ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์)

ปริญญา

วิศวกรรมคอมพิวเตอร์ วิศวกรรมคอมพิวเตอร์

สาขา **ภาควิชา**

เรื่อง การตรวจสอบการขัดแย้งกันของการให้บริการในระบบเครือข่ายบ้านอัจฉริยะด้วยสปินโมเดลเช็คกิ้งโดยการประยุกต์ใช้เทคโนโลยีกริด

Detection of Feature Interaction in Home Network System with SPIN Model Checking by Adapting Grid Technology

นามผู้วิจัย ว่าที่ร้อยตรีวรวุทธ นัมคนิสรณ์

ได้พิจารณาเห็นชอบโดย

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก .....

(อาจารย์ภัทร ลีลาพฤทธิ์, Ph.D.)

อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม .....

(ผู้ช่วยศาสตราจารย์อานนท์ รุ่งสว่าง, Ph.D.)

หัวหน้าภาควิชา .....

(ผู้ช่วยศาสตราจารย์ภุชงค์ อุทโยภาศ, Ph.D.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

.....

(รองศาสตราจารย์กัญจนา ธีระกุล, D.Agr.)

คณบดีบัณฑิตวิทยาลัย

วันที่ ..... **เดือน** ..... **พ.ศ.** .....

วิทยานิพนธ์

เรื่อง

การตรวจสอบการขัดแย้งกันของการให้บริการในระบบเครือข่ายบ้านอัจฉริยะด้วยสปินโมเดลเช็คกิ้งโดยการประยุกต์ใช้เทคโนโลยีกริด

Detection of Feature Interaction in Home Network System with SPIN Model Checking by Adapting Grid Technology

โดย

ว่าที่ร้อยตรีวรวุทธ นัมคนิสรณ์

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์
เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์)
พ.ศ. 2554

วรวุทธ นัมคนิสรณ์, ว่าที่ร้อยตรี 2554: การตรวจสอบการขัดแย้งกันของการให้บริการในระบบเครือข่ายบ้านอัจฉริยะด้วยสปินโมเดลเช็คกิ้งโดยการประยุกต์ใช้เทคโนโลยีกริด ปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์) สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก: อาจารย์ภัทร ลีลาพฤทธิ์, Ph.D. 56 หน้า

งานวิจัยนี้นำเสนอการประยุกต์ใช้เทคโนโลยีกริดในการตรวจหาปัญหาฟีเจอร์อินเตอร์แอคชั่นหรือปัญหาการขัดแย้งกันระหว่างการให้บริการของระบบเครือข่ายบ้านอัจฉริยะด้วยเครื่องมือ โมเดลเช็คกิ้งสปิน เนื่องจากในการใช้งานจริงของระบบเครือข่ายบ้านอัจฉริยะนั้นเมื่อจำนวนของบริการหรือเซอร์วิสและจำนวนของผู้ใช้งานหรือยูสเซอร์ในระบบมีเพิ่มมากขึ้นจะส่งผลให้การตรวจหาฟีเจอร์อินเตอร์แอคชั่นซึ่งเป็นปัญหาการขัดแย้งกันระหว่างฟังค์ชั่นของเซอร์วิสเป็นไปด้วยความยากลำบากและเกิดความล่าช้าซึ่งอาจนานจนไม่สามารถตรวจหาให้แล้วเสร็จได้ในเวลาจริง ดังนั้นงานวิจัยนี้จึงนำเสนอการประยุกต์ใช้เทคโนโลยีกริดซึ่งเป็นหนึ่งในการประมวลผลแบบขนาน ในการลดระยะเวลารวมทั้งทรัพยากรในการตรวจหาฟีเจอร์อินเตอร์แอคชั่น เพื่อให้ได้ผลการตรวจหาที่รวดเร็วและสมบูรณ์

_____ _____ ___ / ___ / ___

ลายมือชื่อนิสิต ลายมือชื่ออาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

Varavuth Numkanisorn, Acting SubLt. 2011: Detection of Feature Interaction in Home Network System with SPIN Model Checking by Adapting Grid Technology. Master of Engineering (Computer Engineering), Major Field: Computer Engineering, Department of Computer Engineering. Thesis Advisor: Mr. Pattara Leelaprute, Ph.D. 56 pages.

This research proposed the application of the grid technology in detecting feature interaction problems or conflicting problems between the services of Home Network System with SPIN model checking tool. During actual use of the Home Network System, when the number of the services and/or the number of users in the system increases, detection of feature interaction, which is the conflict between the functions of the services, would become more difficult and time-consuming, making it not possible to complete the task in real time. Therefore, this research proposes the application of Grid Technology, which is one of the parallel processing methods, to reduce the time and resources in detecting feature interaction so as to achieve faster and more exhaustive detection.

_____ _____ ___/___/___

Student's signature Thesis Advisor's signature

กิตติกรรมประกาศ

กราบขอบพระคุณ อาจารย์ภัทร ลีลาพฤทธิ์ ประธานกรรมการที่ปรึกษาวิทยานิพนธ์ และผู้ช่วยศาสตราจารย์ อานนท์ รุ่งสว่าง กรรมการที่ปรึกษาวิชาเอก ที่ให้คำปรึกษาในการเรียน การค้นคว้าการวิจัย ตลอดจนการตรวจแก้ไขวิทยานิพนธ์จนกระทั่งเสร็จสมบูรณ์และขอกราบขอบพระคุณผู้แทนบัณฑิตวิทยาลัย ที่ให้ความกรุณาตรวจแก้ไขวิทยานิพนธ์ให้สมบูรณ์ยิ่งขึ้น

วิทยานิพนธ์นี้ได้รับการสนับสนุนจากห้องปฏิบัติการวิจัยระบบเครือข่ายบ้านอัจฉริยะและการขัดแย้งกันของเซอร์วิส ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์

วรวุทธ นัมคนิสรณ์

พฤษภาคม 2554

สารบัญ

หน้า

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

คำอธิบายสัญลักษณ์และคำย่อ

HNS	=	Home Network System
FI	=	Feature Interaction
Promela1	=	Process or Protocol Meta Language
LTL	=	Linear temporal logic
MPI	=	Message Passing Interface

การตรวจสอบการขัดแย้งกันของการให้บริการในระบบเครือข่ายบ้านอัจฉริยะด้วยสปิน โมเดลเช็คกิ้งโดยการประยุกต์ใช้เทคโนโลยีกริด

Detection of Feature Interaction in Home Network System with SPIN Model Checking by Adapting Grid Technology

คำนำ

เทคโนโลยีในยุคปัจจุบันล้วนแต่มี Software ที่ใช้ในการควบคุมและจัดการอุปกรณ์ต่างๆ ให้เป็นไปตามคำสั่งที่ผู้ใช้บริการต้องการ แต่เราจะรู้ได้อย่างไรว่า Software หรืออุปกรณ์ต่างๆ เหล่านั้น จะสามารถทำงานได้สอดคล้องหรือตรงตามที่ผู้ใช้ต้องการ ในโครงการวิจัยนี้ได้เสนอแนวคิดรูปแบบหนึ่งที่ใช้ในการตรวจสอบการทำงานของ Integrated Services of Home Network System โดยอาศัยการจำลองการทำงานผ่าน Tool ที่เรียกว่า Model checking tool เพื่อทำการตรวจสอบการทำงานว่าเกิดการขัดแย้งกันขึ้นหรือไม่

เราได้จำลองการทำงานของระบบการให้สั่งงานผ่านอุปกรณ์เครื่องใช้ไฟฟ้าที่เชื่อมต่อกันภายในบ้าน เราเรียกอุปกรณ์เครื่องใช้ไฟฟ้าที่เชื่อมต่อกันภายในบ้านว่า Home Network System (HNS) (Nakamura *et al*., 2004) และ Service คือ การรวมเอาคุณสมบัติและความสามารถหรือฟีเจอร์ (Feature) ที่มีในอุปกรณ์ภายในบ้านต่างๆ มาใช้งานร่วมกัน ทำให้เกิดการบริการที่เป็นไปตามความต้องการของผู้ใช้ ถึงแม้ Services จะถูกเรียกว่า Integrated Services ในระบบ HNS และก็ทำงานได้เป็นอย่างดี แต่เราไม่รู้ว่าเมื่อไรการทำงานเหล่านั้นจะเกิดการขัดแย้งกัน เมื่อมีการทำงานของ Services พร้อมกัน เราเรียกการขัดแย้งกันนั้นว่า Feature Interactions (FI) ซึ่งการขัดแย้งกันนั้นอาจทำให้เกิดการทำงานที่ผิดพลาด อาจทำให้อุปกรณ์เครื่องใช้ไฟฟ้าเสียหายหรือการทำงานของอุปกรณ์เหล่านั้นผิดไปจากที่ผู้ใช้บริการคาดหวังไว้ ซึ่งในการตรวจสอบและวินิจฉัยความถูกต้อง (Verification & Validation) เราอาจจะตรวจสอบจาก code ที่ใช้เขียน Software นั้นได้ แต่จะทำการตรวจสอบได้ยากและช้ามากเราจึงต้องใช้หลักการที่เรียกว่า Symbolic Model Checking มาช่วยตรวจสอบการทำงานนั้นผ่าน Software Tool ที่ใช้ในการจำลองพฤติกรรม (behavior) ของระบบ

การตรวจสอบการขัดแย้งกันของเซอร์วิสที่อยู่ใน HNS ใช้แนวทางการตรวจหา FI แบบโมเดลเช็คกิ้ง (Model checking technique) ซึ่งใช้เครื่องมือชื่อสปิน (Model checking tool; SPIN)

ในการจำลองโมเดล การจำลองอาศัยการเขียนตามรูปแบบเฟรมเวิร์ก (framework) ของ HNS ด้วยภาษาโปรเมลา (promela) ที่ SPIN รองรับ แต่การตรวจหา FI นั้นต้องจำลองความเป็นไปได้ทั้งหมดของเหตุการณ์ที่สามารถเกิดขึ้นได้ในระบบที่ยูสเซอร์แต่ละคนจะเลือกใช้งานเซอร์วิสในเวลาหนึ่ง เช่นในระบบนั้นมีจำนวนเซอร์วิส 2 เซอร์วิส มีจำนวนยูสเซอร์ 2 คน ดังนั้นเหตุการณ์ที่จะเกิดขึ้นได้คือ ยูสเซอร์ที่ 1 ใช้เซอร์วิส A ยูสเซอร์ที่ 2 ใช้เซอร์วิส B เขียนแทนด้วย U1 = A & U2 = B เพราะฉะนั้นเหตุการณ์ทั้งหมดที่สามารถเกิดขึ้นได้คือ U1 = A & U2 = A, U1 = B & U2 = B, U1 = A & U2 = B, U1 = B & U2 = A เหตุการณ์ที่สามารถเกิดขึ้นได้ในระบบนี้จึงมี 4 เหตุการณ์ ซึ่งต้องรอผลการตรวจหาไปทีละเหตุการณ์ ดังนั้นหากมีจำนวนเซอร์วิสและยูสเซอร์เพิ่มขึ้น เหตุการณ์ที่จะเกิดขึ้นได้ทั้งหมดก็จะเพิ่มขึ้นเป็นทวีคูณ ทำให้ต้องรอผลการตรวจหานานขึ้นจนไม่สามารถรอได้ หรือไม่สามารถตรวจหาได้เพราะทรัพยากรในระบบอาจไม่เพียงพอต่อการตรวจหา

ในโครงการวิจัยนี้ได้ตั้งสมมุติฐานว่า การจำลอง Services ที่เกิดในระบบ HNS ด้วย SPIN นั้น จะทำงานช้าลงไปอีกและจะช้าลงอย่างมาก เมื่อมีการเพิ่มขึ้นของ Services ที่ต้องจำลองและสาเหตุที่ Services ในระบบ HNS เพิ่มเพราะมีอุปกรณ์เครื่องใช้ไฟฟ้าที่สามารถเชื่อมกันได้ด้วยระบบ HNS มากขึ้น เนื่องจากการพัฒนาในระบบ HNS และอุปกรณ์เครื่องใช้ไฟฟ้าที่ถูกคิดขึ้นมาเพื่ออำนวยความสะดวกภายในบ้านมีเพิ่มมากขึ้น เช่น จากการพัฒนาการติดต่อสื่อสารข้อมูลในอุปกรณ์ไฟฟ้า (Kolberg *et al*., 2003) การพัฒนาหน่วยประมวลผลที่ใช้ในอุปกรณ์เครื่องใช้ไฟฟ้า เช่น เครื่อง Playstation 3 ที่มีหน่วยประมวลผล Cell (Riley *et al*., 2007) ที่ Cell สามารถแบ่งทรัพยากรในการประมวลผลร่วมกันเมื่อมี Cell ในอุปกรณ์ภายในบ้านหลายๆ เครื่องและการพัฒนาไลบรารี่มาช่วยในการเขียนโปรแกรมบนเครื่องใช้ไฟฟ้าทั้งของบริษัท SUN(J2ME) (2009) บริษัท MICROSOFT(.NET) (2009) ที่สามารถเขียนในอุปกรณ์ไฟฟ้าให้ทำงานตามคำสั่ง ทำให้มีอุปกรณ์ที่สามารถต่อรวมเข้ากับระบบ HNS มีมากขึ้น และเขียนเป็น Service ในระบบมากยิ่งขึ้น ด้วยเหตุนี้จะทำให้สมมุติฐานที่ว่า Services ในระบบ HNS เพิ่มขึ้นมีความเป็นจริงและทำให้เวลาในการตรวจสอบใน SPIN ช้าลงจนรอไม่ได้

ด้วยเหตุนี้โครงการวิจัยเพื่อพัฒนาการตรวจสอบการขัดแย้งกันของการให้บริการในระบบเครือข่ายบ้านอัจฉริยะด้วย SPIN Model checking โดยการนำแนวคิดการประมวลผลแบบขนาน (Parallel Computing) ด้วยการประยุกต์ใช้เทคโนโลยีกริด เพื่อแก้ไขการทำงานที่ช้าลงของ SPIN เมื่อมีการทำงานที่หนักขึ้นโดยจะนำเทคโนโลยีกริดมาช่วยในการแบ่งงานในการประมวลผลของหน่วยประมวลผลที่ SPIN ทำงานอยู่ในเครื่องเดียวมาช่วยกันประมวลผลผ่านเทคโนโลยีกริดและลดต้นทุนในการซื้อระบบใหม่ที่มีประสิทธิ์ภาพมากขึ้นแต่ใช้การทำงานจากหลายเครื่องแทนการ

ทำงานบนระบบ multi-core ที่ต้องซื้อใหม่ทำให้ลดต้นทุนในการทำงานลงไปได้ งานวิจัยนี้ได้การสนับสนุนจากศูนย์ไทยกริดแห่งชาติในการพัฒนา

วัตถุประสงค์

เพื่อพัฒนาการตรวจสอบการขัดแย้งกันของ Integrated Service Services ในระบบ Home Network System ด้วยเทคนิค Symbolic Model Checking โดยใช้โปรแกรม SPIN ที่นำเทคโนโลยีกริดมาช่วยในการประมวลผลโดยวัดประสิทธิภาพที่เพิ่มขึ้นจาก Execution time โดยเปรียบเทียบกับระบบการตรวจหาเดิมที่ทำการตรวจหาแบบ Sequential และ เปรียบเทียบระหว่างระบบ Multi-core กับ Cluster (กริด) โดยงานวิจัยนี้มีความคาดหวังว่าการตรวจสอบจะเร็วขึ้นตามจำนวนการทำงานแบบขนานที่เพิ่มขึ้น

การตรวจเอกสาร

ในบทนี้จะกล่าวถึงความรู้พื้นฐานและงานวิจัยที่เกี่ยวข้องกับการทำงานวิจัย โดยจะแบ่งเป็นส่วนคือ ส่วนที่ 1 อธิบายการทำงานของ Integrated Service ในระบบ Home Network System ส่วนที่ 2 อธิบายเทคนิคการทำการตรวจสอบและวินิจฉัยความถูกต้องของ Integrated Service ในระบบ Home Network System ส่วนที่ 3 อธิบายการทำงานและการพัฒนาใน SPIN ส่วนที่ 4 การนำเทคโนโลยีกริดมาใช้ในการตรวจสอบและวินิจฉัยความถูกต้องของ Integrated Service ในระบบHome Network System

ส่วนที่ 1 การทำงานของ Integrated Service ในระบบ Home Network System

Home Network System เป็นงานวิจัยที่ถูกพัฒนามาแล้วหลายๆ รูปแบบหลายๆ แนวความคิด แต่ในงานวิจัยนี้จะหยิบแนวความคิดของ Nakamura *et al.* (2004) และ Leelaprute *et al.* (2005) คือการเชื่อมต่อกันของอุปกรณ์ไฟฟ้าภายในบ้านเพื่อดึงเอาคุณสมบัติและการทำงาน (Feature & Functions) มาใช้ร่วมกันทำให้เกิด Service

ตัวอย่างของ Service

1. HVAC Service: เป็น Service ที่ใช้ควบคุมอุณหภูมิภายในห้อง

2. Air-Cleaning Service: เป็น Service ที่ใช้เพื่อทำให้อากาศภายในห้องสะอาด

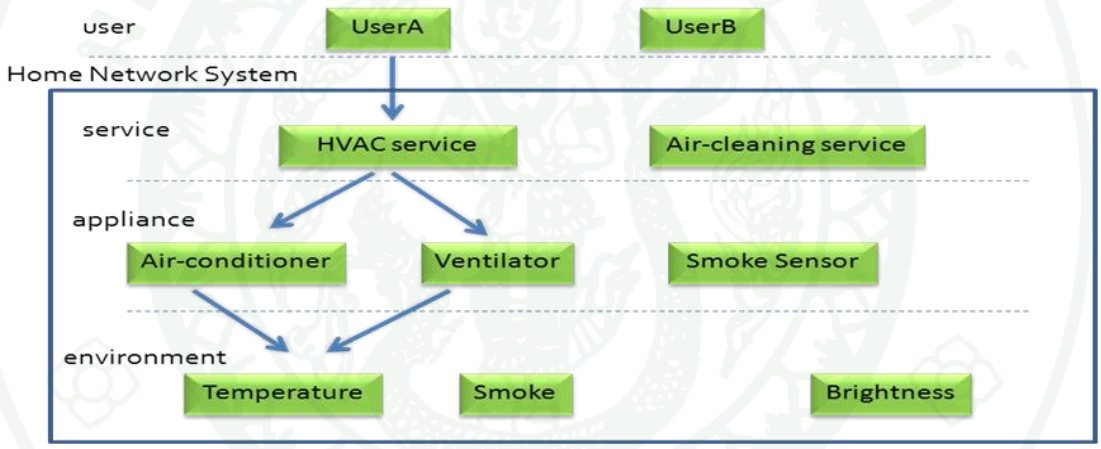
3. DVD Theater Service: เป็น Service ที่ใช้ควบคุมเครื่องเล่น DVD

4. Energy-saving Service: เป็น Service ที่ใช้ควบคุมอุปกรณ์เครื่องใช้ไฟฟ้าให้ประหยัดพลังงาน

5. Auto-Light Service: เป็น Service ที่ใช้ควบคุมความสว่างภายในห้อง

โดยที่นำรูปแบบนี้มาพัฒนาต่อก็เพราะ รูปแบบนี้เป็น Object Oriented ที่มองอุปกรณ์เครื่องใช้ไฟฟ้าเป็นวัตถุ (Object) ที่ประกอบไปด้วย property และ method สามารถนำไปพัฒนาต่อได้ง่าย และไม่ขึ้นกับ protocol ใดๆ ที่ใช้ในการเชื่อมต่ออุปกรณ์ไฟฟ้า

โดยรูปแบบนี้ได้อธิบายความสัมพันธ์ขององค์ประกอบต่างๆของระบบเป็นระดับชั้น ดังภาพที่ 1 อธิบายการทำงานของระบบ HNS ในแต่ละระดับชั้น ตัวอย่างเช่น HVAC Service และ Air-cleaning Service อยู่ในชั้นของ service ที่ถูกเรียกใช้งานจาก UserA และUserB ที่ชั้นของ User เช่น HVAC Service ถูก UserA เรียกใช้ ซึ่ง Service จะไปควบคุมอุปกรณ์เครื่องใช้ไฟฟ้าในชั้น Appliance เช่น HVAC Service จะไปควบคุมอุปกรณ์เครื่องใช้ไฟฟ้า Air-conditioner และ Ventilator อุปกรณ์ทั้งสองมีผลต่อสภาพแวดล้อมอุณหภูมิในชั้น environment เป็นต้น



ภาพที่ 1 ลำดับชั้นองค์ประกอบ HNS

โดยความสัมพันธ์จะเป็นไปตามลำดับชั้นดังภาพที่ 1 การเขียนเพื่อให้เข้าใจในการทำงานของระบบเราจะเขียนด้วยภาษา promela code ในทุกๆรายละเอียดของระบบ HNS ตัวอย่าง promela code ที่ใช้เขียนการทำงานของ HVAC Service แสดงได้ดังตารางที่ 1

ตารางที่ 1 แสดงตัวอย่าง promela code ที่ใช้เขียนการทำงานของ HVAC Service

บรรทัดที่	Code
1	proctype HVAC(){
2	int rvalue;
3	tTemp Ti_temp,To_temp; /* Local variables*/
4	(HVAC_state == START);
5	do ::!(HVAC_state == STOP)->
6	Thermometer_in_SetPower(ON);
7	Thermometer_out_SetPower(ON);
8	AC_SetPower(ON);
9	Thermometer_in_Measure(); Ti_temp = r_value;
10	Thermometer_out_Measure();
	To_temp = r_value;
11	do ::(Ti_temp < user_temp)->
12	AC_SetMode(HEATER);

ส่วนที่ 2 เทคนิคการทำการตรวจสอบและวินิจฉัยความถูกต้องของ Integrated Service ในระบบ Home Network System

Integrated Service ในระบบ Home Network System เป็น Service ที่สามารถเกิดการขัดแย้งกัน Feature Interactions (FI) ขึ้นได้เนื่องจาก HNS มี Service ที่ควบคุมอุปกรณ์เครื่องใช้ไฟฟ้าต่างๆที่ใช้ Feature & Functions ของอุปกรณ์เหล่านั้นในการทำงาน เมื่อ Service ที่ควบคุมอุปกรณ์เหล่านั้นทำงานพร้อมกันอาจทำให้มี Feature & Functions ของอุปกรณ์นั้น เกิดขัดแย้งกันทำให้ไม่สามารถทำงานหรือทำงานผิดพลาด ตัวอย่างเช่น UserA เรียกใช้ HVAC Service เพื่อควบคุม Air-conditioner ให้อุณหภูมิภายในห้องให้เป็นตามต้องการ ในขณะที่ UserB เรียกใช้ Energy-saving Service ในการปิด Air-conditioner เพื่อประหยัดพลังงานไฟฟ้า Service ทั้งสองจึงขัดแย้งกันขึ้น จึงมีงานด้าน FI ที่มาตรวจสอบการขัดแย้งกันนี้

ในงานของ Nakamura *et al*. (2004) ได้ทำการแบ่งการเกิด FI ใน HNS ดังนี้

1. FI ที่เกิดจากการขัดแย้งกันจากสภาพแวดล้อม (Environment)

เกิดจากสภาพแวดล้อมที่ถูกกำหนดใน HNS ถูกอุปกรณ์เครื่องใช้ไฟฟ้าบางเครื่องใน HNS ควบคุมสภาพแวดล้อมเดียวกันอยู่ เช่น Air-conditioner กับ Ventilator มีผลทำให้อุณหภูมิในสภาพแวดล้อมเปลี่ยนแปลง และถ้าอุปกรณ์เครื่องใช้ไฟฟ้าทั้งสองไม่ได้ถูก Service เดียวกันใช้งานอยู่ อาจทำให้แย่งกันควบคุมอุณหภูมิได้

2. FI ที่เกิดจากการขัดแย้งกันจากอุปกรณ์เครื่องใช้ไฟฟ้า (Appliances)

เกิดจากอุปกรณ์เครื่องใช้ไฟฟ้า ถูก Services หลายตัวควบคุมพร้อมกัน ทำให้อุปกรณ์เครื่องใช้ไฟฟ้าทำงานผิดพลาด

3. FI ที่เกิดจากการขัดแย้งกันจาก Service

เกิดจาก Service ถูกเรียกใช้จาก User หลายคนแล้วการเรียกใช้นั้นขัดแย้งกันอยู่

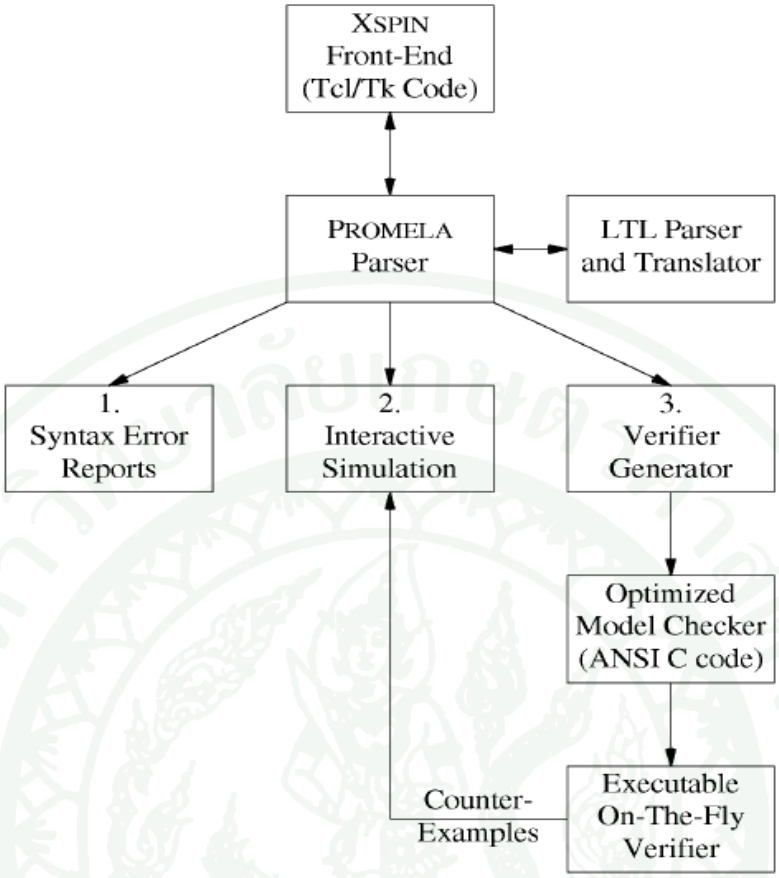
เทคนิคการทำการตรวจสอบและวินิจฉัยความถูกต้อง ในงาน FI มีเทคนิคหลายอย่างที่ใช้ในการตรวจสอบ ในงานวิจัยนี้ยังคงใช้เทคนิคเดียวกับ Leelaprute *et al*. (2005) ในการตรวจสอบและวินิจฉัย โดยจะใช้เทคนิค Symbolic Model Checking ก่อนจะกล่าวถึงจะกล่าวก่อนว่าทำไมถึงใช้เทคนิคนี้ ในการตรวจสอบและวินิจฉัยความถูกต้องของ Service ในระบบ Home Network System มีวิธีการตรวจสอบจากการตรวจจากคำสั่งของ Service โดย Service ถูกเขียนและอธิบายด้วยภาษา Promela code และเป็นไปตามรูปแบบของ HNS ตามแนวคิดต้องเขียนทุกๆ สิ่งใน Service ทั้งตัวอุปกรณ์เครื่องใช้ไฟฟ้า สภาพแวดล้อม ตัว Service ผู้ใช้และยังสามารถพัฒนาให้วิเคราะห์สิ่งอื่นๆได้อีกเช่น เวลา วิธีติดต่อ Service ความสำคัญของผู้ใช้ ซึ่งมีผู้พัฒนา ทำให้รายละเอียดของ code มีมากมายและวิธีการตรวจสอบ คือ การดูทุกๆ บรรทัดทุกๆ คำสั่งของ Service มีผลขัดแย้งกันกับอีก Service ที่ต้องไล่ดูจึงกินเวลาในการตรวจสอบมากและยากในการตรวจสอบ ทั้งนี้ Leelaprute *et al*. (2005) จึงเสนอวิธีในการตรวจสอบโดยใช้เทคนิค Symbolic Model Checking

เทคนิค Symbolic Model Checking เป็นการจำลองเหตุการณ์จำลองคุณสมบัติและพฤติกรรม (behavior) ต่างๆของระบบด้วย Promela code กล่าวคือ ระบบ HNS รวมถึง service ที่เขียนด้วย Promela จะถูกจำลองขึ้นใน software tool ที่มีชื่อว่า SPIN (Holzmann *et al.*, 1997)

เทคนิคนี้จะทำให้ไม่ต้องไล่ code หรือคิดเหตุการณ์ทั้งหมดเองซึ่งทำให้การทำการตรวจสอบและวินิจฉัยความถูกต้องของ Service ในระบบ Home Network System เร็วขึ้นและถูกต้องขึ้นด้วย

ส่วนที่ 3 การทำงานของ SPIN

ลักษณะการทำงานของ SPIN เป็นในลักษณะของ Finite state machine (FSM) โดยจะสร้าง state ต่างๆ ขึ้นตามเงื่อนไขทั้งหมด และจะตรวจสอบเงื่อนไขทั้งหมดแบบ Depth-First Search โดยไล่ไปตามแนวลึกซึ่งนำเสนอโดย Holzmann *et al*. (1997) ภาพที่ 2 แสดงภาพโครงสร้างของ SPIN จะใช้ Tcl/Tk เป็น Code ในการพัฒนาเป็น Front-end และภาษา C ในการพัฒนาตัว Model Checker มีภาษา Promela ในการจำลองคุณสมบัติและพฤติกรรม (behavior) ของระบบและใช้ภาษา LTL กำหนดเงื่อนไขในการตรวจสอบ โดย SPIN จะสร้าง Model ขึ้นตาม Promela ในระหว่างที่สร้างก็ทำการตรวจสอบ Model แล้วครั้งหนึ่ง และใช้ LTL ตรวจสอบเหตุการณ์ที่เกิด FI ใน Model อีกครั้งแบบ Depth-First Search SPIN จะแสดงผลการตรวจสอบออกมาแบบ on the fly ในครั้งนั้นจากทรัพยากรในเครื่องและจะทำใหม่ในครั้งต่อไป

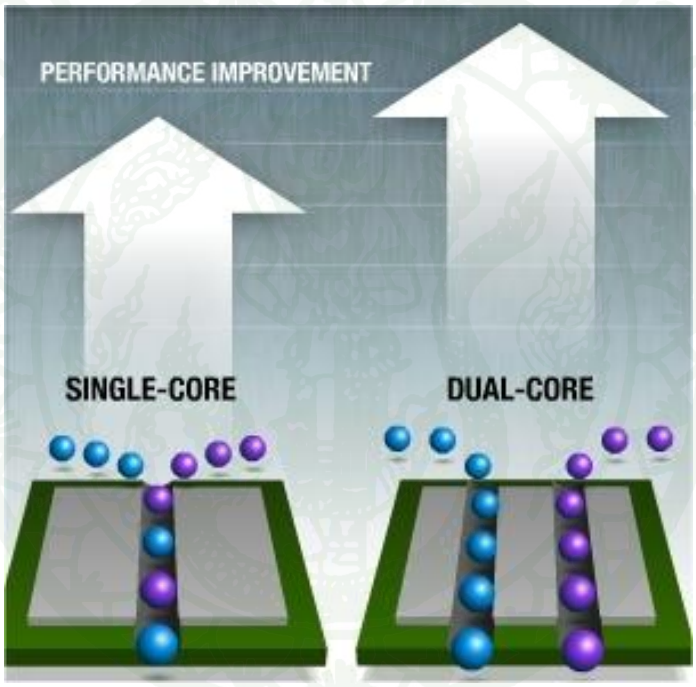


ภาพที่ 2 การทำงานของ SPIN

แต่ถึงเพิ่มความเร็วในการตรวจสอบด้วย SPIN อย่างไร Service ในระบบ Home Network System ก็มีการพัฒนาให้มีความหลากหลายมากขึ้นและมีมากขึ้น ซึ่งการเพิ่มของ Service นั้นมีผลทำให้การจำลองต้องจำลอง state ที่เพิ่มขึ้นตามด้วยเพราะต้องจำลองเหตุการณ์ที่เกิดใน Service ทุกๆ ตัว เมื่อมีการใช้งานพร้อมๆ กัน ทำให้มี state มากขึ้นและเวลาในการคำนวณก็มากขึ้นตาม และผลจากการวิเคราะห์แล้วการเพิ่มขึ้นของ Service มีผลทำให้ state และเวลาเพิ่มขึ้นเป็น Exponential ทุกๆ 1 Service ที่เพิ่มขึ้นดังสมการ (State of Service1) x (State of Service2)… x (State of ServiceN) = Total State ทั้งหมด เมื่อ State of Service คือเงื่อนไขที่เกิดได้ใน Service นั้น และ N คือจำนวน Service ทั้งหมด เวลาก็เพิ่มขึ้นจนเรารอไม่ได้หรือมีทรัพยากรในระบบไม่พอให้ประมวลผลต่อไปได้ จึงมีการพัฒนา SPIN ต่อไปอีกด้วยเทคโนโลยีที่จะนำเสนอต่อไป

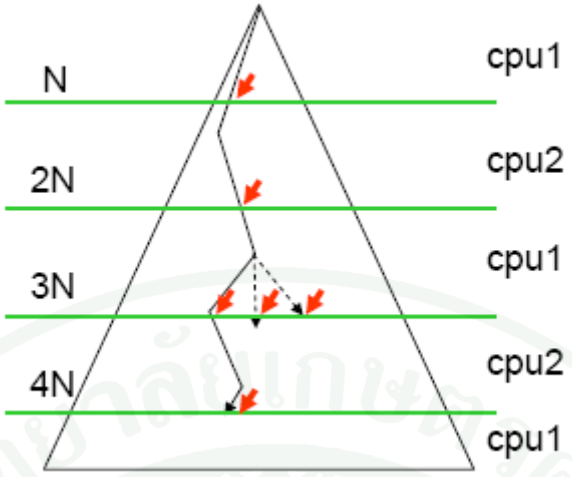
SPIN ถูกพัฒนาต่อเนื่องจนปี 2007 CPU ที่เป็น Multi-Core มีใช้งานอยู่ในตลาดอย่างมาก ภาพที่ 3 Multi-Core คือ CPU ที่ถูกพัฒนาให้สามารถประมวลผลหลายๆ คำสั่งได้ในเวลาเดียวกัน

โดยจะมีการใส่มวลประมวลผลหลายตัว เช่น DUAL-Core, QUAD-Core ช่วยกันประมวลผล โดยขึ้นอยู่กับ Application จะแบ่งให้ประมวลผลแบบไหน ดังภาพที่ 3 จะเห็นลูกกลมสีฟ้ากับสีม่วงแทนชุดคำสั่งที่ต้องให้ CPU ประมวลผล โดยถ้าเป็น SINGLE-CORE จะสลับกันไป ในความเป็นจริงแล้วโปรแกรมที่ทำงานจะใช้ชุดคำสั่งจำนานมากในการทำงาน สมมุติให้ลูกกลมสีฟ้ากับสีม่วงเป็นชุดคำสั่งสองชุดที่พลัดกันทำงานได้จะเห็นได้ว่าโปรแกรมหนึ่งจะต้องรอการทำงานของอีกโปรแกรมทุกครั้งถ้าเป็น SINGLE-CORE แต่ถ้าเป็น Multi-Core โปรแกรมทั้งสองสามารถประมวลผลพร้อมกันได้ หมายถึงในหนึ่งหน่วยเวลาจะมี 2 คำสั่งเสร็จไปเป็น 2 เท่าเมื่อเทียบกับ SINGLE-CORE

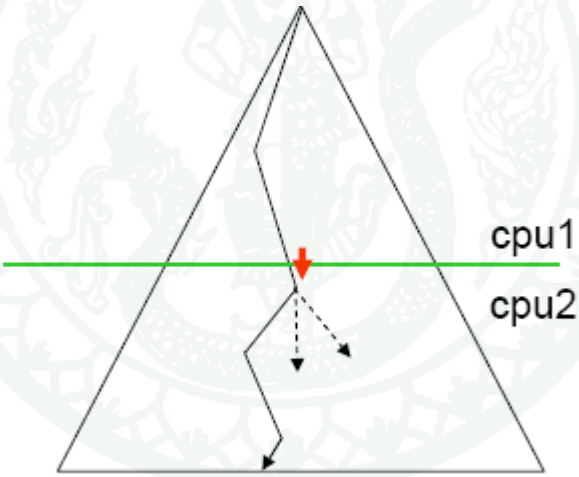


ภาพที่ 3 เปรียบเทียบการทำงาน DUAL-CORE กับ SINGLE-CORE

Multi-Core ทำให้เขียนโปรแกรมให้ทำงายแบบขนานได้ดังการแบ่งสถาปัตยกรรมของการทำงานแบบขนาน ทำให้ SPIN เลือกพัฒนาจนทำงานแบบขนานบน CPU Multi-Core (Holzmann *et al.*, 2007, 2008; SPIN Formal Verification, 2010) และออก SPIN V.5.1.6 ใน MAY 2008 ทำให้การทำงานของ SPIN เร็วขึ้น



ภาพที่ 4 Spin ที่แบ่งการทำงานแบบ safety



ภาพที่ 5 Spin ที่แบ่งการทำงานแบบ liveness

ภาพที่ 4 และภาพที่ 5 เป็นภาพการทำงานของ SPIN ที่เป็น Multi-Core โดยจะแบ่งการทำงานได้สองแบบ Safety กับ liveness โดยหลังจากที่ SPIN ได้ทำการสร้าง Model ตาม Promela แล้ว จะทำการตรวจสอบตามเงื่อนไขที่เขียนโดย LTL โดย SPIN จะสร้าง Model ในโครงสร้างของ Tree แล้วทำการตรวจสอบ Tree ตามเงื่อนไข LTL ที่ละ State ของ tree แบบ Depth First Search ตรวจสอบจากแนวลึก โดยปกติ CPU จะทำการไล่ตรวจสอบ Tree จนครบทุก State หรือจนเจอเงื่อนไขที่ทำการตรวจสอบ LTL ใน CPU แบบ Single-Core CPU ตัวเดียวจะทำการตรวจสอบเงื่อนไขทั้งที่ Tree แต่ CPU Multi-Core แบ่งทำงานได้สองแบบ คือ แบบ liveness จะแบ่ง Tree

เป็นสองส่วนดังภาพที่ 5 เพื่อให้ CPU สองตัวช่วยกันตรวจสอบ Tree คนละส่วน และแบบที่สอง แบบ Safety แบ่งให้ CPU ทำงานในแต่ละชั้นความลึกของ Tree ดังภาพที่ 4 การทำงานทั้งสองแบบจะทำให้การตรวจสอบเงื่อนไขตาม LTL หาได้เร็วขึ้นกว่าทำงานบน CPU ตัวเดียว

ซึ่งทั้งสองแบบต่างกันที่ แบบ Safety จะหาเงื่อนไขตาม LTL จนเจอ แล้วจึงหยุดตรวจสอบ แต่แบบ liveness จะตรวจสอบทั้ง Tree ในการหาตามเงื่อนไข LTL ที่เป็นแบบนี้เพราะเงื่อนไขใน LTL ที่แตกต่างกันในสองแบบ liveness LTL จะเป็น [ ] “always” หมายถึงทุกๆ กรณีต้องเป็นตามเงื่อนไขจึงต้องตรวจสอบทุกๆกรณี และแบบ Safety LTL จะเป็น < > “eventually” มีกรณีที่เป็นตามเงื่อนไขก็จะหยุดตรวจสอบ ซึ่ง SPIN ใช้เงื่อนไขแบบ “eventually” เพราะเราจะไม่ให้มีเงื่อนไขที่ Service ขัดแย้งกันอยู่ในระบบเลย

แต่ด้วยการทำงานแบบขนานบน CPU Multi-Core ยังคงขึ้นกับการทำงานของ CPU บนเครื่องคอมพิวเตอร์เพียงเครื่องเดียว ถึงจะมี CPU แบบ DUAL-CORE หรือ QUAD-CORE ตาม ทั้งหมดยังขึ้นกับข้อจำกัดในทรัพยากรของเครื่องคอมพิวเตอร์เครื่องนั้น อีกทั้ง CPU Multi-Core ที่จำนวน Cores มากยังมีราคาที่สูงมาก

Intel® Core™ i7-990X Processor Extreme Edition 6 Cores / 12 Threads 3.46 GHz $999.99
Intel® Core™ i7-975 Processor Extreme Edition 4 Cores / 8 Threads 3.33 GHz $1,049.99
Intel® Core™ i7-970 Processor 6 Cores / 12 Threads 3.20 GHz $599.99
Intel® Core™ i7-2600K Processor 4 Cores / 8 Threads 3.40 GHz $319.99

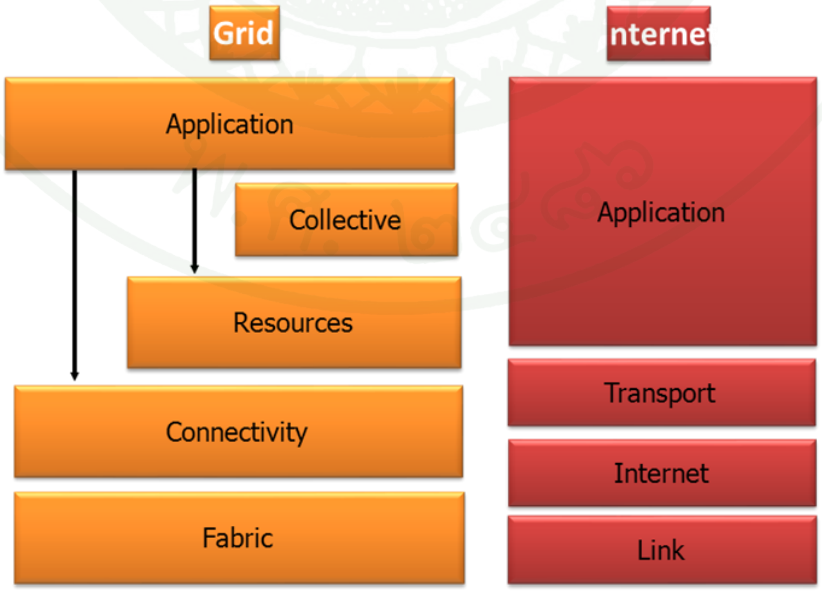
และการประมวลผลของ SPIN จะเน้นที่ปริมาณในการประมวลผลที่มากๆ ดังนั้นจึงเสนอให้ใช้เทคโนโลยีกริด มาช่วยในการลดปริมาณการประมวลผลที่มากๆ ให้แบ่งประมวลผลใน CPU Single–Core หลายเครื่องๆ ช่วยในการคำนวณโดยไม่จำเป็นต้องมี CPU ที่เป็น Multi-Core

วิธีในการแบ่งการประมวลผลแบบการใช้เทคโนโลยี Multi-Core ไม่สามารถไปใช้ในกับการแบ่งแบบเทคโนโลยีกริดได้ เพราะการแบ่งแบบ Multi-Core นั้นต้องทำการแบ่งแบบ share memory จึงไม่สามารถนำไปใช้ในการทำงานบนเทคโนโลยีกริดได้

ส่วนที่4 การนำเทคโนโลยีกริดมาใช้

เทคโนโลยีกริดที่แสดงแบบจำลองได้ด้วยภาพที่ 6 คือการเชื่อมโยงเอาทรัพยากรที่มีอยู่ในเครือข่ายมาใช้งานให้มีประสิทธิ์ภาพที่สูงสุดเช่น การแบ่งเอาหน่วยประมวลผลของเครื่องที่ไม่ได้ใช้อยู่มาช่วยเครื่องที่ต้องการทำงานที่มากๆ งานวิจัยนี้ได้นำเอาเทคโนโลยีกริดมาทำให้ SPIN แบ่งการทำงานบนกริดได้

ภาพที่ 6 แสดงแบบจำลองเทคโนโลยีกริด



ภาพที่ 7 สถาปัตยกรรมของกริด

การแบ่งของสถาปัตยกรรมของกริดดังภาพที่ 7 เป็นการเปรียบเทียบการทำงานของกริดเป็น layerต่างๆ ดังเช่น layer ของ Internet โดยมีรายละเอียดดังนี้

1. ชั้น Fabric เป็นทรัพยากรที่ถูกเรียกใช้โดยโปรโตคอลของกริด เช่น หน่วยประมวลผล ที่เก็บข้อมูล

2. ชั้น Connectivity เป็นโพรโตคอลสำหรับการเชื่อมต่อเครือข่ายเข้าด้วยกันให้เป็นกริด บนความปลอดภัยแบบ GSI (Grid Security Infrastructure) ซึ่งเป็นความปลอดภัยระดับสูง

3. ชั้น Resources เป็นโปรโตคอลและเอพีไอสำหรับการเข้าใช้ทรัพยากรหนึ่งๆ

3.1 GRAM (Grid Resource Allocation Management) ทำหน้าที่จองทรัพยากรสำหรับประมวลผล และทำหน้าที่แสดงสถานการณ์ทำงาน และสามารถควบคุมงานที่ประมวลผลบนทรัพยากรนั้น

3.2 GridFTP ทำหน้าที่ในการโอนถ่ายข้อมูล

3.3 Grid Resource Information Service ทำหน้าที่แสดงข้อมูลของทรัพยากร ในหนึ่งหน่วยทรัพยากรว่าประกอบด้วยหน่วยประมวลผล, หน่วยจัดเก็บข้อมูลใดบ้าง

4. ชั้น Collective เป็นบริการสำหรับการจัดการกลุ่มของทรัพยากร เช่น การกระจายงาน การตรวจสอบสถานะของกลุ่มทรัพยากร เป็นต้น

5. ชั้น Application คือโปรแกรมที่ทำงานในระบบกริดและเรียกใช้บริการต่างๆ ในชั้นที่อยู่ต่ำกว่า

งานวิจัยนี้ทำส่วน Applicationและ Collective นั้นคือ การแบ่งงานในการประมวลSPINที่ทำงานบนกริด

ทั้งนี้ประเภทของกริดแบ่งได้ดังนี้

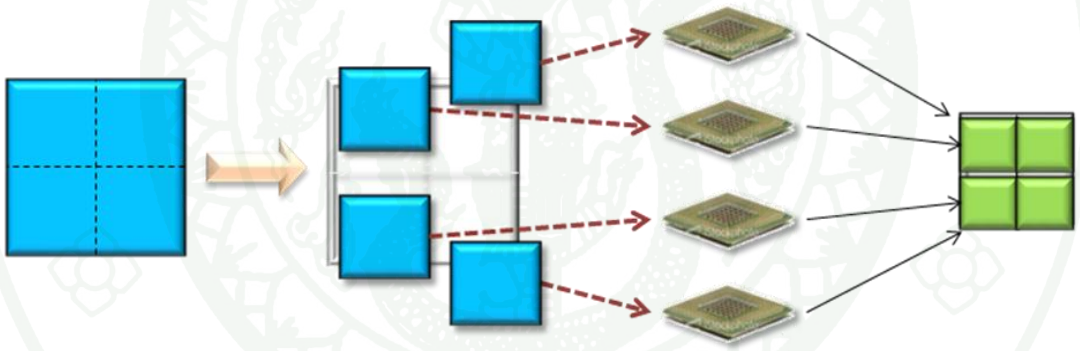
1. คอมพิวเตชั่นนอลกริด (Computational Grid) กริดที่เน้นการประมวลผลมาก

2. ดาต้ากริด (Data Grid) กริดที่เน้นการใช้ข้อมูลมหาศาล

3. แอคเซสกริด (Access Grid) กริดสำหรับติดต่อสื่อสารระยะไกล

แต่ในงานวิจัยนี้จะเน้นการใช้งานประเภทคอมพิวเตชั่นนอลกริด (Computational Grid) คือการนำกริดมาช่วยในการประมวลผล ให้ประมวลผลในปริมาณงานที่มากๆ โดยการแบ่งทรัพย์ให้หน่วยประมวลทำงานหลายตัวพร้อมกันในระบบกริดจะช่วยในการแบ่งการทำงานของ SPIN ทำให้ SPIN ตรวจสอบ Model ได้เร็วขึ้น โดยวิธีการทำงานแบบขนาน

Parallel Programming การทำงานแบบขนานเดิมทีหน่วยประมวลทั่วไปจะมีการทำงานแบบตามลำดับคือจะทำงานทีละคำสั่ง หรือในหนึ่งหน่วยเวลาจะมีเพียงคำสั่งเดียวเท่านั้นที่คำนวณอยู่ในหน่วยประมวลผล แต่วิธีการคำนวณแบบขนานจะช่วยให้ในหนึ่งหน่วยเวลาสามารถคำนวณสองคำสั่งหรือมากกว่านั้นได้ดังภาพที่ 8 การแบ่งการทำงานแบบขนาน



ภาพที่ 8 Parallel Programming

รูปสี่เหลี่ยมซ้ายมือคือคำสั่งที่ต้องนำไปประมวลผลในหน่วยประมวลผล เดิมที่อาจเป็นคำสั่งที่ใหญ่มากๆ หรือมีหลายคำสั่งที่ต้องรอการประมวลผลแบบตามลำดับ เราทำการแบ่งคำสั่งนั้นเป็น 4 ส่วนแล้วกระจายให้หน่วยประมวลผล 4 ตัวทำงานร่วมกัน โดยไม่ต้องรอให้หน่วยประมวลผลประมวลที่ละคำสั่ง เราสามารถคำนวณ 4 คำสั่งพร้อมกัน และรวมผลการประมวลทั้ง 4 นั้นได้เหมือนกับที่ประมวลในหน่วยประมวลผลเดียวกัน ดังรูปสี่เหลี่ยมทางขวามือ

Structure of Parallel สถาปัตยกรรมของการทำงานแบบขนานยังแบ่งได้ตาม Instruction stream และ Data stream

1. SIMD (Single Instruction stream Multiple Data stream) เป็นคอมพิวเตอร์ที่มีซีพียูมากกว่าหนึ่งตัวกระทำการกับคำสั่งเดียวกัน แต่ใช้ข้อมูลต่างกัน มีหน่วยความจำที่เก็บเฉพาะคำสั่งร่วมกัน และมีหน่วยที่ใช้ควบคุมซีพียูแต่ละตัว แต่ละซีพียูจะมีหน่วยความจำเก็บเฉพาะข้อมูลเป็นของตัวเอง ซีพียูจะมีลักษณะพิเศษกว่าซีพียูโดยทั่วๆ ไป ตัวอย่างของคอมพิวเตอร์แบบนี้ ได้แก่ DADO และ CM-5

2. MISD (Multiple Instruction stream Single Data stream)) ยังไม่มีคอมพิวเตอร์ใดจัดอยู่ในกลุ่มนี้แต่อาจจะมีในอนาคต

3. MIMD (Multiple Instruction stream Multiple Data stream) เป็นคอมพิวเตอร์ที่มีหลายซีพียู ซีพียูแต่ละตัวกระทำการกับคำสั่งและข้อมูลของตัวเอง ซีพียูจะเป็นแบบเดียวกันกับที่มีใช้อยู่แล้วในปัจจุบัน แบ่งออกได้เป็นสองพวกย่อย คือ พวกที่ใช้หน่วยความจำร่วมกัน เรียกว่า มัลติโพรเซสเซอร์ และพวกที่ไม่ใช้หน่วยความจำร่วมกัน เรียกว่า มัลติคอมพิวเตอร์ (multicomputer) ซึ่งก็คือการทำงานแบบกริด

สถาปัตยกรรมของงานวิจัยนี้เป็นแบบ MIMD เนื่องจากเราต้องการให้ตัวโปรแกรม SPIN กระจายคำสั่งที่ต้องประมวลผลตามหน่วยประมวลผลที่อยู่ในแต่ละเครื่อง และคำสั่งที่กระจายไปนั้นเป็นคำสั่งไม่ซ้ำกัน จะตรงกับ Multiple Instruction stream Multiple Data stream ซึ่งได้อธิบายไปแล้วว่าเป็น สถาปัตยกรรมแบบเดียวกับกริด

และการทำให้หน่วยประมวลผลแบ่งงานและคุยกันได้นั้น ต้องอาศัย Massage Passing Interface (MPI) ในการติดกันระหว่างหน่วยประมวลผลการแลกเปลี่ยนข้อมูลให้ติดต่องานกันได้ ซึ่งMPI มีอยู่ในไลบรารี่ใน C/C++, Java

Algorithm ที่ใช้ในการพัฒนาในงานการคำนวณแบบขนาน

1. Functional Parallelism เป็นอัลกอริทึ่มที่ใช้แบ่งแต่ละส่วนของฟังก์ชั่นออกจากกัน โดยส่วนที่แบ่งออกมานั้นจะต้องเป็นอิสระต่อกันและไม่เกี่ยวข้องกัน แล้วนำแต่ละส่วนมาประมวลผลพร้อมกัน

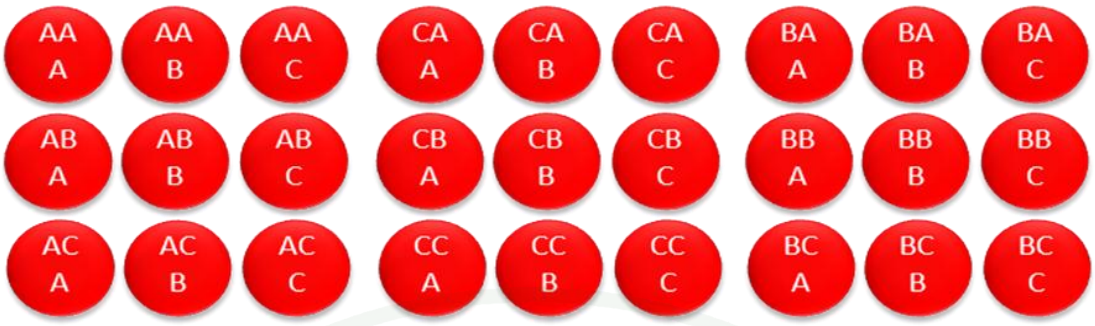
2. Data Parallelism เป็นอัลกอริทึ่มที่จะกระจายฟังก์ชั่นเดียวกันแต่มีข้อมูลขาเข้าแตกต่างกันไป แล้วประมวลผลพร้อมกัน

3. Partition Algorithm (Fine-grain Tasks) แบ่งส่วนเล็กสุด โดยแบ่งส่วนการคำนวณที่เล็กที่สุดและเป็นการคำนวณที่ซ้ำๆ กันจะเห็นถึงส่วนที่เล็กที่สุดและเอาส่วนนั้นมาแบ่งการคำนวณ

4. Agglomeration (Coarse-grain Tasks) แบ่งเป็นกลุ่มโดย นำส่วนที่เล็กที่สุด Fine-grain Tasks มาจัดกลุ่มเพื่อแบ่งให้การคำนวณดีขึ้น

ทั้งนี้งานวิจัยนี้จะใช้ Algorithm แบบ Agglomeration ในการแบ่งการทำงานให้หน่วยประมวลผลทำงาน โดยคิด Fine-grain Tasks จากการจำลองการเกิด Service จาก User เช่นถ้ามี Service 3 Services {A,B,C} มี User 3 คน {X,Y,Z} SPIN จะต้องจำลองการทำงานทั้งหมด ดังภาพที่ 9 โดยวงกลมในภาพต่างละวง แทนการประมวลผลจาก Model ที่เกิดได้ทั้งหมดจากการทำเรียกใช้ Service 3 Services จาก User 3 คน ตามลำดับ ในภาพที่ 10 User {X,Y,Z} เช่น ภาพที่ 11 UserX เรียกใช้ Service C, UserY เรียกใช้ Service B, UserZ เรียกใช้ Service B ตามลำดับ

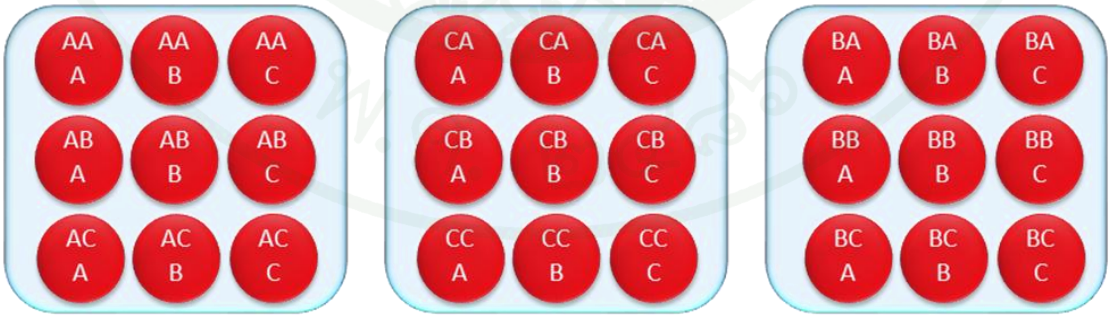
หลักการทำงานของ SPIN จะเป็นแบบ interleaving และ atomic execution กล่าวคือ ถึงแม้จะมี user N คน สั่งงาน service จำนวน N service ณ เวลาเดียวกันจริงๆ (simultaneously) ก็ตาม แต่เวลาสร้าง model, SPIN จะสร้าง process ขึ้นมา N process สำหรับรองรับการ run service ของแต่ละ user แล้วจะทำการ run คำสั่ง (method) ของ service ใน process ใดๆ ได้แค่ครั้งละ 1 คำสั่งเท่านั้น ยกตัวอย่างเช่นถ้ามี process A, B และ C ถูกสร้างขึ้นมาเพื่อ run service A, B และ C ขณะที่ SPIN run คำสั่งใน process A (สมมุติว่าเป็น TV.POWER (ON) อยู่, process B และ C จะไม่สามารถ run คำสั่งใดๆ ได้จนกว่า process A จะ run คำสั่ง TV.POWER (ON) เสร็จสิ้น (แต่โดยทั่วไป เนื่องจากแต่ละคำสั่งสามารถทำงานแล้วเสร็จภายในเวลาอันรวดเร็วมาก ทำให้ผู้ใช้รู้สึกเสมือนว่ามีหลายคำสั่งของหลาย process ทำงานอยู่พร้อมกันได้) เพราะฉะนั้นจากคุณสมบัติในการทำงานของ SPIN ในลักษณะนี้แสดงให้เห็นว่าผลลัพธ์ของการ detect Feature Interaction ในcombination ของ (A, B, C) อาจไม่เหมือนกับ (B, A, C) ซึ่งสามารถตีความได้ว่า (A, B, C) เป็นcombination ของการเรียกใช้ service ตามลำดับโดยเริ่มจาก user X เรียกใช้ service A, user Y เรียกใช้ service B และ user Z เรียกใช้ service C ซึ่งจะไม่เหมือนกับ (B, A, C) ซึ่งหมายถึงcombination ของการเรียกใช้ service ตามลำดับโดยเริ่มจาก user X เรียกใช้ service B, user Y เรียกใช้ service A และ user Z เรียกใช้ service C



ภาพที่ 9 การแบ่งแบบ Fine-grain เมื่อ มี Service 3 Services และUser 3 คน

ภาพที่ 10 ตำแหน่ง User ที่เรียกใช้ Service

ภาพที่ 11 ตัวอย่าง Model CBB

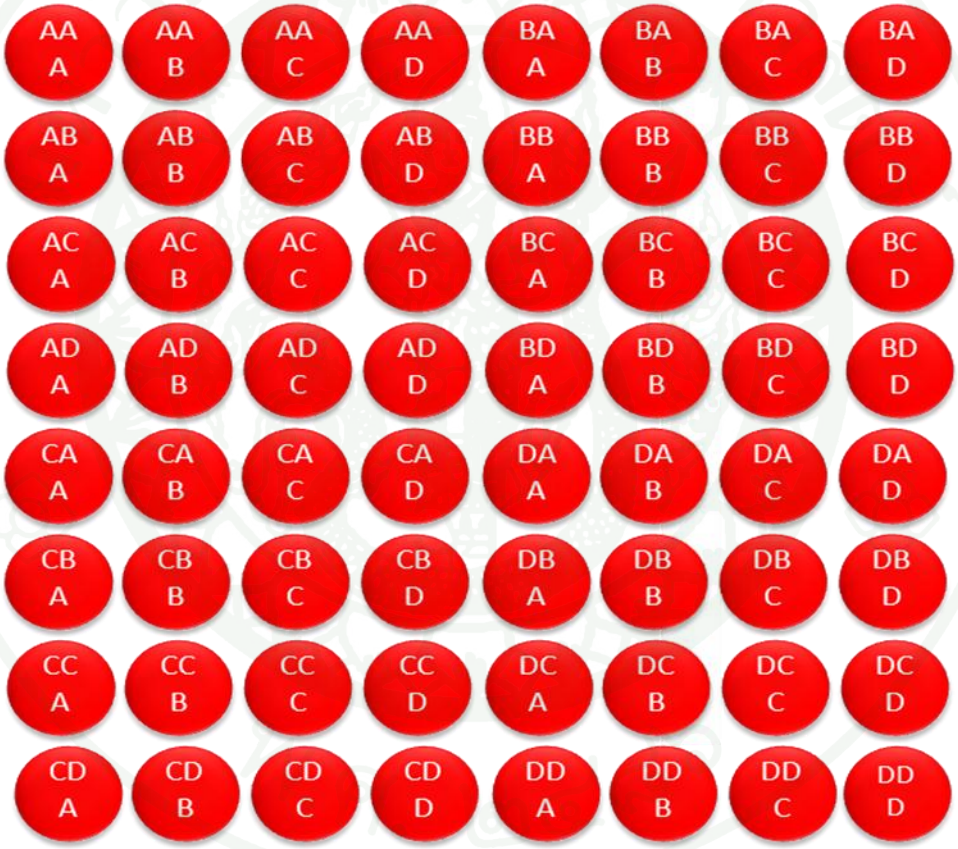


ภาพที่ 12 การแบ่ง Coarse-grain Tasks

หลังจากแบ่งการทำงานในแต่ละการประมวลผลแบบละเอียด Fine-grain Tasks เราทำการจัดกลุ่มการทำงานเพื่อการแบ่งไปประมวลผลไปตามหน่วยประมวลตามแบบ Coarse-grain Tasks

โดยเราแบ่งตาม Service ตัวแรกที่ User อันดับแรกเรียกใช้ จะได้ตามภาพที่ 12 เนื่องจากการประมวลผลที่เราแบ่งแบบ Fine-grain Tasks ทำให้เราประมวลน้อยลงอยู่แล้ว แล้วมาจัดกลุ่มตามที่กำหนดทำให้การประมวลผลในต่างละกลุ่มใช้เวลาไม่มาก

เมื่อเทียบการทำงานที่ใช้เทคโนโลยีกริดกับเทคโนโลยี Multi-Core SPIN ที่ใช้ในการตรวจสอบ FI ใน HNS นั้นมี การประมวลผลย่อยๆ ที่ซ้ำกันและมากขึ้นแบบ Exponential เมื่อ Service เพิ่มขึ้น ดังภาพที่ 13 เปรียบเทียบกับ ภาพที่ 9 มี Service 4 services มีคนอยู่ 3 คน ซึ่งถ้าใช้ SPIN Multi-Core เป็น Tree ที่มีขนาดใหญ่มากๆ



ภาพที่ 13 การแบ่งแบบ Fine-grain เมื่อ มี Service 4 Services และ User 3 คน

การวัดผลของกริดจะได้ดูจาก

1. Load balance Processor แต่ละตัวมีการทำงานที่สมดุลกันหรือเปล่า

2. Concurrency Processor ทุกตัวมีการทำงานไปพร้อมๆ กันหรือเปล่า

3. Overhead ปริมาณงานที่ได้รับเพิ่มขึ้นจากเดิมมากหรือไม่

4. Execution time เวลาที่ใช้ในการประมวลผลตั้งเริ่มต้นจนถึงสิ้นสุด

5. Processor speed จำนวนครั้งของการดำเนินการต่อหนึ่งหน่วยเวลา

ทั้งในงานวิจัยนี้จะวัดจาก Execution time เพราะงานวิจัยนี้เน้นการประมวลผลมากๆและจะวัดจากเวลาที่ใช้ในการประมวลผลว่าจะลดลงอย่างไรเมื่อใช้เทคโนโลยีกริด

อุปกรณ์และวิธีการ

อุปกรณ์

1. Hardware

1.1 คอมพิวเตอร์ส่วนบุคคล CPU Intel Core i5 processor 2.4GHz, 4GB DDR2, HD 500 GB จำนวน 4 เครื่อง

1.2 คอมพิวเตอร์ในศูนย์ ThaiGRID จำนวน 5 เครื่อง

2. Software

2.1 JAVA jdk1.5.0_22	สำหรับพัฒนาโปรแกรม
2.2 Eclipse IDE	สำหรับพัฒนาโปรแกรม
2.3 MinGW Compiler	สำหรับพัฒนาโปรแกรม
2.4 C-Free 5 IDE	สำหรับพัฒนาโปรแกรม
2.5 Spin Version 5.2.5	สำหรับสร้าง Model Checker
2.6 library MPJ	สำหรับการติดต่อสื่อสาร-ระหว่างหน่วยประมวลผล
2.7 Microsoft Excel	สำหรับแสดงผลลัพธ์

3. Operating System

3.1 ระบบปฏิบัติการ Windows XP,Windows 7

วิธีการ

รายละเอียดวิธีการวิจัย

วิธีการตรวจสอบการขัดแย้งกันของการให้บริการในระบบเครือข่ายบ้านอัจฉริยะด้วยสปินโมเดลเว็คกิ้งโดยการประยุกต์ใช้เทคโนโลยีกริดมีกระบวนการวิจัยเพื่อพัฒนาสามขั้นตอนหลักคือ 1) การนิยามระบบบ้านอัจฉริยะ 2) การจำแนกจำนวนเหตุการณ์ที่ต้องตรวจหาฟีเจอร์อินเตอร์แอคชั่นทั้งหมด 3) การตรวจหาฟีเจอร์อินเตอร์แอคชั่นแบบขนาน

1. การนิยามระบบบ้านอัจฉริยะ

ในขั้นตอนที่ 1 เป็นการเขียน Tag นิยามระบบบ้านอัจฉริยะเพื่อให้โปรแกรมควบคุมการทำงานแบบขนานเข้าใจค่าต่างๆของระบบบ้านอัจฉริยะ และการนำเข้าข้อมูลอื่นๆที่จำเป็นในระบบควบคุมการแบบขนานโดยมีขั้นตอนดังนี้

1.1 การนิยามระบบบ้านอัจฉริยะด้วยภาษาโปรเมลา

การเขียนนิยามบ้านอัจฉริยะในงานวิจัยนี้ใช้ Framework ของ Nakamura *et al.* (2004) และ Leelaprute *et al.* (2005) ในการนิยามบ้านอัจฉริยะด้วยภาษาโปรเมลาซึ่งเป็นการนิยามความสัมพันธ์เชิงวัตถุ (object oriented) มีการระบุความสัมพันธ์ตามระดับชั้นของ object ที่ปรากฏดังภาพที่ 1

1.2 การใช้สูตรตรรกศาสตร์เวลาแบบเชิงเส้น

การตรวจหาฟีเจอร์อินเตอร์แอคชั่นในระบบบ้านอัจฉริยะ ในการตรวจหานั้นจะตรวจหาตามการจำแนกประเภทของการเกิดการฟีเจอร์อินเตอร์แอคชั่นในระบบบ้านอัจฉริยะตามการนิยามของ Leelaprute *et al*. (2005) โดยเขียนสูตรตรรกศาสตร์เวลาแบบเชิงเส้น (Linear Temporal Logic; LTL) มาคำนวณหาว่าเป็นจริงตามสูตรตรรกะหรือไม่

1.3 การ Tag บนไฟล์โปรเมลาที่นิยามระบบบ้านอัจฉริยะ

การเขียน Tag บนไฟล์โปรเมลาที่นิยามระบบบ้านอัจฉริยะให้โปรแกรมควบคุมการทำงานแบบขนานสามารถเข้าใจรายละเอียดของโปรเมลาโค้ดเพื่อนำไปใช้คำนวณค่าต่างๆในระบบ

ต่อไป เนื่องจากโปรแกรมควบคุมการทำงานแบบขนานไม่สามารถเข้าใจภาษาโปรเมลาได้จึงต้องมีการเขียน Tag ขึ้นมา ซึ่ง Tag สามารถเขียนได้ด้วยสัญลักษณ์ดังต่อไปนี้

1.) การ Tag โปรเมลาโค้ดส่วน LTL ดังตัวอย่างในภาพที่ 14

```
1.  /*@LTL Begin*/
2.  #define TI_U (Ventilation_power == ON && env_temperature_in <
    env_temperature_out)
3.  /*@LTL End*/
```

ภาพที่ 14 ตัวอย่างการ Tag LTL บนโปรเมลาโค้ด

บรรทัดที่ 1 /*@LTL Begin*/ ใช้เพื่อบอกให้โปรแกรมควบคุมการทำงานแบบขนานรู้ว่าโปรเมลาโค้ด LTL เริ่มจากส่วนนี้ ซึ่งเป็นการระบุการเริ่มประกาศค่า LTL ในระบบ

บรรทัดที่ 2 เป็นส่วนของโปรเมลาโค้ด LTL

บรรทัดที่ 3 /*@LTL End*/ ใช้เพื่อบอกให้โปรแกรมควบคุมการทำงานแบบขนานรู้ว่าโปรเมลาโค้ด LTL สิ้นสุดการประกาศค่า LTL ในระบบที่ส่วนนี้

2.) การ Tag โปรเมลาโค้ดส่วน Service ดังตัวอย่างในภาพที่ 15

```
1.  /*@Service Begin:Saving*/
2.  int ES_State = START;
3.  chan MC_ES_State = [4] of {mtype, int};
4.  int ES_E = 0;
5.  active proctype ES_MC(){
6.  mtype user;
7.  do
8.  :: MC_ES_State?user,ES_State;
…
…
…
9.  od;
10. EXIT:skip;
11. }
12. /*@Service End:Saving*/
```

ภาพที่ 15 ตัวอย่างการ Tag Service บนโปรเมลาโค้ด

บรรทัดที่ 1 /*@Service Begin: Service Name */ ใช้ระบุการเริ่มนิยาม Service ตาม Service Name นั้นเช่น /*@Service Begin:Saving */

บรรทัดที่ 2-11 เป็นการนิยาม Service HVAC

บรรทัดที่ 12 /*@Service End: Service Name */ ใช้ระบุการสิ้นสุดการนิยาม Service ตาม Service Name นั้นเช่น /*@Service End: Saving */ หลังจากบรรทัดนี้จะจบการนิยาม Service HVAC

1.4) การนิยามระบบบ้านอัจฉริยะด้วย Master Calling path

เพื่ออธิบายเพิ่มเติมในรายละเอียดของ Service งานวิจัยนี้ได้ใช้ลักษณะการเขียนนิยามระบบบ้านอัจฉริยะด้วยโค๊ดที่มีลักษณะคล้ายภาษา XML มาอธิบายให้โปรแกรมควบคุมการทำงานแบบขนานเข้าใจโดยเขียนเป็นไฟล์ที่มีชื่อว่า Master Calling path มีองค์ประกอบดังตัวอย่างในภาพที่ 16

```
1.  <flow>
2.  <service name="Saving" enable="true">
3.  <function name="MC_ES_State" step="1">
4.  <parameter name="user" value="@USER"/>
5.  <parameter name="state" value="START"/>
6.  </function>
7.  </service>
8.  <executeService>
9.  <functions>
10. <function name="set_env_temperature_in"/>
11. </functions>
12. </executeService>
13. <sourceCode>
14. <![CDATA[
15. never {
16. If
17. :: ((x)) -> goto accept_all
18. :: (1) -> goto T0_init
19. fi;
20. }
21. ]]>
22. </sourceCode>
23. </flow>
```

ภาพที่ 16 ตัวอย่างการเขียน Master Calling path เพื่อนิยามนิยามระบบบ้านอัจฉริยะ

บรรทัดที่ 1,23 <flow>…</flow> ระบุบรรทัดแรกและบรรทัดสุดท้ายของไฟล์

บรรทัดที่ 2,7 <service name=" service name " enable="true">…</Service> ระบุชื่อของService และสถานะในระบบว่า true:มีการใช้งาน false:ไม่มีการใช้งาน

บรรทัดที่ 3,6 <function name=" function name " step="1">… </function> ระบุฟังก์ชันที่ User ใช้งานได้จะอยู่ใต้ Tag <service…> เพื่อระบุว่าเป็นฟังก์ชันของ Service ใด ระบุชื่อฟังก์ชันและลำดับการเรียกใช้

บรรทัดที่ 4,5 <parameter name=" parameter name " value=" value "/> ระบุชื่อและค่าของ parameter ที่ต้องกำหนดในฟังก์ชันจะอยู่ใต้ Tag < function…> เพื่อระบุว่าเป็น parameter ของฟังก์ชันใด

บรรทัดที่ 8,12 <executeService>…</executeService> ระบุค่าอื่นที่ระบบเรียกใช้งานได้

บรรทัดที่ 9,11 <functions>…</functions> ระบุ function อื่นที่ระบบเรียกใช้

บรรทัดที่ 10 <function name="set_env_temperature_in"/> ชื่อ function อื่นที่ระบบเรียกใช้

บรรทัดที่ 13,22 <sourceCode>… </sourceCode> ใช้ใส่รายละเอียดอื่นที่จำเป็นในการตรวจสอบ FI

บรรทัดที่ 14-21 ตัวอย่าง Code การเรียกใช้ LTL ในการตรวจหา FI ทำให้ SPIN รู้จัก LTL และนำ LTL ไปตรวจหา FI โดยไม่ต้องแยกเป็นไฟล์ LTL ออกมา

2. การจำแนกจำนวนเหตุการณ์ที่ต้องตรวจหาฟีเจอร์อินเตอร์แอคชั่นทั้งหมด

ในขั้นตอนที่ 2 เป็นการประมวลผลของโปรแกรมควบคุมการทำงานแบบขนานเพื่อคำนวณหาค่าต่างๆ ที่โปรแกรมควบคุมการทำงานแบบขนานใช้เพื่อการตรวจหา FI ค่าต่างๆ ที่ต้องคำนวณแสดงได้ด้วย Flow chart ดังภาพที่ 17

ภาพที่ 17 Flow chart แสดงการจำแนกและเขียนไฟล์โปรเมลาตามจำนวนเหตุการณ์ทั้งหมด

2.1 คำนวณหาจำนวนเหตุการณ์ที่ต้องจำลองการตรวจหาทั้งหมดในระบบ HNS

การจำลองเหตุการณ์ในระบบ HNS เพื่อตรวจหา FI นั้นจะต้องจำลองทุกเหตุการณ์ที่สามารถเกิดขึ้นได้ โดยเหตุการณ์ที่เกิดขึ้นได้นั้นมาจากการที่ User เรียกใช้ Service ระบบ HNS ในระบบ HNS ดังนั้นเมื่อ User 1 คนสามารถเรียกใช้ Service ใดๆก็ได้ในเวลาช่วงเวลาหนึ่ง 1 Service เพราะฉะนั้น จำนวนเหตุการณที่มี User 1 คนจึงเกิดตามจำนวน Service ที่มีในระบบ แต่ถ้ามี User หลายคน ก็จะมีการเรียกใช้ service ได้พร้อมๆ กันตามจำนวน User ดังนั้น จำนวนเหตุการณ์ที่สามารถเกิดได้ในระบบ HNS ที่ต้องจำลองเป็นไปตามสมการที่ 1 ดังนี้

$$\text{Events occur} = Number\ of\ Service^{Number\ of\ User} \quad (1)$$

โปรแกรมควบคุมการทำงานแบบขนานจะนำจำนวน User จากการป้อนค่าจะผู้ใช้โปรแกรม และนำค่าจำนวน Service จากโปรเมลาโค้ดโดยวิเคราะห์จากชื่อ Sevice ที่ได้ทำการเขียนไฟล์ Master Calling path นิยาม Service เอาไว้ เอาไปคำนวณตามสูตรจะได้ค่าจำนวนเหตุการณ์ที่สามารถเกิดขึ้นได้ทั้งหมด

2.2 จำแนกจำนวนเหตุการณ์ที่ต้องตรวจหาฟีเจอร์อินเตอร์แอคชั่นทั้งหมด

การตรวจหา FI จำเป็นต้องทราบรายละเอียดเหตุการณ์ที่สามารถเกิดขึ้นทั้งหมดจากเหตุการณ์ที่ User คนที่ 1 ถึง User คนสุดท้ายเรียกใช้ Service ใดๆจนครบทุก service ซึ่ง User แต่ละคนสามารถใช้ Service เดียวกันได้ในเวลาเดียวกัน โดยมีรูปแบบการจำแนกเหตุการณ์ทั้งหมดด้วยการเขียนชื่อ User ตามด้วยชื่อ Service ที่ถูกเรียกใช้สลับไปจนครบทุกเหตุการณ์ ตารางที่ 2 แสดงตัวอย่างการจำลองเหตุการณ์การเรียกใช้ Service ในระบบ HNS เมื่อมีจำนวน User ในระบบ 3 คน มี Service ให้เรียกใช้ 2 service ดังนี้

ตารางที่ 2 การเรียนใช้ Service ในระบบ HNS แบบ 3 User 2 Service

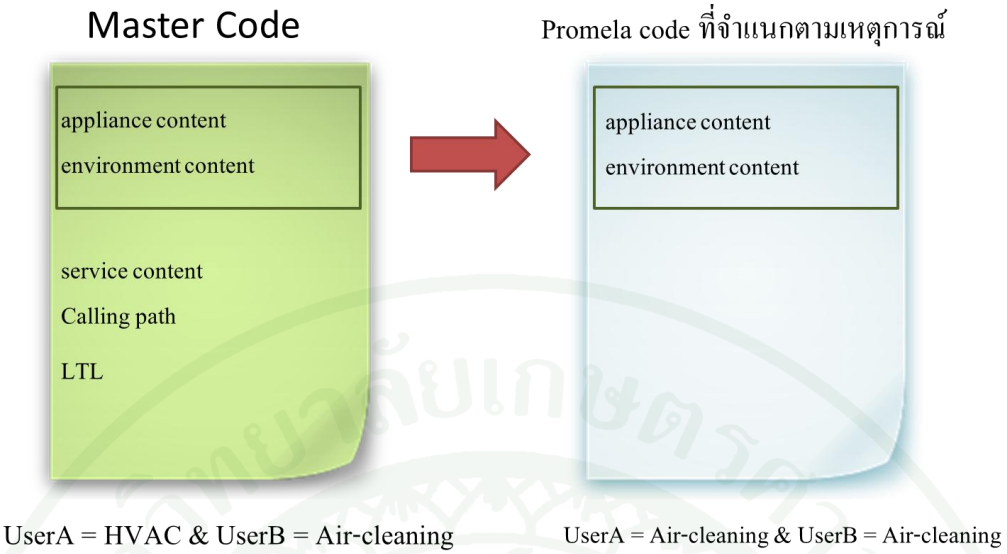
Event	User A	User B	User C
1	HVAC_Service	HVAC_Service	HVAC_Service
2	HVAC_Service	HVAC_Service	Air_Cleaning_Service
3	HVAC_Service	Air_Cleaning_Service	HVAC_Service
4	HVAC_Service	Air_Cleaning_Service	Air_Cleaning_Service
5	Air_Cleaning_Service	HVAC_Service	HVAC_Service
6	Air_Cleaning_Service	HVAC_Service	Air_Cleaning_Service
7	Air_Cleaning_Service	Air_Cleaning_Service	HVAC_Service
8	Air_Cleaning_Service	Air_Cleaning_Service	Air_Cleaning_Service

2.3 การเขียนไฟล์โปรเมลาโค้ดตามเหตุการณ์ที่ต้องตรวจหาฟีเจอร์อินเตอร์แอคชั่น

การตรวจหา FI ในระบบ HNS ด้วยการใช้สปินโมเดลเช็คกิ้ง ต้องใช้ไฟล์โปรเมลาโค้ดในการจำลองเพื่อตรวจหา FI เมื่อได้เหตุการณ์ทั้งหมดที่สามารถเกิดขึ้นได้ โปรแกรมควบคุมการทำงานแบบขนานจะทำการสร้างนิยามเหตุการณ์โดยอัตโนมัติหลังจากที่ได้รับ ไฟล์โปรเมลาโค้ดที่ทำการ Tag แล้ว และไฟล์ Master Calling path ที่นิยาม HNS โดยจะมีขั้นตอนในการทำงานดังนี้

1.) โปรแกรมควบคุมการทำงานแบบขนานสร้างโปรเมลาส่วนที่นิยาม HNS ที่เหมือนกันของทุกๆ เหตุการณ์

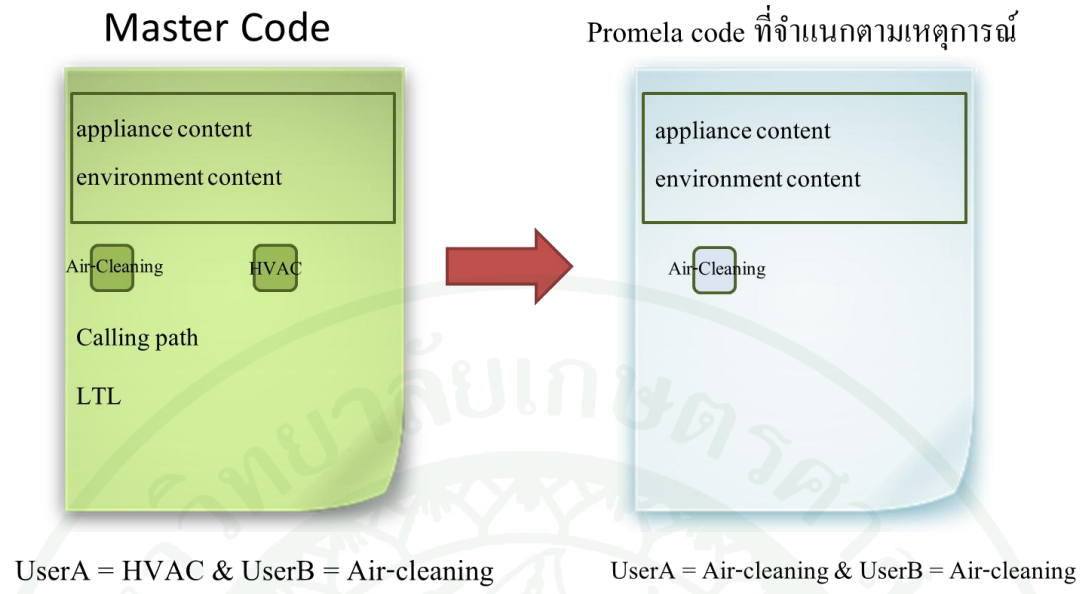
โปรเมลาโค๊ดที่ถูกสร้างจะมีบางส่วนที่นิยาม HNS เอาไว้เหมือนกันทุกๆ เหตุการณ์ที่เกิดขึ้นได้ ซึ่งจะเป็นโครงสร้างหลักของ HNS นั่นคือ ส่วนนิยามสภาพแวดล้อม (Environment) เครื่องใช้ไฟฟ้า (Appliance) และค่าตัวแปรหลักในระบบ HNS โดยโปรแกรมควบคุมการทำงานแบบขนานจะคัดลอกส่วนดังกล่าวจากไฟล์โปรเมลาโค้ดที่ทำการ Tag แล้วข้างต้นเพื่อนำมาเตรียมเขียนไฟล์โปรเมลาโค้ดตามเหตุการณ์ต่างๆ อีกครั้งดังแสดงในภาพที่ 18



ภาพที่ 18 แสดงการสร้างโปรเมลาส่วนที่นิยาม HNS ที่เหมือนกันของทุกๆ เหตุการณ์

2.) เขียนโปรเมลาส่วน Service

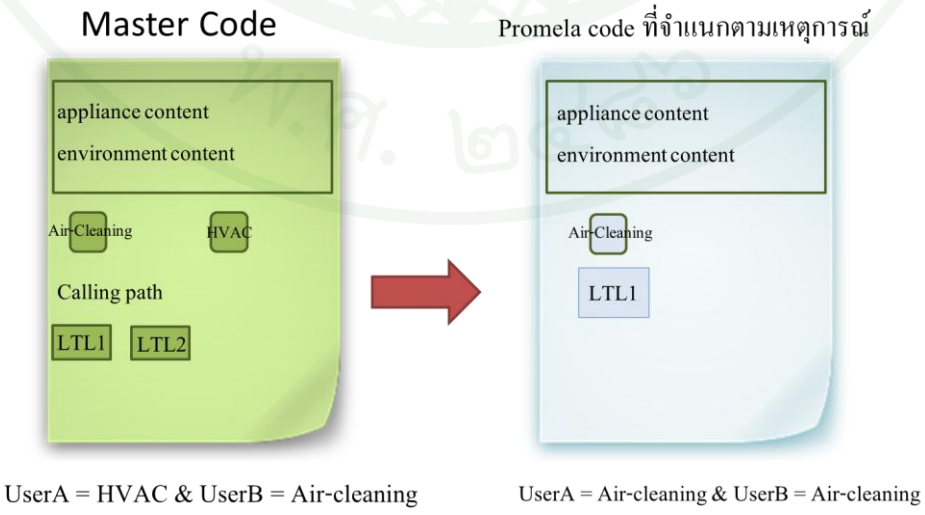
เมื่อทราบเหตุการณ์ทั้งหมดว่ามีการเรียกใช้ Service ต่างๆ ในแต่ละเหตุการณ์อย่างไรบ้าง โปรแกรมควบคุมการทำงานแบบขนานจะคัดลอกนำเอาส่วนที่เป็น Service ที่ได้จากการ Tag Service มาเขียนลงไฟล์โปรเมลาโค๊ดต่อจากไฟล์ที่เตรียมไว้ในข้อ 1.) โดยแยกไฟล์ตามแต่ละเหตุการณ์ซึ่งขึ้นอยู่กับพฤติกรรมการเรียกใช้ service ของเหตุการณ์นั้นๆ เช่น ถ้าเหตุการณ์นั้นมีการเรียกใช้ HVAC เพียง service เดียวก็จะนำส่วน HAVC service ที่ถูก Tag นั้นมาเขียนลงไฟล์โปรเมลาโค้ดตามเหตุการณ์การนั้น แสดงได้ดังภาพที่ 19



ภาพที่ 19 แสดงการสร้างโปรเมลาส่วน Service

3.) เขียนโปรเมลาส่วน LTL

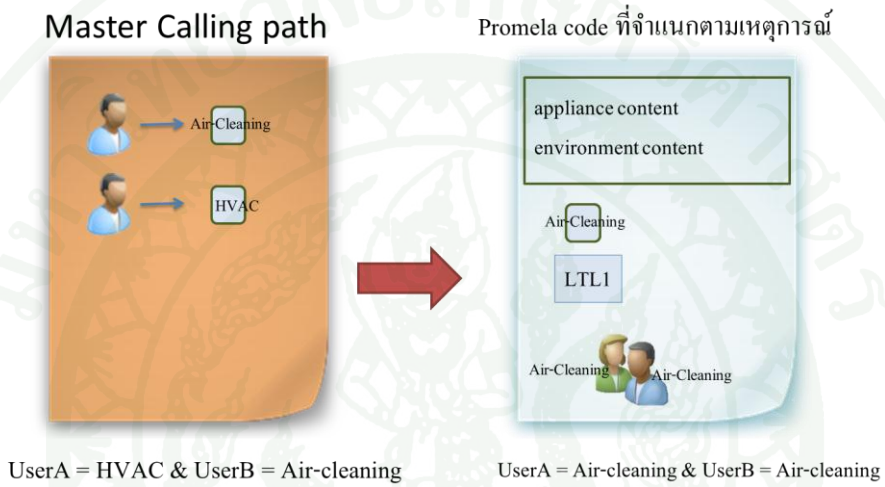
เมื่อทำการ Tag ไฟล์โปรเมลาไว้แล้ว โปรแกรมควบการทำงานแบบขนานจะทราบบรรทัดที่มีการนิยาม LTL เอาไว้แล้วจะเขียนนิยาม LTL ลงไปพร้อมแบ่งไฟล์ตามจำนวน LTL ที่ต้องการตรวจหา FI ตามประเภทการเกิด FI ซึ่งจะได้ไฟล์โปรเมลาแบ่งตามจำนวน LTL อีกครั้ง แสดงได้ดังภาพที่ 20



ภาพที่ 20 แสดงการสร้างโปรเมลาส่วน LTL

4.) เขียนโปรเมลาส่วน User เรียกใช้ Service

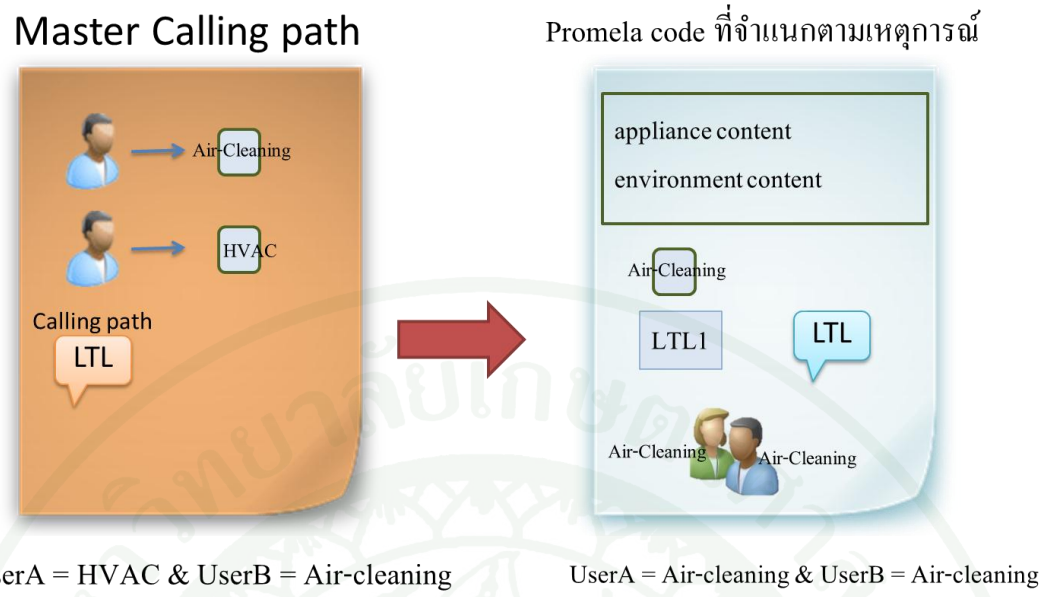
เมื่อทราบว่า User ใดใช้ Service ใด โปรแกรมควบคุมการทำงานแบบขนานจะกำหนด User ให้เรียกใช้ Service นั้นจาก Master Calling path ซึ่งข้อมูลในไฟล์จะรู้วิธีเรียกใช้งาน Service นั้นรวมทั้ง function และ parameter จะถูกนำมาเขียนการเรียกใช้งาน Service นั้นอย่างถูกต้อง ส่วนนี้ได้จากไฟล์ Master Calling path ส่วน service แสดงได้ดังภาพที่ 21



ภาพที่ 21 แสดงโปรเมลาส่วน User เรียกใช้ Service

5.) เขียนโปรเมลาส่วนเรียกใช้งาน LTL ในไฟล์โปรเมลา

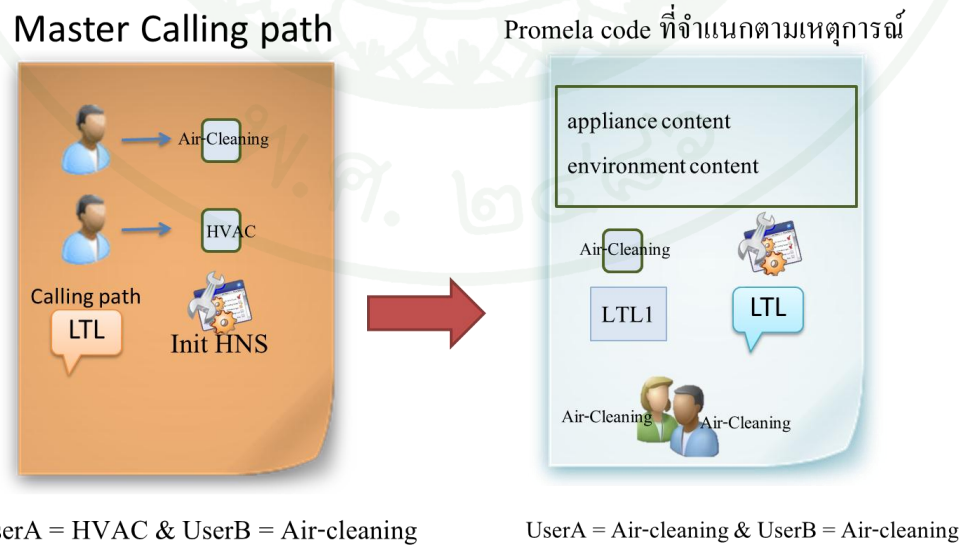
โปรแกรมควบการทำงานแบบขนานจะคัดลอกส่วน sourceCode ในไฟล์ Master Calling path มาใส่เพื่อให้สปินโมเดลเช็คกิ้งรู้ว่ามี LTL อยู่ในไฟล์และนำ sourceCode ไปใช้ตรวจหา LTL แสดงได้ดังภาพที่ 22



ภาพที่ 22 แสดงโปรเมลาส่วนเรียกใช้งาน LTL ในไฟล์โปรเมลา

6.) เขียนโปรแกรมโปรเมลาส่วน Functions อื่นๆที่ถูกเรียกใช้ในระบบ HNS

ส่วน Functions อื่นๆหรือส่วนใดๆ ที่ผู้ใช้ต้องการให้ระบุเพิ่ม เมื่อรันระบบ HNS บน SPIN ที่ต้องเขียนเพิ่มจะถูกคัดลอกจากไฟล์ Master Calling path ส่วน executeService เช่น การ set สภาพแวดล้อมเบื้องต้น แสดงได้ดังภาพที่ 23

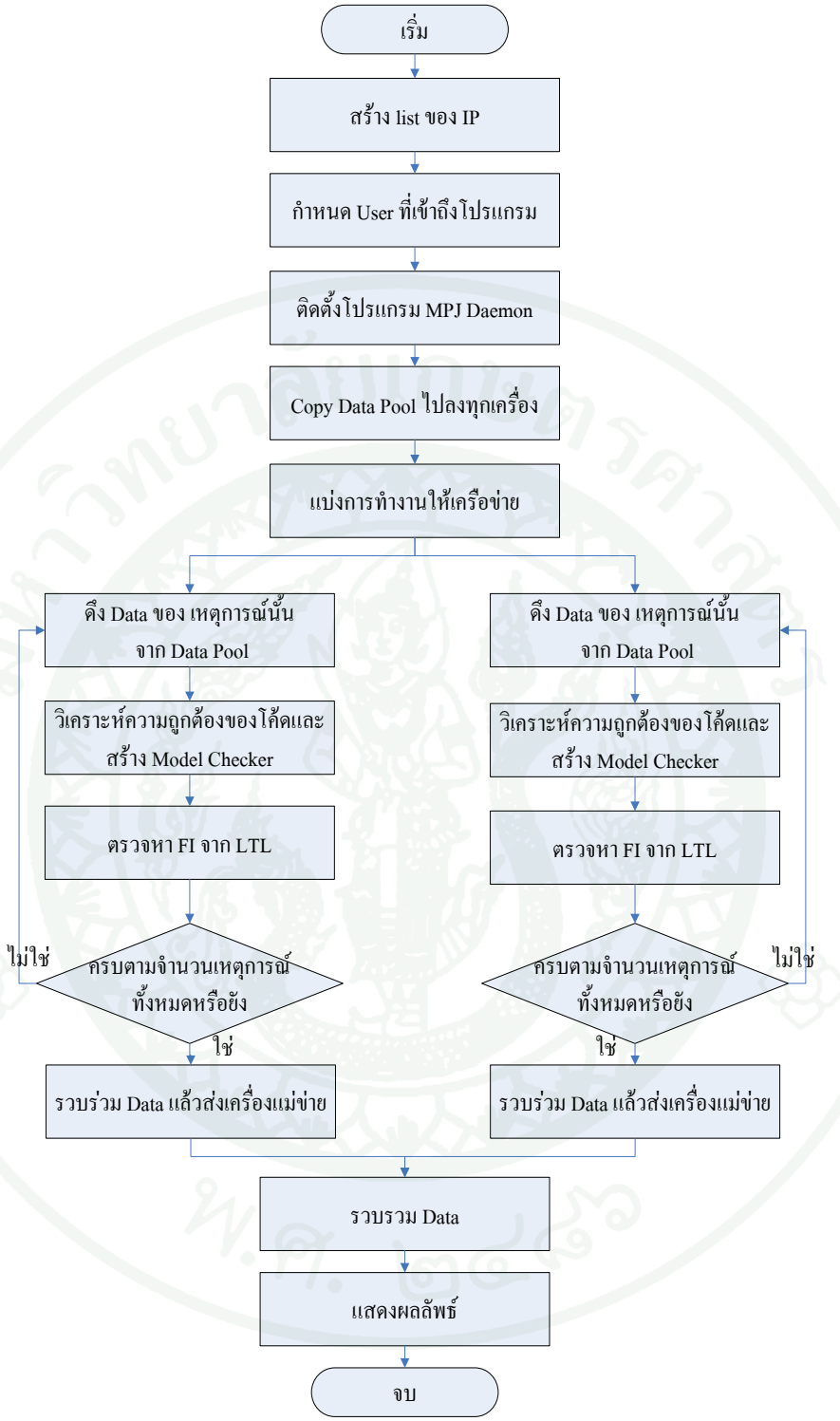


ภาพที่ 23 แสดงโปรแกรมโปรเมลาส่วน Functions อื่นๆที่ถูกเรียกใช้ในระบบ HNS

โปรแกรมควบคุมการทำงานแบบขนานจะทำตามลำดับการเขียนโปรเมลาโค้ดจนครบทุกเหตุการณ์ที่สามารถเกิดขึ้นได้ในระบบและรวบรวมไฟล์โปรเมลาเหล่านั้นไว้ใน Data Pool เพื่อเตรียมทำกระบวนการต่อไป

3. วิธีการตรวจหาฟีเจอร์อินเตอร์แอคชั่นแบบขนาน

ในขั้นตอนที่ 3 เป็นขั้นตอนวิธีในการตรวจหาฟีเจอร์อินเตอร์แอคชั่นโดยโปรแกรมควบการทำงานแบบขนานจะเริ่มทำงานแบบขนานตามขั้นตอนดังนี้

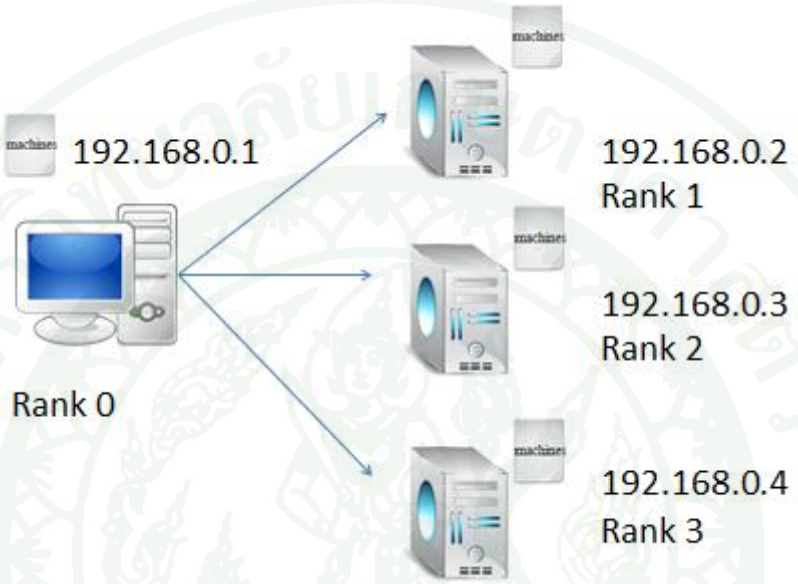


ภาพที่ 24 แสดงขั้นตอนวิธีในการตรวจหาฟีเจอร์อินเตอร์แอคชั่นโดยโปรแกรมควบการทำงานแบบขนาน

3.1 การเชื่อมโยงเครื่องประมวลผลในเครือข่าย

การประผลแบบขนานที่เขียนด้วย library MPJ ได้มีขั้นตอนในการเตรียมระบบเครือข่ายดังนี้

1.) นำ ip หรือ Host Name ของทุกเครื่องของในเครือข่ายมาเขียนไว้ในไฟล์ machine ซึ่ง จะถูกระบุเป็น rank ของเครื่องเครือข่ายตามลำดับที่ถูกเขียนในไฟล์ เริ่มจาก rank 0



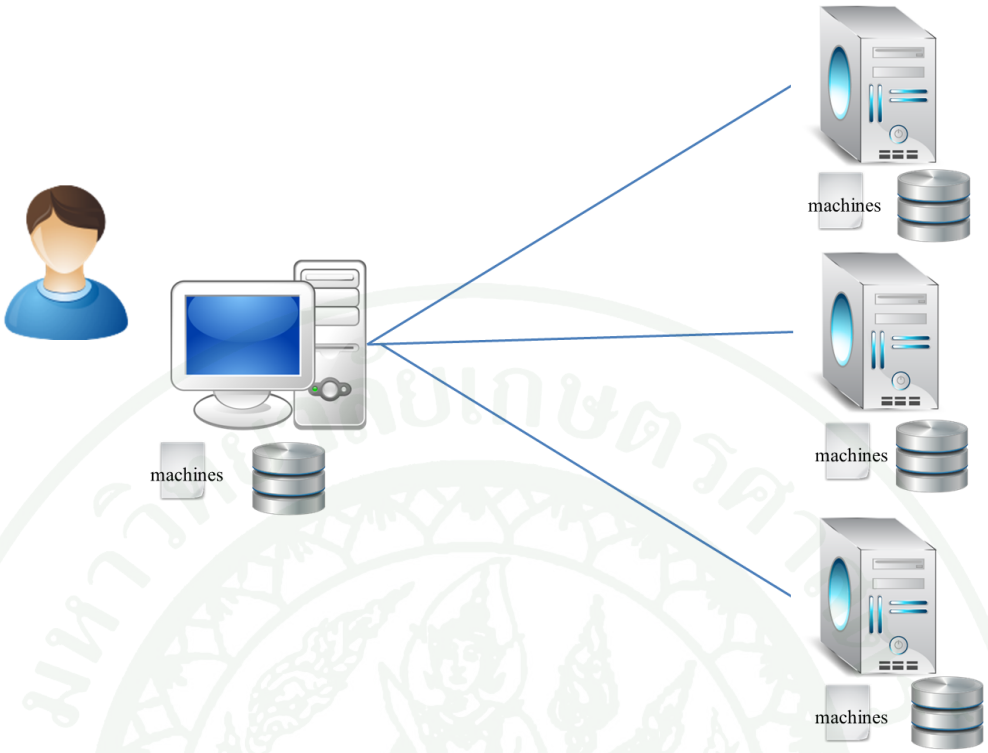
ภาพที่ 25 แสดงการเชื่อมโยงเครื่องประมวลผลในเครือข่าย

2.) สร้าง User ที่เข้าถึงโปรแกรม MPJ Daemon ได้ในทุกเครื่องในเครือข่าย

3.) ติดตั้งโปรแกรม MPJ Daemon โดยใช้คำสั่ง installmpjd-windows.bat ที่อยู่ใน MPJ_HOME

3.2 เตรียมข้อมูลและโปรแกรมที่ต้องมีในเครื่องประมวลผลอื่นๆในเครือข่าย

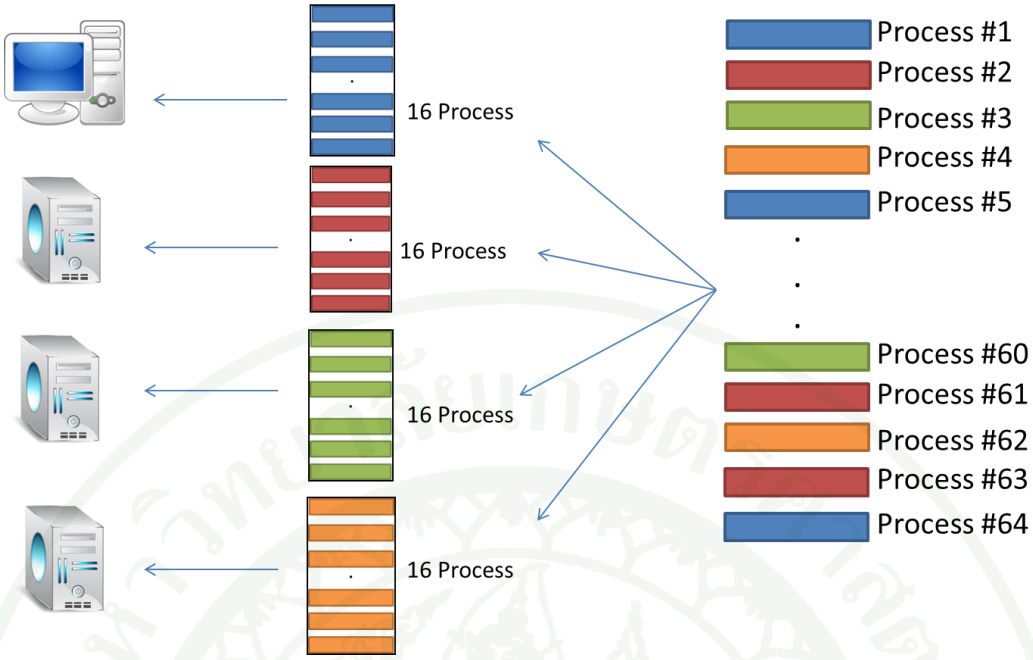
โปรแกรมควบคุมการทำงานแบบขนานเขียนตามแนวการทำงานแบบ MIMD ซึ่งจะแยกข้อมูลและคำสั่งที่ใช้ประมวลผลออกตาม เครื่องต่างจะไม่มีการรอข้อมูลหรือคำสั่งใดจากเครื่องอื่นผู้วิจัยจึงได้ทำการนำไฟล์โปรเมล่าโคดที่เตรียมไว้แล้วใน Data Pool และโปรแกรม SPIN ที่ต้องใช้ในการ สร้าง Model ตรวจหา FI Copy ส่งให้ทุกเครื่องประมวลผลที่อยู่ในเครือข่ายทุกๆ เครื่อง



ภาพที่ 26 แสดงการเตรียมข้อมูลและโปรแกรมที่ต้องมีในเครื่องประมวลผลอื่นๆในเครือข่าย

3.3 การแบ่งการทำงานแบบขนาน

โปรแกรมควบคุมการทำงานแบบขนานจะทำการแบ่ง process การทำงานโดยคิดจากจำนวนเหตุการณ์ที่สามารถเกิดได้ในระบบ HNS ที่ได้คำนวณก่อนหน้านั้น เช่นมีจำนวนเหตุการณ์ที่สามารถเกิดขึ้นได้ทั้งหมด 8 เหตุการณ์โปรแกรมจะแบ่งเหตุการณ์แต่ละเหตุการณ์ไปแต่ละ process และเครื่องในเครือข่ายจะดึง process ไปทำงานอย่างเท่าๆกัน เช่น มีเครื่องในเครือข่าย 4 เครื่องแต่ละเครื่องจะดึง process ไปทำงาน 2 process เท่าๆกัน และไม่ซ้ำ process กันโดยจะมีการเก็บข้อมูลว่าเครื่องในเครือข่ายเครื่องใดดึง process ใดไปทำงาน

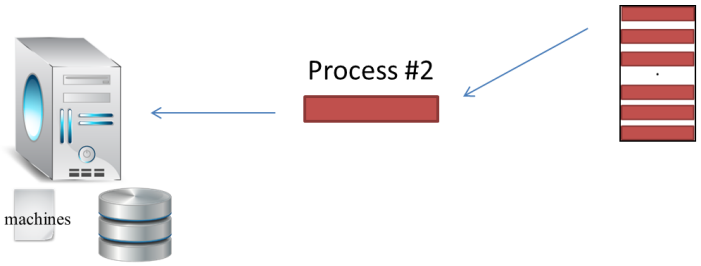


ภาพที่ 27 แสดงการแบ่งการทำงานแบบขนาน

3.4 ขั้นตอนการทำงานในแต่ละ process

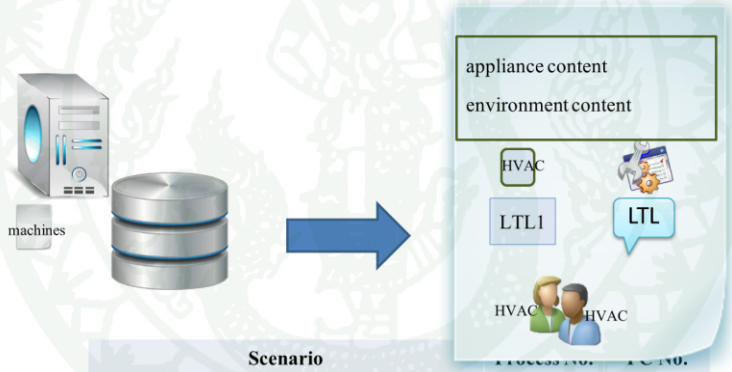
ลักษณะการทำงานในแต่ละ process ที่โปรแกรมควบคุมการทำงานแบบขนานแบ่งการทำงานจะมีลักษณะการทำงานที่เหมือนๆกัน ต่างกันที่ process นั้นทำงานอยู่ที่ข้อมูลใดของเหตุการณ์ที่สามารถเกิดขึ้นได้ทั้งหมดโดยมีการทำงานดังนี้

1.) เมื่อทราบ process ที่เครื่องประมวลได้ดึงมาทำงานว่าเป็นข้อมูลของเหตุการณ์ใดเครื่องประมวลผลก็จะดึงข้อมูลของเหตุการณ์นั้นออกมาจาก Data Pool ที่อยู่ในเครื่องประมวลผลแต่ละเครื่อง



Scenario	Process No.	PC No.
UserA = HVAC & UserB = Air-cleaning	Process 1	PC 0
UserA = HVAC & UserB = HVAC	Process 2	PC 1
UserA = Air-cleaning & UserB = HVAC	Process 3	PC 2
UserA = Air-cleaning & UserB = Air-cleaning	Process 4	PC 3

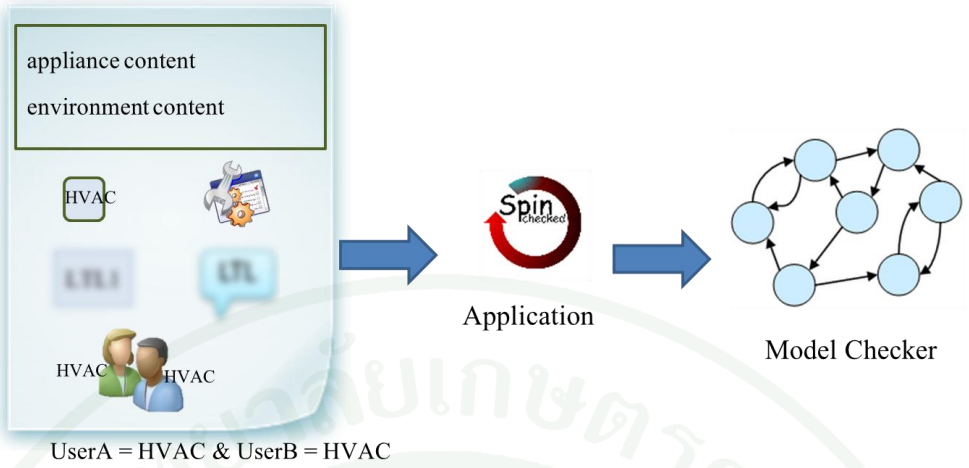
ภาพที่ 28 แสดงการดึงข้อมูลของเหตุการณ์นั้นออกมาจาก Data Pool



Scenario		
UserA = HVAC & UserB = Air-cleaning	Process 1	PC 0
UserA = HVAC & UserB = HVAC	Process 2	PC 1
UserA = Air-cleaning & UserB = HVAC	Process 3	PC 2
UserA = Air-cleaning & UserB = Air-cleaning	Process 4	PC 3

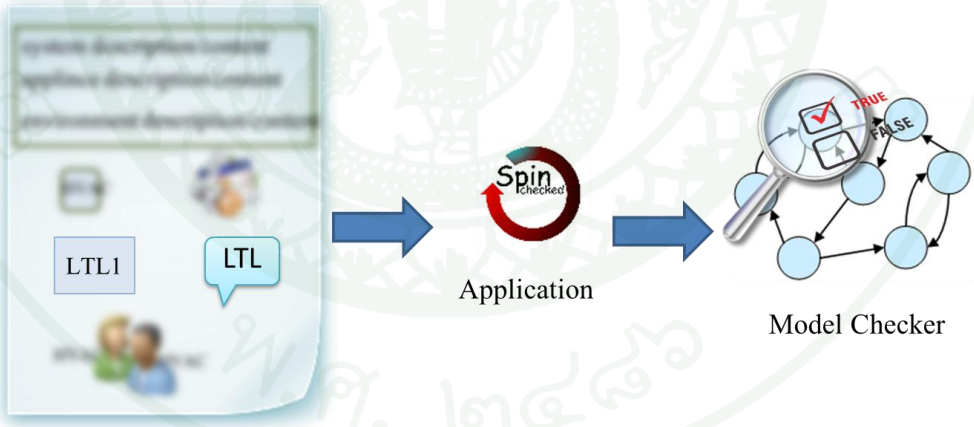
ภาพที่ 29 แสดงข้อมูลของ process

2.) นำไฟล์โปรเมลาของเหตุการณ์นั้นมาวิเคราะห์ความถูกต้องโดยสั่งงานให้โปรแกรม SPIN ที่อยู่ในเครื่องตรวจสอบความถูกต้องของไฟล์โปรเมลาและทำการสร้าง Model Checker ขึ้นตามไฟล์โปรเมลาของเหตุการณ์นั้น



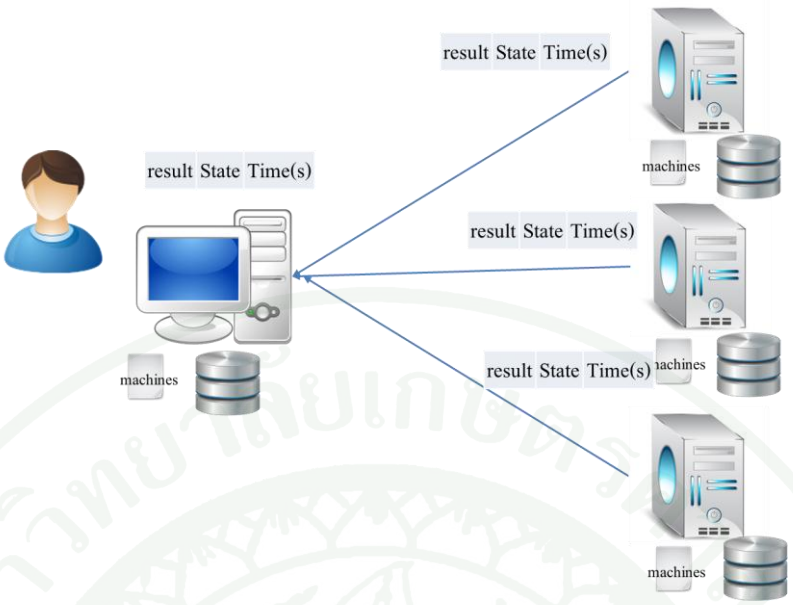
ภาพที่ 30 แสดงการสร้าง Model Checker

3.) เมื่อได้ Model Checker โปรแกรมควบคุมการทำงานแบบขนานจะดึง LTL ที่ต้องการตรวจสอบจากไฟล์โปรเมลาของเหตุการณ์นั้นมาตรวจสอบสมการ LTL จาก Model Checker ในโปรแกรม SPIN



ภาพที่ 31 แสดงการตรวจสอบสมการ LTL จาก Model Checker

4.) ผลลัพธ์ของ LTL และข้อมูลเหตุการณ์ที่ตรวจหา FI นั้นจะถูกส่งกลับมาที่เครื่องแม่ข่ายแต่ละเครื่องในเครือข่าย จะทำจะครบทุก process ที่เครื่องดึงมาทำงานแล้วส่งผลกับเครื่องแม่ข่ายจนครบทุกเกตุการณ์ที่สามารถเกิดขึ้นได้ใน HNS



ภาพที่ 32 แสดงการส่งผลกับเครื่องแม่ข่าย

3.5 วิธีการรวบรวมและการแสดงผลลัพธ์

เมื่อได้ผลลัพธ์ครบทุกๆ process ที่ได้สร้างขึ้นโปรแกรมจะนำผลลัพธ์และข้อมูลของแต่ละ process มาสร้างเป็นตารางแสดงผลลัพธ์ดังตารางที่ 3

ตารางที่ 3 แสดงผลลัพธ์ผลการตรวจหา FI

A	*B*	*C*	*File*	*type*	*result*	*State*	*Time(s)*
Cleaning	Cleaning	Cleaning.txt	HNS#TI_U#A-Cleaning#B-Cleaning#C-Cleaning.txt	TI_U	false	455685	9.11E-01
Cleaning	Cleaning	HVAC.txt	HNS#TI_U#A-Cleaning#B-Cleaning#C-HVAC.txt	TI_U	false	913	9.00E-03
Cleaning	HVAC	Cleaning.txt	HNS#TI_U#A-Cleaning#B-HVAC#C-Cleaning.txt	TI_U	false	913	9.00E-03
Cleaning	HVAC	HVAC.txt	HNS#TI_U#A-Cleaning#B-HVAC#C-HVAC.txt	TI_U	false	1117	1.10E-02
HVAC	Cleaning	Cleaning.txt	HNS#TI_U#A-HVAC#B-Cleaning#C-Cleaning.txt	TI_U	false	913	1.10E-02
HVAC	Cleaning	HVAC.txt	HNS#TI_U#A-HVAC#B-Cleaning#C-HVAC.txt	TI_U	false	1117	1.00E-02
HVAC	HVAC	Cleaning.txt	HNS#TI_U#A-HVAC#B-HVAC#C-Cleaning.txt	TI_U	false	1117	1.00E-02
HVAC	HVAC	HVAC.txt	HNS#TI_U#A-HVAC#B-HVAC#C-HVAC.txt	TI_U	false	1323	1.10E-02

ผลและวิจารณ์

ผล

1. การทำงานแบบขนาน

งานวิจัยนี้ได้พัฒนาการตรวจหา FI โดยใช้ SPIN ให้สามารถทำการตรวจหาแบบขนานได้ 2 วิธีคือ แบบ Multi-core และแบบ Cluster ด้วยกระบวนการพัฒนาดังนี้คือ

1.1 กระบวนการออกแบบการตรวจสอบจำนวนเหตุการณ์ทั้งหมดที่สามารถเกิดขึ้นได้

การคำนวณจำนวนเหตุการณ์ทั้งหมดเพื่อแบ่งเหตุการณ์ให้ทำงานแบบขนานนั้น จากสูตรที่กล่าวข้างต้นผู้วิจัยได้เขียนโปรแกรมเพื่อคำนวณหาจำนวนเหตุการณ์ทั้งหมดที่สามารถเกิดขึ้นได้โดยวิเคราะห์จำนวนของเซอร์วิสและยูสเซอร์จากภาษาโปรเมลา

1.2 กระบวนการออกแบบการทำงานของ SPIN แบบขนาน

สถาปัตยกรรมของการทำงานแบบขนานที่แบ่งตาม Instruction stream และ Data stream แบบ MIMD (Multiple Instruction stream, Multiple Data stream) ซึ่งเป็นการแบ่งการทำงานของงานวิจัยนี้ โดยแบ่งคำสั่งการทำงานออกเป็นคำสั่งย่อยๆ (Fine-grain Tasks) และแบ่งการทำงานออกไป โดยเขียนโปรแกรมเพื่อเพิ่มการทำงานของ SPIN ให้เป็นลักษณะ Multitasking เพิ่มการทำงานของ SPIN ได้ตามจำนวนที่ต้องการ โดยงานวิจัยนี้พัฒนาบนภาษา JAVA โดยใช้ MPI (Message Passing Interface) ในการติดต่อสื่อสารถึงกันระว่าง Processor ซึ่งมีอยู่ใน library ชื่อ MPJ เพื่อให้ส่งข้อมูลและรันการทำงานของ SPIN พร้อมทั้งจัดสรรทรัพยากรที่จำเป็นในการรัน SPIN จึงสามารถรัน SPIN แบบขนานในเวลาเดียวกันได้ คำสั่งในการทำงานแสดงได้ดังภาพที่ 6

```
1. MPI.Init(args);
2. MPI.COMM_WORLD.Recv(fileName, 0, fileName.length,
MPI.CHAR, 0, 99);
3. MPI.COMM_WORLD.Send(new Object[]{spinResult},0, 1,
MPI.OBJECT, 0, 99);
4. MPI.Finalize();
```

ภาพที่ 33 Code ในการสั่งงาน MPJ

อธิบายเพิ่มเติมจากภาพที่ 17

1.) บรรทัดที่ 1 MPI.Init เป็นการประกาศค่าเริ่มใช้ MPI ในการติดต่อระหว่าง Processor

2.) บรรทัดที่ 2 MPI.COMM_WORLD.Recv เป็นการระบุ Data ในการรับข้อมูลระหว่าง Processor

3.) บรรทัดที่ 3 MPI.COMM_WORLD.Send เป็นการระบุ Data ในการส่งข้อมูลระหว่าง Processor

4.) บรรทัดที่ 4 MPI.Finalize เป็นประกาศคืนค่าทรัพยากรแล้วจบการทำงาน

1.3 กระบวนการปรับเปลี่ยน Mode ของ SPIN

การทำงานแบบขนานแบบ MIMD นั้นใช้เทคโนโลยีได้ 2 แบบ คือ Multi-core และ Cluster ความสามารถอีกอย่างของ MPJ คือสามารถปรับ MODE ในการเลือกการทำงาน การคุยกันระหว่าง Processor ไม่ว่าจะเป็นบน core เดียวกัน หรือแบบ Processor คนละเครื่อง ผู้พัฒนาการแบ่งการทำงานขอ SPIN ออกเป็นสองแบบด้วยกันซึ่งทำให้ได้ลักษณะการทำงานแบบหลากหลายและเพิ่มประสิทธิ์ภาพให้เห็นผลลัพธ์แตกต่างกัน วิธีการเลือก Modeใช้คำสั่งดังนี้

Multi-core: mpjrun.bat -np 2 –dev multicore

Cluster: mpjrun.bat -np 2 –dev niodev

ภาพที่ 34 คำสั่งในการเลือก Mode การทำงานของ SPIN

1.4 กระบวนการรวมผลการประมวลผลแบบขนาน

บันทึกผลลัพธ์ที่ได้ในแต่ละ Processor เปรียบเทียบผลแต่ละเหตุการณ์เพื่อให้ได้ผลการตรวจสอบการขัดแย้งกันของเซอร์วิส ซึ่งผู้วิจัยได้พัฒนาโปแกรมให้สามารถดึงข้อมูลที่ได้จาก Multitasking นั้นๆ มารวมกันและแสดงผลในรูปแบบตารางแล้วบันทึกลงไฟล์

2. การทดลอง

ผู้วิจัยได้ออกแบบการทดลองจากการตรวจสอบการขัดแย้งกันของเซอร์วิสโดยเปรียบเทียบการทำงานของแต่ละรูปแบบด้วยการวัดจาก Execution time ของเวลาในการตรวจหา FI ได้ดังนี้

2.1 Sequence

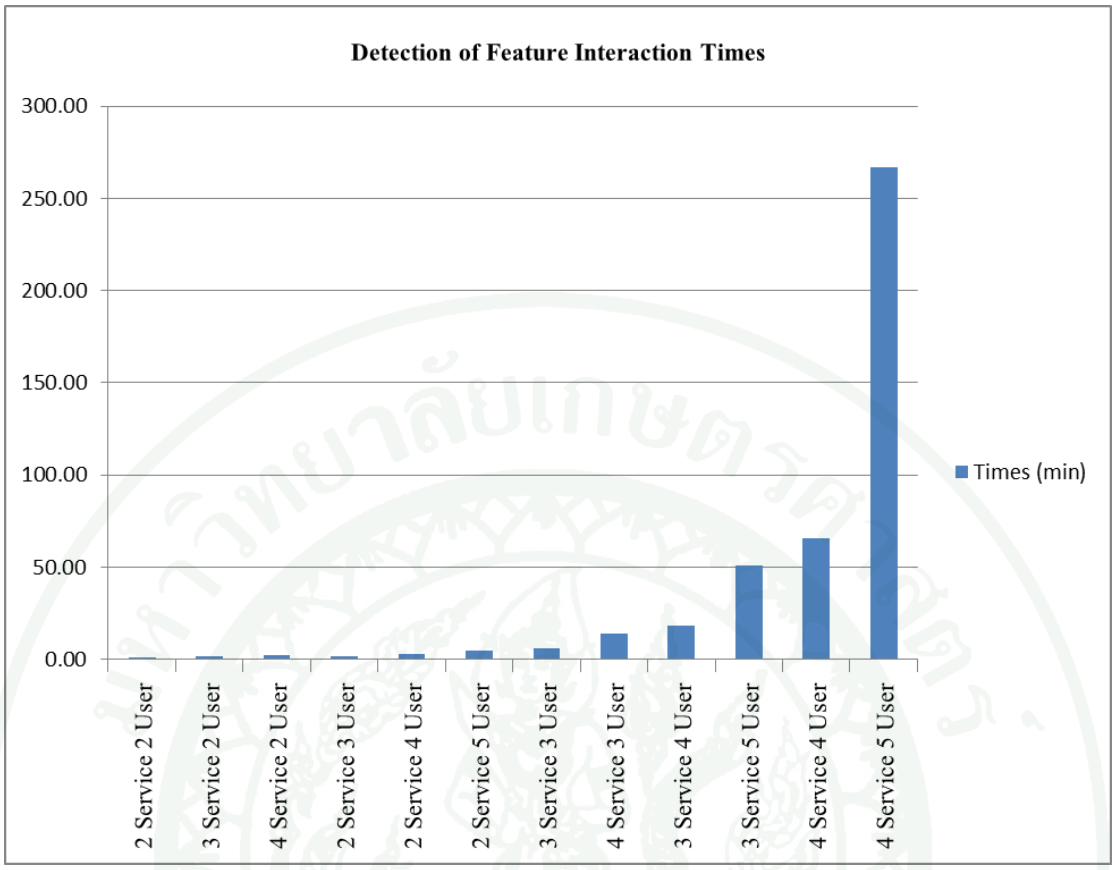
จากการทดลองให้ SPIN ทำงานแบบ Sequence กล่าวคือรัน SPIN ไปทีละ process จนครบตามจำนวนเหตุการณ์ทั้งหมดผลลัพธ์ที่ได้แสดงในตารางที่ 4

ซึ่งจำนวนเหตุการณ์จะเพิ่มขึ้นเป็นแบบ Exponential โดยในงานวิจัยนี้ต้องการจำลองทุกๆ เหตุการณ์แบบอัตโนมัติ ทำให้เวลาในการตรวจสอบเซอร์วิสของ HNS มากขึ้นผลลัพธ์ที่ได้แสดงตามตารางที่ 4

ตารางที่ 4 เวลาที่ใช้ในการตรวจหา FI เมื่อเซอร์วิสและ ยูสเซอร์ เปลี่ยนแปลง

Service and User	*Scenario*	*State*	*Time (minutes)*
2 Service 2 User	4	8529322	0.84
3 Service 2 User	9	9719370	1.48
4 Service 2 User	16	11421118	2.32
2 Service 3 User	8	15064031	1.62
2 Service 4 User	16	20259243	2.82
2 Service 5 User	32	25116237	4.83
3 Service 3 User	27	56311398	5.71
4 Service 3 User	64	120227860	13.72
3 Service 4 User	81	181444617	18.53
3 Service 5 User	243	463086481	51.05
4 Service 4 User	256	645031861	65.92
4 Service 5 User	1024	2595486598	266.72

จากตารางที่ 2 เป็นการตรวจหา FI เพียงตัวเดียวในระบบซึ่งใช้เวลาในการตรวจหามากขึ้นเมื่อยูสเซอร์กับเซอร์วิสเพิ่มขึ้น และเริ่มที่จะรอการตรวจหาไม่ได้เมื่อยูสเซอร์กับเซอร์วิสมีค่าเป็น 4 ทั้งคู่ และไม่สามารถทำการทดลองให้เสร็จในครั้งเดียวได้เมื่อมี 4 เซอร์วิส 5 ยูสเซอร์ จึงต้องแบ่งการทำงานออกเป็น 4 รอบการทำงาน จะเห็นผลได้ชัดเจนยิ่งขึ้นจากภาพที่ 19 ซึ่งเวลาที่ใช้ในการตรวจหามากขึ้นเป็นเท่าตัวเมื่อแสดงผลเปรียบเทียบโดยกราฟแท่ง



ภาพที่ 35 กราฟแสดงเวลาในการตรวจหา FI ตามจำนวน เซอร์วิส และ ยูสเซอร์ ที่เพิ่มขึ้น

2.2 GRID

จากการทดลองให้ SPIN ทำงานแบบขนานบนเทคโนโลยีกริด โดยใช้ Node ในการทำงานจำนวน 2 Node และ 4 Node ผลลัพธ์ที่ได้แสดงในตารางที่ 5

ตารางที่ 5 ผลการตรวจหา FI โดยใช้ GRID

Service and User	*Scenario*	*State*	*Time (minutes)*	
			GRID 2 Node	**GRID 4 Node**
2 Service 2 User	4	8529322	0.30	0.24
3 Service 2 User	9	9719370	0.33	0.29
4 Service 2 User	16	11421118	0.36	0.30
2 Service 3 User	8	15064031	0.37	0.31
2 Service 4 User	16	20259243	0.48	0.40
2 Service 5 User	32	25116237	0.56	0.48
3 Service 3 User	27	56311398	0.81	0.64
4 Service 3 User	64	120227860	1.89	1.19
3 Service 4 User	81	181444617	2.72	1.52
3 Service 5 User	243	463086481	7.74	4.64
4 Service 4 User	256	645031861	9.26	5.80
4 Service 5 User	1024	2595486598	43.23	24.15

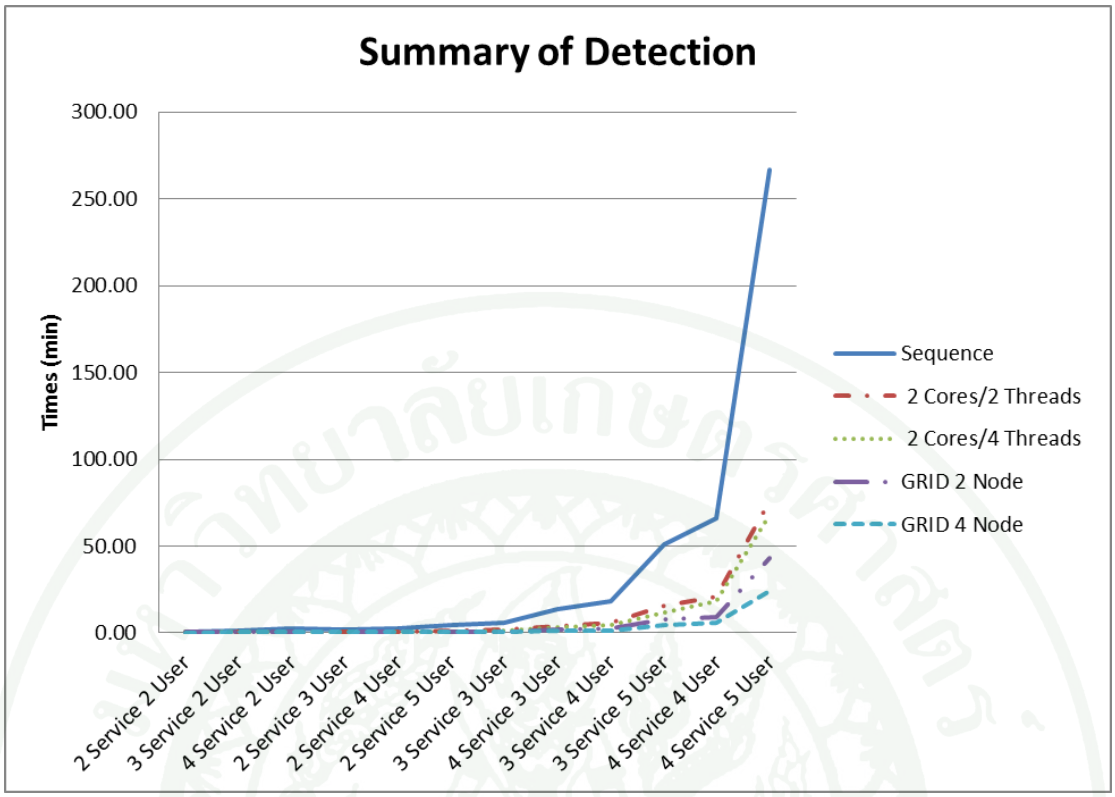
2.3 Multi-core

จากการทดลองให้ SPIN ทำงานแบบขนานบนเทคโนโลยี Multi-core ผลลัพธ์ที่ได้แสดงในตารางที่ 6

ตารางที่ 6 ผลการตรวจหา FI โดยใช้ Multi-Core

Service and User	*Scenario*	*State*	*Time (minutes)*	
			2 core / 2 Threads	**2 core / 4 Threads**
2 Service 2 User	4	8529322	0.40	0.38
3 Service 2 User	9	9719370	0.40	0.41
4 Service 2 User	16	11421118	0.48	0.45
2 Service 3 User	8	15064031	0.55	0.46
2 Service 4 User	16	20259243	0.75	0.63
2 Service 5 User	32	25116237	1.01	0.82
3 Service 3 User	27	56311398	1.76	1.43
4 Service 3 User	64	120227860	3.86	3.02
3 Service 4 User	81	181444617	5.75	4.44
3 Service 5 User	243	463086481	15.68	11.82
4 Service 4 User	256	645031861	21.15	18.05
4 Service 5 User	1024	2595486598	75.97	68.20

เพื่อให้สามารถเปรียบเทียบและวิเคราะห์ผลลัพธ์ที่ได้จากการทำงานของแต่ละรูปแบบได้อย่างชัดเจน สามารถนำผลลัพธ์ที่ได้ของแต่ละการทดลองมาแสดงผลด้วยกราฟเส้นแสดงได้ดังภาพที่ 20



ภาพที่ 36 กราฟสรุปเวลาในการตรวจหา FI ด้วยวิธีการต่างๆ

การทดลองแบบ Sequence ใช้การประมวลผลบน CPU แบบ MULTI-CORE แต่รันการทำงานตรวจหา FI ไปที่ละเหตุการณ์ และการทดลองแบบ Paralle คือรันการทำงานตรวจหา FI ไปพร้อมๆ กันหลายเหตุการณ์เท่าที่เครื่องสามารถรันพร้อมกันได้ โดยแบ่งการทดลองเป็น แบบ MULTI-CORE โดยใช้การประมวลผลบน CPU แบบ MULTI-CORE ซึ่งเรียกใช้งานได้ 4 Threads และ ได้ 2 Threads สำหรับการทดลองแบบกริดใช้การประมวลผลบนเครื่องคอมพิวเตอร์ที่เชื่อมต่อกัน 2 เครื่องและ 4 เครื่อง โดยใช้เครื่องที่มี CPU รุ่นเดียวกันคือ i5 ซึ่งความเร็วของ CPU 2.4 GHz ใช้เครื่องมือตรวจสอบแบบโมเดลเช็คกิ้งสปิน รูปแบบ CODE ภาษาซีและพัฒนาโปรแกรมควบคุมด้วยภาษา JAVA โดยการทำงานของ library MPJ

วิจารณ์

การทดลองโดยวิเคราะห์เวลาที่ใช้ในการประมวลผลตั้งแต่เริ่มต้นจนสิ้นสุดจากการตรวจหา FI จนแล้วเสร็จทุกเหตุการณ์ในทุกรูปแบบของการทดลอง ผลการทดลองทำให้ทราบถึงประสิทธิภาพที่เพิ่มขึ้นเมื่อเปรียบเทียบเวลาในการตรวจหา FI แสดงได้ดังภาพที่ 36 จะเห็นว่าเวลาในการตรวจหา FI ลดลงไปตามจำนวน Threads ที่เพิ่มขึ้นของการทำงานแบบขนาน ทำให้การตรวจหา FI ใช้เวลาน้อยลงดังภาพที่ 36 ซึ่งเวลาในการตรวจหา FI ของระบบ HNS ที่มี 4 เซอร์วิส 5 ยูสเซอร์ จากการทำงานแบบ Seq ใช้เวลา 266.72 นาทีในการตรวจหา แต่การทำงานแบบ Multi-Core บน CPU 2 Cores/2 Threads ใช้เวลาเพียง 75.97 นาที ลดลง 71.52% หรือประมาณ 70% จากผลการทดลองทั้งหมดซึ่งเป็นเวลาที่ช้าที่สุดในรูปแบบ Paralle และการทดลองที่ได้ผลเร็วที่สุดคือการทำงานบน GRID แบบ 4 เครื่อง ใช้เวลาเพียง 24.15 นาที ลดลง 90.94% หรือประมาณ 86% จากผลการทดลองทั้งหมด

จากผลการทดลองจะเห็นได้ว่าการเพิ่มขึ้นของ Threads ทำให้การตรวจหาเร็วขึ้นอย่างมากตามการแบ่ง Process ที่เพิ่มขึ้น แต่จากการทดลองหลายครั้งทำให้ผู้วิจัยเห็นปัญหาบางกรณีที่จำเป็นต้องเอาเครื่องที่มีประสิทธิภาพไม่เท่ากันมาทำการทดสอบทำให้ผลการตรวจหานั้นช้าลง ก็เพราะมีการแบ่งการทำงานที่เท่าๆกันในแต่ละเครื่อง ทำให้เกิดการรอผลลัพธ์ของการตรวจหาจากเครื่องที่ช้าที่สุดในระบบ ทำให้เวลาที่ใช้ในการประมวลผลตั้งแต่เริ่มต้นจนสิ้นสุดจากการตรวจหา FI จนแล้วเสร็จทุกเหตุการณ์ เป็นเวลาของเครื่องที่ช้าที่สุดซึ่งเวลานั้นอาจจะช้ากว่ารูปแบบการทำงานบนเครื่อง Multi-Core ซึ่งมีการประมวลผลเท่ากันทุก Core

สรุปและข้อเสนอแนะ

สรุป

สรุปผลที่ได้จากงานวิจัยนี้ เนื่องจากการตรวจหา FI ในระบบเครือข่ายบ้านอัจฉริยะด้วยสปินโมเดลเช็คกิ้งนั้นมีความยากลำบากมากขึ้น อันเกิดจากการเพิ่มขึ้นของจำนวน Service และจำนวน User ในระบบทำให้จำนวนเหตุการณ์ที่ต้องตรวจหา FI นั้นเพิ่มมากขึ้น จนทำให้เวลาในการตรวจหานั้นมากจนรอการตรวจหาไม่ได้ในเวลาจริง จึงต้องพัฒนาการตรวจหา FI โดยประยุกต์เอาวิธีประมวลผลแบบขนานมาช่วยในการตรวจหา โดยการเขียนโปรแกรมควบคุมการทำงานแบบขนานด้วย library MPJ ที่ใช้สำหรับการติดต่อสื่อสารระหว่างหน่วยประมวลผล โดยโปรแกรมควบคุมแบ่งการทำงานออกเป็น process ย่อยๆ ตามจำนวนเหตุการณ์ที่สามารถเกิดขึ้นได้จากจำนวน Service และจำนวน User ในระบบที่ต้องการตรวจหา จึงสามารถแยกการตรวจหา IF ไปตามเหตุการณ์และตรวจหา FI ไปพร้อมๆกันหลายๆเหตุการณ์ได้ ทำให้ผลการตรวจหา FI นั้นตรวจหาได้เร็วขึ้นตามจำนวน Threads ที่สามารถนำเอา process ไปทำงานได้ การตรวจหา FI นั้นจึงใช้เวลาในการตรวจหาน้อยลงและจะน้องลงไปอีกเมื่อมี Threads มาช่วยแบ่งการทำงานเพิ่มมากขึ้น งานวิจัยนี้จึงสามารถตรวจหา FI ให้แล้วเสร็จได้ในเวลาจริงและสามารถนำการตรวจหา FI แบบขนานนี้ไปใช้งานบนระบบที่มีเครื่องคอมพิวเตอร์หลายเครื่องได้โดยไม่ต้องซื้อเครื่องคอมพิวเตอร์ที่มีประสิทธิภาพสูงและราคาแพงแต่งานวิจัยนี้ใช้ประโยชน์จากจำนวนเครื่องหลายๆ เครื่องมาช่วยกันประมวลผลพร้อมกันได้ทำให้ประหยัดงบประมาณในการลงทุนไปได้

ข้อเสนอแนะ

แนวทางพัฒนาต่อสามารถจะนำกระบวนการวิจัยนี้ไปพัฒนาการตรวจหา FI ในระบบอื่นๆ ที่มีโอกาสในการเพิ่มขึ้นของจำนวนเหตุการณ์ที่ต้องตรวจหา FI ทำให้การตรวจหา FI พัฒนาเป็นแบบขนานเหมือนกันได้ ขึ้นอยู่กับการปรับส่วนต่างๆที่มีอยู่ในงานวิจัย เช่น ปรับส่วนโค้ดโปรเมลาให้มีการนิยามระบบอื่นที่ต้องการตรวจหา FI เหมือนกับระบบ HNS ก็จะสามารถตรวจหา FI ในระบบอื่นแบบขนานได้เช่นกัน และสามารถเพิ่มความเร็วในการตรวจหา FI ได้จากการใช้เครื่องคอมพิวเตอร์ที่มีอยู่แล้วในระบบมาเชื่อมโยงกันให้มีจำนวนมากขึ้น ทำให้คุ้มค่าในการลงทุนในการสร้างระบบในการตรวจหา FI นี้

จากปัญหาในการวิจัยพบว่าปัญหาในการแบ่ง process ในการทำงานนั้นหากแบ่งให้เท่ากันในแต่ละ Node อาจทำให้เกิดปัญหาในการรอ Node ที่ช้าที่สุด เพราะความเป็นจริง Node ที่มีอยู่ในระบบจะมีประสิทธิภาพในการประมวลผลไม่เท่ากันเพราะฉะนั้นควรมีการทำ load balance ในการแบ่ง process ไปตามประสิทธิภาพในการประมวลในแต่ละ Node ให้เวลาในตรวจหา FI ของทุกๆ Node เสร็จพร้อมกันจะได้เวลาที่เร็วที่สุด

เอกสารและสิ่งอ้างอิง

Holzmann, G.J. 1997. The model checker SPIN. **IEEE Trans. Software Eng**. 23(5): 279–295.

_____ and D. Bošnački. 2007. The Design of a Multi-Core Extension of the SPIN Model Checker. **Trans. Software Eng**. 33(10): 659-674. TSE-0004-0107

_____ and _____. 2008. The Design of a Multi-Core Extension of the SPIN Model Checker. **Electronic Notes in Theoretical Computer Science (ENTCS)**. 198(1): 3-16. ISSN: 1571-0661.

Kolberg, M., E. H. Magill and M. Wilson. 2003. Compatibility Issues between Services Supporting Networked Appliances, pp. 136-147. *In* **IEEE Communications Magazine, vol. 41, no. 11**. November 2003.

Leelaprute, P., M. Nakamura, T. Tsuchiya, K. Matsumoto and T. Kikuno. 2005. Describing and verifying integrated services of home network systems, pp. 549–558. *In* **The12th Asia-Pacific Software Engineering Conference (APSEC2005)**. 15-17 December 2005. Taipei, Taiwan.

Matsuo, T., P. Leelaprute, T. Tsuchiya, T. Kikuno, M. Nakamura, H. Igaki and K. Matsumoto. 2006. Automatically verifying Integrated Services in Home Network Systems, pp.173-176. *In* **Proceedings of 21st International Technical Conference on Circuits/Systems, Computer and Communications (ITC-CSCC 2006),** 10-13 July 2006. Chiang Mai, Thailand.

Microsoft Corporation. 2009. **EXPLORE The .NET Framework**. Available Source: http://www.microsoft.com/NET/, August 18, 2008.

Nakamura, M., H. Igaki., H. Tamada and K. Matsumoto. 2004. Implementing integrated services of networked home appliances using service oriented architecture, pp. 269-278. *In* **Proceedings of 2nd International Conference on Service Oriented Computing (ICSOC2004)**. 15-18 November 2004. New York City, USA.

Riley, M.W., J.D. Warnock and D.F. Wendel. 2007. Cell Broadband Engine processor: Design and implementation, pp.545-557. *In* **IBM J. RES. & DEV. VOL. 51 NO. 5**. SEPTEMBER 2007.

Sun Microsystems. 2005. **Java ME Technology**. Available Source: http://developers.sun.com/mobility/j2me/, August 18, 2008.

SPIN Formal Verification. 2010. **ON-THE-FLY, LTL MODEL CHECKING with SPIN**. Available Source: http://spinroot.com, July 25, 2010.

Thai National Grid Center. **Grid Technology**. Available Source: http://www.thaigrid.or.th, August 19, 2008.

ประวัติการศึกษา และการทำงาน

ชื่อ –นามสกุล	ว่าที่ร้อยตรีวรวุทธ นัมคนิสรณ์
วัน เดือน ปี ที่เกิด	วันที่ 18 กันยายน 2526
สถานที่เกิด	กรุงเทพมหานคร
ประวัติการศึกษา	ปริญญาตรี หลักสูตรวิทยาศาสตรบัณฑิต คณะวิทยาศาสตร์ สาขาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยราชภัฏพระนคร ปี 2548
ตำแหน่งหน้าที่การงานปัจจุบัน	นักคอมพิวเตอร์ 4
สถานที่ทำงานปัจจุบัน	บริษัท ทีโอที จำกัด (มหาชน)
ผลงานดีเด่นและรางวัลทางวิชาการ	2011 Eighth International Joint Conference on Computer Science and Software Engineering (JCSSE)
ทุนการศึกษาที่ได้รับ	-