ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ

เรื่อง การทนต่อความผิดพร่องในไดนามิกเว็บโดยใช้แคช

โดย นายศุภวัฒน์ แก้วมงคล

ได้รับอนุมัติให้นับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

_____คณบดีบัณฑิตวิทยาลัย
(อาจารย์ ดร.มงคล หวังสถิตย์วงษ์)
17 พฤษภาคม 2550

คณะกรรมการสอบวิทยานิพนธ์

_____ประธานกรรมการ
(อาจารย์ ดร.มารอง ผดุงสิทธิ์)

_____กรรมการ
(รองศาสตราจารย์ ดร.วรา วราวิทย์)

_____กรรมการ
(อาจารย์ ดร.เบญจพร ลิ้มธรรมาภรณ์)

การทนต่อความผิดพร่องในไดนามิกเว็บโดยใช้แคช

นายศุภวัฒน์ แก้วมงคล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
บัณฑิตวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ
ปีการศึกษา 2549

ชื่อ : นายศุภวัฒน์ แก้วมงคล
ชื่อวิทยานิพนธ์ : การทนต่อความผิดพร่องในไดนามิกเว็บโดยใช้แคช
สาขาวิชา : วิศวกรรมไฟฟ้า
สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ
ที่ปรึกษาวิทยานิพนธ์ : อาจารย์ ดร.มารอง ผดุงสิทธิ์
ปีการศึกษา : 2549

บทคัดย่อ

ไดนามิกเว็บ (DynamicWeb) สามารถจะส่งผลให้เว็บเซิร์ฟเวอร์ (Web Server) นั้นมีประสิทธิภาพลดลงเนื่องจากมีการประมวลผลสคริปต์ทางฝั่งเซิร์ฟเวอร์ (Server-Side Script) เทคนิคทั่วไปที่ใช้ในการบรรเทาปัญหาดังกล่าวคือการแคชหน้าไดนามิกเว็บ (Dynamic Web Caching) ซึ่งส่งผลให้การร้องขอหน้าเว็บเดิมในคราวถัดไปไม่จำเป็นต้องมีการประมวลผลสคริปต์ซ้ำอีก เนื่องด้วยแคชนั้นมักจะถูกเก็บไว้ในหน่วยความจำคอมพิวเตอร์ไว้เป็นระยะเวลานาน จึงมีโอกาสเกิดความผิดพร่อง (Fault) ในหน่วยความจำและอาจส่งผลให้มีการสร้างหน้าเว็บที่ผิดพลาด สำหรับเว็บแอปพลิเคชัน (Web Application) บางประเภทที่ต้องการความน่าเชื่อสูง (Reliability) จำเป็นจะต้องมีการหลีกเลี่ยงไม่ให้เกิดความผิดพร่องในระหว่างการสร้างหน้าเว็บ หรือทำให้สามารถทนทานต่อความผิดพร่อง (Fault-tolerance) ดังกล่าวได้ เพื่อให้ไดนามิกเว็บสามารถทนทานต่อความผิดพร่องได้ ผู้วิจัยจึงได้นำเสนอเทคนิคซึ่งใช้หลักการเพิ่มส่วนที่ซ้ำซ้อนของข้อมูลข่าวสาร (Information Redundancy) เข้ากับโปรแกรมประยุกต์ที่ใช้ในการแคชไดนามิกเว็บซึ่งมีอยู่แล้ว โดยผลที่ได้ปรากฏว่าสามารถทนทานต่อความผิดพร่องได้ โดยไม่ส่งผลกระทบต่อประสิทธิภาพความเร็วในการสร้างหน้าเว็บมากนัก

(วิทยานิพนธ์มีจำนวนทั้งสิ้น 50 หน้า)

คำสำคัญ : ไดนามิกเว็บ, ส่วนขยาย PHP, ทนต่อความผิดพร่อง, การแคช, ส่วนที่ซ้ำซ้อน

_____อาจารย์ที่ปรึกษาวิทยานิพนธ์

Name : Mr. Supawat Kawmongkol
Thesis Title : Fault-Tolerant Dynamic Web Using Cache
Major Field : Electrical Engineering
King Mongkut's Institute of Technology North Bangkok
Thesis Advisor : Dr. Marong Phadoongsidhi
Academic Year : 2006

Abstract

Dynamic Web can adversely affect the performance of web servers due to Server-Side script processing. A common technique employed in order to reduce performance degradation is to cache dynamic web pages so that subsequent requests of the same pages do not result in unnecessary script processing. Since cache is often stored in computer memory for possibly an extended duration its content is more exposed to memory faults, possibly resulting in erroneous web page generations. When high reliability is critical for certain classes of web applications, faults occurred during web page generations must be avoided or tolerated. To achieve fault tolerance in dynamic web applications, we have incorporated a technique based on information redundancy concept with an existing dynamic web caching application. The results show that faults can be tolerated without significant effect on the speed of page generation.

(Total 50 pages)

Keywords : Dynamic Web, PHP Extension, Fault-tolerance, Caching, Information redundancy

_____ Advisor

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สามารถดำเนินการสำเร็จได้ตามวัตถุประสงค์ทุกประการนั้น ก็โดยการได้รับคำแนะนำและความช่วยเหลือและดูแลอย่างดียิ่งของ อาจารย์ ดร.มารอง ผดุงสิทธิ์ ซึ่งได้ให้คำแนะนำตลอดการทำวิทยานิพนธ์ ทำให้งานวิจัยชิ้นนี้บรรลุตามวัตถุประสงค์ที่ตั้งใจไว้ ซึ่งผู้วิจัยขอขอบพระคุณท่านอาจารย์เป็นอย่างยิ่งไว้ ณ โอกาสนี้

ขอขอบคุณเพื่อนพี่น้องในห้องวิจัยคอมพิวเตอร์ที่ให้คำแนะนำในด้านต่างๆ และขอขอบคุณอาจารย์ประจำภาควิชาวิศวกรรมไฟฟ้าและเจ้าหน้าที่ของภาควิชาทุกๆท่าน ซึ่งได้อำนวยความสะดวกในด้านต่างๆ จนทำให้วิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยดี

ท้ายสุดซึ่งขาดเสียมิได้ คือขอขอบพระคุณ บิดา มารดา ซึ่งสนับสนุนในทุกๆ ด้านจนสำเร็จการศึกษา

ศุภวัฒน์ แก้วมงคล

สารบัญ

สารบัญตาราง

สารบัญภาพ

ภาพที่ หน้า

บทที่ 1
บทนำ

ส่วนหนึ่งของโลกไร้พรมแดน (Cyberspace) นั้นก็คือเวิลด์ไวด์เว็บ (World Wide Web, WWW) ในที่นี้ผู้วิจัยขอใช้คำที่เรียกกันสั้นๆว่าเว็บ (Web) ซึ่งเว็บเป็นรูปแบบหนึ่งของการสื่อสารบนอินเทอร์เน็ต (Internet) โดยใช้มาตรฐานการสื่อสารไฮเปอร์เท็กซ์ (HyperText Transfer Protocol, HTTP) เป็นตัวกำหนดลักษณะการสื่อสารระหว่างเว็บบราวเซอร์ (Web Browser) และเว็บเซิร์ฟเวอร์ (Web Server) ลักษณะการทำงานแบบนี้ถูกเรียกว่าไคลเอนต์เซิร์ฟเวอร์ (Client-Server) ในโลกของอินเทอร์เน็ตคงไม่มีใครปฏิเสธได้ว่าการใช้เว็บนั้นได้รับว่าความนิยมมากที่สุด หรืออาจกล่าวได้ว่าเป็นการปลุกกระแสการใช้บริการอินเทอร์เน็ต ให้มีการขยายกลุ่มผู้ใช้งานอินเทอร์เน็ต ในยุคเริ่มแรกของการใช้งานอินเทอร์เน็ตเป็นการใช้งานที่ค่อนข้างจะเฉพาะทาง และเฉพาะกลุ่ม ต่อมาก็เริ่มมีการขยายตัวไปสู่กลุ่มผู้ใช้งานคอมพิวเตอร์ทั่วไป เนื่องจากสามารถเป็นเครื่องมือที่ใช้ในการค้นหาข้อมูล และยังมีบริการต่างๆของอินเทอร์เน็ตอีกมากมายที่สามารถใช้ได้อย่างสะดวกรวดเร็ว ดังนั้นจะเห็นได้ว่าบริการของเว็บได้รับการพัฒนาอย่างต่อเนื่อง จากโปรแกรมเว็บบราวเซอร์ที่สามารถแสดงได้เฉพาะตัวอักษร เช่น โปรแกรม lynx จนมีการพัฒนาให้สามารถแสดงรูปภาพหรือมัลติมีเดียอื่นๆได้ เช่น โปรแกรม Internet Explorer หรือ Firefox ส่วนเว็บเซิร์ฟเวอร์ก็จะมีซอฟต์แวร์ (Software) เช่น Apache หรือ IIS โดยปกติจะติดต่อทางพอร์ต 80

รูปแบบการบอกตำแหน่งทรัพยากร (Uniform Resource Locator ,URL) เป็นระบบมาตรฐานที่ใช้กำหนดตำแหน่งที่อยู่ของหน้าเว็บ (Web Page) แต่ละหน้า ซึ่งเมื่อมีหน้าเว็บหลายหน้าก็จะเป็นกลุ่มข้อมูลที่อยู่บนเว็บนั้นจะถูกเรียกรวมๆกันว่าเว็บไซต์ (Web Site) ซึ่งเนื้อหาที่แสดงในหน้าเว็บของการใช้งานเว็บในช่วงเริ่มแรกเป็นการนำเสนอข้อมูลในทิศทางเดียวของผู้ที่ต้องการจะนำเสนอเนื้อหาต่างๆ ไปสู่ผู้ที่เปิดหน้าเว็บนั้นๆ โดยสามารถถูกเรียกดูด้วยโปรแกรมที่เรียกว่าเว็บบราวเซอร์ ซึ่งอ่านข้อมูลหน้าเว็บมาจากเว็บเซิร์ฟเวอร์ เมื่อข้อมูลถูกอ่านมาจากเว็บเซิร์ฟเวอร์จะมาแสดงผลบนจอคอมพิวเตอร์โดยผ่านทางเว็บบราวเซอร์เพื่อให้ผู้ใช้สามารถเข้าถึงข้อมูลเว็บได้ เมื่อผู้ใช้ต้องการดูหน้าเว็บอื่น ก็สามารถเลือกไฮเปอร์ลิงค์ (Hyperlink) เพื่อเชื่อมโยงไปยังหน้าเว็บอื่นๆ หรือทำการเปลี่ยนรูปแบบการบอกตำแหน่งทรัพยากร หรือที่เรียกกันสั้นๆว่ายูอาร์แอล (URL) เพื่อไปยังหน้าเว็บอื่นที่ผู้ใช้งานต้องการ ในการอ้างตำแหน่งถึงในแต่ละเว็บไซต์นั้นต้องมีเลขที่อยู่ไอพี (IP Address) เช่น 202.44.xx.xx ซึ่งการใช้เลขที่อยู่ไอพีเพื่อที่จะอ้างอิงตำแหน่งของเว็บไซต์นั้นถ้าเป็น

ผู้ใช้โดยทั่วไปแล้วการที่จะจดจำหมายเลขไอพี (IP Number) เพื่ออ้างอิงเว็บไซต์ต่างๆที่ต้องการเข้าไปเยี่ยมชมก็จะดูลำบากและมักจะสับสน ดังนั้นจึงมีวิธีการทีใช้ชื่อขึ้นแทนหมายเลขไอพี นั้นก็คือ ระบบการตั้งชื่อโดเมน (Domain Name System, DNS)

ภาษาที่ใช้ในการสร้างหน้าเว็บนั้นคือ HTML (HyperText Markup Language) ซึ่งถูกพัฒนามาจากภาษา SGML (Standard Generalized Markup Language) เป็นภาษาที่ใช้กันในการนำเสนอหน้าเว็บ แต่การสร้างหน้าเว็บใช้เฉพาะภาษา HTML ซึ่งเป็นสแตติกเว็บ (Static Web) โดยมีลักษณะของเนื้อหาเป็นแบบสถิต (Static Content) การพัฒนาของเว็บนั้นมีมาอย่างต่อเนื่องจนเป็นรูปแบบของแอปพลิเคชัน (Application) ซึ่งแอปพลิเคชันที่ใช้อยู่บนเว็บนั้นจะเรียกว่าเป็นเว็บแอปพลิเคชัน (Web Application) โดยมีการพัฒนาภาษาสคริปต์ขึ้นมาเพื่อใช้ร่วมกับ HTML เช่น จาวาสคริปต์ (JavaScript) ซึ่งมีการใช้กันอย่างแพร่หลาย ลักษณะสคริปต์ของจาวาสคริปต์นั้นเป็นสคริปต์ฝั่งไคลเอนต์ (Client-Side Script) ซึ่งเป็นสคริปต์ที่มีการประมวลผลทางด้านฝั่งไคลเอนต์ (Client side) จาวาสคริปต์เป็นภาษาที่มีโครงสร้างของภาษาและไวยากรณ์อยู่บนพื้นฐานของภาษาซี ปัจจุบันมีการใช้จาวาสคริปต์ที่ฝังอยู่ในเว็บบราวเซอร์ในหลายรูปแบบเพื่อสร้างเนื้อหาพลวัต (Dynamic Content) ภายในหน้าเว็บ เช่น การใช้เพื่อตรวจสอบความถูกต้องของข้อมูลที่ผู้ใช้กรอกก่อนเข้าสู่ระบบ

ส่วนภาษาสคริปต์ฝั่งเซิร์ฟเวอร์ (Server-Side Script) เช่น JSP, ASP หรือ PHP ซึ่งเป็นเทคโนโลยีที่สคริปต์ทำการประมวลผลที่ฝั่งเซิร์ฟเวอร์ (Server Side) ซึ่งแตกต่างกับสคริปต์ฝั่งไคลเอนต์อย่างจาวาสคริปต์ที่ทำการประมวลผลที่ฝั่งไคลเอนต์ สำหรับภาษาสคริปต์ที่ผู้วิจัยใช้ในวิทยานิพนธ์นี้คือ PHP ซึ่งเป็นภาษาสคริปต์ฝั่งเซิร์ฟเวอร์ที่มีผู้ใช้กันมากที่สุดทั่วโลกและยังใช้สำหรับสร้างหน้าเว็บแบบมีการตอบสนองกับผู้ใช้งาน ซึ่งมีลักษณะเป็นไดนามิกเว็บ (Dynamic Web) โดยอยู่ในรูปแบบของเว็บแอปพลิเคชัน ไดนามิกเว็บหมายถึงเว็บที่มีเนื้อหาเปลี่ยนไปได้ในแต่ละครั้งที่ผู้ใช้งานเปิดดู ซึ่งขึ้นอยู่กับเงื่อนไขต่างๆของโปรแกรม เว็บแอปพลิเคชันนั้นเป็นโปรแกรมประยุกต์ (Application Software) ที่มีการใช้งานกันอย่างกว้างขวาง เช่น การพาณิชย์อิเล็กทรอนิกส์ (E-Commerce) หรือในงานที่ต้องการความน่าเชื่อถือสูง (Reliability) เช่น ทางด้านการแพทย์ (Medical) [1] และการเงินการธนาคาร (E-Banking) เว็บแอปพลิเคชันนั้นสามารถทำงานอยู่บนฮาร์ดแวร์ (Hardware) ที่แตกต่างกันและระบบปฏิบัติการที่แตกต่างกันด้วย ทั้งนี้ยังสามารถทำงานรวมกันในหลายๆภาษา แล้วยังมีฐานข้อมูล (Database) ที่ใช้ในการเก็บข้อมูลต่างๆของผู้ใช้งาน หรืออาจจะอยู่บนเครือข่ายที่แตกต่างกันไม่ว่าจะเป็นอินเทอร์เน็ตหรืออินทราเน็ต (Intranet) จากที่กล่าวมาจะสังเกตได้ว่าการพัฒนาเว็บแอปพลิเคชันนั้นมีความซับซ้อน การพัฒนาเว็บแอปพลิเคชันนั้นมีหลายกลุ่มบุคคลที่ร่วมในการพัฒนาไม่ว่าจะเป็นโปรแกรมเมอร์

(Programmer) ผู้ออกแบบรูปภาพ (Graphic Designer) ผู้ร่างข้อมูล (Information Layout) ผู้ดูแลฐานข้อมูล (Database Administrator)

ผู้ใช้งานเว็บแอปพลิเคชันสามารถเข้ามาใช้งานได้พร้อมกันได้ในปริมาณมากๆ ขึ้นอยู่กับความเร็วของเครือข่ายและประสิทธิภาพของเซิร์ฟเวอร์ วิธีการอย่างหนึ่งในการเพิ่มประสิทธิภาพของคอมพิวเตอร์คือการแคช (Caching) แคช (Cache) คือส่วนของข้อมูลที่ถูกเก็บไว้เพื่อใช้ในการประมวลผลหรือการอ่านข้อมูลครั้งต่อไปโดยไม่ต้องเรียกข้อมูลจากต้นแหล่ง (Source) เมื่อแคชถูกสร้างขึ้น การเรียกใช้ข้อมูลในครั้งต่อไปจะถูกอ่านข้อมูลจากแคชแทนที่จะอ่านข้อมูลจากต้นฉบับหรือต้นแหล่งเพื่อประหยัดเวลารวมถึงการเพิ่มความเร็วในการเรียกใช้งาน แคชนิยมใช้เมื่อรูปแบบการใช้ข้อมูลมีลักษณะที่ใกล้เคียงกันและมีการใช้ซ้ำบ่อย

สำหรับหน้าไดนามิกเว็บ (Dynamic Web Page) ทำให้เว็บเซิร์ฟเวอร์มีประสิทธิภาพลดลง [2] เนื่องจากมีการประมวลผลสคริปต์ทางฝั่งเซิร์ฟเวอร์ วิธีการอย่างหนึ่งที่ใช้ในการเพิ่มประสิทธิภาพของเซิร์ฟเวอร์คือการเก็บสคริปต์ทางฝั่งเซิร์ฟเวอร์ที่ได้ทำการประมวลผลแล้ว เพื่อนำมาใช้งานซ้ำในครั้งต่อไป โดยวิธีการนี้ถูกเรียกกันว่าเป็นการแคชไดนามิกเว็บ (Dynamic Web Caching) ข้อมูลแคชสามารถที่จะเก็บไว้ได้ทั้งที่หน่วยความจำหรือว่าฮาร์ดดิสก์ ขึ้นอยู่กับว่าหน่วยความจำที่ถูกแบ่ง (Shared Memory) นั้นถูกแบ่งมากน้อยเพียงใด โดยวัตถุประสงค์ของการแคชไดนามิกเว็บคือการลดภาระการประมวลผลของเว็บเซิร์ฟเวอร์ จากการที่ต้องสร้างหน้าเว็บไดนามิก

ในการใช้งานแอปพลิเคชันต่างๆ ไม่ว่าจะเป็นเว็บแอปพลิเคชันหรือไม่ก็ตามย่อมมีโอกาสที่สามารถจะเกิดความผิดพร่องได้ (Fault) ไม่ว่าจะเป็นทางด้านซอร์ฟแวร์หรือว่าฮาร์ดแวร์ ความผิดพร่องเกิดได้จากความพร่อง (Defect) หรือความไม่สมบูรณ์แบบที่เกิดขึ้นกับบางส่วนของฮาร์ดแวร์หรือซอร์ฟแวร์ ส่วนข้อผิดพลาด (Error) นั้นคือสิ่งที่เกิดขึ้นจากความผิดพร่อง ข้อผิดพลาดนั้นคือความเบี่ยงเบนไปจากความถูกต้องหรือความแม่นยำของค่าที่ควรจะเป็น ทั้งนี้ข้อผิดพลาดที่เกิดขึ้นจากส่วนหนึ่งส่วนใดของระบบยังสามารถทำให้เกิดความล้มเหลว (Failure) ของระบบ จากที่กล่าวมาในข้างต้นเกี่ยวกับเรื่องของการเพิ่มประสิทธิภาพของเซิร์ฟเวอร์โดยการใช้แคช โดยข้อมูลแคชนั้นได้ถูกเก็บไว้ในหน่วยความจำ ซึ่งหน่วยความจำเป็นส่วนหนึ่งของระบบคอมพิวเตอร์ที่สามารถจะเกิดความผิดพร่องได้ เนื่องจากหลายปัจจัยทั้งภายนอกและภายในหน่วยความจำ ถ้าหากในการใช้งานแอปพลิเคชันที่สำคัญบางอย่าง เช่น ทางด้านการเงิน การแพทย์หรือทางด้านการทหารที่ไม่อาจยอมรับได้แม้ความผิดพร่องจะเกิดขึ้นเพียงเล็กน้อย และถ้าหากไม่สามารถที่จะกำจัดความผิดพร่องจากความพร่องทางกายภาพ (Physical Defect) ของหน่วยความจำได้ทั้งหมด ฉะนั้นก็ควรที่จะคิดค้นและสร้างหรือพัฒนาวิธีการที่สามารถทนต่อความผิดพร่อง (Fault-Tolerant) เพื่อเพิ่มความน่าเชื่อถือของแอปพลิเคชัน

วิทยานิพนธ์ได้นำเสนอวิธีการเพิ่มความน่าเชื่อถือในไดนามิกเว็บ ซึ่งทำให้เว็บแอปพลิเคชันทนต่อความผิดพร่องที่เกิดขึ้น โดยมุ่งเน้นไปในการแก้ไขข้อผิดพลาดชั่วคราว (Soft Error) ที่เกิดขึ้นในหน่วยความจำที่ใช้ในการเก็บแคช ซึ่งแคชของไดนามิกเว็บอาจมีข้อผิดพลาดเกิดขึ้นในหน่วยความจำส่งผลให้ของข้อมูลที่อยู่ในแคชนั้นเกิดความเสียหาย จากการเกิดข้อผิดพลาดเพียงเล็กน้อยนั้นอาจจะไม่ทำให้ระบบเกิดความล้มเหลว ระบบอาจจะทำงานได้อยู่แต่ผู้ใช้งานอาจจะได้รับข้อมูลที่ไม่ถูกต้อง แต่ถ้าหากข้อผิดพลาดจำนวนมากแล้วเว็บแอปพลิเคชันก็อาจจะไม่มีการตอบสนองใดๆเลยกับผู้งาน

โดยวิธีการปรับปรุงไดนามิกเว็บแคชที่ได้ทำในวิทยานิพนธ์นี้คือ การเพิ่มส่วนที่ซ้ำซ้อนหรือส่วนขยายเพิ่มของข้อมูล (Information Redundancy) ของแคช เพื่อเป็นการเพิ่มความน่าเชื่อถือของแคชที่เก็บในหน่วยความจำ แนวทางหนึ่งในการแก้ปัญหาคือเมื่อมีการร้องขอข้อมูลหน้าเว็บจากเว็บบราวเซอร์ในครั้งแรกก็จะมีการเก็บแคชและทำสำเนาของแคช (Duplicate Cache) ซึ่งเป็นการสำรองข้อมูลเพื่อใช้ในการตรวจสอบหาข้อผิดพลาดของแคช เมื่อมีการร้องขอข้อมูลจากเว็บบราวเซอร์ในครั้งต่อไป การที่จะนำข้อมูลจากแคชมาใช้งานนั้นจะมีการเปรียบแคชและสำเนาของแคชที่ได้ถูกสร้างขึ้นก่อนที่จะนำไปใช้งานทำให้สามารถทราบได้ว่ามีข้อผิดพลาดเกิดขึ้นกับแคชที่ถูกเก็บอยู่ในหน่วยความจำหรือไม่ เพื่อทำการแก้ไขและนำแคชที่ไม่มีความผิดพลาดไปใช้งานในครั้งต่อๆไป

โดยงานทั้งหมดเริ่มจากการศึกษาการแคชของหน้าไดนามิกเว็บของเว็บ โดยศึกษาในเรื่องของการพัฒนาส่วนขยายของ PHP (PHP Extension) และลักษณะการทำงานของเว็บเซิร์ฟเวอร์ จากนั้นศึกษาลักษณะรูปแบบการเกิดความผิดพร่องของแคชที่ถูกเก็บไว้ในหน่วยความจำ เพื่อให้ทราบถึงข้อผิดพลาดที่จะเกิดขึ้นอีกทั้งยังได้ศึกษาวิธีของการทดสอบแคชหน้าเว็บไดนามิกที่กำหนดขึ้นด้วยการฉีดความผิดพร่อง (Fault Injection) ลงไปในข้อมูลแคชที่ได้ถูกเก็บไว้ในหน่วยความจำเพื่อทดสอบความทนทานต่อความผิดพร่องของแคชในไดนามิกเว็บ สุดท้ายคือการนำไดนามิกเว็บแคชที่ได้รับการปรับปรุง จากที่กล่าวมาแล้วข้างต้นมาทำการทดสอบประสิทธิภาพกับหน้าเว็บที่ได้สร้างขึ้นเองและหน้าเว็บของเว็บแอปพลิเคชันที่ได้มีการใช้งานกันอย่างแพร่หลาย และทำการเปรียบเทียบประสิทธิภาพกับการใช้ไดนามิกเว็บแคชที่ยังไม่ได้รับการปรับปรุง และก็ยังเปรียบเทียบประสิทธิภาพกับหน้าเว็บไดนามิกที่ไม่ใช้ไดนามิกเว็บแคชด้วย ทั้งในสภาวะที่ปกติและในสภาวะของการเกิดความผิดพร่อง โดยการฉีดความผิดพร่องลงไปในแคชที่ถูกเก็บอยู่ในหน่วยความจำ แล้วบันทึกผล และวิเคราะห์ข้อมูลจากนั้นก็สรุปผล

ในบทที่ 2 จะกล่าวถึงเทคโนโลยีที่ใช้ในการสร้างไดนามิกเว็บและเว็บแคช โดยจะเริ่มกล่าวถึงสแตติกเว็บและไดนามิกเว็บ โดยภาษาที่ใช้ในการสร้างไดนามิกเว็บคือภาษา PHP จากนั้น

จะกล่าวรายละเอียดของเทคโนโลยีที่ใช้ในการสร้างไดนามิกเว็บแคชที่ผู้วิจัยใช้ ซึ่งเป็นส่วนขยายของ PHP

ในบทที่ 3 จะกล่าวถึงความผิดพร่องที่มีผลกับไดนามิกเว็บ ซึ่งความปัจจัยที่ทำให้เกิดข้อผิดพลาดชั่วคราว และผลกระทบที่เกิดจากความผิดพลาดนั้นจะส่งผลไปถึงความพึ่งพิงได้ของเว็บ (Dependability) และยังจะกล่าวถึงเทคนิคที่ในการพัฒนาระบบที่ทนต่อความผิดพร่องเพื่อเป็นการป้องกันความผิดที่จะเกิดขึ้น

ในบทที่ 4 เป็นกระบวนการในการสร้างไดนามิกเว็บแคชที่มีการทนต่อความผิดพร่อง ซึ่งจะกล่าวถึงเทคนิคและวิธีการตรวจสอบความถูกต้อง โดยการใช้การเปรียบเทียบแคชกับสำเนาของแคช

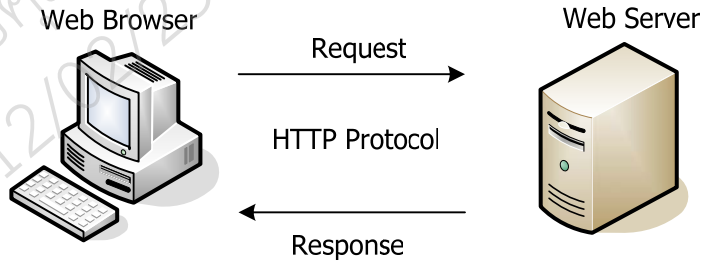
โดยผลที่ได้จากการทดสอบแสดงให้เห็นได้ว่าไดนามิกเว็บแคชที่ได้รับการปรับปรุงแล้วนั้นมีความสามารถทนต่อความผิดพร่องได้ดีกว่าแคชที่ยังไม่ได้ปรับปรุง โดยถึงแม้ว่าไดนามิกเว็บแคชที่ได้ทำการแก้ไขแล้วนั้น ทำให้ช่วงเวลาสำหรับการประมวลผลเพิ่มขึ้นมากกว่าไดนามิกเว็บแคชที่ยังไม่ได้รับการปรับปรุง แต่ก็เป็นช่วงเวลาที่ดีกว่าไม่ได้ใช้ไดนามิกเว็บแคชเลย ผู้วิจัยได้แสดงผลการทดสอบในบทที่ 5 และในบทสุดท้าย (บทที่ 6) จะกล่าวถึงการวิเคราะห์และสรุปผลการทดลองรวมถึงงานในทิศทางของอนาคตซึ่งคาดว่าจะเป็นประโยชน์สำหรับผู้ที่ต้องการจะนำงานชิ้นนี้ไปวิจัยและพัฒนาต่อไป

บทที่ 2
เทคโนโลยีที่ใช้ในการสร้างไดนามิกเว็บและเว็บแคช

บทนี้จะเป็นบทที่กล่าวถึงเทคโนโลยีที่ใช้ในการสร้างไดนามิกเว็บและเว็บแคช โดยจะกล่าวถึงหลักการของสแตติกเว็บและไดนามิกเว็บ ซึ่งหน้าเว็บไดนามิกถูกสร้างจากสคริปต์ PHP และจะกล่าวถึงเทคโนโลยีการแคชไดนามิกเว็บสำหรับ PHP ซึ่งเป็นส่วนขยาย PHP (PHP Extension) การพัฒนาส่วนขยาย PHP เป็นการเพิ่มความสามารถให้กับ PHP แล้วแต่ความต้องการของผู้ใช้งาน สำหรับส่วนขยาย PHP ที่ใช้ในวิทยานิพนธ์นี้คือ eAccelerator (eA) เป็นโปรแกรมที่ผู้วิจัยนำมาพัฒนาเพื่อให้แคชมีความทนต่อความผิดพร่อง eAccelerator เป็นโปรแกรมหนึ่งที่จะเพิ่มความเร็วในประมวลผลสคริปต์ PHP โดยการแคชสคริปต์ที่ได้ทำการประมวลผลแล้ว

2.1 สแตติกและไดนามิกเว็บ

บางคนอาจเข้าใจว่าอินเทอร์เน็ตกับเว็บคือสิ่งเดียวกัน แต่ที่แท้จริงแล้วเว็บเป็นเพียงบริการหนึ่งของอินเทอร์เน็ตเท่านั้น อินเทอร์เน็ตยังมีบริการอื่นๆอีกด้วย เช่น จดหมายอิเล็กทรอนิกส์ (E-mail) ห้องคุย (Chat Room) และอื่นๆ อีกมากมาย



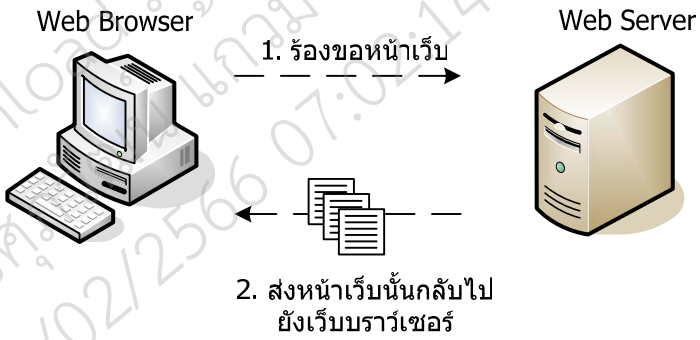
ภาพที่ 2-1 การติดต่อสื่อสารระหว่างเว็บบราวเซอร์กับเว็บเซิร์ฟเวอร์

2.1.1 สแตติกเว็บ (Static Web) : การทำงานของสแตติกเว็บจะมีลักษณะที่อยู่ในรูปแบบไคลเอนต์เซิร์ฟเวอร์ โดยมีโปรแกรมเว็บไคลเอนต์ทำหน้าที่เป็นผู้ร้องขอหน้าเว็บและมีโปรแกรมเว็บเซิร์ฟเวอร์หรือบางครั้งถูกเรียกว่า HTTP เซิร์ฟเวอร์ทำหน้าที่เป็นผู้ให้บริการ สำหรับโปรแกรมเว็บไคลเอนต์ก็คือโปรแกรมเว็บบราวเซอร์ในเครื่องของผู้ใช้นั้นเอง ส่วนโปรแกรมเว็บเซิร์ฟเวอร์

นั้นจะถูกติดตั้งไว้ในเครื่องของผู้ให้บริการ การติดต่อระหว่างโปรแกรมเว็บบราวเซอร์และเว็บเซิร์ฟเวอร์จะกระทำผ่านโปรโตคอล HTTP ดังแสดงในภาพที่ 2-1

โดยลักษณะเนื้อหาของหน้าเว็บที่เป็นแบบสแตติกนั้น หน้าเว็บจะมีเนื้อหาแบบสถิต โดยทั่วไปแล้วจะมีนามสกุลไฟล์เป็น htm หรือ html เมื่อใช้เว็บบราวเซอร์เปิดดูหน้าเว็บใดเว็บหนึ่ง เว็บเซิร์ฟเวอร์จะส่งหน้าเว็บนั้นกลับมายังเว็บบราวเซอร์ จากนั้นเว็บบราวเซอร์ก็จะแสดงผลไปตามคำสั่งภาษา HTML ที่อยู่ในไฟล์

จะเห็นได้ว่าหน้าเว็บดังภาพที่ 2-2 เป็นหน้าเว็บที่มีลักษณะเนื้อหาเป็นแบบสถิต กล่าวคือผู้ใช้จะพบกับหน้าเว็บเดิมทุกครั้งจนกว่าผู้ดูแลเว็บจะทำการปรับปรุงหน้าเว็บนั้นๆ ข้อจำกัดนี้มีสาเหตุมาจากภาษา HTML ซึ่งเป็นภาษาที่ใช้อธิบายหน้าตาของหน้าเว็บ HTML จัดเป็นภาษาที่อยู่ในกลุ่มที่เรียกว่าภาษาลักษณะหน้า (Page Description Language) หรือกล่าวอีกนัยหนึ่งคือ HTML สามารถกำหนดให้หน้าเว็บมีหน้าตาอย่างที่ผู้ให้บริการต้องการได้ แต่ถ้าจะให้หน้าเว็บมีความสามารถในการแสดงผลหรือโต้ตอบกับผู้ใช้มากขึ้น จึงต้องสร้างหน้าเว็บที่สามารถทำให้แสดงข้อมูลพลวัต (Dynamic Content)



ภาพที่ 2-2 การทำงานของหน้าเว็บแบบสแตติก

2.1.2 ไดนามิกเว็บ (Dynamic Web) : หน้าเว็บแบบไดนามิกคือหน้าเว็บที่สามารถแสดงรายละเอียดพลวัตได้โดยสามารถทำได้หลายวิธี หนึ่งในวิธีการนั้นคือการฝังสคริปต์หรือชุดคำสั่งที่ทำงานทางฝั่งเซิร์ฟเวอร์ไว้ในหน้าเว็บ ไดนามิกเว็บโดยส่วนมากอยู่ในรูปแบบของเว็บแอปพลิเคชัน ไดนามิกเว็บนั้นหมายถึงเว็บที่มีเนื้อหาเปลี่ยนไปได้ในแต่ละครั้งที่ผู้ใช้งานเปิดดู ซึ่งขึ้นอยู่กับเงื่อนไขต่างๆของโปรแกรม หรืออาจเปลี่ยนแปลงไปตามข้อมูลต่างๆของผู้ใช้งานที่เก็บอยู่ในฐานข้อมูล เว็บแอปพลิเคชันนั้นเป็นโปรแกรมประยุกต์ซึ่งสามารถทำงานอยู่บนฮาร์ดแวร์ที่แตกต่างกันและระบบปฏิบัติการที่แตกต่างกันได้ด้วย ทั้งนี้ยังสามารถทำงานรวมกันในหลายๆภาษา สำหรับ

ภาษาสคริปต์ที่สามารถประมวลทางฝั่งเซิร์ฟเวอร์ก็มีหลายภาษา เช่น JSP, ASP หรือ PHP ซึ่งเป็นเทคโนโลยีที่สคริปต์ทำการประมวลผลที่ฝั่งเซิร์ฟเวอร์ สำหรับภาษาสคริปต์ที่ผู้วิจัยใช้ในวิทยานิพนธ์นี้คือ PHP ซึ่งจะกล่าวถึงในหัวข้อต่อไป

2.2 ภาษา PHP

ประวัติของภาษา PHP มาจากนาย Rasmus Lerdorf ได้สร้างภาษา PHP ขึ้นมาในปี 1994 เนื่องจากเขาต้องการพัฒนาโปรแกรมเพื่อเก็บข้อมูลของผู้ใช้ที่แวะเวียนเข้ามาเยี่ยมชมเว็บไซต์ของเขา เขาเรียกโปรแกรมนี้ว่า PHP ซึ่งย่อมาจาก **P**ersonal **H**ome **P**age ในเวอร์ชันแรกสุด PHP ยังไม่มีความสามารถอะไรมากนักโดยประกอบด้วยกลไกการแปลภาษาอย่างง่ายๆ

พอกลางปี 1995 ได้มีการพัฒนาตัวแปลภาษา PHP ขึ้นมาใหม่โดยใช้ชื่อว่า PHP/FI ออกมาเป็นเวอร์ชันที่สอง FI นั้นย่อมาจาก Form Interpreter ซึ่งได้เพิ่มความสามารถในการรับส่งข้อมูลจากฟอร์มของ HTML นอกจากนั้นยังเพิ่มความสามารถในการติดต่อกับฐานข้อมูลอีกด้วย

ในปี 1997 มีผู้ร่วมพัฒนา PHP เพิ่มอีก 2 คน คือ **Ze**ev Suraski และ A**nd**i Gutmans เป็นกลุ่มที่เรียกตัวเองว่า Zend ซึ่งมาจาก **Ze**ev และ A**nd**i โดยแก้ไขข้อบกพร่องต่างๆ และเพิ่มเครื่องมือของ PHP ให้เพิ่มมากขึ้นจนกลายเป็น PHP ในเวอร์ชัน 3 และพัฒนาต่อมาเรื่อยๆจนถึงเวอร์ชัน 4 และ 5 ที่ใช้งานกันอยู่ในปัจจุบัน

ปัจจุบันกลุ่มผู้พัฒนา PHP ได้กำหนดให้ PHP นั้นย่อมาจาก **P**HP **H**ypertext **P**reprocessor ซึ่งเป็นคำย่อในลักษณะเรียกซ้ำเพราะชื่อเต็มของ PHP ก็ยังคงมีตัวอักษรย่อ PHP ปรากฏอยู่

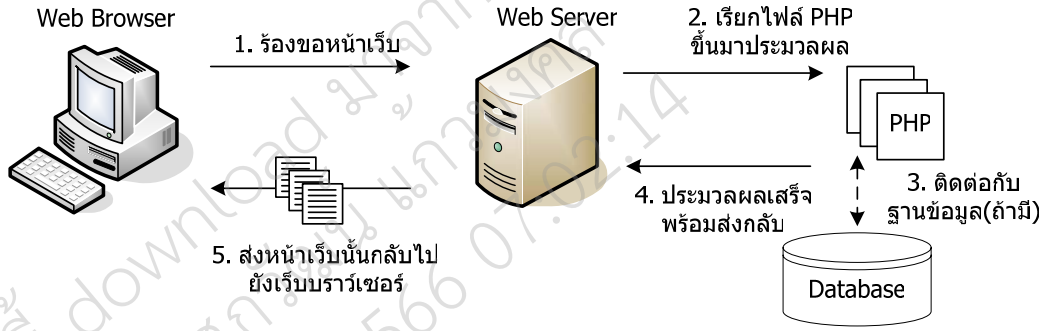
2.2.1 ลักษณะของภาษา PHP (Characteristic of PHP) : ภาษา PHP เป็นภาษาสคริปต์ที่ทำงานทางฝั่งเซิร์ฟเวอร์ ซึ่งมีลักษณะเป็นสคริปต์ที่ถูกฝัง (Embedded Script) หมายความว่าผู้พัฒนาสามารถฝังคำสั่ง PHP ไว้ในหน้าเว็บร่วมกับคำสั่งของ HTML ได้ดังที่แสดงอยู่ในตัวอย่างที่ 2-1

ตัวอย่างที่ 2-1

```
<HTML>
    <HEAD>
        <TITLE> Hello </TITLE>
    </HEAD>
    <BODY>
        <?php
            echo "Hello, world";
        ?>
    </BODY>
</HTML>
```

บรรทัดที่อยู่ระหว่างแท็ก <?php กับ ?> คือคำสั่ง PHP ซึ่งในที่นี้เป็นการสั่งให้ส่งข้อความ Hello, world กลับไปยังเว็บบราวเซอร์ การฝังคำสั่งของ PHP ไว้ในหน้าเว็บนั้นจะต้องบรรจุคำสั่งที่ต้องการไว้ภายในแท็ก <?php ?> ดังเช่นตัวอย่างที่ 2-1

ภาพที่ 2-3 แสดงการทำงานเมื่อมีการร้องขอหน้าเว็บแบบไดนามิกโดยใช้ PHP เมื่อเว็บบราวเซอร์ร้องขอ PHP ไฟล์สคริปต์ใดๆก็ตาม เว็บเซิร์ฟเวอร์จะเรียกเครื่องประมวลผล PHP (PHP Engine) ขึ้นมาทำการแปล (Interpret) และประมวลผลคำสั่งที่อยู่ในไฟล์ PHP ดังกล่าว โดยอาจมีการดึงข้อมูลจากฐานข้อมูลหรือเขียนข้อมูลลงไปยังฐานข้อมูลด้วย หลังจากนั้นผลลัพธ์ในรูปแบบของ HTML และสคริปต์ที่ทำงานทางฝั่งเว็บบราวเซอร์ เช่น จาวาสคริปต์จะถูกส่งไปยังเว็บบราวเซอร์ เว็บบราวเซอร์ก็จะแสดงผลตามคำสั่ง HTML ที่ได้รับมา โดยที่ไม่มีคำสั่งของ PHP ใดๆหลงเหลืออยู่เลย เนื่องจากถูกแปลและประมวลผลโดยเครื่องประมวลผล PHP ที่ฝั่งเซิร์ฟเวอร์ไปหมดแล้ว



ภาพที่ 2-3 การทำงานของหน้าเว็บแบบไดนามิกโดยใช้ PHP

การทำงานของเว็บบราวเซอร์ในกรณีนี้ไม่มีความแตกต่างจากกรณีของหน้าเว็บแบบสแตติกที่ได้กล่าวถึงไปก่อนหน้านี้ เพราะสิ่งที่เว็บบราวเซอร์ต้องกระทำคือการร้องขอหน้าเว็บจากเว็บเซิร์ฟเวอร์แล้วก็รอผลลัพธ์กลับมาแสดงผล แต่ความแตกต่างนั้นอยู่ที่การทำงานทางฝั่งเซิร์ฟเวอร์ในขั้นตอนที่ไฟล์หน้าเว็บนั้นมีการประมวลผลก่อนที่จะถูกส่งกลับไปยังเว็บบราวเซอร์

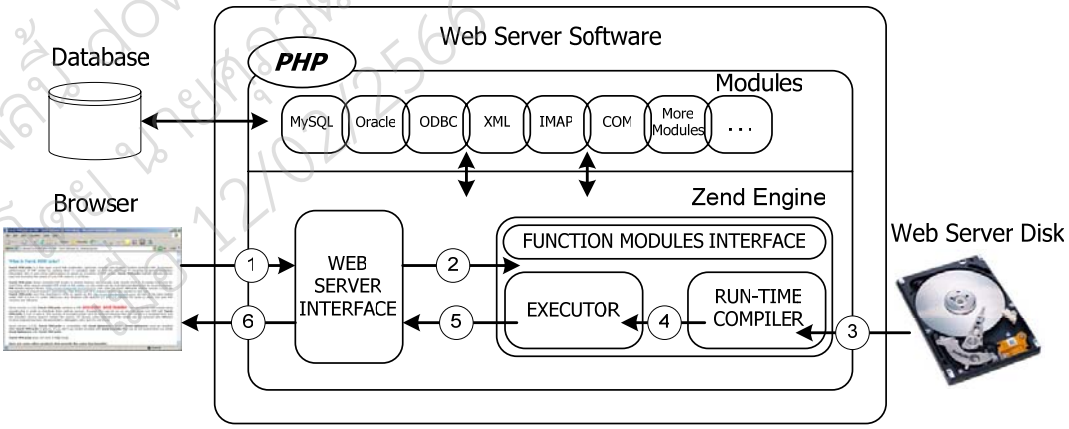
การฝังสคริปต์ PHP ไว้ในหน้าเว็บจะช่วยให้สามารถสร้างหน้าเว็บที่มีเนื้อหาแบบพลวัตได้ ซึ่งหมายถึงหน้าเว็บที่มีเนื้อหาสาระ และหน้าตาของเว็บเปลี่ยนแปลงไปได้ในแต่ละครั้งที่ผู้ใช้เปิดดู โดยขึ้นอยู่กับเงื่อนไขต่างๆ เช่น ข้อมูลที่ผู้ใช้ส่งมาให้ทางเว็บบราวเซอร์หรือข้อมูลที่อยู่ในฐานข้อมูล

2.2.2 โครงสร้างภายในของ PHP (PHP Internal) : ส่วนประกอบของโครงสร้างภายใน PHP นั้นมีอยู่ด้วยกันสามส่วนหลักดังที่แสดงในภาพที่ 2-4

ส่วนแรกคือเซ็นเอ็นจิ้น (Zend Engine) นั้นคือแกนของ PHP ซึ่งทำหน้าที่แจงส่วน (Parse) และ คอมไพล์ (Compile) จากภาษาสคริปต์ไปเป็นภาษาเครื่องในลักษณะของไบท์โค้ด (Bytecode) เซ็นเอ็นจิ้นนั้นจะทำการ Execute ไบท์โค้ดบนเครื่องจักรเสมือน (Virtual Machine) เซ็นเอ็นจิ้นยังควบคุมในส่วนของการอ่านและเขียนตัวแปรต่างๆ รวมถึงจัดการลำดับขั้นตอนของโปรแกรมแล้วยังควบคุมส่วนประกอบต่างๆภายในเช่นส่วนจำเพาะ (Module) หรือที่เรียกว่าส่วนขยาย เซ็นเอ็นจิ้นนั้นยังจัดการเกี่ยวกับทรัพยากรหน่วยความจำสำหรับสภาวะแวดล้อม PHP อีกด้วย

ส่วนที่สองที่อยู่เหนือเซ็นเอ็นจิ้นดังที่แสดงในภาพที่ 2-4 คือส่วนจำเพาะ (Module) หรือที่เรียกว่าส่วนขยาย PHP โดยที่ส่วนขยายนั้นยังแบ่งได้ออกเป็นสองประเภท [3] คือส่วนขยายที่เป็นส่วนประกอบติดมา (Built-in Modules) กับ PHP โดยเป็นค่าปริยาย (Default Value) ถ้าต้องการปรับแต่งค่าเช่นการใช้คำสั่ง `./configure --with-mysql` และส่วนขยายที่เพิ่มเข้ามา (Shared Modules) จะต้องปรับแต่งที่ไฟล์ php.ini

ส่วนสุดท้ายคือส่วนที่ติดต่อกับเว็บเซิร์ฟเวอร์ (Web Server Interface) หรือที่เรียกกันสั้นๆว่า Server API (SAPI) เป็นส่วนที่ควบคุมการติดต่อกันระหว่างเซ็นเอ็นจิ้นกับเว็บเซิร์ฟเวอร์เช่น Apache เว็บเซิร์ฟเวอร์ที่ใช้ในวิทยานิพนธ์นี้คือ Apache



ภาพที่ 2-4 โครงสร้างภายในของ PHP

2.2.3 ขั้นตอนการทำงานของ PHP (Workflow of PHP) : ขั้นตอนการทำงานของ PHP ที่แสดงในภาพที่ 2-4 มีดังต่อไปนี้

2.2.3.1 เมื่อมีการร้องขอหน้าเว็บเข้ามาจากเว็บบราวเซอร์ เว็บเซิร์ฟเวอร์จะทำการติดต่อกับ SAPI ของ PHP

2.2.3.2 จากนั้นจะส่งการร้องขอต่อมาที่เซ็นเอ็นจิ้น

2.2.3.3 เซ็นเอ็นจิ้นจะทำการอ่านสคริปต์ PHP มาทำการคอมไพล์

2.2.3.4 เมื่อทำการคอมไพล์เสร็จแล้วส่งไปทำการ Execute ต่อไป

2.2.3.5 เมื่อทำการ Execute เรียบร้อยแล้วจะทำการติดต่อกลับไปยัง SAPI เพื่อทำการส่งต่อข้อมูลที่ได้จากการประมวลผลแล้วส่งไปที่เว็บเซิร์ฟเวอร์

2.2.3.6 เว็บเซิร์ฟเวอร์ก็จะส่งข้อมูลหน้าเว็บที่ได้ทำการร้องขอมากลับไปยังเว็บบราวเซอร์ เพื่อนำไปแสดงผล

2.3 เทคโนโลยีที่ใช้ในการแคชไดนามิกเว็บ

2.3.1 การแคชไดนามิกเว็บ (Dynamic Web Caching) : สาเหตุที่ต้องมีการแคชไดนามิกเว็บเพราะว่าหน้าไดนามิกเว็บนั้นทำให้เว็บเซิร์ฟเวอร์มีประสิทธิภาพลดลง เนื่องจากมีการประมวลผลสคริปต์ทางฝั่งเซิร์ฟเวอร์ วิธีการอย่างหนึ่งที่ใช้ในการเพิ่มประสิทธิภาพของเว็บเซิร์ฟเวอร์คือการเก็บสคริปต์ทางฝั่งเซิร์ฟเวอร์ที่ได้ทำการประมวลผลแล้ว เพื่อนำมาใช้งานซ้ำในครั้งต่อไปโดยวิธีการนี้ถูกเรียกกันว่าเป็นการแคชไดนามิกเว็บ ซึ่งเป็นสิ่งที่ผู้วิจัยนำมาประยุกต์ใช้ในวิทยานิพนธ์นี้ โดยวัตถุประสงค์ของการแคชไดนามิกเว็บ คือการลดภาระการประมวลผลของเว็บเซิร์ฟเวอร์จากการที่ต้องสร้างหน้าเว็บไดนามิก การแคชไดนามิกเว็บสำหรับ PHP นั้นต้องทำการพัฒนาส่วนขยาย PHP (PHP Extension) เพื่อเป็นเพิ่มความสามารถให้กับ PHP สำหรับสคริปต์ PHP ที่จะทำการแคชสามารถทำได้โดยการพัฒนาโปรแกรมประยุกต์ในรูปแบบของส่วนขยาย PHP จะทำการกล่าวในหัวข้อต่อไป

2.3.2 ส่วนขยาย PHP (PHP Extension) : เหตุผลหลักๆสี่อย่างที่ต้องพัฒนาส่วนขยาย PHP เหตุผลแรกคือเพื่อทำเชื่อมต่อกับไลบรารีภายนอกที่แสดงออกมาในรูปแบบของฟังก์ชัน API (Application Program Interface) ไปยังสคริปต์ PHP ซึ่งจะเห็นได้จากส่วนขยาย MYSQL เพื่อจะเชื่อมต่อกับไลบรารีของ MYSQL นั้นต้องใช้ฟังก์ชัน `mysql_*();` ที่เขียนอยู่บนสคริปต์ PHP เพื่อทำการเชื่อมต่อฐานข้อมูล

สำหรับเหตุผลที่สองคือต้องการทำงานภายในแบบเฉพาะเจาะจง เช่น การประกาศตัวแปรแบบ super globals ตัวอย่างเช่น การประกาศ `$_GET, $_POST, $_SERVER` ซึ่งไม่สามารถประกาศได้จากพื้นที่ใช้งาน (User Space) เนื่องจากปัญหาด้านความมั่นคง

เหตุผลที่สามคือต้องการให้การประมวลผลสคริปต์ PHP มีความเร็วมากขึ้นโดยการคอมไพล์ผ่านทางสภาวะแวดล้อมเครื่องจักรเสมือน ซึ่งสามารถทำให้การประมวลผลเร็วยิ่งขึ้น วิธีการนี้ถูกเรียกว่า Opcode Cache [4] หรือไบท์โค้ดแคช นั้นคือยอมให้ข้ามขั้นตอนของการคอมไพล์และไปทำการ Execute เช่นเดิมหลังจากที่ได้เคยถูกทำการคอมไพล์ไปแล้วจากการร้องขอหน้าเว็บในครั้งแรก

เหตุผลสุดท้ายคือถ้านักพัฒนาได้ทำการพัฒนาแอปพลิเคชันด้วย PHP แล้วต้องการจัดจำหน่ายโดยที่ไม่ต้องการเปิดเผยซอร์สโค้ดก็สามารถทำการเข้ารหัสได้ โดยการพัฒนาส่วนขยายหรือทำการแก้ไขส่วนขยาย PHP

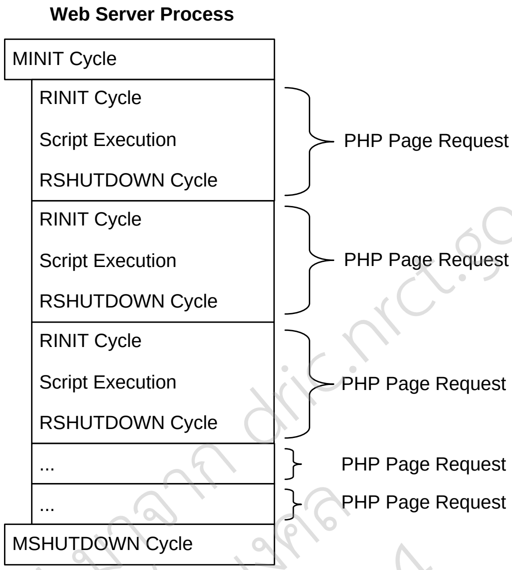
ส่วนในการที่จะติดต่อกับส่วนขยายนั้น สามารถทำได้โดยเมื่อมีการเข้ามาถึงในขั้นตอนที่ 2.2.3.2 ถึงขั้นตอนที่ 2.2.3.5 จากที่ได้กล่าวไปแล้วในหัวข้อที่ 2.2.3 โดยติดต่อผ่านทาง Zend API ซึ่งเซ็นเอ็นจิ้นได้มีการจัดเตรียมฟังก์ชัน API ไว้แล้วเพื่อที่จะทำการติดต่อกันระหว่างเซ็นเอ็นจิ้นกับส่วนขยาย อีกสิ่งหนึ่งที่ต้องทำความเข้าใจเกี่ยวกับการพัฒนาส่วนขยายของ PHP นอกจากโครงสร้างภายในแล้วกับขั้นตอนการทำงานของ PHP ที่ได้กล่าวไปแล้วยังต้องเข้าใจเกี่ยวกับวงจรชีวิต PHP ด้วย

วงจรชีวิต PHP (PHP Life Cycle) นั้นประกอบด้วยการเริ่มขึ้นและการจบลง (Starting Up and Shutting Down) [4] กระบวนการเริ่มขึ้นและการจบลงนี้ถูกแยกออกเป็นสี่ส่วนหลักๆ ดังที่แสดงไว้ในภาพที่ 2-5 วงรอบหนึ่งของการแปลคำสั่ง PHP นั้นจะถูกควบคุมด้วย SAPI

ระหว่างขั้นตอนของการเริ่มขึ้นก่อนที่จะมีการร้องขอหน้าเว็บ PHP ได้ทำการเรียกทุกๆส่วนขยาย ขั้นตอนนี้ถูกเรียกว่า MINIT (Module Initialization) ในขั้นตอนนี้ส่วนขยายจะมีการประกาศค่าคงที่ต่างๆและลงทะเบียนทรัพยากร เพื่อที่จะนำค่าหรือทรัพยากรต่างๆไปใช้เมื่อมีการร้องขอเข้ามาที่จะเกิดขึ้นในขั้นตอนต่อไป (RINIT) ซึ่งขั้นตอน MINIT จะเกิดขึ้นตอนที่มีการเริ่มขึ้นของเว็บเซิร์ฟเวอร์ เช่นเมื่อมีการใช้คำสั่ง `apacheclt start`

หลังจากนั้นเมื่อมีการร้องขอเข้ามา PHP จะมีการสร้างสภาวะแวดล้อมการทำงาน โดยการจองทรัพยากรและประกาศตัวแปรเริ่มต้นต่างๆที่ต้องใช้ในแต่ละครั้งของการร้องขอหน้าเว็บ ซึ่งในแต่ละครั้งที่มีการร้องข้อเข้ามาก็จะมีการเรียกวิธีการ RINIT (Request Inititialization)

หลังจากกระบวนการของการร้องขอเรียบร้อยแล้ว PHP จะมีการเริ่มกระบวนการล้าง (Cleanup) โดยการเรียกวิธีการ RSHUTDOWN (Request Shutdown) ของแต่ละส่วนขยาย เพื่อเป็นการล้างตัวแปรหรือทรัพยากรอื่นๆที่ถูกประกาศหรือจองในขั้นตอนของ RINIT



ภาพที่ 2-5 วงจรชีวิตของ PHP

ในขั้นตอนสุดท้ายในช่วงที่เว็บเซิร์ฟเวอร์หยุดการทำงาน ส่วนขยายนั้นจะมีการเรียกวิธีการ MSHUTDOWN (Module Shutdown) ในขั้นตอนนี้จะมีการคืนหน่วยความจำที่ได้ถูกจองไว้ในขั้นตอน MINIT

เมื่อผู้พัฒนาส่วนขยาย PHP มีเข้าใจโครงสร้างภายในและขั้นตอนการทำงานของ PHP รวมถึงวงจรชีวิตของ PHP แล้วจะทำให้สามารถทำการพัฒนาและปรับปรุงแก้ไขส่วนขยายที่ใช้ในการแคชหน้าไดนามิกเว็บสำหรับ PHP

2.3.3 ไดนามิกเว็บแคชสำหรับ PHP (Dynamic Web Cache for PHP) : โปรแกรมที่ใช้ในการแคช PHP นั้นมีหลายโปรแกรมเช่น Afterburner, Alternative PHP Cache (APC), PHPAccelerator, Zend Performance Suite, Turck MMCache และ eAccelerator (eA) ซึ่งในวิทยานิพนธ์นี้ผู้วิจัยได้เลือกโปรแกรม eAccelerator [5] เพราะเป็นโปรแกรมที่ไม่ต้องเสียค่าใช้จ่าย และเปิดซอร์สโค้ดแล้วยังมีการพัฒนาอยู่อย่างต่อเนื่อง

โปรแกรม eAccelerator ก็คือโปรแกรมที่ใช้ในการแคชสคริปต์ PHP เกิดขึ้นใน ปี ค.ศ. 2004 ซึ่งแยกตัวออกมาจากโครงการ Turck MMCache เพื่อทำการพัฒนาต่อหลังจากที่โครงการ Turck

MMCache ได้มีการหยุดพัฒนาไปแล้ว โครงการ Turck MMCache ถูกสร้างโดย Dmitry Stogov สำหรับซอร์สโค้ดที่อยู่ในโปรแกรม eAccelerator โดยส่วนมากก็ยังคงใช้ของเดิมเป็นหลัก ได้มีการทดสอบประสิทธิภาพของโปรแกรมต่างๆ [6] ที่ได้กล่าวมาในขั้นต้น ซึ่งเป็นเหตุผลหนึ่งที่ทำให้ผู้วิจัยเลือกใช้โปรแกรม eAccelerator ในวิทยานิพนธ์นี้

โปรแกรม eAccelerator คือตัวเร่งความเร็ว PHP ซึ่งเป็นโปรแกรมที่ไม่ต้องเสียค่าใช้จ่ายและเปิดซอร์สโค้ด ทั้งยังทำหน้าที่ออปติไมส์และแคชรายละเอียดไดนามิก ซึ่งเป็นการเพิ่มประสิทธิภาพของการประมวลผลสคริปต์ PHP โดยการแคชส่วนของสคริปต์ที่ได้ทำการคอมไพล์แล้ว ยิ่งไปกว่านั้นการออปติไมส์ยังช่วยให้ความเร็วในขั้นตอนของการ Execute มีความเร็วมากขึ้นด้วย eAccelerator คือสิ่งที่ช่วยให้ภาระของเว็บเซิร์ฟเวอร์ลดลงและช่วยเพิ่มความเร็วให้กับการประมวลผลซอร์สโค้ด PHP ได้มากถึง 1 ถึง 10 เท่า [5] ของการทำงานปกติ

โปรแกรม eAccelerator ที่ใช้ในวิทยานิพนธ์นี้เป็นเวอร์ชัน 0.9.4 ซึ่งโปรแกรม eAccelerator นั้นสามารถเก็บสคริปต์ที่ถูกคอมไพล์แล้วได้ในหน่วยความจำที่ถูกแบ่ง (Shared Memory) หรือเก็บไว้ในฮาร์ดดิส อย่างใดอย่างหนึ่งก็ได้ โปรแกรม eAccelerator นั้นยังสามารถ เข้ารหัสและถอดรหัสไฟล์ PHP ได้อีกด้วย แต่ในส่วนของวิทยานิพนธ์นี้ผู้วิจัยได้ทำการปรับปรุงแก้ไขโปรแกรม eAccelerator ในส่วนของการแคชที่ถูกเก็บไว้ในหน่วยความจำเท่านั้น

บทที่ 3
ความผิดพร่องที่มีผลกระทบกับไดนามิกเว็บ

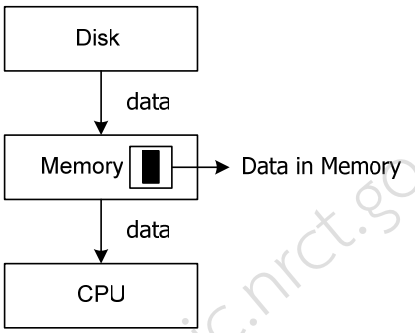
การทำงานของเว็บแอปพลิเคชันสามารถเกิดความผิดพร่องขึ้นได้ จากความไม่สมบูรณ์ทางกายภาพของฮาร์ดแวร์ เมื่อสคริปต์ทางฝั่งเซิร์ฟเวอร์ได้เริ่มทำการประมวลผลหลังจากที่มีการร้องข้อหน้าเว็บเข้ามา เซิร์ฟเวอร์จะมีการอ่านสคริปต์เข้ามาเพื่อทำการประมวลผล ข้อมูลจากสคริปต์ก็ต้องผ่านหน่วยความจำ (RAM) โดยส่วนใหญ่แล้วหน่วยความจำหลักที่ใช้ในระบบคอมพิวเตอร์คือหน่วยความจำเข้าถึงโดยสุ่มที่เรียกว่า RAM (Random Access Memory) เป็นหน่วยความจำชั่วคราวที่ใช้เก็บข้อมูลต่างๆ โดยหน่วยความจำ RAM มีอยู่ 2 ชนิดหลักๆ คือ

DRAM (Dynamic Random Access Memory) เมื่อมีการเริ่มต้นการใช้โปรแกรมหรือโปรแกรมจากดิสก์เก็ตหรือฮาร์ดดิสมาเก็บไว้ในคอมพิวเตอร์ ข้อมูลส่วนใหญ่จะเก็บไว้ใน DRAM เสมอ ซึ่ง DRAM เป็นหน่วยความจำที่มีความจุสูง โดยมีโครงสร้างภายในประกอบด้วยชั้น (Layer) ของตัวเก็บประจุ (Capacitor) ซึ่งมีคุณสมบัติในการเก็บประจุไฟฟ้า การบันทึกข้อมูลลงบน DRAM ก็คือการถ่ายประจุไฟฟ้าไปเก็บไว้ที่เซลล์ (Cell) ของตัวเก็บประจุบนตัว DRAM แต่การที่ภายใน DRAM ประกอบไปด้วยเซลล์ของตัวเก็บประจุนี้เอง ซึ่งจะมีปัญหาเกี่ยวกับการรั่วซึมอันเป็นลักษณะสมบัติของตัวเก็บประจุ ดังนั้นภายในระยะเวลาที่กำหนดหากไม่ได้มีการเติมประจุให้กับเซลล์ดังกล่าว หรือหากไฟดับข้อมูลก็จะเริ่มสูญหายไป ดังนั้นจึงต้องมีกระตุ้นการทำงานซ้ำใหม่เรื่อยๆ (Refresh)

SRAM (Static Random Access Memory) เป็นหน่วยความจำชนิดหนึ่งที่สามารถเก็บข้อมูลได้เหมือนกับ DRAM เมื่อมีแหล่งจ่ายพลังงานไฟฟ้ามาเลี้ยงให้กับหน่วยความจำ SRAM โดยที่โครงสร้างภายในจะเป็น Flip Flop มาเรียงต่อกัน ไม่จำเป็นต้องกระตุ้นการทำงานซ้ำใหม่เรื่อยๆ SRAM มีอัตราความเร็วมาก แต่มีราคาแพงกว่า DRAM โดยปกติแล้ว SRAM นิยมนำมาใช้ทำเป็นหน่วยความจำสำรอง (Cache Memory) เพราะมีความเร็วสูง

ซึ่งเมื่อมีการอ่านสคริปต์จากฮาร์ดดิสมาเก็บไว้ในหน่วยความจำจากนั้นก็ส่งไปประมวลผลที่หน่วยประมวลผล ดังภาพที่ 3-1 ความผิดพร่องนั้นสามารถที่จะเกิดขึ้นได้ในหน่วยความจำเช่น ความผิดพร่องคงอยู่ (Permanent Fault) หรือความผิดพร่องชั่วคราว (Transient Fault) และยังมีก็ความผิดพร่องบางครั้งคราว (Intermittent Fault) ความผิดพร่องที่เกิดขึ้นเหล่านี้ เป็นความผิดพร่องที่เกิดขึ้นในระดับฮาร์ดแวร์ซึ่งความผิดพร่องที่เกิดขึ้นกับข้อมูลในหน่วยความจำที่พบเห็นบ่อยที่สุดก็

คือ ความผิดพร่องที่เกิดขึ้นจากอนุภาคแอลฟา [7] ซึ่งเป็นความผิดพร่องที่ทำให้เกิดผิดพลาดชั่วคราว สามารถที่จะเกิดขึ้นได้กับ DRAM และ SRAM ความผิดพร่องที่เกิดขึ้นเหล่านี้ทำให้ข้อมูลที่เก็บไว้ในหน่วยความจำเกิดความผิดพลาด ซึ่งในวิทยานิพนธ์นี้มุ่งเน้นการแก้ความผิดพร่องที่ทำให้เกิดความผิดพลาดชั่วคราว



ภาพที่ 3-1 การประมวลผลแบบปกติ

เนื่องด้วยความต้องการของผู้ใช้งานเว็บ โดยส่วนมากแล้วต้องการใช้งานเว็บที่มีความน่าเชื่อถือ (Reliability) และไม่ต้องการให้มีความผิดพร่องเกิดขึ้น ซึ่งเป็นลักษณะของเว็บคุณภาพ (Quality of Web) เว็บคุณภาพนั้นต้องมีองค์ประกอบของความพึ่งพิงได้ (Dependability) ในการใช้งานเว็บถ้ามีความผิดพร่องจะส่งผลกระทบกับลักษณะของความพึ่งพิงได้ ซึ่งความพึ่งพิงได้ของเว็บมีองค์ประกอบต่างๆ ดังที่จะกล่าวในหัวข้อถัดไป

3.1 ความพึ่งพิงได้ของเว็บ (Dependability of Web)

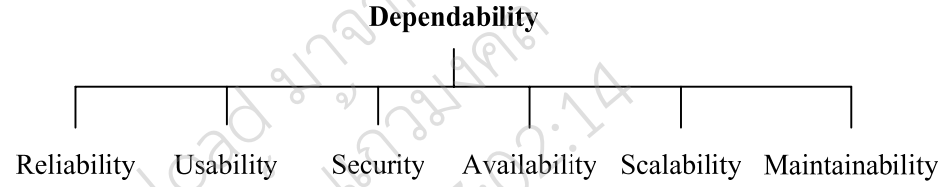
3.1.1 ความน่าเชื่อถือ (Reliability) : เว็บแอปพลิเคชันที่ต้องการความน่าเชื่อถือนั้นหมายถึงแอปพลิเคชันที่ต้องการความปลอดภัยสูง เช่นแอปพลิเคชันทางด้านการบินและการแพทย์ เนื่องจากถ้าเกิดความผิดพลาดขึ้นแล้วจะทำให้เกิดความเสียหายอย่างมาก

3.1.2 การใช้งานได้ (Usability) : คือเว็บแอปพลิเคชันที่สามารถใช้งานได้ง่าย โดยที่ผู้ใช้งานสามารถใช้งานได้อย่างถูกต้อง ไม่ยุ่งยาก ไม่มีความซับซ้อนในการใช้งาน ใช้งานได้สะดวก (User Friendly) ทั้งยังต้องสามารถใช้งานได้อย่างรวดเร็ว

3.1.3 ความมั่นคง (Security) : คือเว็บแอปพลิเคชันที่สามารถป้องกันการมุ่งร้ายในรูปแบบต่างๆจากการทำลายข้อมูล ขโมยข้อมูล หรือแก้ไขข้อมูลจากบุคคลภายนอกที่ไม่ได้รับสิทธิ์ในการเข้าถึงข้อมูล

3.1.4 สภาพพร้อมใช้งาน (Availability) : เว็บแอปพลิเคชันนั้นต้องการสภาพพร้อมใช้งานที่สูงมาก เพราะว่าผู้ใช้งานเว็บนั้นคาดหวังว่าจะเข้าไปใช้งานเว็บได้ตลอดเวลา เนื่องจากผู้ใช้งานเว็บแอปพลิเคชันนั้นสามารถเข้าถึงได้จากทั่วโลกไม่ว่าจากทวีปอเมริกาหรือทวีปเอเชีย เนื่องด้วยระยะเวลากลางวันกลางคืนที่แตกต่างกัน ไม่ว่าจะเป็นวันหยุดทางราชการหรือศาสนาที่แตกต่างกันของแต่ละประเทศ เนื่องจากเหตุผลที่กล่าวไปแล้วสภาพพร้อมใช้งานข้อเว็บแอปพลิเคชันนั้นต้องใช้งานได้ตลอด 24 ชั่วโมงต่อวัน 7 วันต่อสัปดาห์ และ 365 วันต่อปี [1] สภาพพร้อมใช้งานของเว็บนั้นยังมีความเกี่ยวข้องกับเว็บบราวเซอร์ เนื่องจากผู้ใช้งานอาจจะใช้เว็บบราวเซอร์ที่แตกต่างกัน เว็บแอปพลิเคชันนั้นต้องสามารถทำงานได้กับทุกเว็บบราวเซอร์

3.1.5 การปรับขนาดได้ (Scalability) : เว็บแอปพลิเคชันนั้นผู้ใช้งานเว็บสามารถเข้าถึงได้จากทั่วโลก การเปิดให้บริการในช่วงแรกนั้นอาจจะมีผู้เข้ามาใช้งานไม่มากแต่เมื่อผู้เพิ่มขึ้นมาเรื่อยๆในระยะเวลาต่อมาเว็บแอปพลิเคชันนั้นต้องสามารถรองรับผู้ใช้บริการจำนวนมากได้



ภาพที่ 3-2 ส่วนประกอบของความพึ่งพิงได้

3.1.6 สภาพบำรุงรักษาได้ (Maintainability) : เว็บแอปพลิเคชันนั้นต้องสามารถบำรุงรักษาได้ตามเวลาที่ผู้พัฒนาได้กำหนดวงรอบของเวลาการบำรุงรักษาเอาไว้ไม่ว่าจะเป็นรอบเดือนหรือรอบปี ทั้งยังต้องปรับปรุงหรือแก้ไขได้อย่างรวดเร็วถ้าเป็นการแก้ไขเพียงเล็กน้อย เพราะเป็นแอปพลิเคชันที่อยู่บนระบบอินเทอร์เน็ต

ผลกระทบที่เกิดกับความพึ่งพิงได้ของเว็บเมื่อมีความผิดพร่องเกิดขึ้น อาจไม่ได้ส่งผลโดยตรงไปถึงความพึ่งพิงได้บางอย่างเช่น การปรับขนาดได้ แต่ส่งผลโดยตรงถึงความน่าเชื่อถือและสภาพพร้อมใช้งาน โดยความผิดพร่องที่ผู้วิจัยสนใจในวิทยานิพนธ์นี้คือความผิดพร่องที่ทำให้เกิดข้อผิดพลาดชั่วคราวในหน่วยความจำ ซึ่งเป็นการเพิ่มความน่าเชื่อถือทำค่าให้ความพึ่งพิงได้ของเว็บนั้นมีระดับที่สูงขึ้น

3.2 ข้อผิดพลาดชั่วคราวที่เกิดขึ้นในหน่วยความจำอิเล็กทรอนิกส์

ในอดีตคอมพิวเตอร์ที่เกิดความบกพร่องเล็กน้อยอาจจะยอมรับได้ เนื่องจากว่าคอมพิวเตอร์ยังไม่ได้ถูกนำไปใช้ในระบบที่ต้องการความน่าเชื่อมากนัก แต่ในปัจจุบันคอมพิวเตอร์นั้นต้องการความน่าเชื่อถืออย่างมากและมากขึ้น เนื่องจากคอมพิวเตอร์ได้ถูกนำไปใช้กับระบบที่ต้องการความน่าเชื่อถือ เช่น ระบบทางด้านการแพทย์และธนาคาร อีกทั้งผู้ใช้งานยังคาดหวังว่าระบบของพวกเขาจะทำงานได้อย่างถูกต้องและเหมาะสมความบกพร่องเล็กน้อยที่เกิดขึ้นไม่สามารถยอมรับได้ แต่มันก็ยังคงเกิดขึ้นอยู่ ยากที่จะคาดเดาการเกิดขึ้น ความผิดพร่องที่เกิดขึ้น เช่น การกลับบิต (Bit Flips) นั้นเป็นการเกิดข้อผิดพลาดชั่วคราว การเกิดขึ้นนั้นทำให้ข้อมูลที่อยู่ในหน่วยความจำผิดพลาดแต่เซลล์ของหน่วยความจำนั้นไม่เกิดความเสียหาย

ข้อผิดพลาดชั่วคราวในหน่วยความจำอิเล็กทรอนิกส์นั้น โดยส่วนมากจะเกิดได้จากอนุภาคแอลฟา (Alpha Particle) ที่ทำการแผ่รังสีจากวัสดุห่อหุ้มชิพ ซึ่งเป็นข้อผิดพลาดที่พบบ่อยที่สุด [7] โรงงานผลิตหน่วยความจำพยายามที่จะขจัดอนุภาคแอลฟาออกไป โดยมีการเปลี่ยนการออกแบบและเพิ่มการป้องกัน การทดสอบมาตรฐานได้มีการพัฒนาการวัดอัตราการเกิดขึ้นของการแตกตัวทางรังสีวิทยาและแก้ไขความต้านทานของชิพหน่วยความจำให้มีค่ามากยิ่งขึ้น เพื่อป้องกันการเกิดอนุภาคแอลฟาแต่ข้อผิดพลาดชั่วคราวที่เกิดจากอนุภาคแอลฟานั้นก็ยังมีเกิดขึ้นอยู่

3.2.1 ข้อผิดพลาดชั่วคราวที่เกิดขึ้นโดยอนุภาคแอลฟา (Alpha Particle, αP) : ข้อผิดพลาดชั่วคราวที่เกิดขึ้นโดยอนุภาคแอลฟานั้นมีความเกี่ยวข้องกับความน่าเชื่อถือของการออกแบบ RAM และการพัฒนาวงจรรวมที่มีความจุสูงมาก (VLSI) โดยเฉพาะอย่างยิ่งอุปกรณ์หน่วยความจำที่เป็นสารกึ่งตัวนำที่ถูกห่อหุ้มด้วยเซรามิค ในไดนามิกแรม (DRAM) ข้อมูลถูกเก็บอยู่นั้นคือการมีอยู่หรือไม่มีอยู่ของประจุพาหะส่วนน้อย (Minority Carrier) บนตัวเก็บประจุ ตัวอย่างเช่นอุปกรณ์หน่วยความจำ n-Channel MOS ประจุพาหะคืออิเล็กตรอนและตัวเก็บประจุคือเป็นบ่อศักย์ไฟฟ้า (Potential Well) ซิลิคอนชนิดพี อนุภาคแอลฟาถูกแผ่กระจายออกมาในระดับที่ไม่มากจากยูเรเนียม (Uranium) และทอเรียม (Thorium) ที่อยู่ในวัสดุห่อหุ้มโดยสามารถแทรกซึมพื้นผิวของแม่พิมพ์สารกึ่งตัวนำได้ เนื่องจากอนุภาคแอลฟานั้นสามารถทะลุผ่านเข้าสู่อุปกรณ์สารกึ่งตัวนำได้โดยที่อิเล็กตรอนนั้นหลุดออกจากโครงผลึกไปตามทิศทางของอนุภาคแอลฟา ถ้าจำนวนรวมของอิเล็กตรอนที่ถูกสะสม ที่เก็บอยู่ในบ่อศักย์ไฟฟ้าเกินกว่าจำนวนของอิเล็กตรอนที่แยกความต่างระหว่าง 1 กับ 0 ประจุอิเล็กตรอนถูกสะสมสามารถกลับจาก 1 ไปเป็น 0 นั้นก็คือข้อผิดพลาดชั่วคราวที่เกิดขึ้นที่เกิดขึ้นในอุปกรณ์หน่วยความจำ

แนวโน้มการเพิ่มความหนาแน่นของจำนวนชิพ ขนาดของอุปกรณ์ที่เล็กลง ทำให้หน่วยความจำมีความรู้สึกไวต่อข้อผิดพลาดชั่วคราวมากขึ้น ข้อผิดพลาดชั่วคราวนั้นไม่มีความ

เกี่ยวข้องกับความเสียหายที่คงอยู่ (Permanent Damage) ของอุปกรณ์และจะถูกลบให้หายไปในการเขียนข้อมูลครั้งต่อไป [2] เหตุที่ทำให้เกิดข้อผิดพลาดชั่วคราวนั้นได้แก่ อุณหภูมิ การรบกวนของแหล่งจ่ายพลังงาน (Power Supply Noise) รังสีคอสมิก (Cosmic Ray) และอนุภาคแอลฟาซึ่งสาเหตุที่เกิดขึ้นจากอนุภาคแอลฟานั้นเป็นสาเหตุที่เกิดข้อผิดพลาดชั่วคราวขึ้นโดยทั่วไปมากที่สุด [2]

อนุภาคแอลฟาคือกลุ่มประจุนิวเคลียสฮีเลียมประกอบด้วย 2 โปรตอน (Proton) และ 2 อิเล็กตรอน (Electron) [7] พวกมันหลุดออกมาจากกลุ่มของอะตอมหลักในระหว่างการสลายตัวทางรังสี (Radioactive Decay) จากนั้นจะทำปฏิกิริยากับอะตอมอื่น เนื่องจากประจุและพลังงานที่รุนแรง ระดับพลังงานของอนุภาคแอลฟาสามารถแผ่ออกมาได้ถึง 1-9 MeV อนุภาคแอลฟาจากการสลายตัวทางรังสีของ U-283 และ T-232 ของวัสดุห่อหุ้มสามารถแทรกซึมอุปกรณ์สารกึ่งตัวนำได้ลึกถึง 25 μm โดยประมาณ [7]

อนุภาคแอลฟาที่มาจากวัสดุห่อหุ้มสารกึ่งตัวนำเป็นเหตุให้มีการ เกิดขึ้นของคู่อิเล็กตรอนโฮล (Electron-hole Pair) พลังงานอนุภาคแอลฟาที่รุนแรงทะลุผ่านไปที่ อุปกรณ์ซิลิคอนสามารถสร้างคู่อิเล็กตรอนโฮลได้ถึง 2.5×10^{6} อิเล็กตรอนโฮล เป็นเวลาหลายพิโควินาที จำนวนของคู่อิเล็กตรอนโฮลที่ถูกสร้างขึ้นอยู่กับพลังงานที่อนุภาคแอลฟาแผ่กระจายออกมา

แนวโน้มโดยทั่วไปของข้อผิดพลาดที่เกิดขึ้นในหน่วยความจำนั้นคืออนุภาคแอลฟา ซึ่งเป็นปัญหาโดยส่วนใหญ่ที่ต้องมีการขจัดออกไป ส่วนอัตราในการเกิดข้อผิดพลาดชั่วคราวของ DRAM นั้นมีแนวโน้มลดลงมากกว่า SRAM ซึ่ง SRAM นั้นมีความรู้สึกไวต่อข้อผิดพลาดชั่วคราวมากกว่า DRAM แต่ทั้งนี้อัตราข้อผิดพลาดจากหน่วยความนำทั้งสองแบบไม่ว่าจะเป็น SRAM หรือ DRAM ถูกคาดการณ์ไว้ว่าจะมีแนวโน้มเพิ่มขึ้น [8]

ปัจจัยในการออกแบบหน่วยความจำนั้นก็ส่วนหนึ่ง ที่มีผลต่อการเกิดขึ้นของข้อผิดพลาดเช่น การออกแบบให้หน่วยความจำนั้นมีความซับซ้อนเพิ่มมากขึ้น การเพิ่มปริมาณความจุของจำนวนเซลล์หน่วยความจำ แรงดันที่ใช้งานของอุปกรณ์มีศักย์ที่ต่ำ ความเร็วของการทำงานที่สูงขึ้น ประจุที่เก็บอยู่ในเซลล์หน่วยความจำมีจำนวนน้อย สิ่งเหล่านี้เป็นเหตุให้หน่วยความจำเกิดความรู้สึกไวต่อข้อผิดพลาดมากขึ้น การทนต่อรังสีสามารถลดอัตราข้อผิดพลาดได้หลายวิธีการแต่วิธีการเหล่านั้นต้องมีต้นทุนที่เพิ่มขึ้น ประสิทธิภาพลดลง ใช้พลังงานมากขึ้นหรือต้องใช้พื้นที่เพิ่มขึ้น

3.2.2 อัตราการเกิดของข้อผิดพลาดชั่วคราว (Soft Error Rate, SER) : อัตราในการเกิดข้อผิดพลาดชั่วคราวนั้นคือจำนวนครั้งที่มีโอกาสจะเกิดข้อผิดพลาดมีหน่วยเป็น FIT (Failures In Time) 1 FIT เท่ากับโอกาสที่จะเกิดข้อผิดพลาด 1 ครั้งในหนึ่งพันล้านชั่วโมงต่อเมกะบิต (Mb)

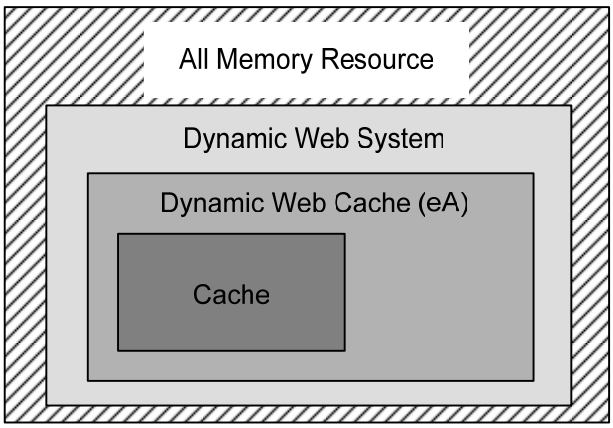
อัตราของการเกิดข้อผิดพลาดชั่วคราวนั้นผู้วิจัยได้มีการค้นคว้าจากเอกสารต่างๆ โดยพิจารณาจากรายงานอัตราข้อผิดพลาดชั่วคราวสำหรับอุปกรณ์หน่วยความจำ ซึ่งอัตราข้อผิดพลาดชั่วคราวของหน่วยความจำสมัยใหม่นั้นอยู่ในช่วง 1000 ถึง 5000 FIT ต่อ Mbit [8]

ข้อผิดพลาดชั่วคราวที่มีอัตราการเพิ่มขึ้นนั้น มีความเกี่ยวข้องกับการเพิ่มความจุของเซลล์หน่วยความจำและเทคโนโลยีที่ต้องการทำให้หน่วยความจำนั้นเล็กลง อัตราข้อผิดพลาดสามารถเกิดขึ้นได้ในอัตรา 500FIT/Mbit ในระบบที่ใช้หน่วยความจำขนาด 1 GByte สามารถที่จะคาดการได้ว่าจะมีข้อผิดพลาดเกิดขึ้นทุกๆสองสัปดาห์ต่อครั้ง [8] มีหลายเทคโนโลยีที่สามารถช่วยลดการเกิดข้อผิดพลาด แต่ก็อาจจะต้องแลกกับต้นทุนที่เพิ่มขึ้น ใช้พลังงานเพิ่มมากขึ้น ความเร็วที่ลดลง และขนาดของความจุของหน่วยจำที่ต้องเสียไป

การเกิดความผิดพร่องขึ้นในหน่วยความจำที่ได้กล่าวไปแล้วนั้นทำให้ข้อมูลแคชเกิดความผิดพลาดและอาจส่งผลทำให้การสร้างหน้าเว็บไดนามิกนั้นไม่ถูกต้อง ถ้าไดนามิกเว็บแคชมีการใช้ทรัพยากรหน่วยความจำมีขนาดที่ใหญ่ขึ้นอัตราของการเกิดข้อผิดพลาดก็จะมีมากขึ้นด้วย ซึ่งในหัวข้อต่อไปผู้วิจัยจะกล่าวถึงการใช้ทรัพยากรหน่วยความของไดนามิกเว็บแคช

3.3 การบริโภคทรัพยากรหน่วยความจำของไดนามิกเว็บแคช

การเกิดความผิดพร่องขึ้นในหน่วยความจำทำให้ข้อมูลแคชเกิดความผิดพลาดอาจส่งผลทำให้การสร้างหน้าเว็บไดนามิกนั้นไม่ถูกต้อง ทรัพยากรหน่วยความจำในคอมพิวเตอร์นั้นรองรับการทำงานของระบบต่างๆ ที่ต้องการเก็บข้อมูลในหน่วยความจำ ในวิทยานิพนธ์นี้เพ่งความสนใจไปที่ระบบของเว็บซึ่งเป็นไดนามิกเว็บและไดนามิกเว็บแคชที่มีอัตราการบริโภคหน่วยความจำเช่นเดียวกันกับแอปพลิเคชันอื่นๆ ดังที่แสดงในภาพที่ 3-3



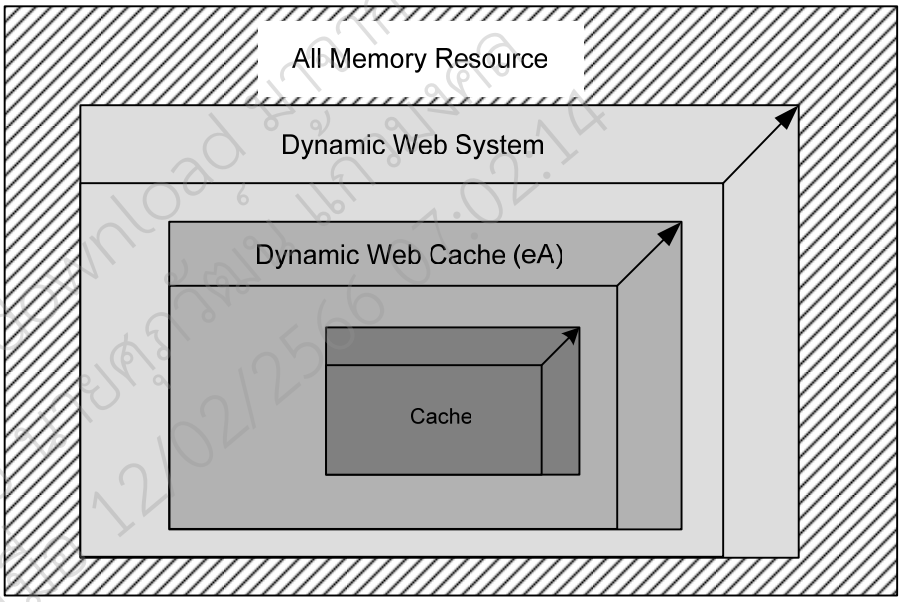
ภาพที่ 3-3 การใช้ทรัพยากรหน่วยความจำของระบบไดนามิกเว็บและไดนามิกเว็บแคช

ซึ่งระบบไดนามิกเว็บนั้นประกอบด้วยเว็บเซิร์ฟเวอร์ และภาษาสคริปต์ที่ทำให้ลักษณะของเนื้อหาของหน้าเว็บมีความเป็นพลวัต เว็บเซิร์ฟเวอร์และภาษาสคริปต์ที่ใช้ในวิทยานิพนธ์นี้คือ Apache และ PHP เว็บเซิร์ฟเวอร์ Apache สามารถทำงานร่วมกับ PHP ได้สองรูปแบบคือในลักษณะของ CGI และ Apache โมดูล ความแตกต่างอยู่ตรงที่ว่าถ้าใช้ PHP เป็นแบบโมดูล PHP จะเป็นส่วนหนึ่งของ Apache จึงทำให้โพสเซสของ Apache และ PHP จะถูกมองเห็นเป็นโพรเซสเดียวกันซึ่งจะทำงานได้เร็วกว่าแบบที่เป็น CGI เพราะว่า ถ้าเป็น CGI แล้วตัวแปลชุดคำสั่งของ PHP ถือว่าเป็นแค่โปรแกรมภายนอกซึ่ง Apache จะต้องเรียกขึ้นมาทำงานทุกครั้งที่ต้องการใช้ PHP ดังนั้นถ้ามองในเรื่องของประสิทธิภาพในการทำงานการใช้ PHP แบบที่เป็นโมดูลหนึ่งของ Apache จะทำงานได้มีประสิทธิภาพมากกว่า แต่เมื่อต้องการให้มีประสิทธิภาพที่มากยิ่งขึ้นจึงมีการใช้ไดนามิกเว็บแคชเข้ามาช่วยในการประมวลผล โดยการเก็บข้อมูลแคชที่ได้รับการประมวลผลแล้วเอาไว้ในหน่วยความจำเพื่อรอการเรียกใช้ซ้ำในครั้งต่อๆไป ซึ่งโปรแกรมไดนามิกเว็บแคชที่ใช้ในวิทยานิพนธ์นี้คือโปรแกรม eAccelerator (eA) ซึ่งเป็นส่วนขยายของ PHP โดยที่โปรแกรม eAccelerator จะทำงานร่วมกับ PHP ได้นั้นต้องใช้ PHP ในลักษณะของ Apache โมดูล

การบริโภคหน่วยความจำที่เพิ่มขึ้นของระบบไดนามิกเว็บมีอยู่หลายปัจจัย เช่น การร้องขอจำนวนมากที่เข้ามาพร้อมกันทำให้การบริโภคหน่วยความจำของ Apache โพรเซสจะเพิ่มขึ้นตามจำนวนของการถูกร้องขอจะสามารถสังเกตเห็นได้จากการส่งการร้องขอที่มีจำนวนมากเช่น การร้องขอ 1000 ครั้งต่อวินาทีของแต่ละการร้องขอไปยังเว็บเซิร์ฟเวอร์ Apache นั้นสามารถกำหนดจำนวนการร้องขอที่เข้ามาเป็นจำนวนมาก เมื่อเกินระดับที่กำหนดไว้ Apache จะทำการปิดโพรเซสที่มากกว่าที่กำหนดไว้เพื่อไม่ให้เกิดการขาดแคลนหน่วยความจำ (Memory Leak) การรองรับโพรเซสได้มากน้อยเพียงใดนั้นขึ้นอยู่กับประสิทธิภาพของเว็บเซิร์ฟเวอร์ด้วย แต่การบริโภคหน่วยความจำที่เกิดจากการร้องขอจำนวนมากนั้นไม่ได้เป็นปัจจัยโดยตรงที่ทำให้การบริโภคหน่วยความจำของไดนามิกเว็บแคชเพิ่มขึ้น แต่ปัจจัยโดยตรงที่ทำให้อัตราการบริโภคของไดนามิกเว็บแคชเพิ่มขึ้นนั้นคือขนาดของเว็บแอปพลิเคชันนั้น

สมมุติว่าเว็บแอปพลิเคชันมีจำนวนไฟล์สคริปต์อยู่ทั้งหมด 2000 ไฟล์ แต่ละไฟล์สคริปต์ใช้ทรัพยากรหน่วยความจำในการทำแคชเป็นจำนวน 10 KB ถ้าไฟล์สคริปต์ทั้งหมดถูกเรียกใช้งานขนาดของหน่วยความจำที่ต้องใช้ในการทำแคชจะมีขนาดโดยประมาณ 20 MB แต่เมื่อขนาดเว็บแอปพลิเคชันมีขนาดของจำนวนไฟล์สคริปต์ที่เพิ่มมากขึ้น ปริมาณการใช้ทรัพยากรหน่วยความจำในการทำแคชก็มากขึ้นด้วยตามลำดับ ซึ่งในการใช้ทรัพยากรหน่วยความจำในการทำข้อมูลแคชนั้นก็มีขนาดที่แตกต่างกันไปขึ้นอยู่กับความซับซ้อนของสคริปต์และขนาดของไฟล์สคริปต์ด้วย

ได้มีการทดสอบการบริโภคหน่วยความจำของ Apache [9] การใช้ทรัพยากรหน่วยความจำของ Apache โพรเซสระหว่างการร้องขอโดยเฉลี่ยนั้นอยู่ที่ประมาณ 5 MB ของหน่วยความจำทั้งหมด ส่วนไฟล์สคริปต์ [9] ที่ได้นำมาทำการทดสอบได้สร้างข้อมูลแคชจากการประมวลผลสคริปต์นั้นมีขนาดประมาณ 5 KB ถ้าคิดสัดส่วนโดยประมาณในการบริโภคหน่วยความจำของ Apache โพรเซสกับการทำแคชนั้นคือ 1000 : 1 อย่างไรตามขนาดของแคชที่อยู่ในหน่วยความจำนั้นยังขึ้นอยู่กับจำนวนของสคริปต์ ความซับซ้อนของสคริปต์และขนาดของสคริปต์ที่ได้ถูกทำการร้องขอเข้ามาด้วย เมื่อจำนวนของสคริปต์มีมากขึ้น หรือมีความซับซ้อนของสคริปต์มากขึ้น และขนาดของไฟล์สคริปต์มีขนาดใหญ่ขึ้นก็จะทำให้ทรัพยากรหน่วยความจำถูกใช้ในการทำแคชมากขึ้นดังที่แสดงในภาพที่ 3-4 ดังนั้นอัตราส่วนของการที่จะเกิดความผิดพร่องกับข้อมูลแคชที่เก็บไว้ในหน่วยความจำก็มีโอกาสที่จะเพิ่มมากขึ้น เพราะฉะนั้นก็ควรจะพัฒนาระบบที่ทนต่อความผิดพร่องซึ่งผู้วิจัยจะกล่าวในหัวข้อต่อไป



ภาพที่ 3-4 การขยายตัวของการใช้ทรัพยากรหน่วยความจำ

3.4 เทคนิคในการพัฒนาระบบที่ทนต่อความผิดพร่อง

ข้อผิดพลาดชั่วคราวนั้นสามารถเกิดขึ้นได้กับระบบหน่วยความจำที่มีขนาดใหญ่ไม่ว่าจะเป็นแอปพลิเคชันที่ต้องการความเชื่อมั่นสูงหรือในพื้นที่ที่มีความสูงมาก วิธีการที่ใช้ในการตรวจสอบและแก้ไขข้อผิดพลาดนั้นมีหลายวิธีการ เช่น การสำรองข้อมูลของแคชด้วยการเพิ่มส่วนที่ซ้ำซ้อนของข้อมูลในหน่วยความจำ ซึ่งเป็นวิธีการที่ผู้วิจัยได้ทำการใช้ในวิทยานิพนธ์นี้

3.4.1 ส่วนที่ซ้ำซ้อน (Redundancy) : วิธีการที่จะใช้เพื่อสร้างความทนทานต่อความผิดพร่องให้กับหน่วยความจำแคช นั้นก็คือการเพิ่มส่วนที่ซ้ำซ้อน (Redundancy) ซึ่งหมายถึงไดนามิกเว็บแคชใหม่ที่ได้รับการปรับปรุง และพร้อมที่จะส่งแคชออกไปใช้งานจะประกอบด้วยส่วนของแคชเดิมและส่วนแคชสำเนา (Duplicate Cache) ที่เพิ่มเติมเข้าไป ทั้งนี้เพื่อเป็นการสร้างความแข็งแกร่งให้กับข้อมูล และสามารถทนต่อความผิดพร่องได้โดยการเพิ่มส่วนที่ซ้ำซ้อนนั้นสามารถแยกประเภทได้ดังนี้

3.4.1.1 การเพิ่มความซ้ำซ้อนของอุปกรณ์ (Hardware Redundancy) : หมายถึงการเพิ่มส่วนของอุปกรณ์เพิ่มเติมเข้าไปเพื่อที่จะตรวจจับความผิดพร่อง หรือเพื่อที่จะสร้างความทนทานต่อความผิดพร่อง ตัวอย่างเช่น การมีส่วนของอุปกรณ์สำรองในระบบหรือมีอุปกรณ์ที่ทำงานประเภทเดียวกันมากกว่าหนึ่งชุด เพื่อทำการเปรียบเทียบผลการทำงาน เช่น TMR (Triple Modular Redundancy) โดย TMR จะมีหลักการทำงานคือ จะมีการรับอินพุตเข้ามาประมวลผลถึงสามหน่วย โดยผลการประมวลผลของแต่ละหน่วยจะถูกคัดเลือก และเอาท์พุตได้จากสองในสามของการประมวลผลจากอินพุตที่เหมือนกัน หรือ NMR (N-Modular Redundancy) โดยผลการประมวลผลของแต่ละหน่วยจะถูกคัดเลือก และเอาท์พุตได้จากจำนวนมากของการประมวลผลจากอินพุตที่เหมือนกัน

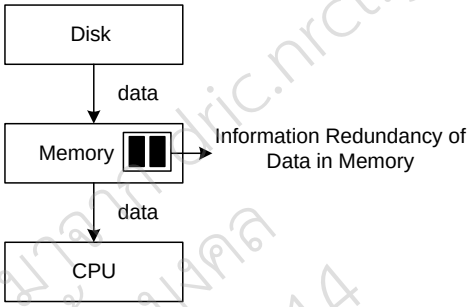
3.4.1.2 การเพิ่มความซ้ำซ้อนของซอฟต์แวร์ (Software Redundancy) : หมายถึงการเพิ่มส่วนของซอฟต์แวร์เพิ่มเติมเข้าไปเพื่อที่จะตรวจจับความผิดพร่อง หรือเพื่อที่จะสร้างความทนทานต่อความผิดพร่อง ตัวอย่างเช่น การเพิ่มส่วนของโปรแกรมโมดูล เพื่อทำการประมวลและนำผลของการประมวลผลมาเปรียบเทียบหรือแม้แต่เพิ่มเติมโปรแกรมที่มีคุณสมบัติเหมือนโปรแกรมเดิม แต่ออกแบบให้มีการประมวลผลที่แตกต่างกันออกไป เพื่อนำผลของการประมวลมาเปรียบเทียบต่อไป

3.4.1.3 การเพิ่มความซ้ำซ้อนของข้อมูลข่าวสาร (Information Redundancy) : หมายถึงการเพิ่มส่วนของข้อมูลข่าวสารเพิ่มเติมเข้าไปเพื่อที่จะตรวจจับความผิดพร่อง หรือเพื่อที่จะสร้างความทนทานต่อความผิดพร่อง ซึ่งการเพิ่มความซ้ำซ้อนของข้อมูลข่าวสารนั้นเป็นที่นิยมมากเช่น Checksum, CRC Check หรือ Duplication Code [10] โดยวิทยานิพนธ์นี้ได้นำหลักการของ Duplication Code มาเพิ่มความซ้ำซ้อนของข้อมูลข่าวสารเข้ามาประยุกต์ใช้ได้อย่างมีประสิทธิภาพ

3.4.1.4 การเพิ่มความซ้ำซ้อนของเวลา (Time Redundancy) : หมายถึงการเพิ่มส่วนของเวลาในการประมวลผลเข้าไปเพื่อที่จะตรวจจับความผิดพร่อง หรือเพื่อที่จะสร้างความทนทานต่อความผิดพร่อง ซึ่งจะเหมาะสมกับระบบการประมวลผลซึ่งสามารถที่จะใช้เวลาในการ

ประมวลผลที่เพิ่มขึ้นมากกว่าปกติได้ ตัวอย่างเช่น การประมวลผลซ้ำๆแล้วเปรียบเทียบผลของการประมวล

ความผิดพร่องที่เกิดขึ้นนั้นทำให้เกิดการกลับบิต (Bit Flip) [8] ซึ่งในบางกรณีจากการเกิดความผิดพลาดของข้อมูลเป็นจำนวนมากซึ่งมากกว่าวิธีการ Error Correction Code จะแก้ไขได้ซึ่งในวิทยานิพนธ์นี้ผู้วิจัยได้มุ่งเน้นไปที่การแก้ไขข้อผิดพลาดชั่วคราวที่เกิดขึ้น ผู้วิจัยจึงได้เสนอแนวคิดในการรับมือความผิดพร่องที่เกิดขึ้นคือการสำรองหน่วยความจำโดยการทำส่วนที่ซ้ำซ้อนของข้อมูล (Information Redundancy) [10] ในการตรวจสอบข้อผิดพลาดที่เกิดขึ้นเพื่อทำการแก้ไขข้อมูลที่ผิดพลาดก่อนที่จะนำไปทำการประมวลผลดังภาพที่ 3-5



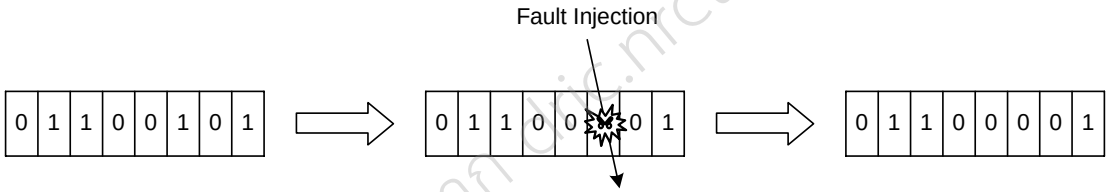
ภาพที่ 3-5 การประมวลผลแบบมีการสำรองหน่วยความจำ

เนื่องจากวิธีการตรวจสอบได้ใช้วิธีการเปรียบเทียบข้อมูลต้นฉบับและข้อมูลสำรองในการตรวจสอบข้อผิดพลาด จึงเป็นสาเหตุที่เลือกใช้แนวคิดการทำส่วนที่ซ้ำซ้อนของข้อมูลเพราะมีความเป็นไปได้น้อยมากสำหรับความผิดพร่องที่เกิดขึ้นของข้อมูลดั้งเดิมกับข้อมูลที่ทำการสำรองที่อยู่ในหน่วยความจำนั้นจะมีความผิดพลาดเหมือนกันทุกประการ ซึ่งข้อมูลต้นฉบับและข้อมูลสำรองนั้นไม่ได้อยู่ในตำแหน่งเดียวกัน วิธีการนี้จึงวิธีการที่เหมาะสม จากภาพที่ 3-5 เมื่อมีการอ่านสคริปต์ผ่านเข้ามาที่หน่วยความจำ เพื่อที่จะทำการประมวลผลนั้นก็ได้มีการสำรองข้อมูลที่ก่อนจะนำไปประมวลผลเพื่อทำการตรวจสอบความผิดพลาด นี้คือสิ่งที่ผู้วิจัยจะนำไปประยุกต์ใช้กับการแคชไดนามิกเว็บซึ่งจะทำการกล่าวในบทต่อไป

การใช้งานจริงของเว็บแอปพลิเคชันนั้นไม่สามารถคาดเดาการเกิดขึ้นของข้อผิดพลาดชั่วคราวได้ว่าจะเกิดขึ้นเมื่อใดและระยะเวลาที่จะเกิดความผิดพร่องขึ้นในระบบนั้นอาจจะต้องใช้เวลานาน การทำการทดสอบวิธีการทำส่วนที่ซ้ำซ้อนของข้อมูลว่าสามารถจะทนต่อความผิดพร่องได้หรือไม่ จึงต้องมีการบังคับให้ปรากฏการดังกล่าวนั้นเกิดขึ้น โดยการฉีดความผิดพร่อง (Fault Injection) เข้าไปในหน่วยความจำ ซึ่งจะกล่าวในหัวข้อต่อไป

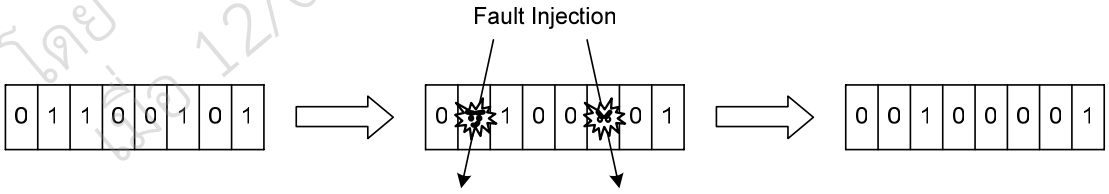
3.4.2 การฉีดความผิดพร่อง (Fault Injection) : คือการเจาะจงที่จะสร้างความผิดพร่องให้เกิดขึ้น หรือนำความผิดพร่องไปฉีดลงไปในระบบ เพื่อจะได้เฝ้าดูและเก็บข้อมูลผลกระทบของระบบหลังจากได้รับความผิดพร่องในลักษณะหรือรูปแบบที่กำหนดขึ้น ซึ่งส่วนที่ต้องการให้เกิดความผิดพลาดขึ้นนั้นก็คือหน่วยความจำ ซึ่งเป็นหน่วยที่ใช้ในการเก็บข้อมูล

ภาพที่ 3-6 เป็นข้อมูลขนาด 1 ไบท์ในหน่วยความจำซึ่งมีขนาด 8 บิตได้ถูกฉีดความผิดพร่องลงไปในบิตที่ 3 จากค่า ‘1’ ไปเป็น ‘0’ เพื่อให้ข้อมูลในหน่วยความจำเกิดปรากฏการณ์ที่เรียกว่าข้อมูลกลับบิต (Bit-flip) แบบบิตเดียว (Single-bit) ทำให้ข้อมูลที่อยู่ในหน่วยความจำนั้นต่างไปจากเดิม



ภาพที่ 3-6 การฉีดความผิดพร่องให้กับข้อมูลแบบบิตเดียว

ภาพที่ 3-7 เป็นข้อมูลขนาด 1 ไบท์ ในหน่วยความจำซึ่งมีขนาด 8 บิตได้ถูกทำการฉีดความผิดพร่องลงไปในบิตที่ 3 และบิตที่ 7 จากค่า ‘1’ ไปเป็น ‘0’ เพื่อให้เกิดปรากฏการณ์ที่เรียกว่าการกลับบิตแบบหลายบิต (Multiple-bit) ทำให้ข้อมูลที่อยู่ในหน่วยความจำนั้นต่างไปจากเดิม



ภาพที่ 3-7 การฉีดความผิดพร่องให้กับข้อมูลแบบหลายบิต

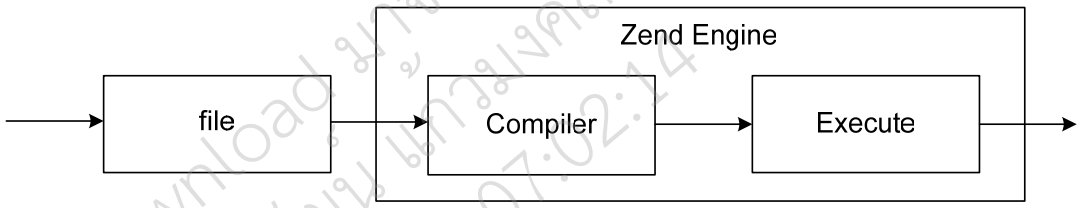
ทฤษฎีที่กล่าวมาทั้งหมดนั้นเป็นสิ่งที่เกี่ยวข้องกับวิทยานิพนธ์ชิ้นนี้ โดยจะเน้นในเรื่องของความผิดพร่องที่เกิดขึ้นกับหน่วยความจำ โดยใช้แนวคิดของการเพิ่มส่วนที่ซ้ำซ้อนเข้ามาช่วยให้ระบบมีความทนทานต่อความผิดพลาดที่จะเกิดขึ้น ซึ่งในการวัดประสิทธิภาพของเว็บเซิร์ฟเวอร์

นั้นผู้วิจัยใช้วิธีการฉีดความผิดพร่องโดยการเปลี่ยนแปลงข้อมูลให้เกิดความผิดพลาด เพื่อลดระยะเวลาในการทดสอบ อีกทั้งยังสามารถกำหนดเงื่อนไข หรือลักษณะของการเกิดความผิดพร่องได้ง่าย ซึ่งการทดลองฉีดความผิดพร่องเข้าไปในหน่วยความจำที่ได้ทำการเก็บแคชไว้ ในโปรแกรมไดนามิกเว็บแคชที่ผู้วิจัยได้ทำการปรับปรุงประสิทธิภาพแล้ว (ด้วยเทคนิคของการเพิ่มส่วนที่ซ้ำซ้อน) และสังเกตรวมถึงบันทึกผลในพฤติกรรมของไดนามิกเว็บแคช ที่ได้ทำการปรับปรุงแล้วว่าสามารถที่จะทนต่อความผิดพร่องที่เกิดขึ้นได้ และทดสอบเว็บเซิร์ฟเวอร์ว่ามีประสิทธิภาพความเร็วว่าเป็นอย่างไร ในส่วนท้ายของบทนี้ยังกล่าวถึงการใช้บริโภคความจำของระบบไดนามิกเว็บเพื่อให้เข้าใจถึงการใช้ทรัพยากรหน่วยความจำไดนามิกเว็บและเว็บแคช ซึ่งในบทต่อไปจะกล่าวถึงการออกแบบให้ไดนามิกเว็บแคชมีความคงทนต่อความผิดพร่อง และความผิดพร่องที่ผู้วิจัยสนใจคือความผิดพร่องที่ทำให้เกิดข้อผิดพลาดชั่วคราวในหน่วยความจำ

บทที่ 4
การสร้างการทนต่อความผิดพร่องโดยใช้ไดนามิกเว็บแคช

การที่จะพัฒนาไดนามิกเว็บแคชให้มีความสามารถในการทนต่อความผิดพร่อง ผู้พัฒนาจะต้องอาศัยความเข้าใจถึงโครงสร้างภายในและขั้นตอนการทำงานของ PHP ซึ่งผู้วิจัยได้กล่าวถึงโครงสร้างภายในและขั้นตอนการทำงานของ PHP ไปแล้วในบทที่ 2 ในบทนี้จะอธิบายส่วนย่อยขั้นตอนของช่วงที่มีการอ่านสคริปต์ไฟล์ การคอมไพล์ และการ Execute ออกมาเพื่อเจาะจงในส่วนของการคอมไพล์

4.1 การประมวลผลภายใน PHP



ภาพที่ 4-1 การประมวลผลภายใน PHP แบบปกติ

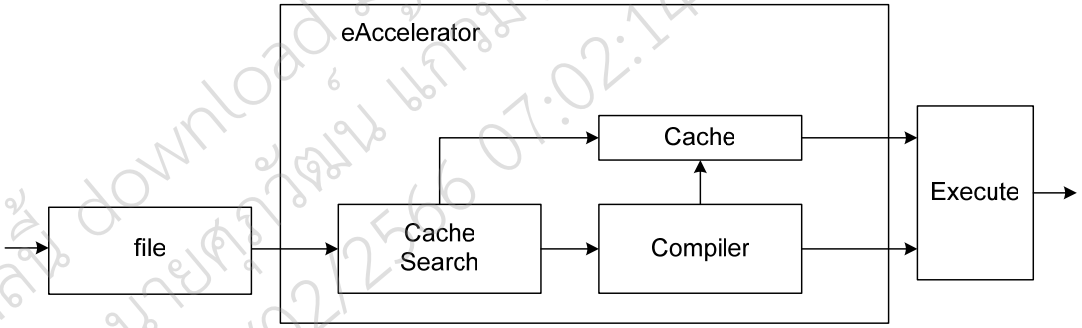
ภาพที่ 4-1 แสดงการประมวลผลในเซ็นเอ็นจิ้น (Zend Engine) เมื่อมีการอ่านไฟล์สคริปต์ PHP มาจากเซิร์ฟเวอร์มาเพื่อทำการคอมไพล์ และเมื่อถูกคอมไพล์เป็นไบท์โค้ดเสร็จเรียบร้อยแล้วก็จะส่งไปทำการ Execute ต่อไป

ในการพัฒนาส่วนขยาย PHP ที่มีการติดต่อกับเซ็นเอ็นจิ้น นั้นสามารถติดต่อผ่านทาง Zend API ที่ Zend ได้มีการจัดเตรียมไว้โดยผ่านทางฟังก์ชันและแมโคร (Macro) ของ Zend เพื่อทำให้สามารถติดต่อกับตัวแปรหรือการทำงานต่างๆ ของเซ็นเอ็นจิ้น ได้ ซึ่งโปรแกรม eAccelerator นั้นก็ติดต่อกับเซ็นเอ็นจิ้น ผ่านทาง Zend API เช่นเดียวกัน

4.2 การทำงานของ eAccelerator

โปรแกรม eAccelerator นั้นเป็นส่วนขยายภายนอก ซึ่งไม่ได้เป็นส่วนประกอบร่วมเข้ามากับ PHP สำหรับโปรแกรมเว็บเซิร์ฟเวอร์ที่ใช้ในวิทยานิพนธ์นี้คือ Apache เวอร์ชัน 1.3.34 เมื่อเว็บเซิร์ฟเวอร์เริ่มต้นทำงาน PHP ก็จะทำการจองทรัพยากรให้กับส่วนขยายต่างๆ

โปรแกรม eAccelerator นั้นเป็นส่วนขยายของ PHP ซึ่งทำงานร่วมกับ PHP ในขั้นตอนที่อยู่ในช่วงการทำงานของเซ็นเอ็นจิ้นเป็นหลัก โดยเมื่อมีการอ่านไฟล์สคริปต์มาจากเซิร์ฟเวอร์แล้ว จะมีการขัดจังหวะการคอมไพล์สคริปต์ ซึ่งโปรแกรม eAccelerator จะมีการค้นหาแคชว่าได้มีการทำแคชเก็บไว้แล้วหรือไม่ ซึ่งการเก็บแคชนั้นในแต่ละไฟล์นั้นจะมีการเก็บแคชแยกกัน ซึ่งเมื่อมีการร้องขอเข้ามาในครั้งแรกนั้นเป็นที่แน่นอนว่ายังไม่มีแคชเก็บอยู่ จึงค้นหาไม่เจอ จากนั้นโปรแกรม eAccelerator จึงทำการคำนวณขนาดของทรัพยากรที่ต้องการเพื่อที่จะใช้ในการจองหน่วยความจำ จากนั้นก็ทำการคอมไพล์ เมื่อคอมไพล์เสร็จ ก็มีการเก็บข้อมูลที่ได้ทำการคอมไพล์แล้ว ไว้ในหน่วยความจำ จากนั้นก็ทำการส่งไป Execute ต่อไป ดังที่แสดงในภาพที่ 4-2

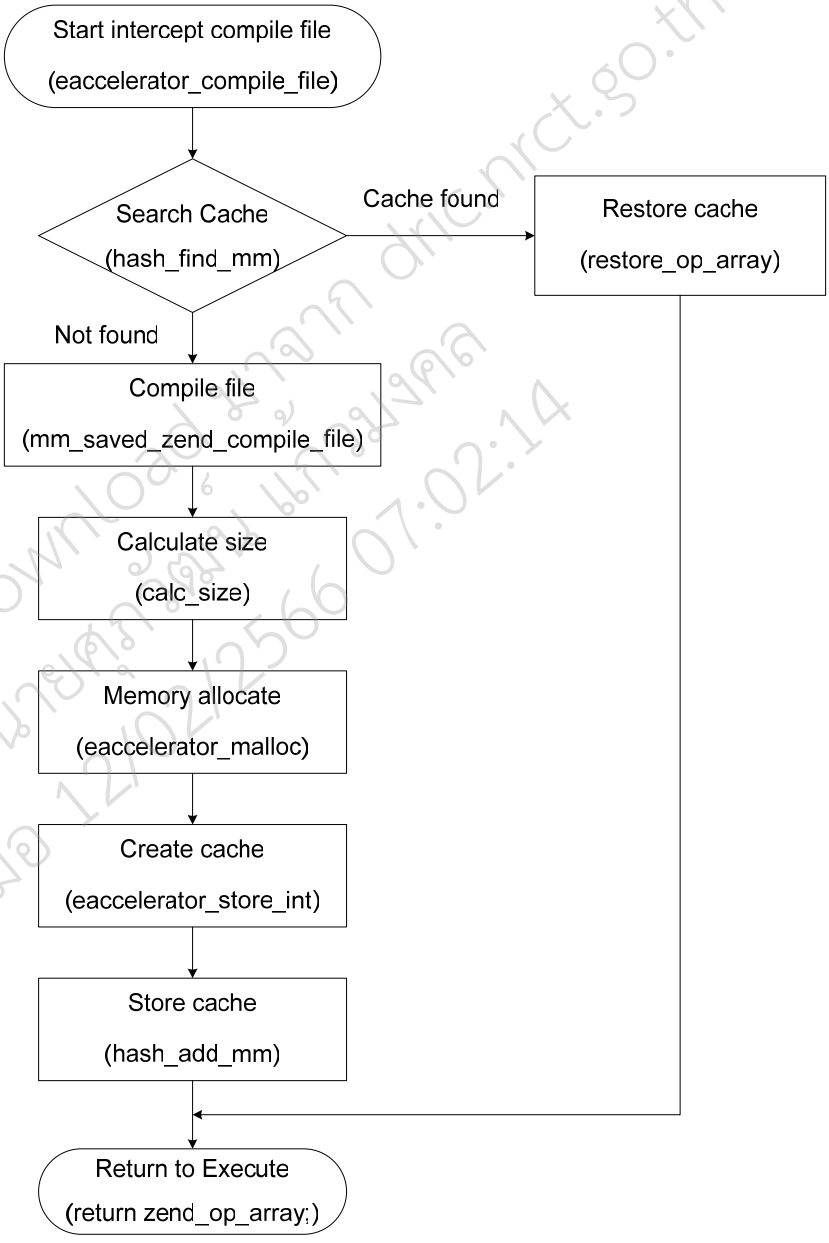


ภาพที่ 4-2 การประมวลผลภายใน PHP เมื่อใช้ร่วมกับส่วนขยาย eAccelerator

เมื่อมีการร้องขอที่ไฟล์เดิมเป็นครั้งที่สอง โปรแกรม eAccelerator ก็ทำการค้นหาแคชเช่นเดิม ซึ่งจะสามารถที่จะหาแคชเจอ เนื่องจากได้มีการทำแคชเก็บไว้ในการร้องขอครั้งแรก จากนั้นก็นำแคชไปใช้โดยที่ไม่ต้องเสียเวลาในช่วงของการคอมไพล์ ซึ่งในการร้องขอที่ไฟล์เดิมในครั้งต่อไปก็จะเป็นเช่นนี้เรื่อยไป

เพื่อเป็นการเพิ่มความเข้าใจในการทำงานภายในของ eAccelerator ให้มากยิ่งขึ้นผู้วิจัยจึงได้แสดงผังงาน (Flowchart) ดังที่แสดงในภาพที่ 4-3 ซึ่งแสดงถึงฟังก์ชันการทำงานหลักของ eAccelerator โดยเมื่อเริ่มต้นทำงานของเว็บเซิร์ฟเวอร์ ซึ่งเป็นการเริ่มต้นวงจรชีวิต PHP ด้วยเช่นกัน

eAccelerator จะเริ่มประกาศตัวเองเป็นตัวคอมไพล์ของ Zend ซึ่งอยู่ในขั้นตอนของ MINIT เพื่อเป็นการขัดจังหวะของการคอมไพล์ (Intercept Compilation) แบบปกติ และเมื่อมีการร้องขอหน้าเว็บเข้ามาส่วนขยายจะเริ่มทำขั้นตอน RINIT ซึ่งเป็นขั้นตอนมาตรฐานที่ได้กล่าวไปแล้วในบทที่ 2 หลังจากนั้นก็จะเริ่มเรียกฟังก์ชันการทำงานในลำดับแรกคือ eaccelerator_compile_file ซึ่งเป็นการเริ่มขัดจังหวะการคอมไพล์ของสคริปต์ PHP และทำการเรียกฟังก์ชัน hash_find_mm เพื่อไปทำการค้นหาแคชว่าได้มีการเก็บแคชอยู่ในหน่วยความจำหรือไม่

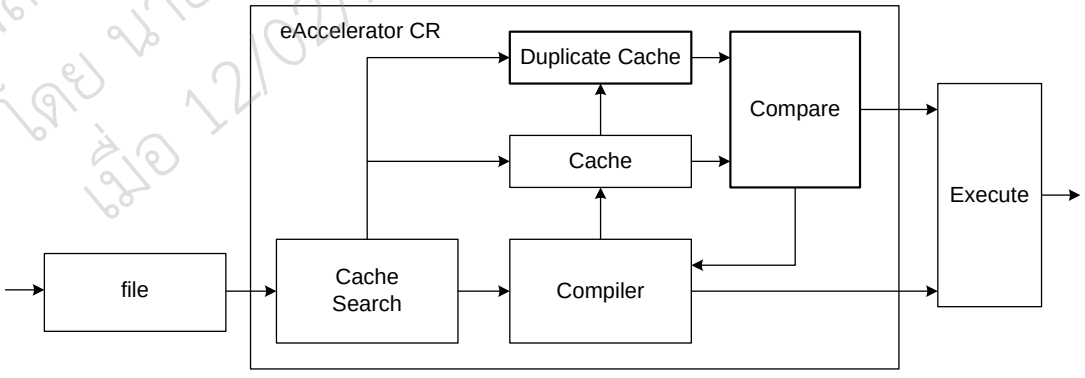


ภาพที่ 4-3 ผังงานของ eAccelerator

ถ้ามีแคชเก็บอยู่แล้วก็จะทำการเรียกฟังก์ชัน restore_op_array เพื่อที่จะนำไบท์โค้ดที่ถูกเก็บอยู่ในแคชไปทำการ Execute แต่ถ้ายังไม่มีการร้องขอหน้าเว็บในครั้งแรกฟังก์ชัน hash_find_mm จะหาแคชไม่เจอเนื่องจากยังไม่มีการทำแคชเก็บไว้ eAccelerator จะทำการคอมไพล์สคริปต์ด้วยฟังก์ชัน mm_saved_zend_compile_file เมื่อคอมไพล์เสร็จจะได้เป็นไบท์โค้ดเพื่อที่จะนำไปเก็บไว้เป็นแคช แต่ก่อนที่จะนำไปเก็บจะต้องมีการคำนวณขนาดของไบท์โค้ดด้วยฟังก์ชัน calc_size ฟังก์ชัน calc_size ทำการคำนวณขนาดของไบท์โค้ดเสร็จแล้วจึงไปทำการจองพื้นที่ในหน่วยความจำด้วยฟังก์ชัน eaccelerator_malloc เมื่อทำการจองพื้นที่ในหน่วยความจำได้แล้วก็จะทำการสร้างแคชด้วยฟังก์ชัน eaccelerator_store_int และนำไปเก็บไว้ในตำแหน่งที่ได้ทำการจองได้ด้วยฟังก์ชัน hash_add_mm หลังจากที่ได้ทำการเก็บแคชเรียบร้อยแล้ว eAccelerator ก็จะส่งไบท์โค้ดไปทำการ Execute เป็นการจบการทำงานในแต่ละการร้องขอ

4.3 ไดนามิกเว็บแคชที่สามารถทนต่อความผิดพร่อง

ในการพัฒนาหรือการแก้ไขส่วนขยาย PHP นั้นผู้พัฒนาเลือกใช้ภาษา C ในการพัฒนาเนื่องจากโครงสร้างภายใน PHP นั้นถูกพัฒนาขึ้นด้วยภาษา C ซึ่งโปรแกรม eAccelerator ก็เช่นเดียวกัน จากภาพที่ 4-4 ผู้วิจัยได้ทำการปรับปรุงและแก้ไขโปรแกรม eAccelerator ให้แคชที่เก็บอยู่ในหน่วยความจำสามารถทนต่อความผิดพร่อง โดยการเพิ่มส่วนแคชเพื่อใช้ในการทำสำเนาและส่วนการตรวจสอบข้อผิดพลาดก่อนที่จะนำแคชที่ทำการเก็บไว้ในการร้องขอครั้งแรก มาทำการ Execute



ภาพที่ 4-4 การประมวลผลภายใน PHP เมื่อใช้ร่วมกับส่วนขยาย eAccelerator CR

เมื่อมีการร้องขอหน้าเว็บเข้ามาในครั้งแรกจะทำการอ่านไฟล์ที่ได้ร้องขอเข้ามา จากนั้นก็ได้ทำการค้นหาแคช ซึ่งในการร้องขอหน้าเว็บครั้งแรกจะหาแคชไม่เจอ เนื่องจากยังไม่ได้ทำแคชเก็บ

ไว้ ในเมื่อหาแคชไม่เจอก็สั่งให้ทำการคอมไฟล์ และทำการจองทรัพยากรหน่วยความจำ ที่ต้องการใช้ในการเก็บแคช แต่จากโปรแกรม eAccelerator ที่ได้มีการปรับปรุงให้ทนต่อความผิดพลาดที่จะเกิดขึ้น นั้นได้มีการเพิ่มในส่วนของแคชเพื่อใช้ในการทำสำเนา การจองหน่วยความจำที่ใช้ในการทำแคชของแต่ละไฟล์นั้นต้องใช้มากขึ้นเป็นสองเท่า เมื่อทำการจองหน่วยความจำที่จะต้องใช้ในการเก็บแคชเรียบร้อยแล้ว ก็ทำการนำข้อมูลที่คอมไฟล์เรียบร้อยแล้วไปเก็บไว้ในแคชและแคชสำเนา เมื่อทำการเก็บแคชแล้วก็ส่งข้อมูลไปทำการ Execute ต่อไป ดังแสดงในภาพที่ 4-4 ในที่นี้ผู้วิจัยขอเรียกโปรแกรม eAccelerator ที่ได้รับการแก้ไขปรับปรุงว่า eAccelerator CR (eAccelerator Cache Redundancy)

เมื่อมีการร้องขอที่ไฟล์เดิมเป็นครั้งที่สอง โปรแกรม eAccelerator CR ก็ทำการค้นหาแคชเช่นเดิมซึ่งสามารถที่จะหาแคชเจอ เนื่องจากได้มีการทำแคชเก็บไว้ในการร้องขอครั้งแรก แต่ก่อนที่จะนำข้อมูลในแคชไปทำการ Execute ก็จะมีการตรวจสอบข้อผิดพลาดที่สามารถเกิดขึ้นในหน่วยความจำแคชได้ โดยการเปรียบเทียบแคชและแคชสำเนาว่ามีความแตกต่างกันหรือไม่ ถ้าไม่มีความแตกต่างเกิดขึ้นหลังจากการเปรียบเทียบ แสดงว่าไม่มีความผิดพลาดเกิดขึ้นในหน่วยความจำแคช ก็จะส่งไปทำการ Execute ต่อไป

แต่ถ้ามีความแตกต่างเกิดขึ้นในขั้นตอนการเปรียบเทียบแคช แสดงว่ามีข้อผิดพลาดเกิดขึ้นในหน่วยความจำแคช โปรแกรม eAccelerator CR จะมีการลบแคชและแคชสำเนาทิ้งไป แล้วทำการคอมไฟล์ใหม่เหมือนเช่นเดียวกับเมื่อตอนที่มีการร้องขอเข้ามาในครั้งแรก จากนั้นก็ทำการเก็บแคชที่ถูกต้องเก็บไว้ในหน่วยความจำเช่นเดิม เมื่อการการร้องขอเข้ามาอีกที่ไฟล์เดิมก็จะมีการนำแคชที่ได้รับการตรวจสอบความผิดพลาดแล้วไป Execute

จากที่กล่าวมาแล้วสามารถอธิบายลำดับขั้นตอนการทำงานของ eAccelerator CR มี 4 ขั้นตอนดังต่อไปนี้

4.3.1 เมื่อมีการร้องขอหน้าเว็บเข้ามาจะมีการทำการค้นหาแคชถ้าพบแคชที่ถูกเก็บไว้ จากการร้องขอในครั้งแรกให้ข้ามขั้นตอนที่ 4.3.2 ไปทำในขั้นตอนที่ 4.3.3

4.3.2 ทำการคอมไพล์สคริปต์แล้วจึงนำไบท์โค้ดที่ได้ไปเก็บเป็นแคชและแคชสำเนาจากนั้นจึงทำการ Execute ไบท์โค้ดนั้นตามปกติ เป็นการสิ้นสุดขั้นตอนการทำงานของ eAccelerator CR (ข้ามขั้นตอนที่ 4.3.3 และ 4.3.4)

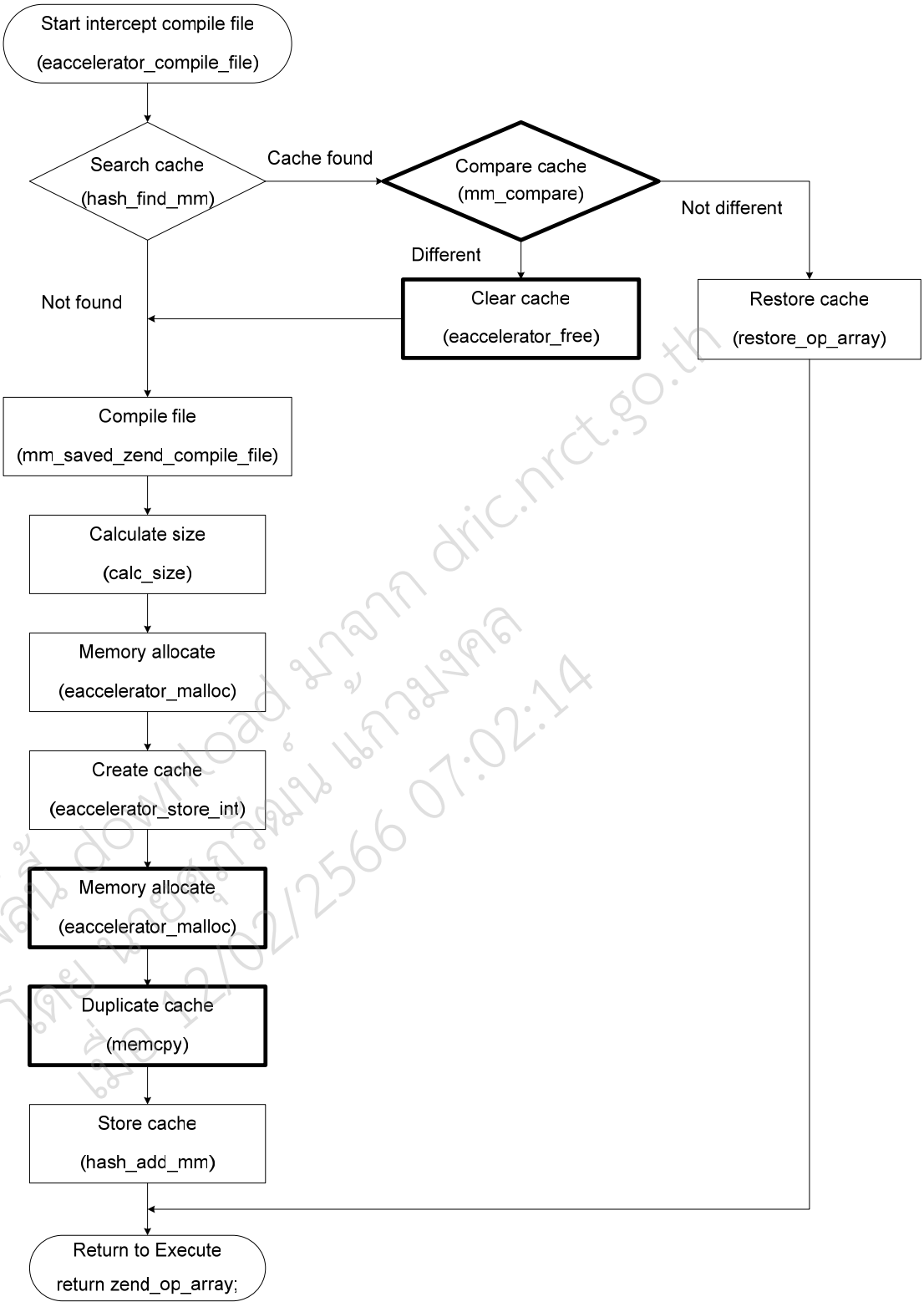
4.3.3 นำแคชและแคชสำเนามาทำการเปรียบเทียบเพื่อหาข้อผิดพลาดถ้ามีข้อผิดพลาดเกิดขึ้นให้ไปทำต่อในขั้นตอนที่ 4.4 ถ้าไม่มีความผิดพลาดเกิดขึ้นก็จะนำแคชไป Execute เป็นการสิ้นสุดขั้นตอนการทำงานของ eAccelerator CR (ข้ามขั้นตอนที่ 4.3.4)

4.3.4 ทำการลบแคชและแคชสำเนาทิ้งหลังจากที่ตรวจสอบพบความผิดพลาด แล้วกลับไปทำในขั้นตอนที่ 4.3.2

เพื่อเป็นการเพิ่มความเข้าใจในการทำงานภายในของ eAccelerator CR ให้มากยิ่งขึ้นผู้วิจัยจึงได้แสดงผังงานดังที่แสดงในภาพที่ 4-5 ซึ่งแสดงถึงฟังก์ชันการทำงานหลักของ eAccelerator CR โดยเมื่อเริ่มต้นทำงานของเว็บเซิร์ฟเวอร์ ซึ่งเป็นการเริ่มต้นวงจรชีวิต PHP ด้วยเช่นกัน eAccelerator CRจะเริ่มประกาศตัวเองเป็นตัวคอมไพล์ของ Zend ซึ่งอยู่ในขั้นตอนของ MINIT เพื่อเป็นการขัดจังหวะของการคอมไพล์แบบปกติ และเมื่อมีการร้องขอหน้าเว็บเข้ามาส่วนขยายจะเริ่มขั้นตอน RINIT ซึ่งเป็นขั้นตอนมาตรฐาน หลังจากนั้นก็จะเริ่มเรียกฟังก์ชัน eaccelerator_compile_file เช่นเดียวกับ eAccelerator โดยเป็นการเริ่มขัดจังหวะการคอมไพล์ของสคริปต์ PHP และทำการเรียกฟังก์ชัน hash_find_mm เพื่อไปทำการค้นหาแคชและแคชสำเนาว่าได้มีการเก็บอยู่ในหน่วยความจำหรือไม่

แต่ถ้าเป็นการร้องขอหน้าเว็บในครั้งแรกฟังก์ชัน hash_find_mm จะหาแคชไม่เจอเนื่องจากยังไม่มีการทำแคชเก็บไว้ eAccelerator CR จะทำการคอมไพล์สคริปต์ด้วยฟังก์ชัน mm_saved_zend_compile_file เมื่อคอมไพล์เสร็จจะได้เป็นไบท์โค้ดเพื่อที่จะนำไปเก็บไว้เป็นแคช แต่ก่อนที่จะนำไปเก็บจะต้องมีการคำนวณขนาดของไบท์โค้ดด้วยฟังก์ชัน calc_size เมื่อทำการคำนวณขนาดของไบท์โค้ดได้แล้ว จึงไปทำการจองพื้นที่ในหน่วยความจำด้วยฟังก์ชัน eaccelerator_malloc เมื่อทำการจองพื้นที่ในหน่วยความจำได้แล้วก็จะทำการสร้างแคชด้วยฟังก์ชัน eaccelerator_store_int แต่เนื่องด้วย eAccelerator CR ได้ทำการปรับปรุงให้มีการเพิ่มส่วนที่ซ้ำซ้อน จึงต้องทำการจองพื้นที่ให้กับแคชสำเนาด้วยฟังก์ชัน eaccelerator_malloc อีกครั้งหนึ่งเพื่อที่จะนำพื้นที่ที่ได้จองไว้ไปเก็บสำเนาแคช จากนั้นจึงสร้างแคชสำเนาด้วยฟังก์ชัน duplicate_cache โดยนำแคชและแคชสำเนาไปเก็บไว้ในตำแหน่งที่ได้ทำการจองได้ด้วยฟังก์ชัน hash_add_mm หลังจากที่ได้ทำการเก็บแคชเรียบร้อยแล้ว eAccelerator CR ก็จะส่งไบท์โค้ดไปทำการ Execute เป็นการจบการทำงานในการร้องขอครั้งแรก

แต่ถ้ามีแคชและแคชสำเนาเก็บอยู่แล้วก็จะทำการเรียกฟังก์ชัน mm_compare เพื่อทำการเปรียบเทียบแคชกับแคชสำเนาเพื่อตรวจสอบความผิดพลาดชั่วคราวที่อาจจะเกิดขึ้นได้ ถ้าไม่มีความแตกต่างจากการเปรียบแสดงว่าไม่มีความผิดพลาดเกิดขึ้น หรืออาจจะเกิดขึ้นแต่เป็นกรณีที่เกิดขึ้นได้ยากมากซึ่งผู้วิจัยจะนำไปกล่าวในบทที่ 5 จากนั้น eAccelerator CR จะทำการเรียกฟังก์ชัน restore_op_array เพื่อที่จะนำไบท์โค้ดที่ถูกเก็บอยู่ในแคชไปทำการ Execute



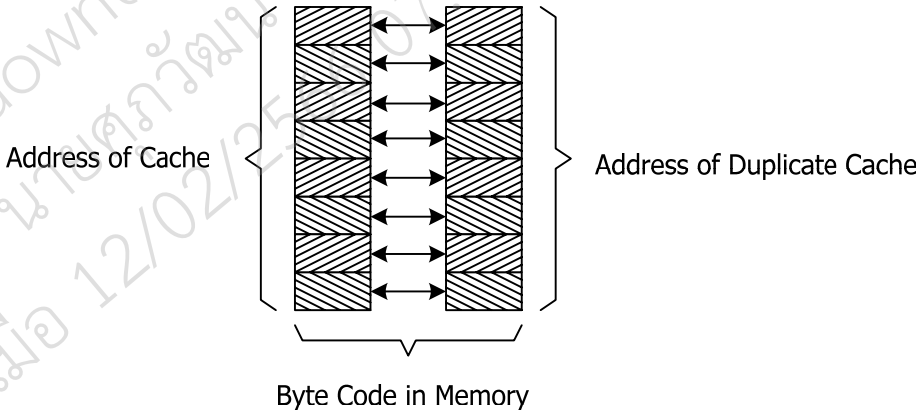
ภาพที่ 4-5 ผังงานของ eAccelerator CR

ถ้ามีความแตกต่างเกิดขึ้นจากการเปรียบแสดงว่ามีความผิดพลาดเกิดขึ้น ซึ่งอาจเกิดที่ตำแหน่งของแคชหรือแคชสำเนา eAccelerator CR จะทำการลบแคชและแคชสำเนาทิ้งไปด้วยฟังก์ชัน eaccelerator_free เพื่อที่จะคืนทรัพยากรหน่วยความจำที่ได้ทำการจองไว้

หลังจากที่ได้ทำการทำการลบแคชและแคชสำเนาทิ้งไปแล้ว จากนั้น eAccelerator CR จึงทำการคอมไพล์ใหม่อีกครั้ง ด้วยฟังก์ชัน mm_saved_zend_compile_file จากนั้นก็จะทำกระบวนการของการนำไบท์โค้ดที่ได้จากการคอมไพล์ไปเก็บเป็นแคชเช่นเดียวกับการร้องขอในครั้งแรก

4.4 การตรวจสอบความผิดพลาด

ในการแก้ไขโปรแกรม eAccelerator มาเป็น eAccelerator CR นั้นผู้วิจัยได้มีการเพิ่มส่วนแคชเพื่อใช้ในการทำสำเนา และส่วนการตรวจสอบข้อผิดพลาด การแก้ไขโปรแกรมนั้นมีความเกี่ยวข้องกับการจัดการหน่วยความจำ เช่นการทำแคชสำเนาเพื่อทำข้อมูลสำเนาที่เหมือนกับข้อมูลดั้งเดิม การทำแคชสำเนานั้นได้ใช้ฟังก์ชัน `memcpy()` ของ C ไลบรารีในการคัดลอกข้อมูลของแคช ฟังก์ชัน `memcpy()` นั้นจะทำการคัดลอกข้อมูลจากพื้นที่ของข้อมูลดั้งเดิมในหน่วยความจำมาใส่ในพื้นที่ ที่ได้ถูกทำการจองไว้สำหรับข้อมูลใหม่ ที่มีข้อมูลและขนาดของข้อมูลเหมือนกับข้อมูลดั้งเดิม



ภาพที่ 4-6 แสดงการเปรียบเทียบหน่วยความจำทั้งสอง

สำหรับส่วนการตรวจสอบข้อผิดพลาดของข้อมูลแคชที่เก็บไว้นั้น เกี่ยวข้องกับการจัดการหน่วยความจำ ผู้วิจัยได้ใช้วิธีการเปรียบเทียบในแต่ละไบท์ของแคชกับแคชสำเนา ดังที่แสดงไว้ในภาพที่ 4-6 การตรวจสอบความถูกต้องของแคช ที่ได้ทำการเก็บไว้ในหน่วยความจำนั้น จะใช้

ฟังก์ชัน memcmp() ของ C ไลบรารี ฟังก์ชัน memcmp() เป็นฟังก์ชันที่ทำการเปรียบเทียบหน่วยความจำที่อยู่ในตำแหน่งที่ต่างกัน แล้วแต่ขนาดที่ต้องการให้ทำการเปรียบเทียบค่าที่คืนกลับมาของฟังก์ชันนี้คือค่าจำนวนเต็มมีสามลักษณะคือค่าที่เป็นบวก ค่าที่เป็นลบ และค่าที่เป็น 0 เมื่อค่าของข้อมูลแคชในหน่วยจำที่ทำการเปรียบนั้นมีค่าไม่เท่ากัน ก็จะมีการคืนค่ากลับมาเป็นบวกหรือเป็นลบ แต่เมื่อค่าของข้อมูลแคชในหน่วยความจำที่ทำการเปรียบเทียบนั้นมีค่าเท่ากัน ฟังก์ชัน memcmp() จะคืนค่ากลับมาเป็น 0 จากค่าที่คืนกลับมาสามารถทำให้รู้ได้ว่าหน่วยความจำที่เก็บข้อมูลแคชมีข้อผิดพลาดเกิดขึ้นหรือไม่

ที่กล่าวมาแล้วนั้น คือโปรแกรม eAccelerator ที่ได้รับการแก้ไขในส่วนของแคชที่ทำการเก็บไว้ในหน่วยความจำ และมีการตรวจสอบข้อผิดพลาด โดยในบทต่อไปผู้วิจัยจะนำโปรแกรม eAccelerator ดังกล่าวมาทำการทดสอบ และประเมินประสิทธิภาพต่อไป

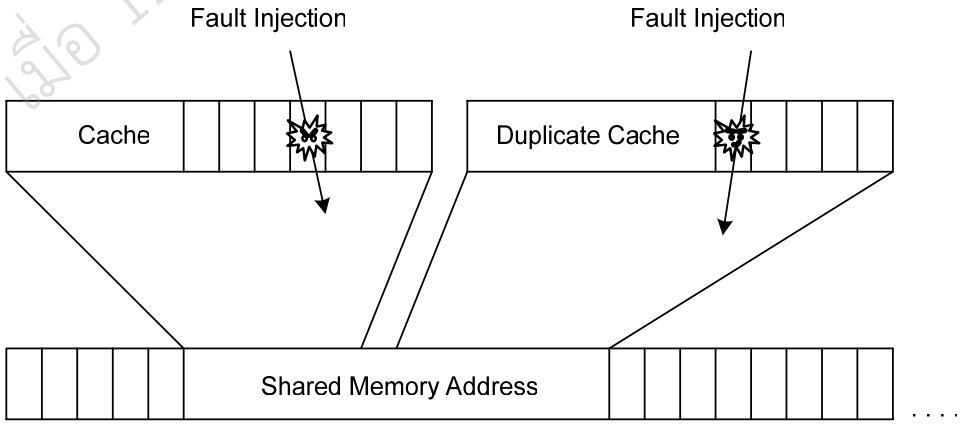
บทที่ 5
การประเมินประสิทธิภาพ

บทนี้ได้นำเสนอการประเมินประสิทธิภาพของ eAccelerator และ eAccelerator CR ที่ได้รับการแก้ไขปรับปรุง เพื่อแสดงให้เห็นว่าความคงทนต่อความผิดพร่องของแคชที่เก็บไว้ในหน่วยความจำ และผลกระทบข้างเคียงที่เกิดขึ้นในด้านการใช้เวลาที่เพิ่มขึ้นจากการแก้ไขโปรแกรม eAccelerator ว่ามีความแตกต่างกันมากน้อยเพียงใด

ระบบเซิร์ฟเวอร์ที่สร้างขึ้นเพื่อที่จะทำการประเมินประสิทธิภาพทำงานบนระบบปฏิบัติการ FreeBSD 5.5 โดยซอฟต์แวร์ประกอบด้วย เว็บเซิร์ฟเวอร์ Apache 1.3.34, PHP 4.5.2 และโปรแกรม eAccelerator 0.9.4 และระบบดังกล่าวใช้งานร่วมกับฮาร์ดแวร์ CPU Intel Pentium III 800 MHz . ซึ่งมีหน่วยความจำ 256MB ทำงานอยู่บนเครือข่าย LAN ความเร็ว 100 Mbit/s ซึ่งเป็นระบบที่จะใช้ในการประเมินประสิทธิภาพต่อไป

5.1 การประเมินประสิทธิภาพของการตรวจสอบความผิดพลาด

การทดสอบในวิทยานิพนธ์นี้เป็นการฉีดความผิดพร่อง (Fault Injection) เพื่อทำให้เกิดความผิดพลาดขึ้นในข้อมูลแคชที่เก็บอยู่ในหน่วยความจำ ซึ่งจะเป็นการจำลองข้อผิดพลาดชั่วคราวที่เกิดขึ้นในหน่วยความจำ เพื่อที่จะทำการทดสอบกลไกในการตรวจสอบความผิดพลาดของข้อมูลที่ได้กล่าวไปในบทที่ 4 ว่าจะสามารถตรวจสอบข้อผิดพลาดของข้อมูลแคชได้หรือไม่



ภาพที่ 5-1 การฉีดความผิดพร่องเข้าไปในแคช

ภาพที่ 5-1 เป็นการฉีดความผิดพร่องเข้าไปในแคช เพื่อทำให้เกิดความผิดพลาดขึ้นในข้อมูลแคชที่เก็บอยู่ในหน่วยความจำ โดยการจงใจที่จะฉีดความผิดพร่องของไปที่ไบท์ใดไบท์หนึ่งของแคชหรือแคชสำเนา เพื่อให้เกิดข้อผิดพลาดในระดับบิต (Bit Error)

ในการทำงานปกติของ eAccelerator CR คือการที่ไม่มีความผิดพร่องเกิดขึ้นกับแคชทั้งสอง ความผิดพร่องที่จะเกิดขึ้นกับหน่วยความจำไม่สามารถคาดเดาได้ว่าจะมีความผิดพร่องเกิดขึ้นเวลาใด ดังนั้นผู้วิจัยจึงได้ทำการจำลองสภาวะที่ทำให้เกิดความผิดพร่องขึ้น โดยการฉีดความผิดพร่องเข้าไปในข้อมูลแคชที่ถูกเก็บไว้ในหน่วยความจำ ข้อมูลแคชที่ถูกเก็บอยู่ในหน่วยความจำมีลักษณะเป็นไบท์โค้ด ดังนั้นเพื่อเป็นการสร้างความผิดพร่องขึ้นในไบท์ใดไบท์หนึ่ง (โดยความผิดพร่องนั้นอาจอยู่ในรูปแบบ Single-bit หรือ Multiple-bit) ผู้วิจัยจึงได้ทำการฉีดความผิดพร่องแบบสุ่ม (Random) เข้าไปในแต่ละส่วนของแคชและแคชสำเนาที่มีขนาดที่แตกต่างกัน ดังนั้นความผิดพร่องที่เกิดขึ้นจากการฉีดความผิดพร่องนั้นสามารถแบ่งได้เป็นสามกรณีดังนี้

กรณีแรก (Case 1) คือความผิดพร่องที่เกิดขึ้นกับแคชหรือแคชสำเนาอย่างใดอย่างหนึ่งเท่านั้น ในกรณีนี้ eAccelerator CR สามารถที่จะตรวจสอบหาข้อผิดพลาดได้ทุกครั้ง

ในกรณีที่สอง (Case 2) คือความผิดพร่องที่เกิดขึ้นกับแคชและแคชสำเนาทั้งคู่หากแต่ความผิดพร่องที่เกิดขึ้นนั้นไม่ได้เกิดขึ้นที่ไบท์เดียวกัน ซึ่งในกรณีนี้ eAccelerator CR สามารถที่จะตรวจสอบหาข้อผิดพลาดได้ทุกครั้งเช่นกัน

ตารางที่ 5-1 แสดงกรณีที่สามารถตรวจสอบความถูกต้องได้และไม่ได้เมื่อมีความผิดพร่องเกิดขึ้น

Fault Injection	Case 1	Case 2	Case 3
Detection	Detectable	Detectable	Not Detectable

ความผิดพร่องในกรณีที่สาม (Case 3) นั้นเกิดขึ้นในไบท์โค้ดของแคชและแคชสำเนา ซึ่งทำให้ไม่สามารถระบุความแตกต่างระหว่างแคชทั้งสองได้ กล่าวคือเป็นความผิดพร่องที่เกิดขึ้นในไบท์เดียวกัน ที่ตำแหน่งบิตเดียวกันและค่าของข้อมูลที่บิตดังกล่าวเหมือนกันทำให้ไม่สามารถตรวจสอบข้อผิดพลาดที่เกิดขึ้นได้ เนื่องจากการตรวจสอบข้อผิดพลาดนั้นเป็นการเปรียบเทียบความแตกต่างของแคชกับแคชสำเนา แต่ในกรณีนี้เป็นกรณีที่เกิดขึ้นได้ยาก ซึ่งจากการฉีดความผิดพร่องแบบสุ่มเข้าไปในไบท์ใดไบท์หนึ่ง ซึ่งมีขนาด 8 บิต ความเป็นไปได้ของข้อมูลในไบท์นั้นๆ ที่จะมีข้อผิดพลาดที่เหมือนกันมีค่าความน่าจะเป็น (Probability) เท่ากับ 0.143 ดังที่ได้แสดงไว้ในสมการที่ 5-1

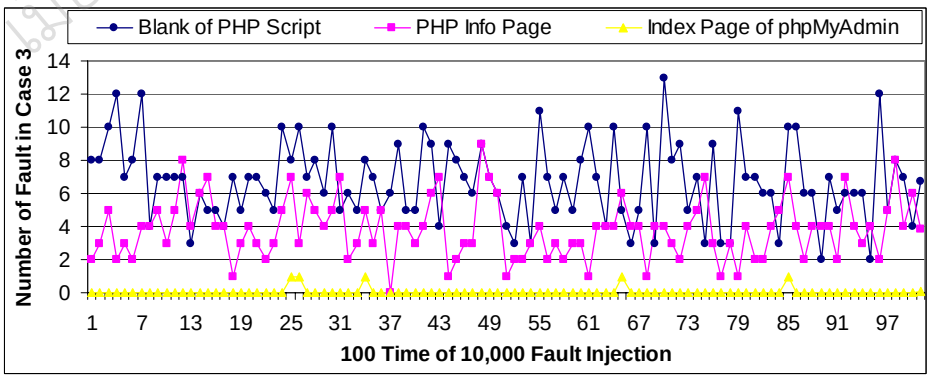
สมการที่ 5-1 แสดงความน่าจะเป็นที่จะเกิดความผิดพลาดในกรณีที่ 3 ของแคชขนาด 1 ไบท์

$$\sum_{i=1}^{8} \frac{1}{\binom{8}{i} \cdot i^2} = 0.143 \qquad (5\text{-}1)$$

แต่เนื่องจากขนาดของแคชและแคชสำเนานั้นสามารถมีขนาดได้เล็กที่สุด 188 ไบท์ ค่าความน่าจะเป็นที่จะเกิดขึ้นทั้งหมดจึงเท่ากับ 0.00076 สามารถคำนวณได้จาก $\frac{1}{188} \cdot 0.143 = 0.00076$

ซึ่งค่าความน่าจะเป็นที่ได้ทำให้ทราบได้ว่าความผิดพร่องที่เกิดขึ้น 10,000 ครั้ง นั้นจะมีความผิดพร่องที่เกิดขึ้นในกรณีที่ไม่สามารถตรวจสอบข้อผิดพลาดได้นั้นมีความเป็นไปได้ 7 ครั้ง โดยประมาณ ยิ่งถ้าเป็นหน้าเว็บไดนามิกที่ใช้กันโดยทั่วไปแล้วนั้นมีขนาดของแคชที่ใหญ่กว่า 188 ไบท์ ทำให้ความเป็นไปได้ที่แคชทั้งสองนั้นจะมีความผิดพลาดเกิดขึ้นเหมือนกันทุกประการมีน้อยลงไปอีก

จากความผิดพร่องในกรณีที่ 3 ผู้วิจัยได้ทำการทดสอบการเกิดขึ้นโดยการฉีดความผิดพร่องกับแคชและแคชสำเนาเพื่อดูแนวโน้มของการเกิดความผิดพร่องในกรณีที่ 3 กับแคชที่มีขนาดต่างกันโดยมีแคชที่ถูกสร้างจากสคริปต์ว่างของ PHP ซึ่งมีขนาดเล็กที่สุดจากที่ได้กล่าวไปแล้วคือ 188 ไบท์ และแคชที่ถูกสร้างจากหน้ารายละเอียดของ PHP (phpinfo();) มีขนาดของแคชเท่ากับ 320 ไบท์ แล้วยังมีแคชที่ถูกสร้างจากสคริปต์หน้าแรกของ phpMyAdmin เวอร์ชั่น 2.9.0.2 มีขนาดของแคชเท่ากับ 20,872 ไบท์ ซึ่งแนวโน้มที่เกิดขึ้นได้แสดงไว้ในภาพที่ 5-2



ภาพที่ 5-2 แสดงการแนวโน้มการเกิดความผิดพร่องของกรณีที่ 3

โดยเส้นกราฟทั้ง 3 เส้นจากภาพที่ 5-2 แสดงการแนวโน้มการเกิดความผิดพร่องของกรณีที่ 3 ผู้วิจัยจำลองการฉีดความผิดพร่องกับแคชและแคชสำเนา เพื่อดูแนวโน้มของการเกิดความผิดพร่องโดยการฉีดความผิดพร่อง 10,000 ครั้งเป็นจำนวนทั้งหมด 100 ครั้ง โดยเส้นวงกลมแทนแคชที่ถูกสร้างจากสคริปต์ว่างของ PHP (Blank of PHP Script) เส้นสี่เหลี่ยมแทนแคชของหน้ารายละเอียด PHP (PHP Info Page) เส้นสามเหลี่ยมแทนแคชที่ถูกสร้างจากสคริปต์หน้าแรกของ phpMyAdmin (Index Page of phpMyAdmin) ซึ่งแคชที่ถูกสร้างจากสคริปต์หน้าแรกของ phpMyAdmin นั้นมีขนาดใหญ่ที่สุด จะสังเกตได้ว่าจำนวนการเกิดความผิดพร่องของกรณีที่ 3 เกิดขึ้นได้ยากยิ่งถ้าเป็นแคชที่มีขนาดใหญ่แนวโน้มของการเกิดความผิดพร่องของแคชทั้งสองจะเหมือนกันทุกประการนั้นเป็นไปได้ยากมาก

ตารางที่ 5-2 แสดงค่าเฉลี่ยของการเกิดความผิดพร่องในกรณีที่ 3

100 Time of 10,000 Fault Injection	**Blank of PHP Script**	**PHP Info Page**	**Index of phpMyAdmin**
Cache Size [Byte]	188	320	20,872
Number of Fault In Case 3 [Mean]	6.71	3.83	0.05
Number of Fault In Case 3 [Probability]	7.60	4.46	0.06

ผู้วิจัยได้นำผลการทดสอบจากการฉีดความผิดพร่อง เพื่อที่จะสังเกตแนวโน้มการเกิดความผิดพร่องในกรณีที่ 3 โดยในตารางที่ 5-2 แสดงค่าเฉลี่ยของการเกิดความผิดพร่องในกรณีที่ 3 (Number of Fault In Case 3 [Mean]) เปรียบเทียบกับจำนวนความน่าจะเป็นที่จะเกิดความผิดพร่องในกรณีที่ 3 (Number of Fault In Case 3 [Probability]) ที่ได้ทำการคำนวณมา มาทำการเฉลี่ยเพื่อดูจำนวนครั้งที่เกิดขึ้น จะสังเกตได้ว่าจำนวนเกิดความผิดพร่องในกรณีที่ 3 ของแคชมีความใกล้เคียงกับค่าของความน่าจะเป็นที่ได้ทำการคำนวณไว้ เช่นการฉีดความความผิดพร่องให้กับแคช ที่ถูกสร้างโดยหน้ารายละเอียดของ PHP ค่าเฉลี่ยเท่ากับ 3.83 ส่วนค่าความน่าจะเป็นที่คำนวณมาได้นั้นเท่ากับ 4.46

5.2 การประเมินประสิทธิภาพความเร็วของการประมวลผล

การทดสอบประสิทธิภาพความเร็วผู้วิจัยได้ใช้โปรแกรม ApacheBench [11] หรือที่เรียกว่า ab ซึ่งเป็นโปรแกรมที่ใช้ในการทดสอบประสิทธิภาพความเร็วในการทำงานของเว็บเซิร์ฟเวอร์ ApacheBench คือโปรแกรมคอมพิวเตอร์ที่ติดต่อผ่านทางบรรทัดคำสั่ง (Command Line Interface)

เป็นโปรแกรมที่ไม่เสียค่าใช้จ่ายและเปิดซอร์สโค้ด โดยอยู่ภายใต้ลิขสิทธิ์ของทีมงาน Apache โปรแกรม ApacheBench ใช้สำหรับวัดประสิทธิภาพของเว็บเซิร์ฟเวอร์โดยเฉพาะอย่างยิ่ง Apache เว็บเซิร์ฟเวอร์มันถูกออกแบบมาในมุมมองของการวัดประสิทธิภาพ ApacheBench เป็นโปรแกรมที่ติดมาด้วยกันกับเว็บเซิร์ฟเวอร์ Apache มันสามารถแสดงจำนวนของการร้องขอได้ว่าเว็บเซิร์ฟเวอร์นั้นสามารถรองรับจำนวนการร้องขอต่อวินาทีได้เป็นจำนวนมากเท่าใด คำสั่งหลักที่ใช้คำสั่ง ab ตามด้วยตัวเลือกต่างๆ ที่ผู้ทดสอบประสิทธิภาพจะเป็นผู้กำหนดแล้วตามด้วยหน้าเว็บที่ต้องการทำการทดสอบ คำสั่งในการทดสอบประสิทธิภาพความเร็วนั้นผู้วิจัยได้ใช้มาตรฐานเดียวกับการทดสอบในโครงการ Turck MMCache [6] ซึ่งเป็นการทดสอบการร้องขอไปที่เว็บเซิร์ฟเวอร์เป็นจำนวน 200 ครั้ง ต่างกันตรงที่ในการทดสอบของวิทยานิพนธ์นี้ไม่ได้มีการเปรียบเทียบประสิทธิภาพกับโปรแกรมไดนามิกเว็บแคชอื่นๆ แต่เป็นการเปรียบเทียบประสิทธิภาพการทำงานปกติกับการทำงานร่วมกับโปรแกรม eAccelerator และ eAccelerator CR แล้วยังมีเงื่อนไขของการฉีดความผิดพร่องร่วมด้วย

เว็บเซิร์ฟเวอร์ที่ใช้คือ Apache เวอร์ชัน 1.3.34 ทำงานอยู่บนเครื่องเลขที่ 202.44.37.34 ติดต่อสื่อสารทางพอร์ต 80 โดยกำหนดระดับการเข้าถึงพร้อมกัน (Concurrency Level) ของการร้องขอเท่ากับ 1 ซึ่งใช้จำนวนการร้องขอทั้งหมดเท่ากับ 200 ครั้ง

5.2.1 การทดสอบความเร็วที่หนึ่ง : ในการทดสอบแรกนี้ได้ทำการทดสอบกับสคริปต์ไฟล์ PHP โดยเป็นไฟล์สคริปต์หน้าแรกของเว็บแอปพลิเคชันที่มีการใช้งานอย่างแพร่หลายนั้นคือ phpMyAdmin ซึ่งเป็นเว็บแอปพลิเคชันที่ใช้ในการจัดการฐานข้อมูล MYSQL ผลการทดสอบประสิทธิภาพของเว็บเซิร์ฟเวอร์อยู่ในตารางที่ 5-3 สำหรับคำสั่งที่ใช้ในการทดสอบคือ

```
$ab –n200 http://202.44.37.34/phpmyadmin/index.php
```

ตารางที่ 5-3 แสดงผลการทดสอบประสิทธิภาพเว็บเซิร์ฟเวอร์กับหน้าแรกของ phpMyAdmin

Benchmarking Finished	**Normal**	**eAccelerator**	**eAccelerator CR**	**eAccelerator CR with One Fault Injection**
Time taken for tests: (seconds)	64.667824	21.680974	28.559148	28.675991
Document Length: (bytes)	13672	13672	13672	13672
Complete requests:	200	200	200	200
Failed requests:	0	0	0	0
Write errors:	0	0	0	0

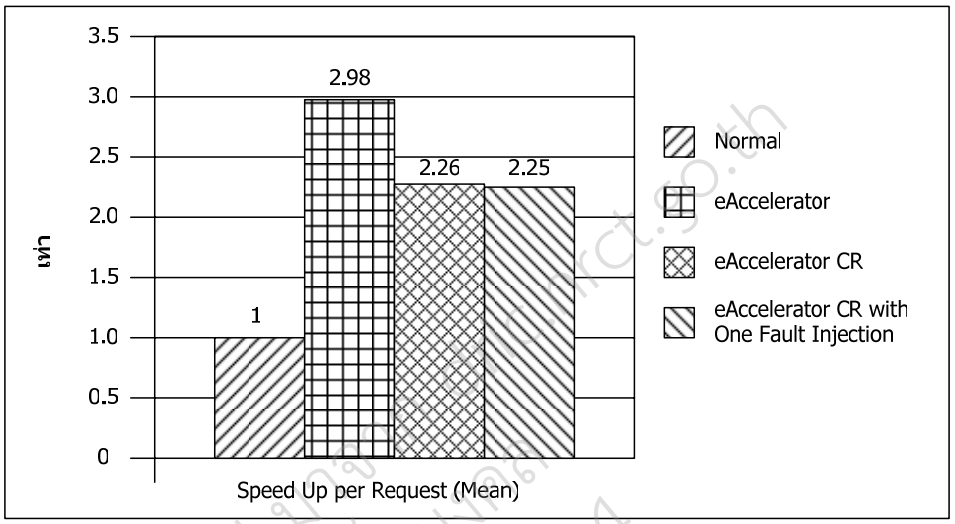
ตารางที่ 5-3 (ต่อ)

Benchmarking Finished	Normal	eAccelerator	eAccelerator CR	eAccelerator CR with One Fault Injection
Total transferred: (bytes)	2832200	2832200	2832200	2832200
HTML transferred: (bytes)	2734400	2734400	2734400	2734400
Requests per second: (mean)	3.09	9.22	7.00	6.97
Time per request: [ms] (mean)	323.339	108.405	142.796	143.380
Transfer rate: [Kbytes/sec]	42.76	127.53	96.82	96.42

จากผลการทดสอบแรกในตารางที่ 5-3 เป็นผลที่ได้จากโปรมแกรม ApacheBench ค่าที่น่าสนใจมี 4 ค่าคือเวลาทั้งหมดที่ใช้ในการทดสอบ (Time Taken for Tests), ปริมาณร้องขอต่อวินาที (Requests per Second) ซึ่งเป็นค่าเฉลี่ยของการร้องขอ 200 ครั้ง, ค่าเวลาที่ใช้ต่อการร้องขอหนึ่งครั้ง ซึ่งเป็นค่าเฉลี่ยเช่นเดียวกัน (Time per Request) และอัตราการส่งข้อมูล (Transfer Rate) แสดงเป็นกิโลไบท์ต่อวินาที ซึ่งค่าเวลาที่ใช้ต่อการร้องขอจากการทำงานปกติ (Normal) เท่ากับ 323.339 ms โดยเป็นการทำงานที่ช้าที่สุด การทำงานที่เร็วที่สุดคือการทำงานร่วมกับ eAccelerator ซึ่งใช้เวลาเท่ากับ 108.405 ms ส่วนเวลาที่ใช้ในการทำงานร่วมกับ eAccelerator CR มีค่าเท่ากับ 142.796 ms และเมื่อฉีดความผิดพร่องร่วมด้วยหนึ่งครั้ง (eAccelerator CR with One Fault Injection) เวลาที่ใช้ทำงานนั้นจะเพิ่มขึ้นเล็กน้อยโดยมีค่าเท่ากับ 143.380 ms โดยสามารถสรุปประสิทธิภาพการทำงานของเว็บเซิร์ฟเวอร์ดังแสดงในภาพที่ 5-3

ภาพที่ 5-3 เป็นการสรุปทดสอบประสิทธิภาพในด้านของความเร็วเปรียบเทียบกับความเร็วในระดับปกติซึ่งเป็นการแสดงประสิทธิภาพความเร็วโดยเฉลี่ยต่อการร้องขอ (Speed Up per Request [Mean]) โดยจะสังเกตได้จากค่าเวลาในระดับปกติ (Normal) ที่ใช้ต่อการร้องขอหนึ่งครั้ง (Time per request) ในตารางที่ 5-3 ซึ่งได้ค่าเวลาที่ได้เท่ากับ 323.339 ms ส่วนค่าเวลาต่อการร้องขอของการทำงานร่วมกับ eAccelerator นั้นเท่ากับ 108.405 ms โดยเมื่อเทียบความเร็วที่ระดับปกติเป็น 1 เท่า ก็จะทราบได้ว่า eAccelerator สามารถทำให้ความเร็วของเว็บเซิร์ฟเวอร์เพิ่มขึ้นเท่ากับ 323.339/108.405 = 2.98 เท่า หรือประมาณ 3 เท่าจากระดับการทำงานปกติ สำหรับการทำงานร่วมกับ eAccelerator CR ความเร็วในการทำงานนั้นลดต่ำลงกว่าการทำงานของ eAccelerator แต่ก็ยังเร็วกว่าการทำงานแบบปกติได้เท่ากับ 323.339/142.796 = 2.26 เท่า ในการทดสอบนั้นได้จงใจฉีด

ความผิดพร่องลงไปหนึ่งครั้งในระหว่างการร้องขอ 200 ครั้ง เพื่อเป็นการทดสอบว่าสามารถทนต่อความผิดพร่องที่เกิดขึ้นได้หรือไม่ ปรากฏว่าเมื่อมีการฉีดความผิดพร่องเข้ามาเพื่อทำให้เกิดความผิดพลาด eAccelerator CR นั้นสามารถตรวจสอบข้อผิดพลาดที่เกิดขึ้นได้ ทำให้เสียเวลาขึ้นอีกเล็กน้อยในการที่จะต้องคอมไพล์สคริปต์ใหม่อีกครั้งหนึ่ง



ภาพที่ 5-3 แสดงความเร็วในการทำงานของเว็บเซิร์ฟเวอร์กับหน้าแรกของ phpmyadmin

โปรแกรม eAccelerator ยังมีหน้าเว็บที่สามารถติดต่อกับโปรแกรมเพื่อแสดงได้เห็นว่าโปรแกรมนั้นได้มีการเปิดการทำงานอยู่หรือไม่ หรือฟังก์ชันใดบ้างที่ทำงานอยู่ ดังที่จะสังเกตได้จากตารางที่ 5-4 ผู้วิจัยได้เปิดการทำงานของแคช (Caching) แต่ไม่มีการเปิดใช้งานออปติไมส์เซอร์ (Optimizer) โปรแกรม eAccelerator ได้มีการใช้ทรัพยากรหน่วยความจำที่ได้ถูกแบ่งออกมา โปรแกรมได้ทำการจองหน่วยความจำ (Memory Size) เป็นจำนวน 32 MByte จากตารางที่ 5-4 ยังมีการแสดงถึง หน่วยความจำที่เหลืออยู่ (Memory Available) และหน่วยความจำที่ได้มีการใช้งานไปแล้ว (Memory Allocated) ซึ่งเป็นหน่วยความจำที่ถูกใช้ไปในตอนเริ่มต้น ซึ่งโปรแกรม eAccelerator ใช้หน่วยความจำเริ่มต้นเท่ากับ 2,816 ไบท์ ส่วน eAccelerator CR ใช้หน่วยความจำเริ่มต้นเท่ากับ 4,560 ไบท์

เมื่อมีร้องขอไฟล์สคริปต์หน้าแรกของ phpMyAdmin เข้ามาก็จะมีการใช้ทรัพยากรหน่วยความจำมากขึ้นเพื่อใช้ในการทำแคชหน้าแรกของโปรแกรม phpMyAdmin คือไฟล์ที่ชื่อว่า index.php เมื่อมีการร้องข้อสคริปต์ไฟล์ index.php เข้ามาก็มีการร้องขอสคริปต์ไฟล์อื่นเข้ามาอีก 24 ไฟล์ ซึ่งเป็นส่วนประกอบเดียวกันกับสคริปต์ไฟล์ index.php เมื่อมีการเปรียบเทียบปริมาณการใช้

ทรัพยากรหน่วยความจำของโปรแกรม eAccelerator และ eAccelerator CR จะสังเกตได้ว่า หน่วยความจำที่ถูกใช้ไปกับ eAccelerator CR ต้องใช้หน่วยความจำเพิ่มขึ้นมากกว่า eAccelerator เป็น 2 เท่า ซึ่งได้แสดงไว้ในตารางที่ 5-5

ตารางที่ 5-4 แสดงการใช้ทรัพยากรหน่วยความจำเริ่มต้น

	eAccelerator	**eAccelerator CR**
Caching	TRUE	TRUE
Optimizer	FALSE	FALSE
Memory Size (Bytes)	33,554,392	33,554,396
Memory Available (Bytes)	33,551,576	33,549,836
Memory Allocated (Bytes)	2,816	4,560

ตารางที่ 5-5 แสดงการใช้หน่วยความจำเมื่อมีการร้องขอจากหน้าแรกของ phpMyAdmin

	eAccelerator	**eAccelerator CR**
Caching	TRUE	TRUE
Optimizer	FALSE	FALSE
Memory Size (Bytes)	33,554,392	33,554,396
Memory Available (Bytes)	31,740,536	29,927,756
Memory Allocated (Bytes)	1,813,856	3,626,640

ทรัพยากรหน่วยความจำที่ใช้เพิ่มขึ้นเป็น 2 เท่านั้นเนื่องจากการทำสำเนาแคช เพื่อใช้ในตรวจสอบข้อผิดพลาดโดยการเปรียบเทียบแคชและแคชสำเนา ซึ่งโปรแกรม eAccelerator ใช้งานหน่วยความจำเริ่มต้นเท่ากับ 1,813,856 ไบท์ ส่วน eAccelerator CR ใช้หน่วยความจำเริ่มต้นเท่ากับ 3,626,640 ไบท์

5.2.2 การทดสอบความเร็วที่สอง : เป็นประเมินประสิทธิภาพความเร็ว ซึ่งเป็นการทดสอบที่สอง ซึ่งเป็นการทำทดสอบจากสคริปต์ไฟล์ที่ผู้วิจัยได้มีการสร้างขึ้นเอง ซึ่งต่างจากการทดสอบแรก (5.2.1) ที่ได้มีการใช้สคริปต์ไฟล์จาก phpMyAdmin ในการทดสอบนี้ได้มีการทดสอบกับสคริปต์ไฟล์ที่ eAccelerator สามารถทำความเร็วได้มากกว่าความเร็วของการทำงานปกติ 1 เท่า โดยประมาณกับสคริปต์ไฟล์ที่ได้มีการสร้างขึ้นเอง เพื่อเปรียบเทียบประสิทธิภาพกับ eAccelerator

CR การที่ eAccelerator จะสามารถทำความเร็วได้มากกว่าการทำงานปกติเท่าใดนั้นขึ้นอยู่การความซับซ้อน และขนาดของไฟล์สคริปต์ ในการทดสอบนี้ผู้วิจัยได้ทำการทดสอบกับสคริปต์ไฟล์ที่ eAccelerator สามารถทำความเร็วได้ 1 เท่าโดยประมาณ โดยให้ไฟล์ที่ชื่อว่า test1.php ผลการทดสอบประสิทธิภาพของเว็บเซิร์ฟเวอร์ อยู่ในตารางที่ 5-6 สำหรับคำสั่งที่ใช้ในการทดสอบคือ

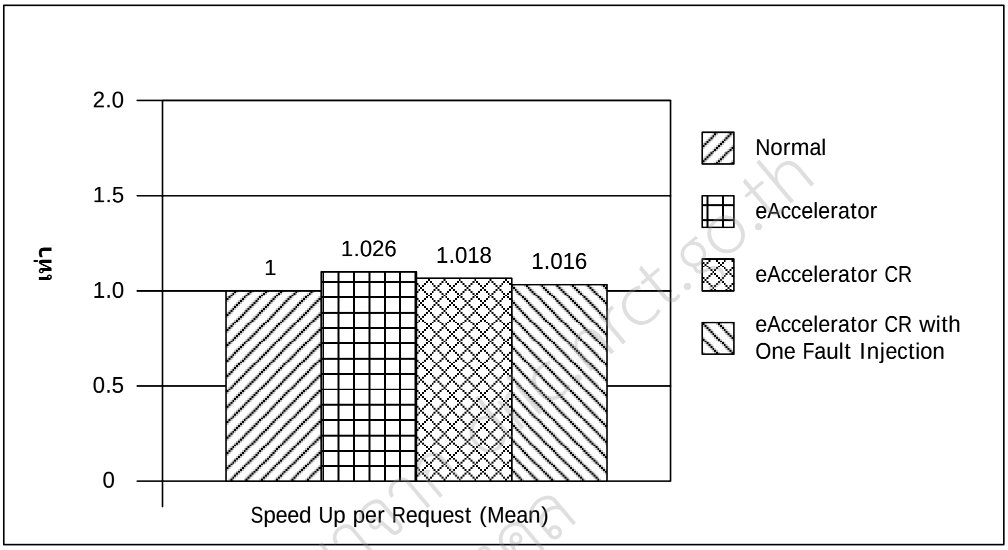
```
$ab -n200 http://202.44.37.34/test1.php
```

ตารางที่ 5-6 แสดงผลการทดสอบประสิทธิภาพเว็บเซิร์ฟเวอร์กับสคริปต์ไฟล์ test1.php

Benchmarking Finished	Normal	eAccelerator	eAccelerator CR	eAccelerator CR with One Fault Injection
Time taken for tests: (seconds)	0.448859	0.437290	0.440834	0.442543
Document Length: (bytes)	12	12	12	12
Complete requests:	200	200	200	200
Failed requests:	0	0	0	0
Write errors:	0	0	0	0
Total transferred: (bytes)	35400	35400	35400	35400
HTML transferred: (bytes)	2400	2400	2400	2400
Requests per second: (mean)	445.57	457.36	453.69	451.93
Time per request: [ms] (mean)	2.244	2.186	2.204	2.213
Transfer rate: [Kbytes/sec]	75.75	77.75	77.13	76.83

จากการทดสอบนี้ได้มีเงื่อนไขของการฉีดความผิดพร่องเข้าไปหนึ่งครั้งจากการร้องขอ 200 ครั้ง เช่นเดียวกับการทดลองที่แล้ว eAccelerator CR นั้นสามารถทนทานต่อความผิดพร่องที่ทำให้เกิดข้อผิดพลาดได้เช่นเดียวกัน ซึ่งค่าเวลาที่ใช้ต่อการร้องขอจากการทำงานปกติ เท่ากับ 2.244 ms โดยเป็นการทำงานที่ช้าที่สุด การทำงานที่เร็วที่สุดคือการทำงานร่วมกับ eAccelerator ซึ่งใช้เวลาเท่ากับ 2.186 ms ส่วนเวลาที่ใช้ในการทำงานร่วมกับ eAccelerator CR มีค่าเท่ากับ 2.204 ms เมื่อมีการฉีดความผิดพร่องร่วมด้วยหนึ่งครั้ง โดยเวลาที่ใช้ทำงานนั้นจะเพิ่มขึ้นเล็กน้อยโดยมีค่าเท่ากับ 2.213 ms โดยสามารถสรุปประสิทธิภาพการทำงานของเว็บเซิร์ฟเวอร์ดังแสดงในภาพที่ 5-4 สำหรับการคำนวณค่าความเร็วในภาพที่ 5-4 นั้นใช้หลักการเดียวกับการทดสอบความเร็วที่หนึ่ง

ภาพที่ 5-4 ความเร็วเฉลี่ยของ eAccelerator CR นั้นก็ยังเร็วกว่าการทำงานปกติอยู่เล็กน้อย เกือบจะไม่มีความแตกต่างจากการทำงานในระดับปกติเนื่องจากสคริปต์ไฟล์ test1.php นั้นมีความซับซ้อนของไฟล์ที่น้อย และมีขนาดที่เล็กมาก



ภาพที่ 5-4 แสดงความเร็วในการทำงานของเว็บเซิร์ฟเวอร์กับสคริปต์ไฟล์ test1.php

ตารางที่ 5-7 การใช้ทรัพยากรหน่วยความจำจากการทดสอบนี้ หลังจากการที่มีการร้องขอหน้าเว็บแล้ว ได้มีการใช้ทรัพยากรหน่วยความจำเพิ่มจากการใช้ทรัพยากรหน่วยความจำในตอนเริ่มต้น ดังที่แสดงในตารางที่ 5-4 เพียงเล็กน้อย

ตารางที่ 5-7 แสดงการใช้ทรัพยากรหน่วยความจำจากการร้องขอไฟล์สคริปต์ test1.php

	eAccelerator	**eAccelerator CR**
Caching	TRUE	TRUE
Optimizer	FALSE	FALSE
Memory Size (Bytes)	33,554,392	33,554,396
Memory Available (Bytes)	33,446,924	33,340,532
Memory Allocated (Bytes)	107,468	213,864

ซึ่งโปรแกรม eAccelerator ใช้หน่วยความจำเริ่มต้นเท่ากับ 107,468 ไบท์ ส่วน eAccelerator CR ใช้หน่วยความจำเริ่มต้นเท่ากับ 213,864 ไบท์ ทรัพยากรหน่วยความจำที่ใช้เพิ่มขึ้นเป็น 2 เท่านั้น เนื่องจากการทำสำเนาแคช เพื่อใช้ในตรวจสอบข้อผิดพลาดโดยการเปรียบเทียบแคชและแคชสำเนา

5.2.3 การทดสอบความเร็วที่สาม : เป็นการประเมินประสิทธิภาพความเร็วที่ได้ทำการทดสอบกับสคริปต์ไฟล์ที่ eAccelerator สามารถทำความเร็วได้มากกว่า 10 เท่าของการทำงานปกติโดยประมาณ โดยให้ไฟล์ที่ชื่อว่า test10.php ผลการทดสอบประสิทธิภาพของเว็บเซิร์ฟเวอร์ อยู่ในตารางที่ 5-8 สำหรับคำสั่งที่ใช้ในการทดสอบคือ

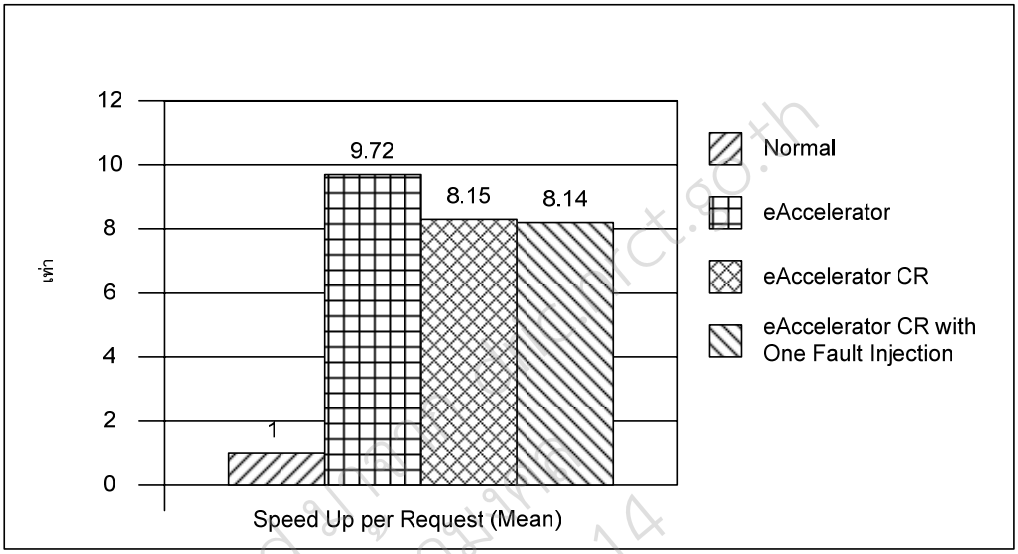
```
$ab -n200 http://202.44.37.34/test10.php
```

ตารางที่ 5-8 แสดงผลการทดสอบประสิทธิภาพเว็บเซิร์ฟเวอร์กับสคริปต์ไฟล์ test10.php

Benchmarking Finished	**Normal**	**eAccelerator**	**eAccelerator CR**	**eAccelerator CR with One Fault Injection**
Time taken for tests: (seconds)	186.643914	19.185719	22.893626	22.928385
Document Length: (bytes)	1045894	1045894	1045894	1045894
Complete requests:	200	200	200	200
Failed requests:	0	0	0	0
Write errors:	0	0	0	0
Total transferred: (bytes)	209211800	209211800	209211800	209211800
HTML transferred: (bytes)	209178800	209178800	209178800	209178800
Requests per second: (mean)	1.07	10.42	8.74	8.72
Time per request: [ms] (mean)	933.220	95.929	114.468	114.642
Transfer rate: [Kbytes/sec]	1094.64	10648.96	8925.23	8910.70

จากการทดสอบนี้ได้มีเงื่อนไขของการฉีดความผิดพร่องเข้าไปหนึ่งครั้งจากการร้องขอ 200 ครั้ง เช่นเดียวกับการทดสอบที่แล้ว eAccelerator CR นั้นสามารถตรวจสอบหาความผิดพลาดที่เกิดขึ้น ทำให้สามารถทนทานต่อความผิดพร่องที่ทำให้เกิดข้อผิดพลาดได้เช่นเดียวกัน ซึ่งค่าเวลาที่ใช้ต่อการร้องขอจากการทำงานปกติเท่ากับ 933.220 ms โดยเป็นการทำงานที่ช้าที่สุด การทำงานที่เร็วที่สุดคือการทำงานร่วมกับ eAccelerator ซึ่งใช้เวลาเท่ากับ 95.929 ms ส่วนเวลาที่ใช้ในการ

ทำงานร่วมกับ eAccelerator CR มีค่าเท่ากับ 114.468 ms และเมื่อมีการฉีดความผิดพร่องร่วมด้วยหนึ่งครั้ง ซึ่งเวลาที่ใช้ทำงานนั้นจะเพิ่มขึ้นเล็กน้อยโดยมีค่าเท่ากับ 114.642 ms โดยสามารถสรุปประสิทธิภาพการทำงานของเว็บเซิร์ฟเวอร์ดังแสดงในภาพที่ 5-5 การคำนวณค่าความเร็วในภาพที่ 5-5 นั้นใช้หลักการเดียวกับการทดสอบความเร็วที่หนึ่ง



ภาพที่ 5-5 แสดงความเร็วในการทำงานของเว็บเซิร์ฟเวอร์กับสคริปต์ไฟล์ test10.php

ภาพที่ 5-5 ความเร็วเฉลี่ยของ eAccelerator CR มีความเร็วน้อยลงมากกว่า eAccelerator แต่ก็ยังเร็วกว่าการทำงานปกติอยู่มากถึง 8.15 เท่า จากการทำงานในระดับปกติเนื่องจากสคริปต์ไฟล์ test10.php นั้นขนาดที่ค่อนข้างใหญ่

ตารางที่ 5-9 แสดงการใช้ทรัพยากรหน่วยความจำจากการร้องขอไฟล์สคริปต์ test10.php

	eAccelerator	**eAccelerator CR**
Caching	TRUE	TRUE
Optimizer	FALSE	FALSE
Memory Size (Bytes)	33,554,392	33,554,396
Memory Available (Bytes)	32,505,424	31,457,532
Memory Allocated (Bytes)	1,048,968	2,096,864

ตารางที่ 5-9 แสดงการใช้ทรัพยากรหน่วยความจำจากการร้องขอไฟล์สคริปต์ test10.php หน่วยความจำที่ใช้ไปในการทำแคชนั้นมากกว่าการทดสอบที่สองประมาณ 10 เท่า และการใช้หน่วยความจำของ eAccelerator CR นั้นมากกว่า eAccelerator เป็น 2 เท่าเช่นเดียวกับ การทดสอบที่แรก (5.2.1) และการทดสอบที่สอง (5.2.2)

การประเมินประสิทธิภาพที่ได้ทำการทดสอบทั้งหมดนั้น ได้มีการเปรียบเทียบในเรื่องของประสิทธิภาพในด้านความเร็วของเว็บเซิร์ฟเวอร์เมื่อมีการร้องขอเป็นจำนวน 200 ครั้งเพื่อเป็นการเฉลี่ยเวลาของการคอมไพล์ในครั้งแรก และเฉลี่ยเวลาของการส่งข้อมูลและรับข้อมูลในแต่ละการร้องขอและการรับข้อมูล ที่อาจมีสภาพแวดล้อมการจราจรของเครือข่ายที่แตกต่างกันอยู่เล็กน้อย สรุปผลของการทดสอบ eAccelerator CR นั้น eAccelerator CR ความทนทานต่อความผิดพร่องที่ทำให้เกิดข้อผิดพลาดได้ และมีการใช้ทรัพยากรหน่วยความจำมากกว่า eAccelerator เป็นจำนวน 2 เท่า ส่วนในด้านของประสิทธิภาพความเร็วของ eAccelerator CR เมื่อเปรียบเทียบกับ eAccelerator และการทำงานปกตินั้น eAccelerator CR มีการทำงานที่ช้ากว่า eAccelerator แต่ก็ยังสามารถทำงานได้เร็วกว่าการทำงานแบบปกติ ซึ่งขึ้นอยู่กับความซับซ้อนและขนาดของสคริปต์ไฟล์

บทที่ 6
สรุปผลการวิจัยและงานในอนาคต

การใช้งานอินเทอร์เน็ตของประชากรอินเทอร์เน็ตมีอัตราที่เพิ่มมากขึ้นเรื่อยๆ การใช้งานเว็บนั้นก็มีอัตราที่เพิ่มขึ้นด้วยเช่นกัน ไม่ว่าจะเป็นเว็บสถิตหรือว่าเว็บไดนามิก ไม่เพียงแต่จำนวนของผู้ใช้งานที่เพิ่มมากขึ้นเท่านั้น แต่ความเชื่อถือได้ของการใช้งานเว็บก็เป็นสิ่งที่ต้องการเพิ่มมากขึ้นเช่นกัน เนื่องจากการในงานเว็บแอปพลิเคชันถ้ามีความผิดพลาดเกิดกับผู้ใช้งาน จะมีประเด็นความน่าเชื่อถือขององค์กรหรือบริษัทที่เป็นผู้ให้บริการแอปพลิเคชันนั้นด้วย ซึ่งถ้าเป็นองค์กรที่มีชื่อเสียงหรือบริษัทที่เกี่ยวข้องกับระบบคอมพิวเตอร์ด้วยแล้ว จะทำให้ความน่าเชื่อถือขององค์กรหรือบริษัทนั้นลดลงอย่างมาก

แต่ก็ยังคงปฏิเสธไม่ได้ว่าความผิดพลาดในระบบคอมพิวเตอร์นั้นก็ยังคงมีอยู่ ไม่ว่าจะเป็นทางด้านซอฟต์แวร์ เนื่องจากการเรียบเร่งในการพัฒนาแอปพลิเคชันให้ทันกับเวลาที่จะต้องส่งงาน หรือทางด้านฮาร์ดแวร์ซึ่งเป็นเหตุผลทางด้านกายภาพซึ่งไม่สามารถควบคุมได้ เช่นการเสื่อมสภาพของวัสดุ หรือการประสบพบเจอกับอุปกรณ์ที่มีคุณภาพต่ำ ซึ่งหน่วยความจำนั้นเป็นหน่วยหนึ่งของระบบคอมพิวเตอร์เช่นกัน ซึ่งใช้ในการเก็บข้อมูลต่างๆ ขึ้นอยู่กับแอปพลิเคชันที่ใช้งานว่าต้องการใช้ทรัพยากรหน่วยความจำนั้นมากน้อยเพียงใด

สำหรับแอปพลิเคชันที่ใช้สำหรับวิทยานิพนธ์นี้มีความเกี่ยวข้องกับเว็บ ซึ่งมีความเป็นพลวัตของรายละเอียดที่อยู่ในหน้าเว็บ ซึ่งเป็นผลจากการทำงานของสคริปต์ทางด้านฝั่งเซิร์ฟเวอร์ ทำให้มีการประมวลผลทางด้านฝั่งเซิร์ฟเวอร์มีมากขึ้น ภาษาสคริปต์ทางฝั่งเซิร์ฟเวอร์ที่ผู้วิจัยใช้ในวิทยานิพนธ์นี้คือ PHP ซึ่งเป็นภาษาสคริปต์ทางฝั่งเซิร์ฟเวอร์ที่มีผู้ใช้มากที่สุดในโลก ทั้งยังสามารถพัฒนาส่วนขยายให้กับ PHP เพื่อให้ PHP มีความสามารถที่เพิ่มมากขึ้น เหตุผลหนึ่งในการพัฒนาส่วนขยายนั้นคือการทำให้ PHP มีการประมวลผลที่เร็วมากยิ่งขึ้น โดยการแคชสคริปต์ PHP ซึ่งได้มีการประมวลผลไว้แล้วในการร้องขอหน้าเว็บครั้งแรก และนำแคชมาใช้ในครั้งต่อไป ส่วนขยายที่ใช้ในการแคชสคริปต์ PHP นั้นก็มีหลายโปรแกรม ในวิทยานิพนธ์นี้ได้มีการเลือกใช้โปรแกรมที่ชื่อว่า eAccelerator ซึ่งเป็นโปรแกรมที่มีการพัฒนาอย่างต่อเนื่อง ซึ่งในปัจจุบันนี้ก็ยังมีการพัฒนาอยู่ โปรแกรม eAccelerator นั้นเป็นเป็นโปรแกรมที่ไม่เสียค่าใช้จ่ายและยังเปิดซอร์สโค้ดอีกด้วย ทำให้สามารถแก้ไขหรือปรับปรุงอยู่ภายใต้อนุสัญญาของ GNU General Public License นี้ก็เป็นอีกเหตุผลหนึ่งที่ทำให้ผู้วิจัยเลือกใช้โปรแกรม eAccelerator กับวิทยานิพนธ์นี้

จากความต้องการในด้านความเร็วของการประมวลผลเพื่อให้มีความเร็วมากยิ่งขึ้น จึงมีการทำแคชเกิดขึ้น ซึ่งแคชที่ถูกสร้างขึ้นมานั้นได้มีการเก็บไว้ในหน่วยความจำ ในวิทยานิพนธ์นี้ผู้วิจัยได้ศึกษาในเรื่องของความผิดพร่องที่ทำให้เกิดความผิดพลาดขึ้นในหน่วยความจำ ซึ่งการจากศึกษาผู้วิจัยได้พบว่าความผิดพร่องที่เกิดขึ้นกับหน่วยความจำ นั้นเป็นความผิดพร่องทางด้านกายภาพในเรื่องของรังสีวิทยา ซึ่งเป็นการแตกตัวทางด้านรังสีวิทยา ทำให้เกิดอนุภาคแอลฟา เมื่อมีการแตกตัวทางด้านรังสีวิทยาจากวัสดุห่อหุ้มสารกึ่งตัวนำแล้ว ก็จะสามารถทำให้ข้อมูลที่อยู่ในหน่วยจำผิดพลาดขึ้นให้ ซึ่งเป็นความผิดพลาดที่เกิดขึ้นชั่วคราว

ผู้วิจัยจึงได้นำเสนอวิธีการที่จะทำให้ ข้อมูลแคชที่เก็บไว้ในหน่วยความจำทนทานต่อความผิดพร่อง และวิธีการหรือเทคนิคที่ใช้คือการการเพิ่มความซ้ำซ้อนของข้อมูลข่าวสาร (Information Redundancy) ซึ่งก็คือการเพิ่มข้อมูลแคชสำเนาและสามารถนำแคชสำเนานั้นมาช่วยในการเพิ่มความทนทานต่อความผิดพร่อง โดยการนำแคชและแคชสำเนามาทำการเปรียบเทียบเพื่อตรวจสอบข้อผิดพลาดที่เกิดขึ้น ก่อนที่จะนำข้อมูลแคชนั้นไปทำการ Execute เมื่อมีการตรวจสอบพบความผิดพลาดขึ้นในแคช ก็จะมีการสั่งให้ลบแคชทั้งสองที่มีการตรวจสอบพบความผิดพลาดและทำการคอมไพล์ ไฟล์สคริปต์ที่มีการร้องข้อเข้ามานั้นใหม่อีกครั้งหนึ่ง

ซึ่งการทดสอบระบบเพื่อหาประสิทธิภาพนั้น ผู้วิจัยได้ประเมินประสิทธิภาพของระบบตรวจสอบความผิดพลาดและประเมินสิทธิภาพความเร็วของการประมวลผล สำหรับการทดสอบระบบตรวจสอบความผิดพลาด (5.1) นั้นผู้วิจัยได้ใช้วิธีการการฉีดความผิดพร่อง (Fault Injection) เข้าไปในหน่วยความจำในตำแหน่งที่เก็บแคช การฉีดความผิดพร่องเข้าไปในแคชนั้นผู้วิจัยได้ทำการที่จะจงใจทำความผิดพลาดในระดับบิต (Bit Error) เข้าไปในแคชที่ได้มีการร้องขอเข้ามาและเป็นลักษณะของการกลับบิตของข้อมูล (Bit Flip) ซึ่งวิธีการนี้จะเป็นการที่ทำให้ข้อมูลวิบัติ (Corrupt) และจากนั้นก็ทำการประเมินผลการทดสอบ

สำหรับในการทดสอบเพื่อประเมินสิทธิภาพความเร็ว (5.2) ได้มีการใช้เครื่องมือมาทำการวัดประสิทธิภาพของเว็บเซิร์ฟเวอร์ มีทั้งหมดสามการทดสอบ สำหรับการทดสอบแรกคือ การทดสอบประสิทธิภาพความเร็วของเว็บเซิร์ฟเวอร์ กับสคริปต์ไฟล์ของเว็บแอปพลิเคชันที่มีใช้งานกันอย่างแพร่หลาย (5.2.1) สำหรับส่วนที่สองได้ทำการทดสอบประสิทธิภาพของเว็บเซิร์ฟเวอร์กับสคริปต์ไฟล์ที่ได้สร้างขึ้นเอง (5.2.2) การทดสอบเพื่อที่จะได้เห็นความแตกต่างระหว่างไฟล์ที่ประมวลผลที่เร็วกับไฟล์ที่ประมวลผลได้ช้ากว่าถึงประมาณ 10 เท่า ซึ่งอยู่ในการทดสอบที่สาม (5.2.3) ทั้งนี้ในการทดสอบทั้งหมดยังได้มีการฉีดความผิดพร่องไปในข้อมูลแคช หรือแคชสำเนากับระบบที่ผู้วิจัยได้ทำการปรับปรุงได้นั้นก็คือ eAccelerator CR เพื่อเป็นการทดสอบระบบตรวจสอบความผิดพลาด

ผลที่ได้สามารถสรุปได้ดังนี้ สำหรับระบบตรวจสอบความผิดพลาดสามารถตรวจสอบความผิดพลาดได้ แต่ไม่ใช่ในกรณีที่มีความผิดพร่องที่เกิดกับแคชและแคชสำเนาเหมือนกันทุกประการ ซึ่งเป็นกรณีที่เกิดขึ้นได้ยากมาก

สำหรับความเร็วในการประมวลผล การทดสอบแรก eAccelerator CR สามารถทำงานได้เร็วว่าการทำงานแบบปกติ 2.26 เท่าโดยเฉลี่ย การทดสอบที่สองนั้น eAccelerator CR สามารถทำงานได้เร็วกว่าการทำงานแบบปกติเล็กน้อย การทดสอบที่สาม eAccelerator CR สามารถทำงานได้เร็วกว่าการทำงานแบบปกติ 8.15 เท่า ทั้งสามการทดสอบนี้สามารถตรวจสอบความผิดพลาดที่เกิดขึ้นได้ สำหรับการใช้หน่วยความจำนั้น eAccelerator CR ใช้หน่วยความจำมากกว่า eAccelerator เป็น 2 เท่า

งานในอนาคตที่น่าสนใจคือการทำให้แคชมีการใช้หน่วยความจำที่น้อยลงกว่าเดิม จากที่ได้มีการจองหน่วยความจำเพิ่มเพื่อทำสำเนาแคช โดยอาจใช้จะเทคนิคอื่นๆที่ใช้ในการตรวจสอบความผิดพลาด เพื่อทำส่วนที่ซ้ำซ้อนในการตรวจสอบความถูกต้องของแคช หรือนำดิสแคชมาทำการตรวจสอบความผิดพลาดร่วมด้วยเพื่อเป็นการลดเวลาในการประมวลผลซ้ำ

เอกสารอ้างอิง

1. J. Offutt. "Quality attributes of Web software applications." **IEEE Software, Special Issue on Software Engineering of Internet Software**. March/April 2002 : 25-32.

2. A. Iyengar and J. Challenger. "Improving Web Server Performance by Caching Dynamic Data." **In Proceedings of the USENIX Symposium on Internet Technologies and Systems**. December 1997.

3. **PHP and Zend Engine Internals**. PHP Manual. [2006 May].
   Available from : http://th2.php.net/manual/en/

4. Sara Golemon. **Extending and Embedding PHP**. Indianapolis : Sams Publishing, 2006.

5. **eAccelerator Project**. [2006 March]. Available from : http://eaccelerator.net

6. **Track MMCache Project**. [2006 March].
   Available from : http://turck-mmcache.sourceforge.net/index_old.html

7. L. Lantz. "Soft Errors Induced By Alpha Particles." **IEEE Transaction on Reliability**. 1996 : 174-179.

8. "Soft errors in electronic memory." **Tezzaron Semiconductor**. January 2004.

9. **Apache Memory Consumption with MOD PHP**. [2007 January].
   Available from : http://www.webmasterworld.com/apache/3141749.htm

10. K. Pradhan Dhiraj K. **Fault-tolerant computer system design**. Upper Saddle River, New Jersey : Prentice Hall, 1996.

11. **ApacheBench**. Apache HTTP Server ManualPage. [2006 August].
    Available from : http://httpd.apache.org/docs/1.3/programs/ab.html

ประวัติผู้วิจัย

ชื่อ : นายศุภวัฒน์ แก้วมงคล
ชื่อวิทยานิพนธ์ : การทนต่อความผิดพร่องในไดนามิกเว็บโดยใช้แคช
สาขาวิชา : วิศวกรรมไฟฟ้า

ประวัติ

ประวัติส่วนตัว เกิดเมื่อวันศุกร์ที่ 14 เดือนมีนาคม พ.ศ. 2523 ปัจจุบันอยู่บ้านเลขที่ 12/6 ถ.เชื่อมสัมพันธ์ แขวงโคกแฝด เขตหนองจอก จ.กรุงเทพมหานคร 10530

ประวัติการศึกษา สำเร็จการศึกษาในระดับประกาศนียบัตรวิชาชีพ หรือ ปวช. ในสาขาอิเล็กทรอนิกส์จากวิทยาลัยเทคนิคฉะเชิงเทรา สำเร็จการศึกษาในระดับปริญญาตรีในสาขาวิศวกรรมไฟฟ้าจากคณะครุศาสตร์อุตสาหกรรม หรือ คอบ. จากสถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ