่รวบรวมได้ในขั้นตอนการทำงานของโรบอตแบบกระจาย ดังที่ได้อธิบายไว้แล้วในการทำงานของโปรแกรมโรบอตแบบกระจาย

การคำนวนน้ำหนักของการจัดเรียงเอกสารจะคำนวนโดย อาศัยค่าความเหมือนของเอกสารกับวลีค้นหาและจำนวนลิงค์ชี้เข้าวิธีการคำนวนจะคำนวนโดยใช้สมการต่อไปนี้

$$RankingWeight(Q,D) = A * Similarity(Q,D) + B * inLink(D)$$

A อัตราส่วนของการให้คะแนนโดยอาศัยความเหมือนของเอกสารกับวลีค้นหา

B อัตราส่วนของการให้คะแนนโดยอาศัยจำนวนลิงค์ชี้เข้า

ในการทดลองเรากำหนดให้ค่าอัตราส่วนของการให้คะแนน A และ B เท่ากับ 0.5 เท่ากัน ทั้งนี้เนื่องเพราะเราตั้งการให้พารามิเตอร์ทั้งสองอันมีความสำคัญเท่า ๆกัน

โมดูลการนับจำนวนลิงค์ชี้เข้า

เนื่องจากเราต้องการออกแบบโรบอตเพื่อรวบรวมเว็บเพจและนำข้อมูลของไฮเปอร์ลิงค์มาใช้สำหรับสร้างดัชนีค้นหาสำหรับเสิร์จเอ็นจิน เราจึงออกแบบโรบอตให้มีหน้าที่เพิ่มอีก 1 หน้าที่คือ เก็บความสัมพันธ์ของเว็บเพจต่าง ๆที่รวบรวมมาได้ โดยในที่นี้จะจัดเก็บจำนวนลิงค์ชี้เข้า (In-Link) ของเว็บเพจ วิธีการทำจะกำหนดให้โรบอตแต่ละเครื่องมีตัวนับ (counter) สำหรับนับว่า เว็บเพจใด ๆมีลิงค์ชี้เข้าทั้งหมดครั้ง

ปัญหาที่ตามมาคือหากลดข้อมูลที่ส่งระหว่างเครื่องโดยเลือกส่งเฉพาะ URL ใหม่ทำให้ไม่สามารถนับจำนวนลิงค์ชี้เข้าของเว็บเพจได้ เพราะฉะนั้นในขั้นตอนการทำงานจริงของโรบอตจะส่ง URL ทั้งหมดระหว่างโรบอตแต่ละเครื่อง โดยไม่สนใจว่า URL ที่ส่งไปมานั้นเคยพบแล้วหรือไม่

สถานที่ทำการทดลอง

สถานที่ที่ใช้ในการทำวิทยานิพนธ์นี้คือ ห้องปฏิบัติการเครือข่ายประยุกต์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์

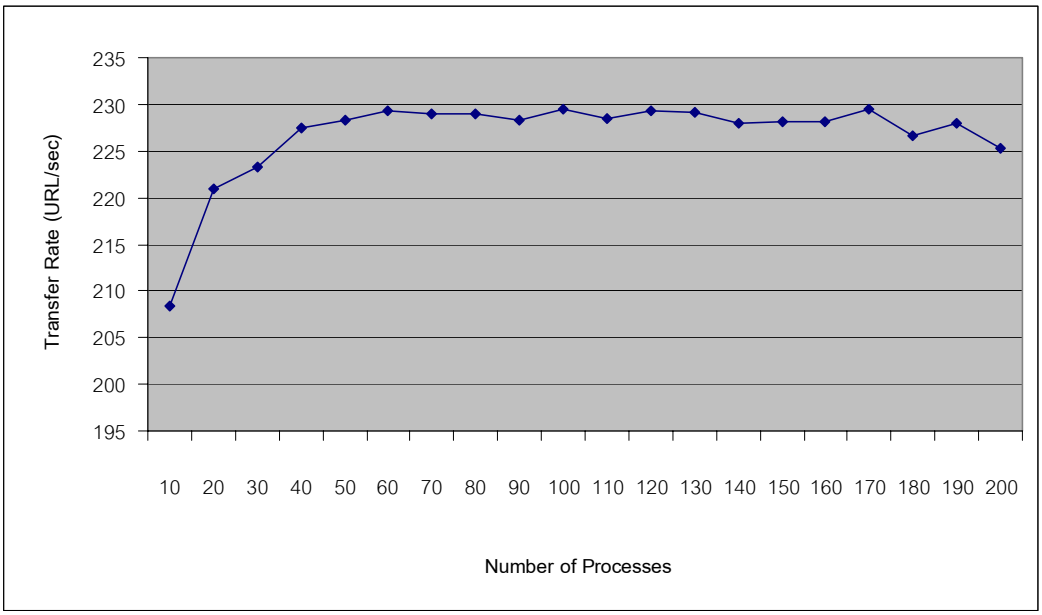
ระยะเวลาทำการวิจัย

ระยะเวลาในการทำงานวิจัย เริ่มต้นพัฒนาโปรแกรมตั้งแต่เดือน กรกฎาคม 2542 จนถึงเดือนกรกฎาคม 2544

ผลและวิจารณ์

ประสิทธิภาพของโปรแกรมโรบอตแบบกระจาย

เนื่องจากประสิทธิภาพของโปรแกรมโรบอตขึ้นอยู่กับปัจจัยต่าง ๆ ทั้งลักษณะของเครือข่ายสภาวะความหนาแน่นของการใช้งานเครือข่าย จำนวนงานของเว็บเซิร์ฟเวอร์ การทดสอบประสิทธิภาพของโรบอตในขั้นแรกจะทดสอบความเร็วของเครือข่ายว่ามีผลต่อประสิทธิภาพของโรบอตหรือไม่

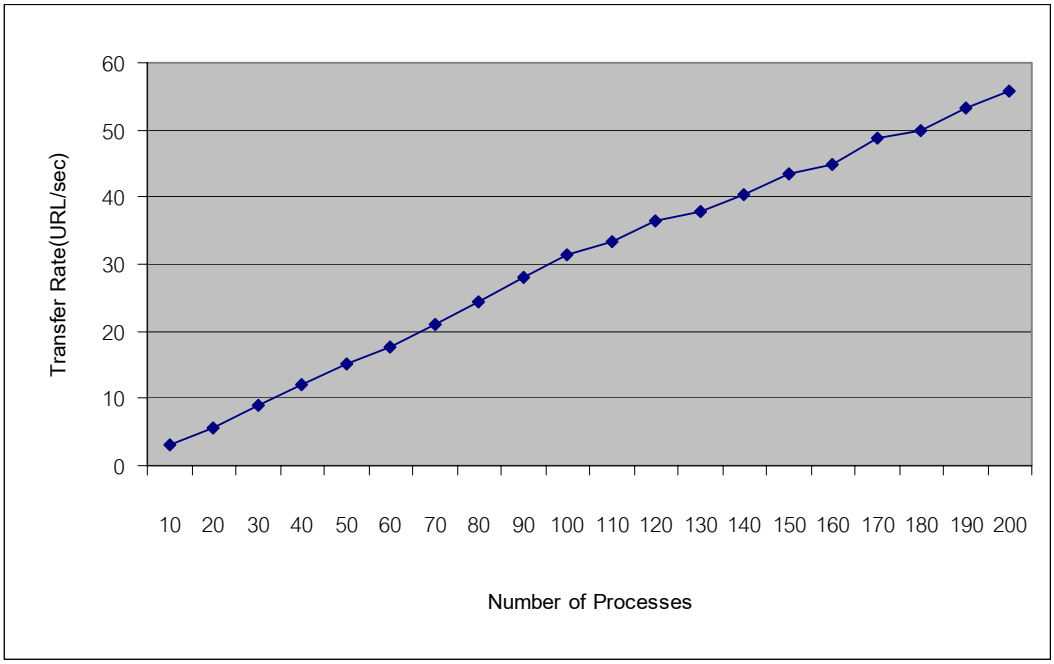


ภาพที่ 23 กราฟแสดงอัตราการอ่านเว็บเพจผ่านเครือข่ายความเร็วสูง

การทดลองจะแบ่งการทดสอบออกเป็น 2 ส่วนคือ ทดสอบประสิทธิภาพของโรบอตโดยให้โรบอตอ่านเว็บเพจผ่านเครือข่ายที่มีความเร็วสูงและทดสอบโดยให้โรบอตอ่านเว็บเพจผ่านเครือข่ายที่มีความเร็วต่ำ ในการทดสอบผ่านเครือข่ายความเร็วสูงนั้น จะทดสอบโดยสั่งให้โปรแกรมเว็บโรบอตทำงานบนเครื่องเซิร์ฟเวอร์ในภาควิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเกษตรศาสตร์ และกำหนดให้โปรแกรมโรบอตรวบรวมเว็บเพจเฉพาะเว็บเพจที่อยู่ภายในมหาวิทยาลัย กล่าวคือรวบรวมเฉพาะเว็บเพจที่อยู่ในโดเมน ku.ac.th ซึ่งได้ผลลัพธ์ดังภาพที่ 23

จากผลการทดลอง พบว่าเมื่อเพิ่มจำนวนโปรเซสถึง 50 โปรเซสแล้ว อัตราการอ่านเว็บเพจจะค่อนข้างคงที่คืออยู่ที่ 230 URLs ต่อวินาที หรือประมาณ 3.4 เมกะไบต์ต่อวินาที สาเหตุที่กราฟค่อนข้างคงที่เพราะโปรแกรมโรบอตอ่านเว็บเพจถึงขีดจำกัดของเครือข่าย และ CPU

ในการทดสอบผ่านเครือข่ายความเร็วต่ำนั้น จะทดสอบโดยให้โปรแกรมเว็บโรบอตทำงานบนเครื่องเซิร์ฟเวอร์ในภาควิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเกษตรศาสตร์ และกำหนดให้โปรแกรมเว็บโรบอตรวบรวมเฉพาะเว็บเพจเฉพาะเว็บเพจที่อยู่ภายนอกมหาวิทยาลัย (นอกโดเมน ku.ac.th) ซึ่งได้ผลดังภาพที่ 24

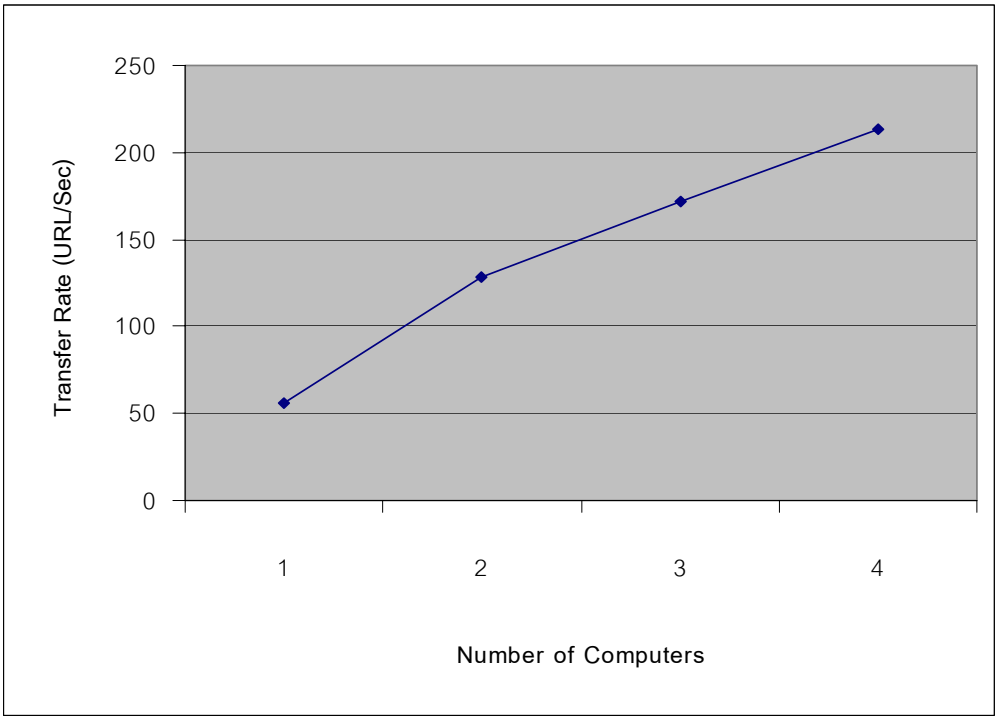


ภาพที่ 24 กราฟแสดงอัตราการอ่านเว็บเพจผ่านเครือข่ายความเร็วต่ำ

จากกราฟ เมื่อจำนวนโปรเซสเพิ่มขึ้นอัตราการอ่านเว็บเพจก็เพิ่มขึ้นด้วยอัตราค่อนข้างคงที่ โดยที่โรบอต 200 โปสเซส โรบอตสามารถอ่านได้ 55 URLต่อวินาที ที่ 1.0 เมกะไบต์ต่อวินาที

จากผลการทดลองจะเห็นได้ว่าสภาวะเครือข่ายระหว่างโรบอตและเว็บเซิร์ฟเวอร์นั้นมีผลต่ออัตราการเพิ่มของความเร็วได้เมื่อเพิ่มจำนวนโปรเซส กล่าวคือ หากเครือข่ายมีความเร็วสูงเมื่อเพิ่มจำนวนโปรเซสเพียงเล็กน้อยอัตราการอ่านเว็บเพจจะคงที่ ต่างจากเน็ตเวิร์คความเร็วต่ำซึ่ง

สามารถเพิ่มโปรเซสเข้าไปได้เยอะกว่าอัตราการเพิ่มจะคงที่ เพราะฉะนั้นเราสามารถสรุปได้ว่าในกรณีที่เน็ตเวิร์คความเร็วสูง การที่จะเพิ่มจำนวนโปรเซสให้มากขึ้น โดยการใช้คอมพิวเตอร์หลายเครื่อง อาจไม่เพิ่มประสิทธิภาพของโรบอต



ภาพที่ 25 กราฟแสดงอัตราการอ่านเว็บเพจของโรบอตแบบกระจายผ่านเครือข่ายความเร็วต่ำ

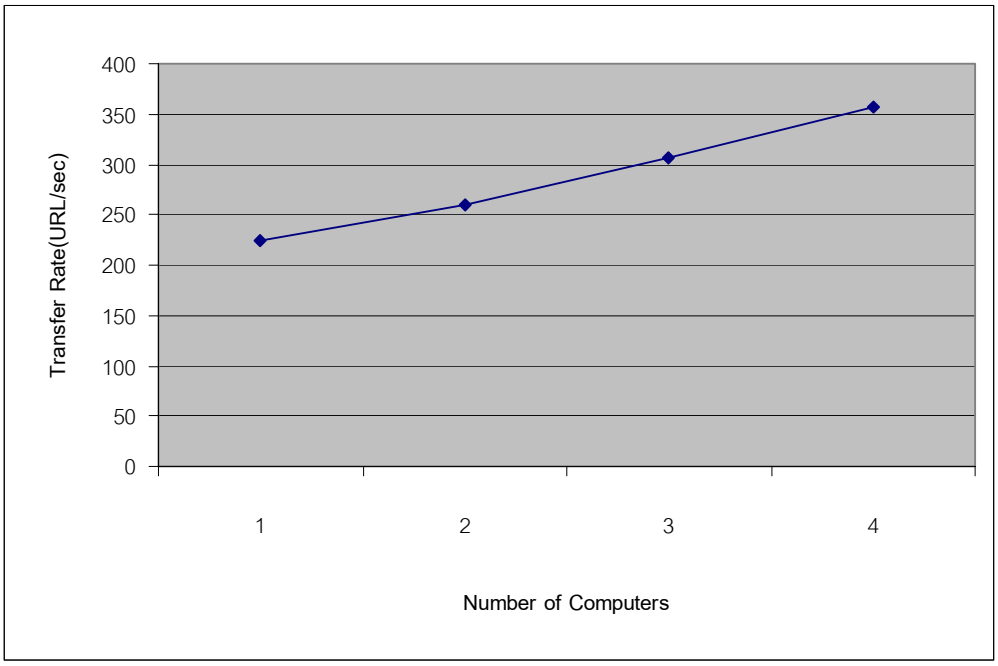
การทดลองอ่านเว็บเพจโดยใช้คอมพิวเตอร์เพียงเครื่องเดียว เราสามารถเพิ่มจำนวนโปรเซสได้จำกัด ทั้งนี้เพราะ มาสเตอร์โปรเซสของโปรแกรมโรบอตนั้นใช้เดสคลิบเตอร์ (descriptor) จำนวนมากในการติดต่อสื่อสารไปยังสเลฟโปรเซสแต่ละตัว แต่ในระบบปฏิบัติการลินุกซ์ได้กำหนด ให้โปรเซสแต่ละโปรเซสสามารถใช้เดสคลิบเตอร์ ได้เพียง 1024 เดสคลิบเตอร์เท่านั้น ทำให้เพิ่มจำนวนโปรเซสได้เพียง 200 กว่าโปรเซสเท่านั้น

การทดสอบประสิทธิภาพของโปรแกรมโรบอตแบบกระจายโดยใช้คอมพิวเตอร์หลายเครื่อง จะแบ่งการทดสอบออกเป็นสองส่วนเช่นเดียวกับการทดสอบเครื่องเดียว ซึ่งจะได้ผลดังกราฟในภาพที่ 25 และ ภาพที่ 26

จากกราฟจะเห็นได้ว่าเมื่อเพิ่มจำนวนเครื่องเข้าไปในระบบ โปรแกรมโรบอตสามารถรวบรวมเว็บเพจได้เร็วมากขึ้น โดยที่อัตราการอ่านเว็บเพจผ่านเครือข่ายความเร็วต่ำอยู่ที่ 55,128,171,213 URLต่อวินาที อัตราการอ่านเว็บเพจผ่านเครือข่ายความเร็วสูงได้ 225,259,307,357 URLต่อวินาที เมื่อใช้คอมพิวเตอร์ 1,2,3 และ4 เครื่องตามลำดับ

เราสามารถคำนวณค่าประสิทธิภาพของการทดสอบทั้งสองการทดลองได้ จากการทดลองผ่านเครือข่ายความเร็วต่ำ เมื่อใช้คอมพิวเตอร์ 1 เครื่องสามารถอ่านได้ 55 URLต่อวินาที เมื่อใช้คอมพิวเตอร์ 4 เครื่องสามารถอ่านได้ 213 URLต่อวินาที เพราะฉะนั้นได้ค่าประสิทธิภาพเป็น (213*100)/(55*4)=96.81%

จากการทดลองผ่านเครือข่ายความเร็วสูง เมื่อใช้คอมพิวเตอร์ 1 เครื่องสามารถอ่านได้ 225 URLต่อวินาที เมื่อใช้คอมพิวเตอร์ 4 เครื่องสามารถอ่านได้ 357 URLต่อวินาที เพราะฉะนั้นได้ค่าประสิทธิภาพเป็น (357*100)/(225*4)=39.67%



ภาพที่ 26 กราฟแสดงอัตราการอ่านเว็บเพจของโรบอตแบบกระจายผ่านเครือข่ายความเร็วสูง

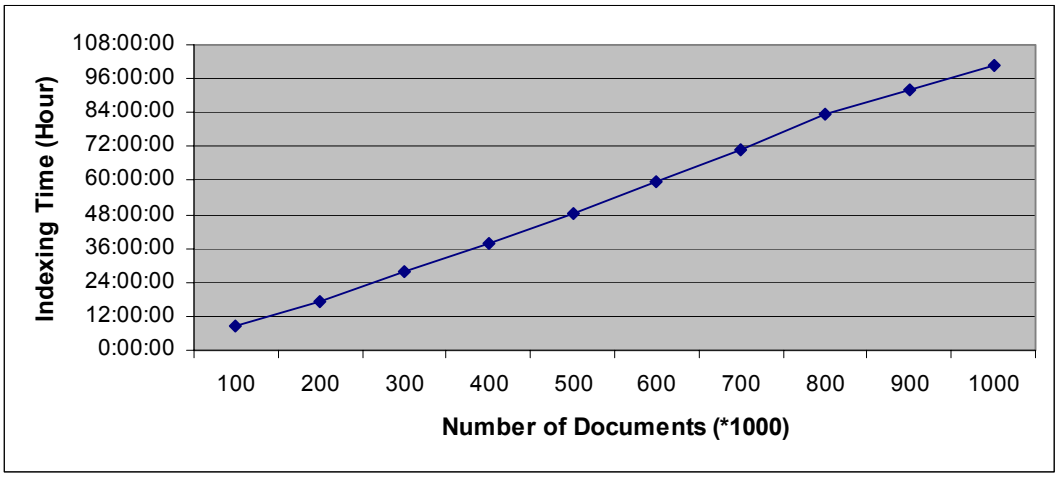
เราสามารถสรุปผลจากการทดลองวัดประสิทธิภาพของโปรแกรมโรบอตได้ว่า เมื่อเพิ่มจำนวนคอมพิวเตอร์เข้าไปในระบบ โปรแกรมโรบอตจะสามารถรวบรวมเว็บเพจได้เร็วขึ้น โดยที่

ค่าประสิทธิภาพของระบบที่จะขึ้นอยู่กับความเร็วของเครือข่ายระหว่างเครื่องที่โรบอตทำงานและเว็บเซิร์ฟเวอร์ กล่าวคือหากเครื่อข่ายมีความเร็วสูงค่าประสิทธิภาพจะน้อยกว่าโรบอตที่ทำงานอยู่บนเครือข่ายความเร็วต่ำ ทั้งนี้เพราะ โรบอตที่ทำงานอยู่บนเครื่อข่ายความเร็วสูง คอมพิวเตอร์แต่ละเครื่องก็จะสามารถอ่านเว็บเพจด้วยอัตราที่สูงมากและหากเพิ่มคอมพิวเตอร์เข้าไปจะทำให้เกิดปัญหาคอขวดที่เน็ตเวิร์คที่โรบอตทำงานมีผลทำให้ประสิทธิภาพต่ำลง แต่หากเครือข่ายระหว่างเครื่องที่โรบอตทำงานกับเว็บเซิร์ฟเวอร์มีความเร็วต่ำ เครื่องแต่ละเครื่องจะรับส่งข้อมูลได้น้อย หากเราเพิ่มจำนวนเครื่องเข้าไปในระบบเพิ่มขึ้น จะช่วยลดค่าความล่าช้า(latency)เฉลี่ยในการส่งข้อมูลลงได้ โดยที่ยังไม่เกิดปัญหาคอขวด

ประสิทธิภาพของโปรแกรมสร้างดัชนีแบบกระจาย

เนื่องจากการทำงานของโปรแกรมสร้างดัชนีค้นหาแบบกระจาย คอมพิวเตอร์แต่ละเครื่องในระบบจะทำงานอิสระแยกจากกัน ไม่มีการอ้างอิงข้อมูลระหว่างกันเลย เพราะฉะนั้นการวัดประสิทธิภาพของโปรแกรม สามารถวัดได้จากการทำงานจากคอมพิวเตอร์เพียงเครื่องเดียว

ตัวอย่างเช่น หากต้องการทราบเวลาที่ใช้ทั้งหมดสำหรับการสร้างดัชนีค้นหา สำหรับเว็บเพจ จำนวน 400,000 เว็บเพจ เมื่อมีคอมพิวเตอร์ในระบบเป็นจำนวน 4 เครื่อง สามารถหาได้โดย ใช้คอมพิวเตอร์เพียงเครื่องเดียวสร้างดัชนีค้นหาสำหรับเอกสารจำนวน 400,000/4 = 100,000 เอกสาร โดยเวลาที่ใช้เครื่องหนึ่งเครื่องสร้างดัชนีค้นหาสำหรับเว็บเพจ 100,000 เว็บเพจจะเท่ากับ เวลาที่ใช้คอมพิวเตอร์ 4 เครื่องสร้างดัชนีค้นหาของเว็บเพจจำนวน 400,000 เว็บเพจ ทั้งนี้เพราะโปรแกรมสร้างดัชนีค้นหานั้นสามารถทำงานได้อย่างอิสระและกระจายภาระงานได้เท่ากัน ภายใต้สมมุติฐานที่ว่าเว็บเพจส่วนใหญ่มีขนาดและจำนวนคำศัพท์เท่า ๆกัน

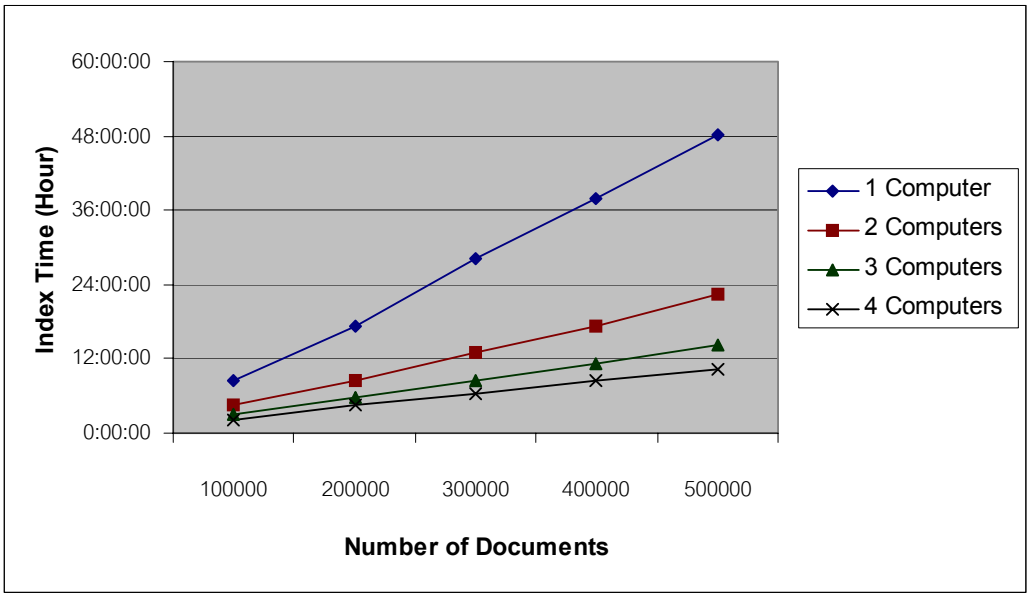


ภาพที่ 27 กราฟแสดงเวลาที่ใช้ในการสร้างดัชนีค้นหาโดยใช้คอมพิวเตอร์ 1 เครื่อง

จากการทดลองจับเวลา โดยใช้คอมพิวเตอร์เพียงเครื่องเดียวสร้างดัชนีค้นหาของเว็บเพจที่มีจำนวนต่างกัน ได้กราฟดังภาพที่ 27 แล้วนำเวลาที่ได้มาวาดกราฟโดยประมาณเวลาของการทำงานที่ใช้คอมพิวเตอร์หลายเครื่องจากการทำงานของคอมพิวเตอร์เพียงเครื่องเดียว ได้กราฟดังภาพที่ 28

จากกราฟในภาพที่ 28 จะเห็นได้ว่าเมื่อเพิ่มจำนวนเว็บเพจ เวลาที่ใช้ในการสร้างดัชนีค้นหาเพิ่มขึ้นในอัตราที่คงที่ เกือบเป็นสมการเส้นตรงซึ่งหมายความว่าจำนวนของเว็บเพจที่นำมาทำดัชนีค้นหานั้นแทบจะไม่มีผลต่ออัตราการสร้างดัชนีค้นหา

จะเห็นได้ว่าเมื่อเพิ่มจำนวนเครื่องเข้าไปในระบบ เวลาที่ใช้ในการสร้างดัชนีค้นหาจะน้อยลงตามลำดับ โดยเมื่อใช้คอมพิวเตอร์จำนวน 1,2,3 และ 4 เครื่องจะได้อัตราเฉลี่ยในการทำดัชนีอยู่ที่ 2.89 6.20 9.71 และ 13.40 เว็บเพจต่อวินาที ตามลำดับ



ภาพที่ 28 กราฟแสดงเวลาที่ใช้ในการสร้างดัชนีค้นหาโดยใช้คอมพิวเตอร์หลายเครื่อง

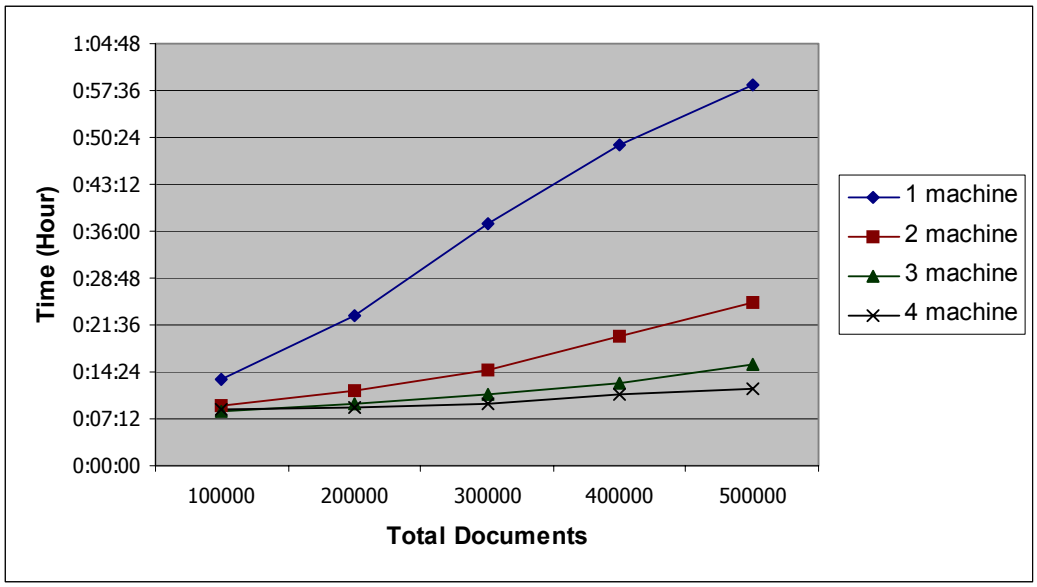
ประสิทธิภาพของโปรแกรมค้นหาแบบกระจาย

การวัดประสิทธิภาพของโรบอตแบบกระจายจะใช้คอมพิวเตอร์ ทั้งหมด 5 เครื่อง โดยใช้คอมพิวเตอร์ 1 เครื่องสำหรับให้โปรแกรม MetaSearch ทำงานโดยโปรแกรม MetaSearch ทำหน้าที่เป็นตัวกระจายคำศัพท์ค้นหาและรวบรวมผลลัพธ์ส่งกลับไปยังผู้ใช้ ส่วนคอมพิวเตอร์ที่เหลือใช้เป็นเครื่องสำหรับเก็บดัชนีค้นหา และรันโปรแกรม MetaServer ซึ่งทำหน้าที่ค้นหาเอกสารตามวลีค้นหาที่โปรแกรม MetaServer กำหนดมา

การทดสอบขั้นแรก จะวัดประสิทธิภาพโดยหาเวลาเฉลี่ยที่ใช้ในการค้นหา โดยทดสอบที่จำนวนเอกสารรวมตั้งแต่ 100,000 ถึง 500,000 เอกสาร สำหรับระบบที่มีเครื่อง MetaServer จำนวน 1 – 4 เครื่อง โดยมีเซ็ตของวลีค้นหาทั้งหมด 1000 ชุด ซึ่งชุดวลีค้นหาเหล่านี้ได้มาจากชุดของวลีค้นหาที่ผู้ใช้ใช้ค้นหามากที่สุดจากเครื่อง search.ku.ac.th

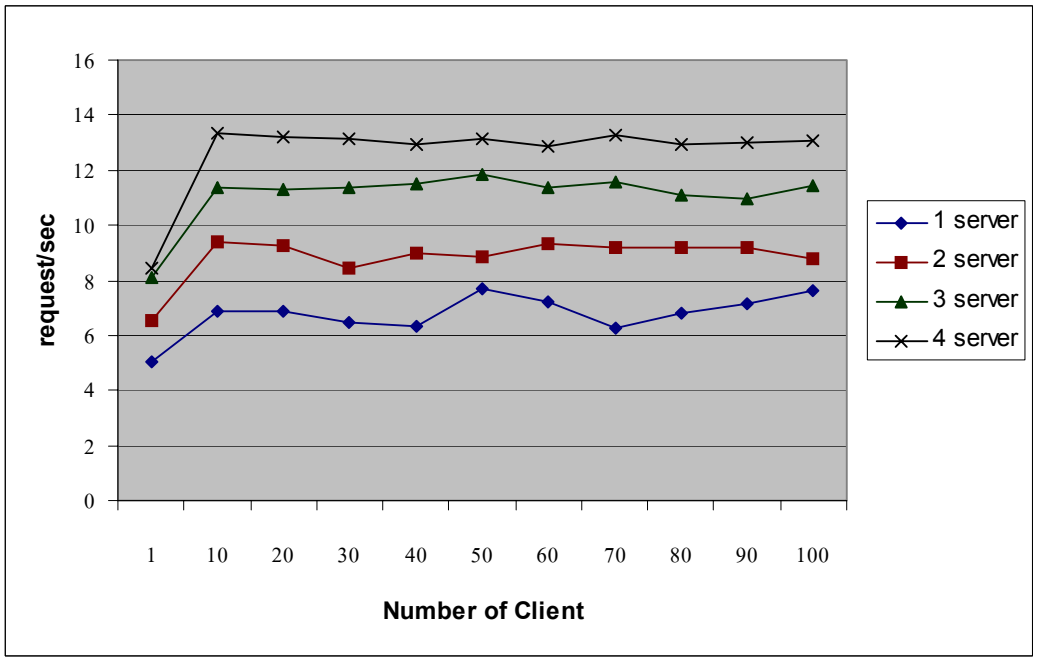
คอมพิวเตอร์ที่ใช้ เป็นสำหรับให้โปรแกรม MetaSearch ทำงานเป็นเครื่อง PentiumIII-800 เมกะเฮิร์ซ หน่วยความจำหลักขนาด 1 จิกกะไบต์ คอมพิวเตอร์อีก 4 เครื่องที่ใช้ให้โปรแกรม MetaServer ทำงานเป็นเครื่อง PentiumIII-800 เมกะเฮิร์ซ หน่วยความจำหลักขนาด 512 เมกะไบต์. จากการทดลองจะได้กราฟดังภาพที่ 29

จากกราฟจะเห็นได้ว่า เมื่อเพิ่มจำนวนเครื่องเข้าไปในระบบจะทำให้เวลาที่ใช้ในการค้นหาลดลง จากการทดลองพบว่า สำหรับเอกสารจำนวน 500,000 เอกสาร โปรแกรม metaServer ใช้เวลาค้นหาโดยเฉลี่ย 3.5, 1.5, 0.93 และ 0.73 วินาทีสำหรับจำนวนเครื่อง 1,2,3 และ 4 ตามลำดับ



ภาพที่ 29 กราฟแสดงเวลาที่ใช้ในการค้นหาชุดของคำศัพท์ค้นหา 1000 ชุด

ผลการทดลองในภาพที่ 29 นั้นใช้คอมพิวเตอร์เพียงเครื่องเดียวแทนผู้ใช้หนึ่งคน แต่ในสภาวะการใช้งานจริงอาจมีผู้ใช้มาใช้งานเสิร์จเอ็นจินพร้อมกันหลายคน จึงได้ทดสอบประสิทธิภาพของโปรแกรมเสิร์จเอ็นจินในสภาวะที่มีไคลเอ็นท์มาใช้งานมาก ๆ โดยการทดลองจะใช้โปรแกรม webbench (Gonopolsky, 2001) จำลองไคลเอ็นท์จำนวนตั้งแต่ 1-100 โปรแกรม



จะหาจำนวนครั้งที่เสิร์จเอ็นจินสามารถค้นหาผลลัพธ์มาได้ โดยจะกำหนดเวลาทดสอบไว้ที่ 180 วินาที ซึ่งจะได้ผลลัพธ์ดังภาพที่ 30

ภาพที่ 30 กราฟแสดงจำนวนการค้นหาที่ทำได้ที่จำนวนไคลเอ็นท์ต่าง ๆกัน

สรุป

การออกแบบโปรแกรมเสิร์จเอ็นจินแบบกระจาย ซึ่งอาศัยแฮชชิ่งฟังก์ชันการออกแบบทำได้ง่ายในการแบ่งงานให้กับคอมพิวเตอร์แต่ละเครื่องมีข้อดีคือ โรบอตที่ทำงานอยู่บนเครื่องแต่ละเครื่องสามารถตัดสินใจได้เองว่าเว็บเพจใดอยู่ในความรับผิดชอบ ทำให้ลดปริมาณข้อมูลที่จะต้องสื่อสารกันในเครือข่าย นอกจากนี้ยังทำให้โปรแกรมสร้างดัชนีค้นหาสามารถทำงานได้โดยอิสระ

ในขั้นตอนการค้นหาได้มีการนำเอาคุณสมบัติด้านต่าง ๆเว็บเพจ เช่น จำนวนลิงค์ชี้เข้าและคุณลักษณะของคำศัพท์ มาช่วยในการจัดลำดับเว็บเพจ ทำให้ผลลัพธ์ที่ได้ที่ดีมากยิ่งขึ้น

จากการทดสอบประสิทธิภาพการทำงานของโปรแกรมเสิร์จเอ็นจินแบบกระจายจะเห็นได้ว่า เมื่อเพิ่มจำนวนเครื่องเข้าไปในระบบจะทำให้การทำงานทั้งในส่วนของการรวบรวมเอกสาร การสร้างดัชนีค้นหา และการค้นหาสามารถทำงานได้เร็วขึ้น รองรับจำนวนเอกสารได้มากขึ้น

ข้อเสนอเสนอแนะ

แม้ว่าโปรแกรมเสิร์จเอ็นจินแบบกระจายจะสามารถทำงานได้ดีในระดับหนึ่งแต่ยังมีข้อเสียและปัญหาที่ทำให้ประสิทธิภาพโดยรวมลดลงอยู่อีกหลายจุด ดังต่อไปนี้

จำนวนเดสคลิบเตอร์ที่ใช้ได้สูงสุด โปรแกรมโรบอตสามารถมีโปรเซสลูกสูงสุดได้เพียง 200 โปรเซส เพราะติดปัญหาเรื่องจำนวนของเดสคลิบเตอร์ที่แต่ละโปรเซสสามารถใช้ได้ หนทางที่จะแก้ไขสามารถทำได้โดยเปลี่ยนวิธีการเขียนโปรแกรมจากที่เป็นมัลติโปรเซส ให้เปลี่ยนเป็นมัลติเทรด(Multi-Thread) ซึ่งถ้าหากโปรแกรมเป็นแบบมัลติเทรดแล้วจะไม่จำเป็นต้องใช้ไฟล์ดิสคริบเตอร์ ในการติดต่อกับเทรดลูกอีกต่อไป เพราะเทรดสามารถแบ่งปันค่าตัวแปรระหว่างเทรดได้

การอ่านเว็บเพจโดยไม่สร้างภาระแก้เว็บเซิร์ฟเวอร์ แม้ว่าจะเปลี่ยนวิธีการเพิ่ม URL เข้าไปในคิวแล้ว แต่สามารถแก้ไขไม่ให้โรบอตติดต่อเว็บไซต์ด้วยกันได้เพียงบางส่วน คือป้องกันได้เฉพาะโปรเซสแต่ละโปรเซสในแต่ละเครื่อง ไม่สามารถป้องกันให้โรบอตต่างเครื่องติดต่อไปที่เว็บเพจเดียวกันได้

การจัดลำดับเอกสารโดยใช้เฉพาะจำนวนลิงค์ชี้เข้า ไม่ได้สามารถแสดงถึงคุณภาพของเว็บเพจโดยสมบูรณ์ สามารถพัฒนาต่อให้มีประสิทธิภาพให้ดีขึ้นได้

เอกสารอ้างอิง

Carriere, J. and R. Kazman. 1997. WebQuery: Searching and visualizing the Web through Connectivity. Proceeding of 1997 International World Wide Web Conference: 701-711.

Croft, W.B. and L. Ruggles. 1979. Using Probabilistic Models of Document Retrieval without Relevance Information. Journal of Documentation 35(4): 285-295.

Faloutsos, C. and S. Christodoulakis. 1984. Signature File: An Access Method for Documents and its Analytical Performance Evaluation. Proceeding of 1984 ACM Transaction on Office Information Systems. 2(4): 237-257.

Fox, E. and W. Lee. 1991. FAST-INV: A Fast Algorithm for Building Large Inverted Files. TR-91-10, VPI&SU Department of Computer Science, Blacksburg. March 1991. 20.

Gonopolsky, O. 2001. Webbench. Ziff Davis Media. ETesting Lab. Available http://www.etestinglabs.com/benchmarks/webbench/webbench.asp, September 8, 2001.

Greg, R. 2002. Search Engine Features by Feature Type. The Users' Guide to Web Searching. Available http://www.searchengineshowdown.com/, January 10, 2002.

Harman, D. 1986. An Experimental Study of Factors Important in Document Ranking. Proceeding of 1986 ACM SIGIR Conference on Research and Development in Information Retrieval: 186-193.

Kleinberg, J. 1998. Authoritative Sources in a Hyperlinked Environment. Proceeding of 1998 ACM-SIAM Symposium on Discrete Algorithms: 668-677

Koster, M. 1994. WebCrawler. Robots Exclusion Protocol. Available http://info.webcrawler.com/mak/projects/robots/index.html,

May 18, 2000: 701-711.

Mauldin, M. 1997 Lycos: Design Choices in an Internet Search Service. Proceeding of 1997 IEEE Expert 12(1):1-8.

Page, L. and S. Brin. 1998. The anatomy of a large-scale hypertextual web search engine. Proceeding of 1998 International Web Conference: 107-117.

Page, L., S. Brin, R. Motwani and T. Winograd. 1998. The PageRank Citation Ranking. Proceeding of 1998 International Web Conference: 161-172.

Randy, D. 2002. Comparative Features of Major WWW Search Engine. Search Engine, Indexes, Directories and Libraries. Available http://www.netstrider.com/search/, January 10,2002.

Salton, G., A. Wong and C. Yang. 1975. A vector Space Model for Automatic Indexing. Proceeding of 1975 ACM Conference on Communication 18 (11): 613-620.

Salton, G. and C. Buckley. 1988. Term-Weighting Approaches in Automatic Text Retrieval. Proceeding of 1998 Information Processing and Management 24(5): 513-23.

Sanguanpong, S., P. Piamsa-nga, Y. Poovarawan and S. Warangrit. 2000. Measuring Thai Web Using NontriSpider. Proceeding of the 2000 International Forum cum Conference on Information Technology and Communication: 123-132.

Sanguanpong, S.,P. Piamsa-nga, S. Keretho, Y. Poovarawan and S. Warangrit. 2000. Measuring and Analysis of the Thai World Wide Web. Proceeding of 2000 Asia Pacific Advance Network: 225-230.

Sanguanpong, S. and S. Warangrit. 1998. NontriSearch: A Search Engine for Campus Networks. Procceeding of 2000 National Computer Science and Engineering Conference. 5.

Sanguanpong, S., S. Warangrit and K. Koht-Arsa. 2000. Facts about the Thai Web. Proceeding of 2000 National Computer Science and Engineering Conference: 102-103.

Sonnenreich, W. and T. Macinta. 1998. Guide to Search Engines. John Wiley & Sons Inc., New York. 441p.

Sornlertlamvanich, V. 1993. Word Segmentation for Thai in Machine Translation System. Proceeding of 2000 National Electronics and Computer Technology Center: 50-56.

Sullivan, D. 1996. How does search engine work? Search Engine Watch: Tip about Internet Search Engine and Search Engine Submission. Available http://www.searchenginewatch.com, July 8, 2000.

Weiss, R., B. Velez, M. Sheldon, C. Manprempre and P. Szilagyi. 1996. HyPursuit: A hierarchical network search engine that exploits content-link hypertext clustering. Proceedings of 1996 ACM Conference on Hypertext: 180-193.

ภาคผนวก

ข้อมูลทางสถิติของเว็บเพจที่อยู่ในโดเมน .th

เสิร์จเอ็นจินนอกจากจะรวบรวมเว็บเพจไว้ใช้สำหรับค้นหาแล้ว เว็บเพจเหล่านี้สามารถนำเป็นใช้เป็นข้อมูลทางด้านสถิติได้อีก เช่น นำไปศึกษาถึงอัตราการเพิ่มของจำนวนเว็บเพจ เป็นต้น

จากการทดลองใช้โปรแกรมโรบอตเพื่อรวบรวมเว็บเพจทั้งหมดที่อยู่ภายใต้โดเมน .th ณ วันที่ 20 เดือนมีนาคม พ.ศ. 2544 สามารถสรุปและจำแนกข้อมูลเว็บเพจที่ได้ในแง่มุมต่าง ๆ (Sanguanpong และคณะ 2000 C) (Sanguanpong และคณะ, 2000 D) ได้ดังต่อไปนี้

ตารางผนวกที่ 1 จำนวนเว็บเพจภายใต้โดเมน .th

ลำดับที่	ประเภท	จำนวน
1	จำนวนเว็บเพจที่เป็นชนิด HTML	1146340
2	จำนวนเว็บเซิร์ฟเวอร์	6200
3	จำนวนเว็บเซิร์ฟเวอร์ที่มีชื่อโฮสต์ขึ้นต้นด้วย www	3860

หมายเหตุ จำนวนเว็บเพจที่มีอยู่จริงอาจจะมีจำนวนมากกว่าที่รวบรวมได้ทั้งนี้อาจมีได้หลายสาเหตุ เช่น เว็บเพจบางเว็บเพจไม่มีเว็บเพจอื่นชี้ไปหาทำให้โปรแกรมโรบอตสามารถเข้าถึงเว็บเพจเหล่านั้น เว็บเซิร์ฟเวอร์หลายแห่งไม่อนุญาตให้บุคคลภายนอกอ่านเว็บเพจบ้างหน้า เป็นต้น

ตารางผนวกที่ 2 จำนวนเว็บเพจภายในโดเมนย่อยภายใต้โดเมน .th

โดเมน	จำนวนเว็บเซิร์ฟเวอร์	จำนวนเว็บเพจ	จำนวนเว็บเซิร์ฟเวอร์มีชื่อขึ้นต้นด้วย www
ac.th	2,072	505,266	505,266
co.th	2,805	318,089	318,089
go.th	508	151,532	151,532
or.th	458	106,716	106,716
net.th	163	64,915	64,915
in.th	173	15,570	15,570
mi.th	21	5,225	5,225
รวม	6,200	1167,411	1,167,411

ตารางผนวกที่ 3 10 อันดับโดเมนที่มีเว็บเซิร์ฟเวอร์มากที่สุดภายใต้โดเมน .th

ลำดับที่	โดเมน	จำนวนเว็บเซิร์ฟเวอร์	จำนวนเว็บเพจ	จำนวนเว็บเซิร์ฟเวอร์มีชื่อขึ้นต้นด้วย www
1	chula.ac.th	179	30,846	71
2	ku.ac.th	126	65,037	29
3	kku.ac.th	126	31,342	8
4	tu.ac.th	117	29,854	45
5	inet.co.th	109	21,257	40
6	cmu.ac.th	101	23,041	42
7	psu.ac.th	86	19,992	24
8	police.go.th	77	3,051	42
9	au.ac.th	76	29,002	24
10	rit.ac.th	67	3,857	26

ตารางผนวกที่ 4 จำนวนของเว็บเพจที่ขนาดไฟล์ต่าง ๆ

ขนาดไฟล์	จำนวนเอกสาร	เปอร์เซ็นต์
1-128	1,498	0.13
129-256	1,060	0.09
257-512	22,396	1.92
513-1K	89,576	7.67
1K-2K	172,048	14.72
2K-4K	184,903	15.82
4K-8K	167,520	14.34
8K-16K	142,989	12.24
16K-32K	312,026	28.70
32K-64K	54,045	4.62
64K-128K	15,630	1.34
128K-256K	3,558	0.30
256K-512K	1,018	0.09
512K-1M	212	0.02
1M-4M	74	0.00

ตารางผนวกที่ 5 จำนวนลิงค์ที่ชี้ไปยังไฟล์ชนิดต่าง ๆ

ชนิด	จำนวนลิงค์	เปอร์เซ็นต์
.html	12,556,908	51.61%
.gif	7,374,846	19.99%
.htm	2,395,818	5.41%
.jpg	818,880	1.76%
.asp	197,183	0.42%
.zip	96,787	0.20%
.js	75,208	0.16%
Other	816,338	1.71%
Total	47,847,598	100.00%