**โครงงานวิศวกรรมคอมพิวเตอร์**
**ภาควิชาวิศวกรรมคอมพิวเตอร์**
**คณะวิศวกรรมศาสตร์**
**มหาวิทยาลัยเกษตรศาสตร์**

**เรื่อง**

# การพัฒนามิดเดิลแวร์สำหรับระบบการประมวลผลแบบกลุ่มเมฆพร้อมระบบเคลื่อนย้ายคอมพิวเตอร์เสมือนแบบอัตโนมัติ

# A Development of Cloud Middleware with Automatic Virtual Machine Migration

**โดย**

**นายประยุกต์ เจตสิกทัต 51052090**

**พ.ศ. 2554**

การพัฒนามิดเดิลแวร์สำหรับระบบการประมวลผลแบบกลุ่มเมฆ
พร้อมระบบเคลื่อนย้ายคอมพิวเตอร์เสมือนแบบอัตโนมัติ
A Development of Cloud Middleware with Automatic Virtual Machine Migration

โดย
นายประยุกต์ เจตสิกทัต 51052090

โครงงานวิศวกรรมคอมพิวเตอร์
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
มหาวิทยาลัยเกษตรศาสตร์

ตามหลักสูตร
วิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมคอมพิวเตอร์

ได้รับการพิจารณาเห็นชอบโดย
อาจารย์ที่ปรึกษาโครงงาน.....วันที่.....เดือน.....พ.ศ.....
(ผศ. ดร. ภุชงค์ อุทโยภาศ)
หัวหน้าภาควิชาวิศวกรรมคอมพิวเตอร์.....วันที่.....เดือน.....พ.ศ.....
(ผศ. ดร. ภุชงค์ อุทโยภาศ)

นายประยุกต์ เจตสิกทัต ปีการศึกษา 2554
การพัฒนามิดเดิลแวร์สำหรับระบบการประมวลผลแบบกลุ่มเมฆ
พร้อมระบบเคลื่อนย้ายคอมพิวเตอร์เสมือนแบบอัตโนมัติ
ปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์

**บทคัดย่อ**

ระบบการประมวลผลแบบกลุ่มเมฆที่ให้บริการระดับโครงสร้างพื้นฐาน มีรากฐานอยู่บนเทคโนโลยีเวอชวลไลเซชั่น ซึ่งทำให้ระบบสามารถให้ที่อยู่กับทรัพยากรคอมพิวเตอร์เป็นจำนวนมากได้ การจัดการพลังงานอย่างมีประสิทธิภาพสามารถช่วยลดค่าใช้จ่ายในการสร้างระบบการประมวลผลแบบกลุ่มเมฆขนาดใหญ่ได้เป็นอย่างมาก การใช้พลังงานในระบบโดยส่วนมากจะขึ้นกับปริมาณภาระงาน รวมถึงลักษณะการใช้งานเครื่องคอมพิวเตอร์ โครงการนี้ได้เสนอขั้นตอนวิธีอันหนึ่ง ที่จะช่วยลดปริมาณการใช้พลังงานของระบบได้ ขั้นตอนวิธีนี้มีแนวคิดพื้นฐานคือ ควบคุมการใช้พลังงานให้ลู่เข้าสู่ระดับการใช้งานจริงของระบบ ด้วยสองขั้นตอนหลัก คือ การปรับเปลี่ยนขนาดของระบบ และการทำให้เกิดความสมดุลของภาระงาน ขั้นตอนวิธีนี้ถูกทดสอบบนมิดเดิลแวร์ของระบบประมวลผลแบบกลุ่มเมฆ ที่ชื่อว่า เมฆินทร์ ซึ่งถูกออกแบบและสร้างขึ้นในโครงการนี้เช่นกัน ผลการทดสอบแสดงถึงการลดลงของการใช้พลังงานตามที่คาดไว้ โครงการนี้ได้สร้างความรู้ ความเข้าใจที่ดีขึ้นเกี่ยวกับการจัดการภาระงานในระบบการประมวลผลแบบกลุ่มเมฆ เพื่อการประหยัดพลังงาน ซึ่งจะเป็นประโยชน์อย่างมากต่อการพัฒนาการ จัดสรรทรัพยากรในระบบการประมวลผลแบบกลุ่มเมฆต่อไป

**คำสำคัญ** การประมวลผลแบบกลุ่มเมฆ, มิดเดิลแวร์, การจัดการพลังงานอย่างมีประสิทธิภาพ, การสมดุลภาระงาน

เลขที่เอกสารอ้างอิงภาควิชา E9006-PU-2-2554

Jatesiktat, Prayook Academic Year 2011
A Development of Cloud Middleware with Automatic Virtual Machine Migration
Bachelor Degree in Computer Engineering, Department of Computer Engineering
Faculty of Engineering, Kasetsart University

**Abstract**

Infrastructure as a Service Cloud system is based on virtualization technology to enables the hosting of massive number of computing resources. The efficient power management can help lower the cost of operating a large cloud substantially. Power usage in a cloud system depends largely on the system load and the proper use of physical machines. This project proposes a new algorithm that can be used to decrease the system power consumption substantially. This algorithm is based on the converging of the power usage to system utilization using 2 phases strategy – scaling phase and balancing phase. The implementation on our cloud system, Maekin, shows the result of lower power consumption as expected. The contribution of this work is a better understanding of workload management in cloud computing environment to achieve good power consumption. This is very useful for the improvement of cloud system resource utilization.

**Keywords:** cloud computing, middleware, efficient power management, load balancing

Department Reference No E9006-PU-2-2554

# กิตติกรรมประกาศ

ในการทำโครงงานวิศวกรรมคอมพิวเตอร์ครั้งนี้ ข้าพเจ้าได้รับความช่วยเหลือและคำแนะนำมากมายจากหลายๆ ท่าน ข้าพเจ้าขอขอบพระคุณ ผศ. ดร. ภุชงค์ อุทโยภาศ ที่เปิดโอกาสให้ข้าพเจ้าได้ใช้ทรัพยากรของห้องปฏิบัติการวิจัยคอมพิวเตอร์สมรรถนะสูงและเครือข่ายมหาวิทยาลัยเกษตรศาสตร์ รวมถึงคำแนะนำต่างๆที่มอบให้

นอกจากนี้ยังขอขอบคุณพ่อแม่ที่ให้กำลังใจ รวมถึงเพื่อนๆ พี่ๆ ในห้องปฏิบัติการวิจัยที่ค่อยให้คำแนะนำสำหรับแนวทางในการดำเนินโครงการนี้ ในแง่มุมต่างๆ

ประยุกต์ เจตสิกทัต

ผู้จัดทำ

# สารบัญ

# สารบัญภาพ

# สารบัญตาราง

# คำอธิบายสัญลักษณ์และคำย่อ

| | |
|---|---|
| VM | ย่อมาจาก Virtual Machine (เครื่องคอมพิวเตอร์แบบเสมือน) |
| OS | ย่อมาจาก Operating System (ระบบปฏิบัติการ) |
| API | ย่อมาจาก Application Programming Interface (ส่วนต่อประสานโปรแกรมประยุกต์) |
| DHCP | ย่อมาจาก Dynamic Host Configuration Protocol เป็นโปรโตคอลที่ใช้ในการกำหนดไอพีแอดเดรสอัตโนมัติแก่เครื่องลูกข่ายบนระบบ |
| PKI | ย่อมาจาก Public Key Infrastructure เป็นระบบที่ใช้สร้างช่องทางการสื่อสารที่ปลอดภัยด้วยการใช้กุญแจคู่ (Key Pair) |
| HTTP | ย่อมาจาก Hypertext Transfer Protocol (เกณฑ์วิธีขนส่งข้อความหลายมิติ) |
| XML | ย่อมาจาก Extensible Markup Language เป็นภาษามาร์กอัปสำหรับการใช้งานทั่วไป |
| Guest | คือ คอมพิวเตอร์แบบเสมือนที่อยู่บนระบบประมวลผลแบบกลุ่มเมฆ |

# 1 บทนำ

ในปัจจุบัน บริษัท หรือผู้ให้บริการประเภทต่างๆ บนอินเทอร์เน็ต เริ่มมีการเปลี่ยนแปลงลักษณะโครงสร้างพื้นฐานของระบบการให้บริการมาใช้ระบบการประมวลผลแบบกลุ่มเมฆมากขึ้นเรื่อยๆ ด้วยเหตุผลด้านความยืดหยุ่น ความสะดวกในการจัดการ ความเสถียรของระบบ หรือแม้กระทั่งความประหยัด ซึ่งทำให้เกิดซอฟต์แวร์สำหรับสร้างระบบการประมวลผลแบบกลุ่มเมฆขึ้นมาจำนวนหนึ่ง ทั้งซอฟต์แวร์โอเพนซอร์สที่มีความสามารถจำกัด และซอฟต์แวร์เพื่อการค้าที่มีราคาค่อนข้างสูง แต่ถึงอย่างไรก็ตามผู้ให้บริการระบบการประมวลผลแบบกลุ่มเมฆ ก็ยังมีความต้องการระบบโอเพนซอร์ส ที่มีความสามารถเทียบเท่า หรือเกือบเทียบเท่ากับระบบที่สร้างมาเพื่อการค้า เช่น ระบบที่มีความสะดวกในการติดตั้งและใช้งาน, ระบบที่ง่ายต่อการบำรุงรักษา, ระบบที่ประหยัดพลังงาน, หรือ ระบบที่บริหารทรัพยากรให้แบบอัติโนมัติ เป็นต้น ซึ่งความสามารถหลายๆ ประการที่กล่าวมานั้น ล้วนแล้วมีความเกี่ยวข้องกับส่วนประกอบที่สำคัญส่วนหนึ่งของระบบการประมวลผลแบบกลุ่มเมฆ ที่โครงการนี้กำลังได้สร้างขึ้นมา ซึ่งก็คือ “มิดเดิลแวร์” โดยความสามารถในการเคลื่อนย้ายคอมพิวเตอร์เสมือนแบบอัตโนมัติ ของมิดเดิลแวร์ จะไปสนับสนุนให้เกิดความสามารถอื่นๆ ที่ช่วยเพิ่มความสะดวกแก่ผู้ให้บริการระบบการประมวลผลแบบกลุ่มเมฆได้อีกมาก

และแม้ว่าแนวโน้มของโลกธุรกิจ และการวิจัย จะเคลื่อนไปทางระบบการประมวลผลแบบกลุ่มเมฆ แต่สำหรับประเทศไทย ยังคงขาดบุคลากรที่มีความรู้ความสามารถเกี่ยวกับระบบการประมวลผลแบบกลุ่มเมฆอีกเป็นจำนวนมาก โครงการนี้ ร่วมกับโครงการ “การพัฒนาดิสทริบิวชั่นสำหรับระบบการประมวลผลแบบกลุ่มเมฆ” โดยนายณัฐ ศรชำนิ และโครงการ “การพัฒนาซอฟต์แวร์สำหรับบริหารจัดการระบบการประมวลผลแบบกลุ่มเมฆ” โดยนายยุทธกร ยุทธกรกิจ จึงได้ร่วมกันสร้าง “เมฆินทร์” ชุดซอฟต์แวร์โอเพนซอร์สสำหรับระบบการประมวลผลแบบกลุ่มเมฆ ขึ้นมาให้ครบทุกส่วนประกอบ เพื่อให้เกิดความรู้ ความเข้าใจ ในเทคโนโลยีการประมวลผลแบบกลุ่มเมฆ และส่งต่อความรู้ให้ผู้ที่มีความสนใจสามารถนำไปศึกษา และพัฒนาต่อยอดได้ ซึ่งเป็นหนึ่งในแรงผลักดันให้เกิดการวิจัย และเกิดการใช้งานระบบการประมวลผลแบบกลุ่มเมฆ มากขึ้นในอนาคต โดยเฉพาะในประเทศไทย

## 1.1. วัตถุประสงค์ของโครงงาน

มิดเดิลแวร์สำหรับโครงการนี้ถูกสร้างขึ้น เพื่อให้เกิดชุดซอฟต์แวร์สำเร็จรูปสำหรับระบบการประมวลผลแบบกลุ่มเมฆ ที่ชื่อว่าเมฆินทร์ โดยวัตถุประสงค์ของการสร้างเมฆินทร์ คือความพยายามที่จะทำให้มีซอฟต์แวร์โอเพนซอร์สสำหรับระบบการประมวลผลแบบกลุ่มเมฆ ที่อนุญาตให้องค์กรหรือธุรกิจ สามารถนำไปใช้ได้โดยไม่เสียค่าใช้จ่าย

## 1.2. ขอบเขตของโครงงาน

โครงงานนี้ จำกัดขอบเขตที่การสร้างมิดเดิลแวร์ของระบบการประมวลผลแบบกลุ่มเมฆเท่านั้น โดยจะต้อง

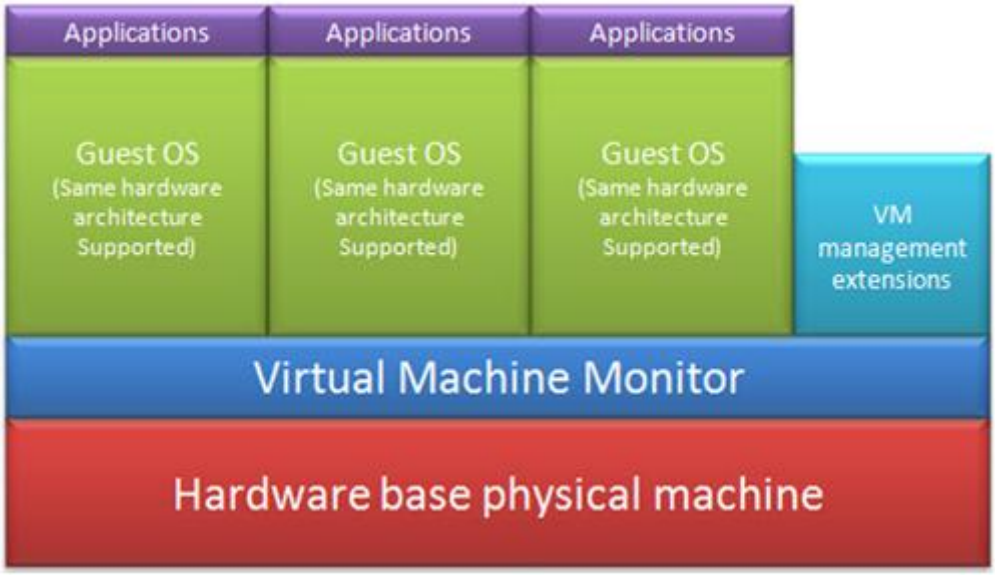
1) สามารถประกอบเข้ากับโครงการ “การพัฒนาดิสทริบิวชั่นสำหรับระบบการประมวลผลแบบกลุ่มเมฆ” โดยนายณัฐ ศรชำนิ และโครงการ “การพัฒนาซอฟต์แวร์สำหรับบริหารจัดการระบบการประมวลผลแบบกลุ่มเมฆ” โดยนายยุทธกร ยุทธกรกิจ เพื่อให้กลายเป็นระบบการประมวลผลแบบกลุ่มเมฆที่สมบูรณ์
2) สามารถเคลื่อนย้ายคอมพิวเตอร์เสมือนได้โดยอัตโนมัติ เพื่อความสะดวกในการใช้งาน และเพื่อการประหยัดพลังงานของระบบ

# 2 ทฤษฎีที่เกี่ยวข้อง

## 2.1. Virtualization Technology

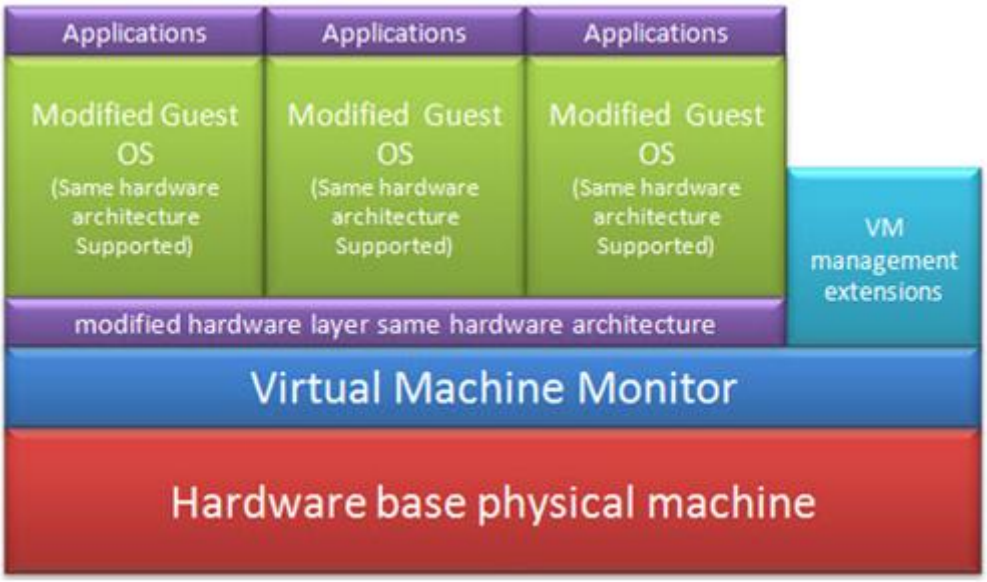
Virtualization Technology เป็นเทคโนโลยีที่ใช้ในการจำลองเครื่องคอมพิวเตอร์ หรือระบบปฏิบัติการ โดยมีซอฟต์แวร์ที่ใช้ในการจำลองฮาร์ดแวร์ และควบคุมเครื่องคอมพิวเตอร์เสมือน เรียกว่า hypervisor ซึ่งการจำลองนี้สามารถแบ่งออกได้เป็น 3 ประเภท คือ

1) Full Virtualization [12] เป็นการจำลองทุกๆ อย่างของฮาร์ดแวร์ เช่น จำลองชุดคำสั่ง, การอ่านหรือเขียน IO, การเข้าถึงหน่วยความจำ เป็นต้น ซึ่งเดิมนั้น การทำ full virtualization บนสถาปัตยกรรม x86 นั้น ต้องใช้ซอฟต์แวร์ที่ซับซ้อนมาก จนกระทั่งบริษัท Intel และ AMD ออกชุดคำสั่ง Intel VT-x และ AMD-V เพื่อรองรับการทำ virtualization จึงทำให้การทำ Full virtualization ง่าย และมีประสิทธิภาพมากขึ้น ด้วยลักษณะที่เรียกว่า hardware-assisted virtualization ตัวอย่างซอฟต์แวร์ที่ทำ full virtualization เช่น VirtualBox, VMware Workstation, Parallels Desktop for Mac หรือ Kernel-based Virtual Machine (KVM)



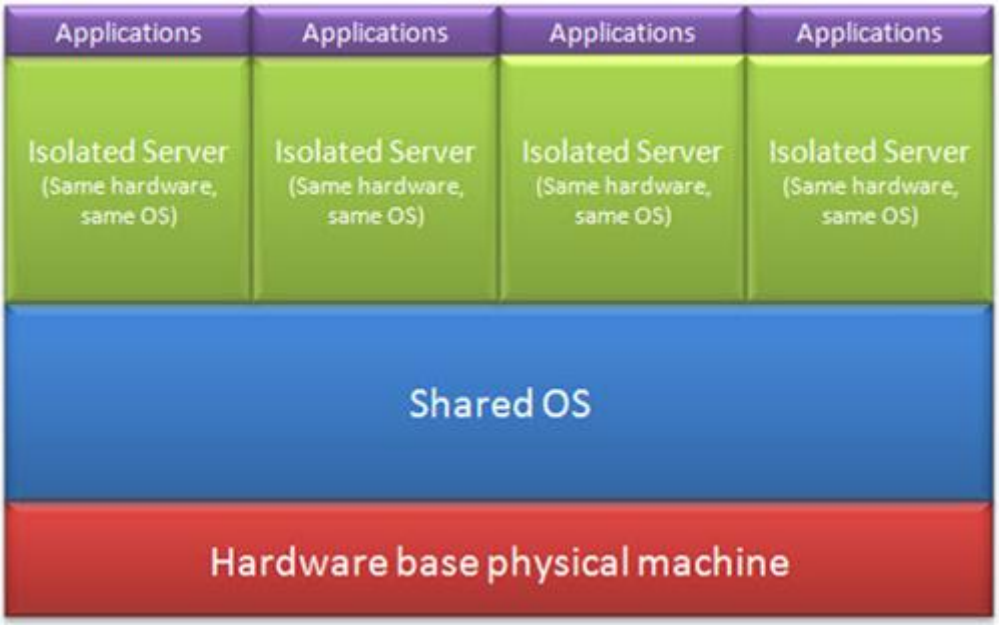

รูปที่ 1 โครงสร้างของ Full Virtualization

2) Paravirtualization [12] เป็นการจำลองที่จะมีอินเตอร์เฟสในระบบปฏิบัติการของเครื่องโฮสต์ (Host) เพื่อช่วยลดเวลาการประมวลผลของเครื่องคอมพิวเตอร์แบบเสมือน (Virtual Machine) ซึ่งจำเป็นต้องมีการแก้ไขระบบปฏิบัติการของเครื่องโฮสต์ เพื่อทำการสร้างอินเตอร์เฟสสำหรับการเข้าถึงฮาร์ดแวร์ของเครื่องคอมพิวเตอร์แบบเสมือน ตัวอย่างซอฟต์แวร์ที่เป็นแบบ Paravirtualization เช่น Xen




รูปที่ 2 โครงสร้างของ Paravirtualization

3) Operating Virtualization [12] เป็นวิธีที่ลินุกซ์เคอร์เนล (linux kernel) จัดสรรพื้นที่ของผู้ใช้แยกออกจากกัน จึงสามารถประมวลผลหลายๆ คอมพิวเตอร์เสมือนได้ในเครื่องเดียว ซึ่งมีข้อจำกัดหนึ่งคือ เครื่องคอมพิวเตอร์เสมือนทุกๆ เครื่องในระบบจะต้องมีระบบปฏิบัติการและฮาร์ดแวร์ที่เหมือนกัน ตัวอย่างซอฟต์แวร์ที่เป็น operating virtualization เช่น FreeBSD jails, OpenVZ และ Linux-VServer เป็นต้น




รูปที่ 3 โครงสร้างของ Operating Virtualization

## 2.2. ระบบการประมวลผลแบบกลุ่มเมฆ (Cloud Computing)

ด้วยความต้องการด้านสมรรถนะในการประมวลผล ที่คอมพิวเตอร์เพียงเครื่องเดียวไม่สามารถให้ได้อย่างเพียงพอ แนวคิดในการสร้างระบบการประมวลผลแบบขนานในลักษณะที่ใช้หลายเครื่องคอมพิวเตอร์ (Multicomputer) จึงเกิดขึ้น โดยในช่วงแรกเป็นการนำคอมพิวเตอร์ (Physical Machine) จำนวนหนึ่งมาเชื่อมต่อ

กันด้วยระบบ Interconnection network เพื่อให้สื่อสารกันได้ ซึ่งเรียกว่าระบบ Cluster แต่ด้วยเหตุที่วิธีการนี้ยังขาดความยืดหยุ่นในการปรับขนาด (Scalability) ของระบบให้มีทรัพยากรตามความต้องการ จึงได้เกิดเทคโนโลยีการจำลองเครื่องคอมพิวเตอร์แบบเสมือนขึ้น (Virtualization Technology) ซึ่งทำให้สามารถสร้างคอมพิวเตอร์แบบเสมือน (Virtual Machine) หลายๆ เครื่อง บนเครื่องคอมพิวเตอร์เครื่องหนึ่งได้ ซึ่งช่วยให้สามารถจัดสรรทรัพยากรที่มีอยู่ในระบบ Cluster ได้อย่างยืดหยุ่นมากยิ่งขึ้น เมื่อนำเทคนิคนี้มาผนวกเข้ากับแนวคิดในการให้บริการ จึงได้เกิดเป็นแนวคิดของระบบการประมวลผลแบบกลุ่มเมฆ ซึ่งมีคุณลักษณะสำคัญอยู่ 5 ประการ คือ

1) On-demand self-service หมายถึง ลูกค้าผู้ใช้บริการระบบสามารถที่จะจัดสรรทรัพยากรจากระบบให้ตนเองได้ตามความต้องการ โดยไม่จำเป็นต้องมีการติดต่อกับตัวบุคคลที่เป็นผู้ให้บริการ
2) Broad network access หมายถึง ทรัพยากรที่เปิดให้บริการ สามารถถูกเข้าถึงได้ผ่านทางระบบเครือข่ายที่มี และกลไกขั้นพื้นฐานรูปแบบต่างๆ
3) Resource pooling หมายถึง ทรัพยากรคอมพิวเตอร์ของผู้ให้บริการถูกมองว่ามีอยู่รวมกัน เพื่อให้บริการลูกค้า ด้วยลักษณะที่มีผู้เช่าหลายราย (multi-tenant model) ซึ่งจะให้บริการแตกต่างกันไปตามความต้องการของลูกค้า โดยที่ลูกค้าไม่สามารถรู้ หรือควบคุมได้ว่าในความเป็นจริง จะให้ทรัพยากรที่ใช้อยู่นั้นอยู่บนเครื่องไหนบ้าง แต่อาจจะสามารถควบคุมได้ในระดับที่สูงขึ้นไป เช่น อยู่ในประเทศใด หรือ ศูนย์ข้อมูลแห่งไหน เป็นต้น
4) Rapid elasticity หมายถึง ทรัพยากรที่ใช้ มีความยืดหยุ่นในการจัดหามาเพิ่ม หรือปล่อยทิ้งไป ซึ่งช่วยให้ระบบของลูกค้าสามารถปรับเปลี่ยนขนาดได้ตลอดเวลา
5) Measured service หมายถึง ระบบสามารถที่จะวัดการให้บริการ จากการใช้ทรัพยากรในระบบ รวมถึงรายงานผล ได้ด้วยตัวเอง ซึ่งจะนำไปสู่ความสะดวกในการคิดค่าบริการจากลูกค้าต่อไป

เมื่อพิจารณาตามลักษณะการให้บริการ ระบบการประมวลผลแบบกลุ่มเมฆสามารถแบ่งได้ 3 รูปแบบ คือ

1) Infrastructure as a Service (IaaS) คือ การให้บริการทรัพยากรคอมพิวเตอร์แก่ลูกค้า ในรูปของ หน่วยประมวลผล, พื้นที่เก็บข้อมูล, ระบบเครือข่าย หรือทรัพยากรพื้นฐานอื่นๆ ตัวอย่างที่มีในปัจจุบัน เช่น AmazonEC2, Amazon S3, Go Grid, Nirvanix, Linode, SunGrid, Flexiscale
2) Platform as a Service (PaaS) คือ การให้บริการที่ลูกค้าสามารถพัฒนา Application หรือ Service ของตนไว้บนระบบ Cloud โดยใช้ภาษาโปรแกรม หรือเครื่องมือที่ได้รับการสนุนจากผู้ให้บริการ ตัวอย่างเช่น Google App Engine, Microsoft Azure, Intel MashMaker, Yahoo Pipes, IBM Mashup Hub, GWT, GME, Force.com เป็นต้น
3) Software as a Service (SaaS) คือ การให้บริการด้วยโปรแกรมประยุกต์ ซึ่งทำงานอยู่บนระบบการประมวลผลแบบกลุ่มเมฆ ให้แก่ลูกค้า ตัวอย่างเช่น Google Apps(Gmail, Doc, Calendar), Hotmail, Picasa, Youtube, facebook, twitter เป็นต้น

เมื่อพิจารณาระบบการประมวลผลแบบกลุ่มเมฆตามรูปแบบการติดตั้ง (Deployment Model) จะสามารถแยกได้ 4 ประเภท ได้แก่

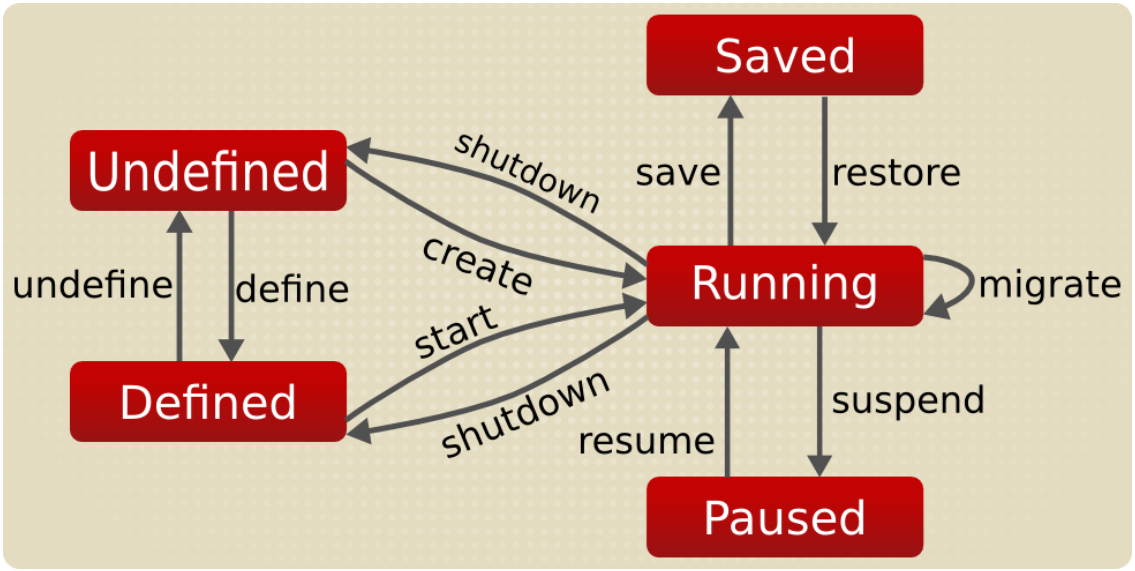
1) Private Cloud คือ ระบบ Cloud ที่หน่วยงานหนึ่งสร้างขึ้นมาเพื่อใช้ภายในองค์กรนั้นเอง ตัวอย่างระบบที่มีให้เห็นในปัจจุบัน เช่น Open Nebula, Eucalyptus, OpenStack เป็นต้น
2) Public Cloud คือ ระบบ Cloud ที่ผู้ให้บริการเปิดให้บุคคลทั่วไปมาใช้บริการได้ ตัวอย่างเช่น Amazon EC2, Google App Engine, Azure เป็นต้น

3) Community Cloud คือ ระบบ Cloud ที่ถูกแบ่งปันโดยกลุ่มบริษัทจำนวนหนึ่ง เพื่อสนับสนุนกลุ่มชุมชนที่มีความเกี่ยวข้องกับบริษัท
4) Hybrid Cloud คือ ระบบ Cloud ที่เกิดจากการผสมผสานกันระหว่างสอง Cloud ขึ้นไป (Public, Community หรือ Private Cloud) ซึ่งสามารถผสานรวมกันได้ ด้วยการมีมาตรฐานร่วมกัน หรือจากการมีเทคโนโลยีที่ทำให้สามารถเคลื่อนย้ายข้อมูลหรือโปรแกรมประยุกต์ระหว่างกันได้

## 2.3. วงจรชีวิตของคอมพิวเตอร์เสมือน (Virtual Machine Life Cycle)

ในการจัดการวงจรชีวิตของคอมพิวเตอร์เสมือน [13] คอมพิวเตอร์เสมือนหนึ่งเครื่องจะมีสถานะได้ 5 สถานะดังต่อไปนี้

1) Undefined เป็นสถานะพื้นฐานที่สื่อว่าคอมพิวเตอร์เสมือนนี้ยังไม่มีอยู่ในระบบ
2) Defined คือ คอมพิวเตอร์เสมือนนี้ได้ถูกสร้างขึ้นแล้ว แต่ยังไม่ถูกเปิดใช้งาน
3) Running คือ คอมพิวเตอร์เสมือนนี้ กำลังถูกจำลอง และใช้งานอยู่บน hypervisor
4) Pause คือ คอมพิวเตอร์เสมือนนี้กำลังถูกหยุดการทำงาน แต่ก็ยังคงอยู่บน hypervisor ซึ่งสามารถกลับไปทำงานต่อในสถานะ running ได้ด้วยการ resume
5) Saved คล้ายกับสถานะ Pause แต่สถานะทั้งหมดของเครื่องเสมือนนี้ จะถูกเก็บไว้ในหน่วยความจำถาวร (Persistent storage)




รูปที่ 4 วงจรการเปลี่ยนแปลงสถานะสถานะของคอมพิวเตอร์แบบเสมือน

## 2.4. การเคลื่อนย้ายคอมพิวเตอร์เสมือน (Virtual Machine Migration)

การย้ายคอมพิวเตอร์เสมือนข้ามเครื่องคอมพิวเตอร์จริง มีอยู่ 2 รูปแบบ คือ

1) Offline Migration คือการเคลื่อนย้ายคอมพิวเตอร์เสมือนที่จะต้องหยุดการทำงานของคอมพิวเตอร์เสมือนนั้นก่อนที่จะทำการเคลื่อนย้าย ซึ่งการหยุดการทำงานนี้อาจยาวนานได้ในหลักนาที
2) Live Migration คือการเคลื่อนย้ายคอมพิวเตอร์เสมือนที่สามารถช่วงลดเวลาการหยุดทำงานขณะเคลื่อนย้ายให้เหลือเพียงระดับมิลลิวินาที แต่ระบบนี้ต้องการ shared storage ถึงจะสามารถใช้งานได้

2.5. **ภาระการประมวลผล**

การคำนวณภาระการประมวลผลบนเครื่องคอมพิวเตอร์ หรือคอมพิวเตอร์เสมือน สามารถคำนวณได้ดังนี้

ให้เซตของเครื่องคอมพิวเตอร์ในระบบ คือ $S = \{H_1, H_2, H_3, \dots, H_{|S|}\}$ และให้ $T_H(t_i)$ เป็นค่า CPU Time สะสมของเครื่องคอมพิวเตอร์ $H$ ณ เวลา $t_i$ และ $C_H$ เป็นจำนวนหน่วยประมวลผล (Processor Core) ของเครื่องคอมพิวเตอร์ $H$ จะสามารถคำนวณภาระการประมวลผลโดยเฉลี่ย ในช่วงเวลาระหว่างเวลา $t_1$ กับ $t_2$ ได้เป็น

$$\overline{W_H}(t_1, t_2) = \frac{T_H(t_2) - T_H(t_1)}{(t_2 - t_1) \times C_H} \quad (1)$$

ให้ $T_G(t_i)$ เป็นค่า CPU Time สะสมที่เครื่องคอมพิวเตอร์เสมือนใช้ไปทั้งหมดจนถึงเวลา $t_i$ และ $Host(G)$ เป็นเครื่องคอมพิวเตอร์ที่กำลังให้ที่อยู่กับเครื่องคอมพิวเตอร์เสมือน $G$ จะสามารถคำนวณภาระการประมวลผลเฉลี่ยของเครื่องคอมพิวเตอร์เสมือน $G$ ระหว่างเวลา $t_1$ กับ $t_2$ ได้เป็น

$$\overline{W_G}(t_1, t_2) = \frac{T_G(t_2) - T_G(t_1)}{(t_2 - t_1) \times C_{Host(G)}} \quad (2)$$

และสามารถคำนวณภาระงานเฉลี่ยสำหรับทั้งระบบระหว่างเวลา $t_1$ กับ $t_2$ ได้เป็น

$$\overline{W_S}(t_1, t_2) = \frac{\sum_{i=1}^{|S|} \overline{W_{H_i}}(t_1, t_2)}{|S|} \quad (3)$$

2.6. **การใช้พลังงานของระบบการประมวลผลแบบกลุ่มเมฆ**

การใช้พลังงานในระบบการประมวลผลแบบกลุ่มเมฆ ย่อมขึ้นกับพลังงานที่คอมพิวเตอร์แต่ละเครื่องใช้เป็นหลัก และการใช้พลังงานของเครื่องคอมพิวเตอร์แต่ละเครื่องก็ขึ้นกับภาระการประมวลผลของเครื่องนั้นเป็นหลัก ซึ่งเนื้อหาในส่วนนี้จะอธิบายแนวคิดในการประหยัดพลังงานในระบบการประมวลผลแบบกลุ่มเมฆ ด้วยแบบจำลองทางคณิตศาสตร์ ดังต่อไปนี้

เพื่อความสะดวกในการสร้างแบบจำลองทางคณิตศาสตร์ จะสมมติให้เครื่องคอมพิวเตอร์ทุกเครื่องในระบบนั้นมีคุณลักษณะเหมือนกัน และสมมติว่าระบบปฏิบัติการให้ภาระการประมวลผลที่ต่ำมาก เมื่อเทียบกับภาระการประมวลผลจากเครื่องคอมพิวเตอร์แบบเสมือน ให้เซตของภาระการประมวลผลของเครื่องคอมพิวเตอร์เสมือนในระบบคือ $D = \{D_1, D_2, D_3, \dots, D_{|D|}\}$ เราจะประมาณผลรวมของค่าภาระการประมวลผลของระบบได้เป็น

$$W_S \cong \sum_{i=1}^{|D|} D_i \quad (4)$$

ซึ่งจะมีค่าเท่ากับผลรวมของค่าภาระการประมวลผลจากเครื่องคอมพิวเตอร์แต่ละเครื่อง คือ

$$W_S = \sum_{i=1}^{|S|} W_{H_i} \quad (5)$$

ให้ $B_0$ เป็นอัตราการใช้พลังสำหรับเครื่องคอมพิวเตอร์ที่ปิดอยู่ ให้ $B_1$ แทนอัตราการใช้พลังงานขั้นต่ำสุดสำหรับเครื่องคอมพิวเตอร์ที่เปิดอยู่ และ $K$ เป็นค่าสัมประสิทธิ์พลังงาน แบบจำลองอย่างง่ายสำหรับการหาค่าพลังงานที่ใช้ไปจากภาระการประมวลผล บนเครื่องคอมพิวเตอร์ $H$ สามารถหาได้ ดังนี้

$$P_H = \begin{cases} B_0 & , H\ is\ shut\ off \\ B_1 + KW_H & , H\ is\ booted \end{cases} \tag{6}$$

สมมติว่าในระบบมีเครื่องคอมพิวเตอร์อยู่ $m$ เครื่อง และมีแค่ $n$ เครื่องที่เปิดใช้งานอยู่ จะสามารถคำนวนอัตราการใช้พลังงานของระบบนี้ในขณะที่กำลังรับภาระการคำนวณรวม $(W_S)$ ได้เป็น

$$\begin{aligned} P_S(n, m, W_S) &= (m-n)B_0 + \textstyle\sum_{i=0}^{n}\left(B_1 + KW_{H_i}\right) \\ &= mB_0 - nB_0 + nB_1 + K\textstyle\sum_{i=0}^{n} W_{H_i} \\ &= mB_0 + n(B_1 - B_0) + KW_s \end{aligned} \tag{7}$$

ดังนั้น เราสามารถคำนวณอัตราการประหยัดพลังงาน เปรียบเทียบกับการเปิดเครื่องคอมพิวเตอร์ทั้งระบบ ได้เป็น

$$\begin{aligned} R_{Save}(n, m, W_S) &= \frac{P_S(m,m,W_S) - P_S(n,m,W_S)}{P_S(m,m,W_S)} \\ &= \frac{(m-n)(B_1 - B_0)}{mB_1 + KW_S} \end{aligned} \tag{8}$$

จากสมการ (8) จะเห็นว่า อัตราการประหยัดพลังงาน ขึ้นกับจำนวนของเครื่องคอมพิวเตอร์ที่ได้รับการปิด $(m-n)$ เป็นอย่างมาก จึงนำมาซึ่งแนวคิดในการวางแผนจัดการทรัพยากรด้วยการปิดเครื่องคอมพิวเตอร์ที่ไม่ได้ใช้งานในระบบ

# 3 เครื่องมือที่ใช้ในการทำโครงงาน

## 3.1. ฮาร์ดแวร์

### 3.1.1. เครื่องคอมพิวเตอร์

เครื่องคอมพิวเตอร์จำนวน 5 เครื่อง มีคุณสมบัติ ดังนี้

1) รุ่น Dell Optiplex 760
2) หน่วยประมวลผล Intel® Core™ 2 Duo E8400 3.00 กิกะเฮิรตซ์
3) ความเร็วบัส 1.333 กิกะเฮิรตซ์
4) หน่วยความจำหลัก 2 กิกะไบต์
5) การ์ดเชื่อมต่อเครือข่ายแบบอีเทอร์เน็ต ความเร็ว 1 กิกะบิตต่อวินาที
6) รองรับการทำ Wake on LAN และ Virtualization (Intel VT-x)

### 3.1.2. สวิตช์

เชื่อมต่อคอมพิวเตอร์ด้วยสวิตช์ แบบอีเทอร์เน็ต รุ่น Linksys SD2008 ความเร็ว 1 กิกะบิตต่อวินาที

## 3.2. ซอฟต์แวร์

### 3.2.1. ระบบปฏิบัติการ CentOS 6.2

เป็นระบบปฏิบัติการแบบโอเพนซอร์ส ที่ได้ปรับแต่งมาจาก Red Hat Enterprise Linux (RHEL) ซึ่งระบบมีความเหมาะสมที่จะนำมาใช้ในการเปิดให้บริการ (Server)

### 3.2.2. Kernel-based Virtual Machine (KVM)

เป็น Hypervisor ที่ใช้การจำลองแบบ full virtualization และรองรับการทำงานแบบ hardware-assisted virtualization ได้ในกรณีที่ฮาร์ดแวร์มีชุดคำสั่ง Intel VT-x หรือ AMD-V โดย KVM มีความสามารถในการทำ Live Migration ในตัวอยู่แล้ว

### 3.2.3. libvirt API

เป็น Virtualization Application Programming Interface ที่ใช้ในการติดต่อกับ Hypervisor ซึ่งสามารถติดต่อได้ทั้ง Xen, QEMU, KVM, LXC, OpenVZ, Virtual Box, VMware ESX, GSX และ VMware Workstation and Player

### 3.2.4. ภาษาโปรแกรมไพธอน (Python)

ภาษาโปรแกรมขั้นสูง ที่ใช้พัฒนาส่วนประกอบทั้งหมดในโครงการนี้

### 3.2.5. ระบบฐานข้อมูล MySQL Database

### 3.2.6. GNU Transport Layer Security Library (Gnu TLS)

เป็นเครื่องมือที่ใช้ในการสร้าง Public Key Infrastructure ในระบบ บน Transport Layer

### 3.2.7. CherryPy

เป็น Python Web Framework ที่ใช้ในการสร้าง Cloud Middleware API ที่เป็นช่องทางติดต่อกับเครื่องมือที่ใช้จัดการระบบ ข้อดีของ CherryPy คือ มีความสามารถในการทำงานแบบ Multithreading

# 4 วิธีการดำเนินโครงงาน

ในบทนี้ จะกล่าวถึงแนวคิดการสร้างระบบในภาพรวม และโครงสร้างโดยละเอียดของระบบว่าประกอบด้วยส่วนประกอบอะไรบ้าง แต่ละส่วนมีหน้าที่อย่างไร จากนั้น จะอธิบายถึงขั้นตอนวิธีที่ใช้ในการจัดการทรัพยากรเพื่อการประหยัดพลังงาน และจบด้วยวิธีการในการวัดผลการประหยัดพลังงานของระบบ

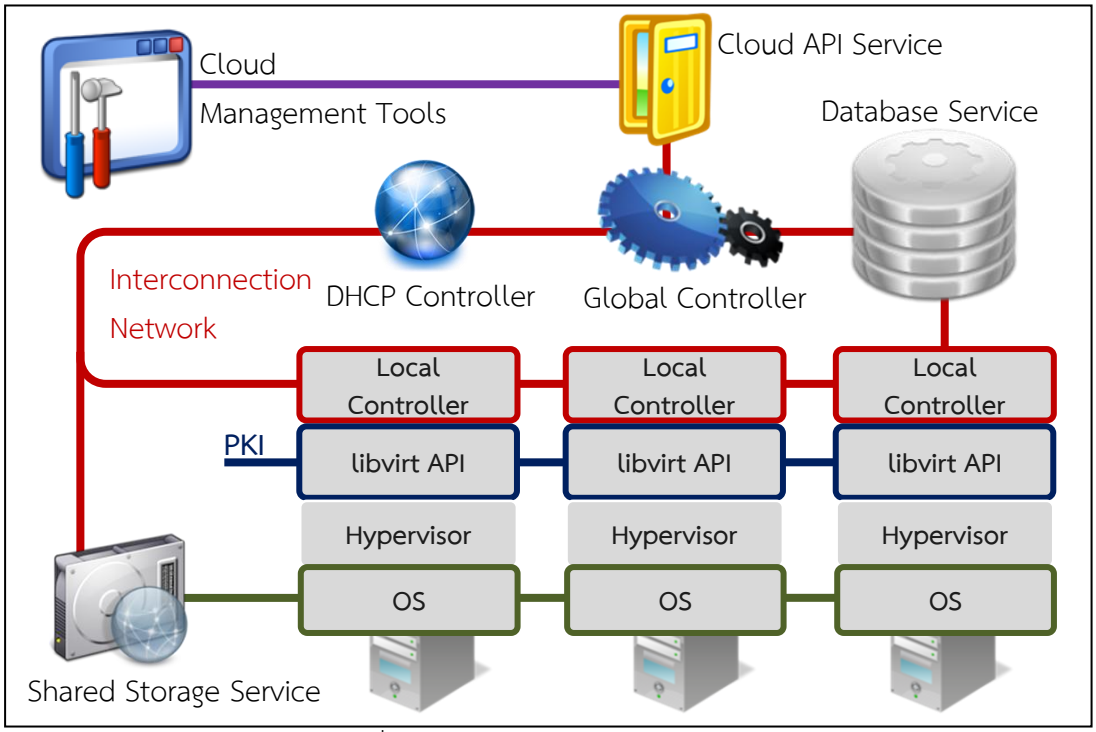
## 4.1. แนวคิด

สำหรับระบบกลุ่มเมฆที่สร้างขึ้นมาใช้ในองค์กร ทรัพยากรคอมพิวเตอร์ที่มีทั้งหมดในระบบนั้นจำเป็นต้องจัดหามาไว้ให้มากกว่าความต้องการทรัพยากรสูงสุดขององค์กร เพื่อให้เพียงพอต่อความต้องการอยู่เสมอ เมื่อเป็นเช่นนี้ ระบบจะต้องสูญเสียพลังงานจากทรัพยากรที่ไม่ได้ถูกใช้งานอยู่ตลอดเวลา โครงการนี้จึงพยายามเสนอแนวทางที่จะลดระดับการใช้พลังงานของระบบ จากทรัพยากรส่วนที่ไม่ได้ถูกใช้งาน ซึ่งแนวทางที่จะสามารถทำได้ก็คือ การโยกย้ายคอมพิวเตอร์แบบเสมือนจนเกิดเครื่องที่ว่างในระบบ แล้วทำการปิดเครื่องที่ว่างนั้นเพื่อลดการใช้พลังงาน และเมื่อระบบมีความต้องการจะใช้ทรัพยากรมากขึ้น ก็จำเป็นที่จะต้องเปิดเครื่องนั้นอีกครั้งโดยอัตโนมัติเพื่อช่วยแบ่งเบาภาระของระบบ

จากแนวคิดข้างต้นนี้ ไม่ใช่แค่เรื่องของพลังงานที่จะต้องคำนึงถึง แต่จะต้องคำนึงถึงผลกระทบที่จะมีต่อผู้ใช้งานด้วยจากระบบที่สร้างขึ้น ซึ่งเครื่องมือที่สำคัญที่จะต้องใช้ก็คือ Live Migration ซึ่งจะไม่ขัดจังหวะการทำงานของการบริการ ในขณะที่กำลังย้ายเครื่องคอมพิวเตอร์เสมือน อีกทั้งยังต้องออกแบบให้การโยกย้ายทำได้อย่างเหมาะสมและไม่ทำให้ภาระงานอัดแน่นที่เครื่องใดเครื่องหนึ่งมากจนเกินไป ซึ่งจะกระทบต่อประสิทธิภาพการใช้งานได้

## 4.2. โครงสร้างระบบ

มิดเดิลแวร์ของระบบเมฆินทร์ จะมองว่าเครื่องคอมพิวเตอร์ทุกเครื่องในระบบนั้น มีลักษณะเหมือนกันทั้งหมดและหน้าที่ต่างๆ ในระบบ จะถูกกระจายไปอยู่ที่แต่ละเครื่อง โดยส่วนประกอบในระบบ สามารถอธิบายได้ ดังนี้




รูปที่ 5 โครงสร้างระบบมิดเดิลแวร์โดยรวม

### 4.2.1. Cloud API Service

ส่วนประกอบนี้มีหน้าที่ในการให้บริการให้เครื่องมือสำหรับการจัดการ (Cloud Management Tool) ใดๆ ก็ตาม สามารถที่จะติดต่อ และควบคุมระบบประมวลผลแบบกลุ่มเมฆทั้งระบบได้ ซึ่ง API นี้จะให้บริการด้วยลักษณะของ Representational State Transfer (REST) Web service ซึ่งจะใช้ Hypertext Transfer Protocol (HTTP) และ Extensible Markup Language (XML) ในการสื่อสาร Cloud API Service ถูกสร้างด้วย CherryPy ซึ่งเป็นกรอบการพัฒนาเว็บ (Web framework) ที่สามารถรับการเชื่อมต่อได้แบบขนานและส่วนประกอบนี้จะถูกบังคับให้ทำงานอยู่บนเครื่องคอมพิวเตอร์ที่รับหน้าที่เป็น Global Controller โดยอัตโนมัติ

เครื่องมือสำหรับการจัดการ จะติดต่อมายัง Cloud API ด้วย HTTP Request ซึ่งการติดต่อนี้สามารถแบ่งแยกได้เป็นสองประเภท คือ

1) การร้องขอให้ทำงานที่สามารถทำเสร็จได้ในระยะเวลาอันสั้น งานประเภทนี้จะถูกทำและตอบกลับในทันที ซึ่งเป็นการทำงานแบบซิงโครนัส (Synchronous) ตัวอย่างของงานประเภทนี้ เช่น การถามสถานะระบบ หรือ การเปลี่ยนการตั้งค่าของระบบบางอย่าง
2) การร้องขอให้ทำงานที่อาจต้องใช้เวลานาน งานประเภทนี้จะถูกนำไปเข้าคิว (Queue) ในฐานข้อมูลของระบบ เพื่อรอที่จะนำออกมาทำงานต่อไป ซึ่งเป็นการทำงานแบบอะซิงโครนัส (Asynchronous) ผู้ร้องขอจะได้รับหมายเลขสำหรับระบุงานกลับไป เพื่อใช้ในการสอบถามระบบว่างานถูกทำไปถึงขั้นไหนแล้ว ตัวอย่างของงานประเภทนี้ เช่น การสร้างคอมพิวเตอร์เสมือน การเปิดเครื่องคอมพิวเตอร์เสมือน, การปิดเครื่องคอมพิวเตอร์ในระบบ เป็นต้น

### 4.2.2. การสื่อสารระหว่างส่วนประกอบต่างๆ ในระบบ

การสื่อสารระหว่างส่วนประกอบต่างๆ ในระบบ จะใช้การเชื่อมต่อแบบ Socket โดยทุกๆ เครื่องคอมพิวเตอร์ในระบบ จะทำการเปิด Socket server ไว้ เพื่อให้เครื่องคอมพิวเตอร์เครื่องอื่นๆ สามารถติดต่อเข้าไปได้ และเพื่อให้เครื่องเครื่องหนึ่งสามารถรองรับการสื่อสารได้จากหลายๆ เครื่องพร้อมกัน จึงได้สร้างให้ Socket server นี้ทำงานแบบขนาน

โดยที่ระบบการสื่อสารผ่าน Socket ทั้งหมดนั้น จะทำในลักษณะ Remote Procedure Call (RPC) คือ ในการสื่อสารหนึ่งครั้ง ก็จะมีการส่งชื่อคำสั่งที่ต้องการให้ทำ พร้อมกับค่าตัวแปร ไปยังปลายทาง เมื่อปลายทางได้รับ ก็จะทำตามชุดคำสั่งที่มีเตรียมไว้อยู่แล้วในเครื่องจนเสร็จ และทำการคืนค่าผลลัพธ์กลับไป เป็นอันเสร็จการสื่อสารหนึ่งรอบ

### 4.2.3. บริการฐานข้อมูล (Database Service)

มิดเดิลแวร์ ใช้ฐานข้อมูลที่แยกจากส่วนประกอบอื่นๆ ของระบบ โดยฐานข้อมูลนี้จะเก็บข้อมูลไว้ 3 ส่วนหลักๆ คือ ข้อมูลทั่วไปของระบบ (ตัวแปรสำหรับปรับค่าระบบ, ไอพีแอดเดรสในระบบ, ตัวแบบของเครื่องคอมพิวเตอร์แบบเสมือน, เครื่องคอมพิวเตอร์จริงในระบบ, เครื่องคอมพิวเตอร์เสมือนในระบบ) คิวงาน และบันทึกภาระการประมวลผลจากระบบดูสถานะ

ทุกๆ ส่วนประกอบในมิดเดิลแวร์ จะสามารถเข้าใช้งานฐานข้อมูลนี้ได้ผ่านทางไลบรารี (Library) ภาษาไพธอนของ MySQL Database โดยฐานข้อมูลนี้จะนำไปไว้บนเครื่องคอมพิวเตอร์เครื่องไหนในระบบก็ได้

### 4.2.4. Shared Storage Service

บริการนี้ใช้ความสามารถของ Network File System (NFS) ที่มีในระบบปฏิบัติการ ในการสร้างระบบเก็บข้อมูล ที่ทำให้ทุกๆ เครื่องในระบบสามารถเข้าถึงข้อมูลชุดเดียวกันได้ เราได้นำระบบนี้ มาใช้เก็บข้อมูลสองอย่าง คือ ไฟล์ตัวแบบของเครื่องคอมพิวเตอร์แบบเสมือน (Virtual Machine Template) และ ไฟล์อิมเมจของเครื่อง

คอมพิวเตอร์แบบเสมือน (Virtual Machine Image) โดยเหตุผลหลักที่ต้องนำระบบนี้มาใช้ก็คือ เพื่อให้ระบบมีความสามารถในการทำ Live Migration

#### 4.2.5. DHCP Controller

ส่วนประกอบนี้มีหน้าที่ควบคุม การแจกจ่ายไอพีแอดเดรสให้กับเครื่องคอมพิวเตอร์ทุกเครื่องในระบบ ทั้งเครื่องคอมพิวเตอร์จริง และเครื่องคอมพิวเตอร์เสมือน โดยระบบจะทำการผูกแมคแอดเดรส (MAC Address) ของเครื่องแต่ละเครื่อง ไว้กับ ไอพีแอดเดรสที่เครื่องนั้นจะได้รับ และเมื่อมีการปลดเครื่องบางเครื่องออกจากระบบ การผูกนี้ก็ต้องนำออกไปจากระบบด้วย ซึ่งส่วนประกอบนี้จะถูกเปิดใช้งานอยู่บนเครื่องเครื่องเดียวกับ Global Controller โดยอัตโนมัติ

#### 4.2.6. Hypervisor

บนเครื่องคอมพิวเตอร์ทุกเครื่อง จะมี Kernel-based Virtual Machine (KVM) ติดตั้งอยู่เป็น hypervisor ซึ่งมีหน้าที่ในการจำลองเครื่องคอมพิวเตอร์เสมือนในระบบทั้งหมด โดยการจำลองเครื่องคอมพิวเตอร์เสมือนขึ้นมาหนึ่งเครื่อง ก็ต้องการการเข้าถึงไฟล์อิมเมจของเครื่องคอมพิวเตอร์แบบเสมือนนั้น ผ่านทาง Shared Storage Service ด้วย และระบบเครือข่ายภายในเครื่องคอมพิวเตอร์เครื่องนั้น จะต้องถูกติดตั้งให้ใช้ระบบบริดจ์ (Bridge) เพื่อให้ทั้งเครื่องคอมพิวเตอร์จริง และเสมือน สามารถที่จะใช้อินเตอร์เฟสเพื่อเชื่อมต่อเครือข่ายร่วมกันได้

#### 4.2.7. libvirt API

API นี้จะถูกใช้งานอยู่บนคอมพิวเตอร์ทุกเครื่องเช่นเดียวกับ Hypervisor ถูกใช้เป็นช่องทางการติดต่อไปยัง Hypervisor เพื่อควบคุมคอมพิวเตอร์เสมือนบนเครื่องคอมพิวเตอร์จริงนั้นๆ ตามวงจรชีวิตของมัน โดยที่ libvirt API บนเครื่องทุกเครื่อง จะถูกเชื่อมต่อกันอยู่ด้วย Public Key Infrastructure (PKI) เพื่อให้สามารถทำการย้ายเครื่องคอมพิวเตอร์เสมือนข้ามเครื่องกันได้สะดวก ผ่านทาง API ตัวนี้ ซึ่งจากการใช้ Public Key Infrastructure นี้ จึงทำให้ต้องมีเครื่องคอมพิวเตอร์เครื่องหนึ่ง ทำหน้าที่เป็น Certificate Authority (CA) ด้วย

#### 4.2.8. Local Controller

ส่วนประกอบนี้ จะถูกติดตั้งไว้เหมือนๆ กันในเครื่องคอมพิวเตอร์ทุกเครื่อง ซึ่งมี 2 หน้าที่หลักๆ คือ

1) เปิด Socket server เพื่อรอรับคำสั่งจากส่วนประกอบอื่นๆ ในระบบ ในลักษณะ Remote Procedure Call และทำตามคำสั่งเหล่านั้น
2) คอยเก็บค่า CPU Time ของเครื่องคอมพิวเตอร์แบบเสมือนที่อยู่บนเครื่องนี้ทั้งหมด รวมถึงเครื่องจริงด้วย ลงในฐานข้อมูล โดยข้อมูลเหล่านี้ จะถูกใช้สำหรับการวางแผนการเคลื่อนย้ายคอมพิวเตอร์เสมือนในภายหลัง

#### 4.2.9. ระบบดูสถานะของเครื่องคอมพิวเตอร์ (Host Monitoring System)

เป็นระบบที่จะคอยสื่อสารกันอยู่ภายใต้ระบบทั้งหมด และเก็บค่าสถานะปัจจุบันเอาไว้ เพื่อให้ส่วนประกอบอื่นๆ สามารถถามค่าสถานะปัจจุบัน ของเครื่องใดๆ ในระบบก็ได้ ซึ่งระบบนี้ไม่ได้อยู่ในขอบเขตของโครงการนี้ แต่ถูกพัฒนาโดยอีกโครงการหนึ่งที่อยู่ภายใต้ระบบเมฆินทร์เช่นกัน

#### 4.2.10. Global Controller

ส่วนประกอบนี้มีหน้าที่หลักคือการนำงานที่อยู่ในคิวออกมาทำ รวมถึงทำการวางแผนในการโยกย้ายคอมพิวเตอร์เสมือน และจัดการการใช้ทรัพยากรในระบบ โดยใช้ข้อมูลจาก CPU Time ย้อนหลังที่ได้มาจาก Local Controller ของเครื่องคอมพิวเตอร์แต่ละเครื่อง

4.3. **การย้ายคอมพิวเตอร์เสมือนแบบอัตโนมัติ**

**4.3.1. นิยามศัพท์**

เพื่อให้สามารถอธิบายขั้นตอนวิธีได้ง่ายขึ้น เราจึงได้นิยามคำต่อไปนี้ขึ้นมา ได้แก่

**โฮสต์ (Host)** คือ เครื่องคอมพิวเตอร์จริงที่เปิดใช้งานอยู่ และได้รับอนุญาตให้สามารถมีคอมพิวเตอร์เสมือนทำงานอยู่บนเครื่องนี้ได้

"**Puller-threshold**" แทนด้วยสัญลักษณ์ $\nabla$ เป็นค่าที่ใช้แบ่งแยกประเภทของโฮสต์ โดยถ้าภาระงานโดยเฉลี่ยของโฮสต์ น้อยกว่าค่าค่านี้ โฮสต์นี้จะถูกจัดประเภทเป็น "**Puller**" ความหมายคือ เป็นโฮสต์ที่มีความสามารถในการดึงภาระงานจากโฮสต์อื่น มาใส่ตัวเองได้

"**Pusher-threshold**" แทนด้วยสัญลักษณ์ $\Delta$ ค่านี้จะต้องมากกว่า "Puller-threshold" เป็นค่าที่ใช้แบ่งแยกประเภทของโฮสต์เช่นกัน โดยถ้าภาระงานโดยเฉลี่ยของโฮสต์ มากกว่าค่าค่านี้ โฮสต์นี้จะถูกจัดประเภทเป็น "**Pusher**" ความหมายคือ เป็นโฮสต์ที่ต้องการจะผลักภาระงานของตนไปสู่โฮสต์อื่นในระบบ

"**Puller-force**" เป็นความสามารถในเชิงปริมาณของโฮสต์เครื่องหนึ่ง ในการดึงภาระงานจากโฮสต์อื่น โดยค่า "Puller-force" ของโฮสต์ $H_i$ นิยามโดย

$$Pull_{H_i}(t_1, t_2) = \Delta - \overline{W_{H_i}}(t_1, t_2) \quad (5)$$

"**Pusher-force**" เป็นความต้องการในเชิงปริมาณของโฮสต์เครื่องหนึ่ง ในการผลักภาระงานไปยังโฮสต์อื่น โดยค่า "Pusher-force" ของโฮสต์ $H_i$ นิยามโดย

$$Push_{H_i}(t_1, t_2) = \overline{W_{H_i}}(t_1, t_2) - \Delta \quad (5)$$

"**Total-puller-force**" คือผลรวมของ "Puller-force" จากทุกๆ "Puller" ในระบบ

"**Total-pusher-force**" คือผลรวมของ "Pusher-force" จากทุกๆ "Pusher" ในระบบ

"**Scale-up-threshold**" คือค่าที่ใช้เปรียบเทียบกับขนาดของภาระการประมวลผลโดยเฉลี่ยของโฮสต์ทุกเครื่องในระบบ (System workload) ซึ่งถ้าค่านี้มากกว่า Scale-up-threshold ระบบก็จะทำการขยายความสามารถโดยการเปิดเครื่องคอมพิวเตอร์ที่ปิดอยู่มาช่วยแบ่งเบาภาระงานเป็นจำนวน 1 เครื่อง

"**Scale-down-threshold**" คือค่าที่ใช้เปรียบเทียบกับขนาดของภาระการประมวลผลโดยเฉลี่ยของโฮสต์ทุกเครื่องในระบบ (System workload) ซึ่งถ้าค่านี้น้อยกว่า Scale-down-threshold สามารถตีความได้ว่าตอนนี้ระบบกำลังเปิดโฮสต์มากเกินจำเป็น และสามารถที่จะปิดโฮสต์ได้ 1 เครื่อง

**4.3.2. ขั้นตอนวิธี**

ขั้นตอนวิธีทั้งหมด แบ่งเป็นขั้นตอนย่อยๆ 3 ส่วน ที่จะทำงานต่างเวลากัน โดยที่ส่วนแรกจะถูกเรียกใช้เมื่อมีการร้องขอให้เปิดเครื่องคอมพิวเตอร์เสมือนเครื่องใหม่ในระบบ ส่วนที่สองและสามจะถูกเรียกใช้งานสลับกันไปมาทุกๆ 5 นาที รายละเอียดของขั้นตอนต่างๆ เป็นดังนี้

1) เมื่อมีการเปิดใช้งานเครื่องคอมพิวเตอร์แบบเสมือนเครื่องใหม่ ระบบจะคำนวณค่าภาระงานเฉลี่ยของโฮสต์แต่ละโฮสต์ในช่วงเวลาสามนาทีหลังสุด แล้วเลือกโฮสต์ที่มีภาระงานต่ำที่สุดในการรับเครื่องคอมพิวเตอร์เสมือนเครื่องใหม่ไปเปิด
2) การปรับเปลี่ยนขนาดของระบบ (Scaling Phase) เริ่มจากการคำนวณค่าเฉลี่ยของภาระงานของระบบ จากโฮสต์ทุกตัวในช่วง 5 นาทีหลังสุด แล้วนำมาเปรียบเทียบ ว่า ถ้าค่านี้มากกว่า Scale-up-threshold ก็

จะต้องเปิดเครื่องที่เปิดอยู่ขึ้นมา 1 เครื่อง แต่ถ้าค่านี้น้อยกว่า Scale-down-threshold ก็จะเลือกโฮสต์ที่มีค่าภาระงานน้อยที่สุดในขณะนั้นมา 1 เครื่อง แล้วทำการย้ายคอมพิวเตอร์เสมือนจากเครื่องนั้น ออกไปสู่เครื่องโฮสต์อื่นๆ โดยสุ่มแบบถ่วงน้ำหนักตามปริมาณทรัพยากรที่เหลือในเครื่อง จากนั้นจึงทำการปิดเครื่องที่เลือก

3) การทำให้เกิดความสมดุล (Balancing Phase) ในส่วนนี้ ระบบจะพยายามโยกย้ายเครื่องคอมพิวเตอร์แบบเสมือนจาก Pusher ไปสู่ Puller โดยระบบจะพยายามลดภาระงานของ Pusher ที่มี Pusher force สูงที่สุดก่อน โดยจะประเมินและป้องกันไม่ให้ Puller เปลี่ยนไปเป็น Pusher เมื่อรับภาระจาก Pusher ซึ่งการโยกย้ายจริงๆ จะเกิดขึ้นหลังจากวางแผนเสร็จแล้วเท่านั้น

**Algorithm**

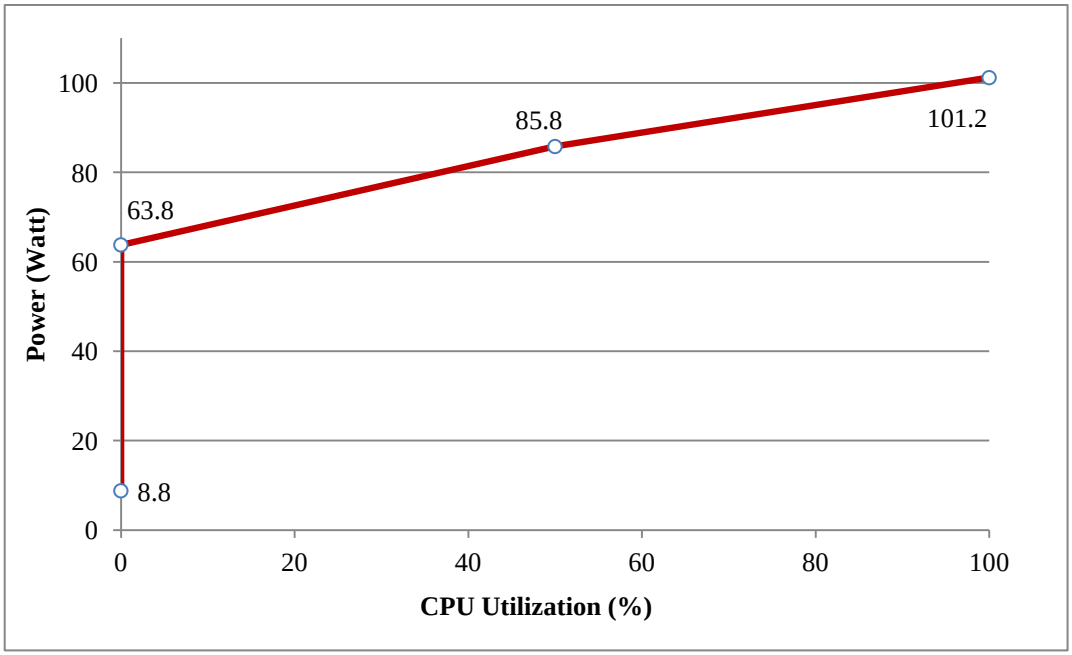
```
Start new VM:
   start VM on a host with least load.

Scaling phase:
   if system average workload < scale-down-threshold
      select a host with least load
      migrate VM out from this host
      shutdown this host
   end if
   if system average workload > scale-up-threshold or total-pusher-force > total-puller-force
      boot a new host
   end if

Balancing phase:
while there are pusher in system
   select puller with maximum puller-force
   sort all pushers by pusher-force (max to min)
   for each pusher
      sort VM in current pusher by VM load (max to min)
      for each VM in current pusher
         if load of VM < puller-force of selected puller
            add task "migrate current VM to current puller"
            minus current puller-force by load of VM
            minus current pusher-force by load of VM
            if current pusher-force < 0
               break
            end if
         end if
      end for
   end for
   if there is no new migration task added along this loop
      execute added tasks
      stop balancing phase
   end if
end while
```

รูปที่ 6 ขั้นตอนวิธีการจัดการทรัพยากรในระบบ

## 4.4. การทดสอบอัตราการประหยัดพลังงาน

### 4.4.1. การวัดอัตราการใช้พลังงาน

เนื่องจากการขาดเครื่องมือที่ใช้ในการวัดค่าและบันทึกการใช้พลังงานของเครื่องคอมพิวเตอร์อย่างต่อเนื่อง เราจึงต้องทำการวัดอัตราการใช้พลังงาน จากเครื่องคอมพิวเตอร์เครื่องหนึ่งที่ได้ติดตั้งระบบเมฆินทร์ลงไป ขณะกำลังรับภาระงานที่ระดับต่ำที่สุดประมาณ 0% ประมาณ 50% และประมาณ 100% รวมถึงพลังงานที่ใช้ในขณะที่เครื่องถูกปิด แล้วนำค่าที่วัดได้มาสร้างฟังก์ชันในการวัดอัตราการใช้พลังงาน ซึ่งได้ผลออกมาคือ ในขณะปิดเครื่อง เครื่องจะใช้พลังงาน 8.8 วัตต์ เมื่อเปิดเครื่อง จะใช้พลังงานขั้นต่ำ 63.8 วัตต์ ขณะเครื่องรับภาระครึ่งหนึ่ง จะใช้พลังงาน 85.8 วัตต์ และเมื่อเครื่องรับภาระอย่างเต็มที่ จะใช้พลังงาน 101.2 วัตต์ สร้างออกมาเป็นฟังก์ชันได้ดังรูปที่ 8




รูปที่ 7 ฟังก์ชันการแปลงพลังงาน จากภาระการประมวลผล

### 4.4.2. การปรับแต่งระบบที่ใช้ทดสอบ

การทดสอบจะใช้ระบบเมฆินทร์ ที่มีเครื่องคอมพิวเตอร์ที่เหมือนกันจำนวน 5 เครื่อง โดยที่ระบบจะถูกปรับแต่งให้มีเครื่องคอมพิวเตอร์เครื่องหนึ่งไม่สามารถรองรับการจำลองคอมพิวเตอร์เสมือนได้ แต่คอมพิวเตอร์เครื่องนี้จะรับหน้าที่เป็นส่วนประกอบต่างๆ ของระบบทั้งหมด ได้แก่ Global Controller, DHCP Controller, Shared storage service, Cloud API service ดังนั้น จะมีเครื่องโฮสต์ในระบบทดสอบเพียงแค่ 4 เครื่องเท่านั้น

### 4.4.3. คอมพิวเตอร์เสมือนที่ใช้ทดสอบ

การทดสอบจะใช้เครื่องคอมพิวเตอร์แบบเสมือนแบบเดียวกันจำนวน 10 เครื่อง โดยที่แต่ละเครื่องจะได้รับหน่วยความจำหลัก 256 เมกะไบต์ และหน่วยประมวลผลจำลอง (VCPU) 2 หน่วย โดยหน่วยประมวลผลจำลองทั้งสองหน่วยนี้จะถูกผูกการใช้งานเข้ากับทั้งสองหน่วยประมวลผลของเครื่องคอมพิวเตอร์จริงด้วย เพื่อให้แน่ใจว่าภาระงานที่เครื่องคอมพิวเตอร์เสมือนนั้นได้รับ จะถูกแบ่งการทำงานไปสู่หน่วยประมวลผลทั้งสองอย่างเท่าเทียมกัน

เครื่องคอมพิวเตอร์เสมือนเหล่านี้จะถูกติดตั้งระบบปฏิบัติการ CentOS 6.2 Minimal และติดตั้งโปรแกรมที่ถูกสร้างขึ้นเฉพาะ ที่จะคอยรับการสั่งงานจากภายนอกมาทำ พร้อมทั้งบันทึกเวลาที่ใช้ในการทำงานลงบนเครื่องคอมพิวเตอร์เสมือนนั้น

### 4.4.4. สถานการณ์ทดสอบ

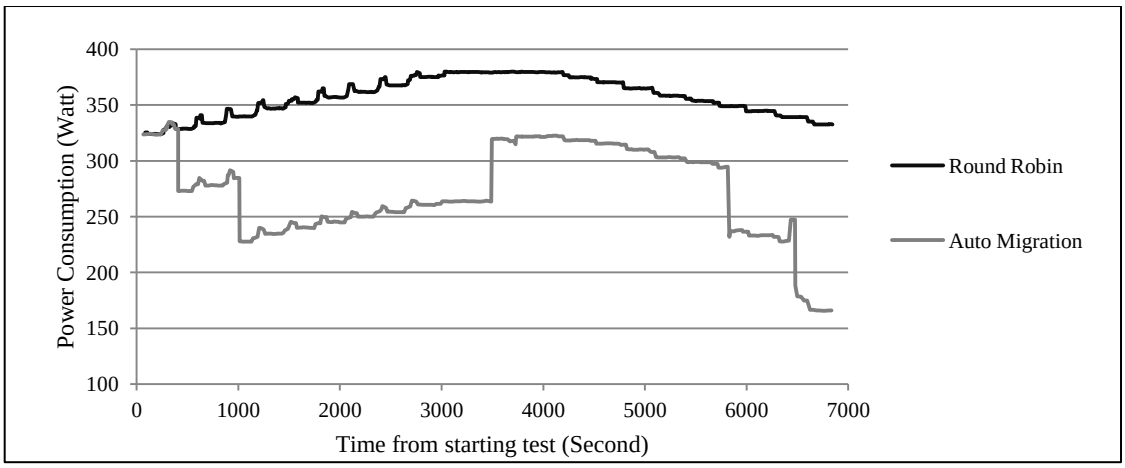
การทดสอบเริ่มต้นจากการที่คอมพิวเตอร์ทุกเครื่องเปิดอยู่ และจะมีคำสั่งให้เปิดเครื่องคอมพิวเตอร์เสมือนขึ้นมาทุกๆ 5 นาที จนครบทั้ง 10 เครื่อง โดยที่คอมพิวเตอร์เสมือนแต่ละเครื่องหลังจากเปิดเครื่องได้ 5 นาที ก็จะได้รับคำสั่งให้คำนวณค่าของ 10,000 แฟคทอเรียล และบันทึกเวลาที่ใช้คำนวณแต่ละครั้งเอาไว้ โดยแต่ละเครื่องเสมือน จะได้รับคำสั่งให้ทำงานด้วยความถี่ที่ต่างกันไปตามแต่หมายเลขเครื่อง ซึ่งจะมีความถี่ในการให้งานอยู่ในช่วงทุกๆ 2 ถึง 3 วินาที และเมื่อเครื่องเสมือนถูกเปิดจนครบ 70 นาที การจ่ายงานสู่เครื่องคอมพิวเตอร์เสมือนนั้นก็จะหยุดลง รวมเวลาการทดสอบระบบ รอบละประมาณ 2 ชั่วโมง โดยจะทำการทดสอบ 2 รอบ รอบที่หนึ่งเป็นการทดสอบโดยเปิดระบบเคลื่อนย้ายคอมพิวเตอร์เสมือนแบบอัตโนมัติ และรอบที่สองจะทดสอบโดยปิดระบบนี้ และแทนที่ด้วยการเลือกโฮสต์ที่จะใช้เปิดเครื่องคอมพิวเตอร์เสมือนเครื่องใหม่แบบวนรอบ (Round Robin) ซึ่งในรอบที่สองนี้ จะไม่มีการเคลื่อนย้ายคอมพิวเตอร์เสมือน หรือการปิดคอมพิวเตอร์จริงแต่อย่างใด

# 5 ผลการดำเนินโครงงานและวิจารณ์

## 5.1. ผลการทดสอบ

### 5.1.1. การใช้พลังงาน

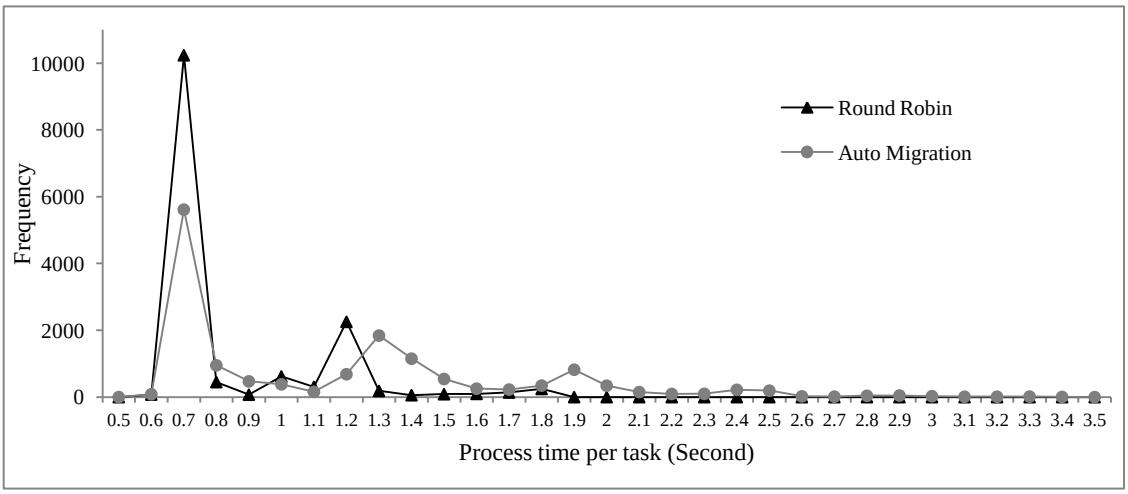
จากมุมมองด้านการใช้พลังงาน วิธีการที่นำเสนอสามารถประหยัดพลังงานในขณะที่ภาระงานในระบบมีไม่มากนัก จากรูปที่ 9 จะเห็นได้ว่า มี 2 โฮสต์ที่ถูกปิดไปหลังจากเริ่มการทดลองได้ไม่นาน ปล่อยให้อีก 2 โฮสต์ในระบบรับภาระจากคอมพิวเตอร์เสมือนในขณะนั้น แต่เมื่อภาระงานของระบบเริ่มเพิ่มมากขึ้น โฮสต์เครื่องหนึ่งจะถูกเปิดขึ้นมาเพื่อแบ่งเบาภาระงานของระบบ ซึ่งในการทดสอบนี้ขั้นตอนวิธีของเราสามารถลดการใช้พลังงานได้ถึง 23.74% โดยสามารถประมาณได้ว่าการปิดโฮสต์ 1 เครื่อง จะช่วยลดการใช้พลังงานในขณะนั้นได้ประมาณ 50 วัตต์




รูปที่ 8 เปรียบเทียบการใช้พลังงานระหว่างสองการทดลอง

### 5.1.2. เวลาที่ใช้ในการประมวลผล

จากตารางที่ 1 จะเห็นเวลาต่ำสุดที่ใช้ในการประมวลผล 10,000 แฟคทอเรียล อยู่ที่ค่าประมาณ 0.6 วินาที และจากรูปที่ 10 จะเห็นว่าช่วงเวลาที่ใช้ในการประมวลผลในกรณีที่เปิดใช้ระบบเคลื่อนย้ายคอมพิวเตอร์เสมือนแบบอัตโนมัตินั้น ขยายกว้างขึ้นออกไปทางขวาเล็กน้อย แสดงถึงการใช้เวลาประมวลผลโดยเฉลี่ยที่เพิ่มขึ้น

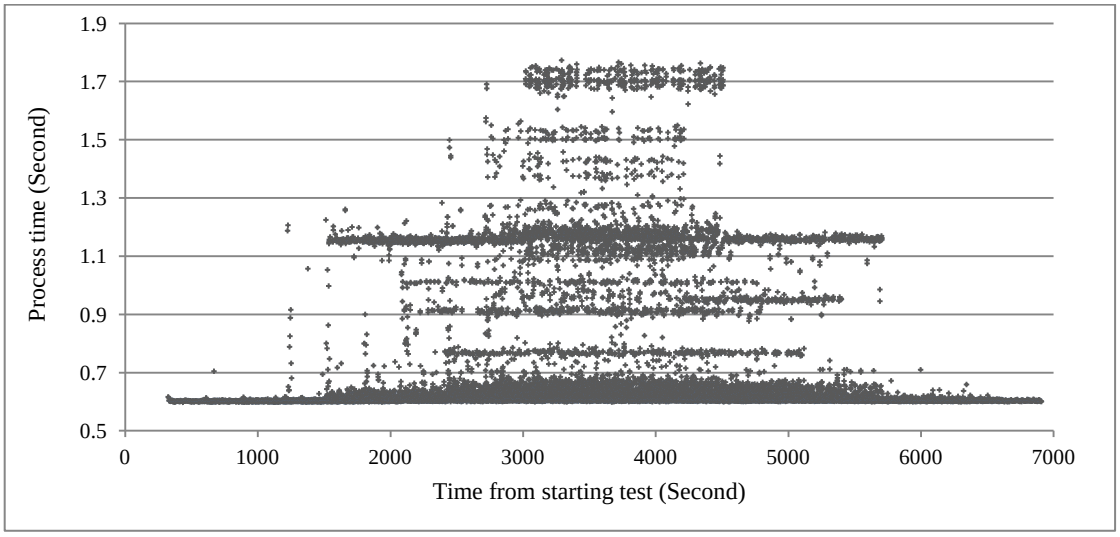



รูปที่ 9 เปรียบเทียบการใช้เวลาในการประมวลผลระหว่างสองการทดลอง

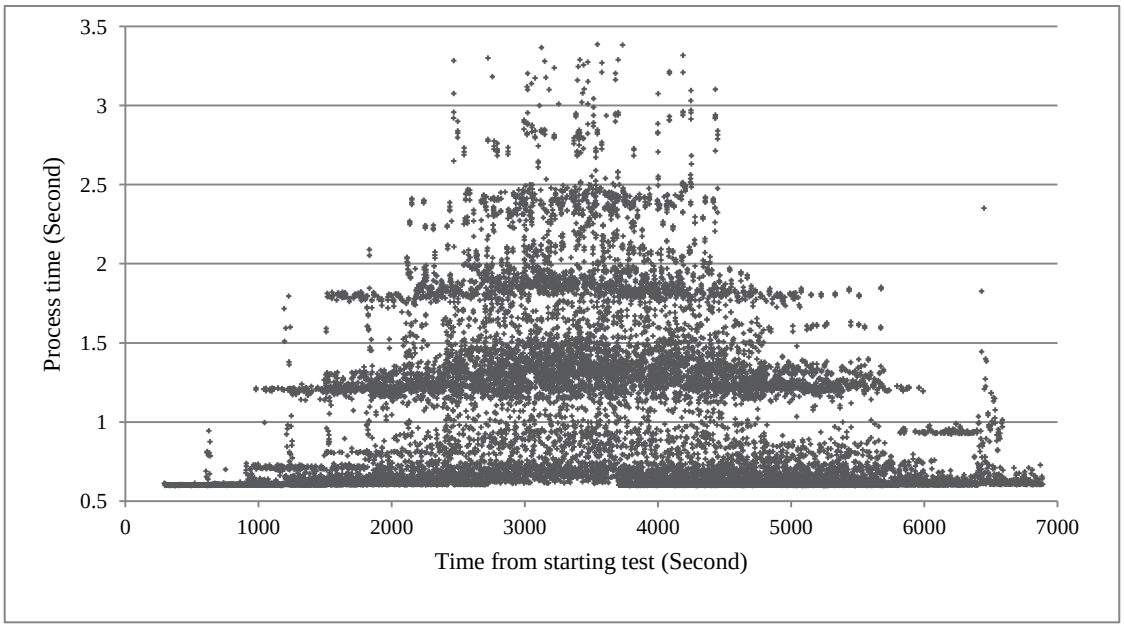
ตารางที่ 1 เวลาสูงสุด ต่ำสุด และโดยเฉลี่ยที่ใช้ในการคำนวน

| | Round Robin | Auto Migration |
|---|---|---|
| Minimum Time (Second) | 0.598 | 0.598 |
| Maximum Time (Second) | 1.773 | 3.503 |
| Average Time (Second) | 0.775 | 1.108 |

ในการทดลองนี้เวลาเฉลี่ยในการคำนวณ เพิ่มขึ้น 43% เมื่อเปิดระบบการเคลื่อนย้ายคอมพิวเตอร์เสมือนแบบอัตโนมัติ ซึ่งเมื่อพิจารณารูปที่ 11 และ 12 จะสามารถเห็นการกระจายตัวของเวลาที่ใช้ในการประมวลผล ณ เวลาใดๆ ของการทดลองได้ จากทั้งสองการทดลอง



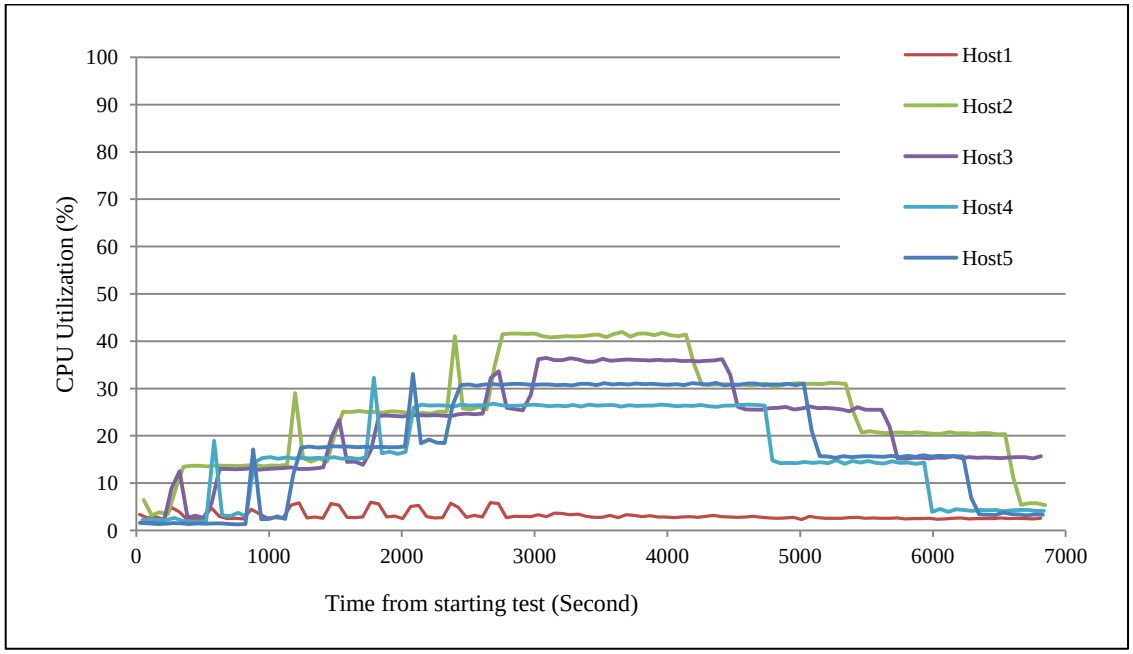

รูปที่ 10 เวลาที่ใช้ในการประมวลผลเมื่อปิดระบบอัตโนมัติ ตลอดการทดลอง



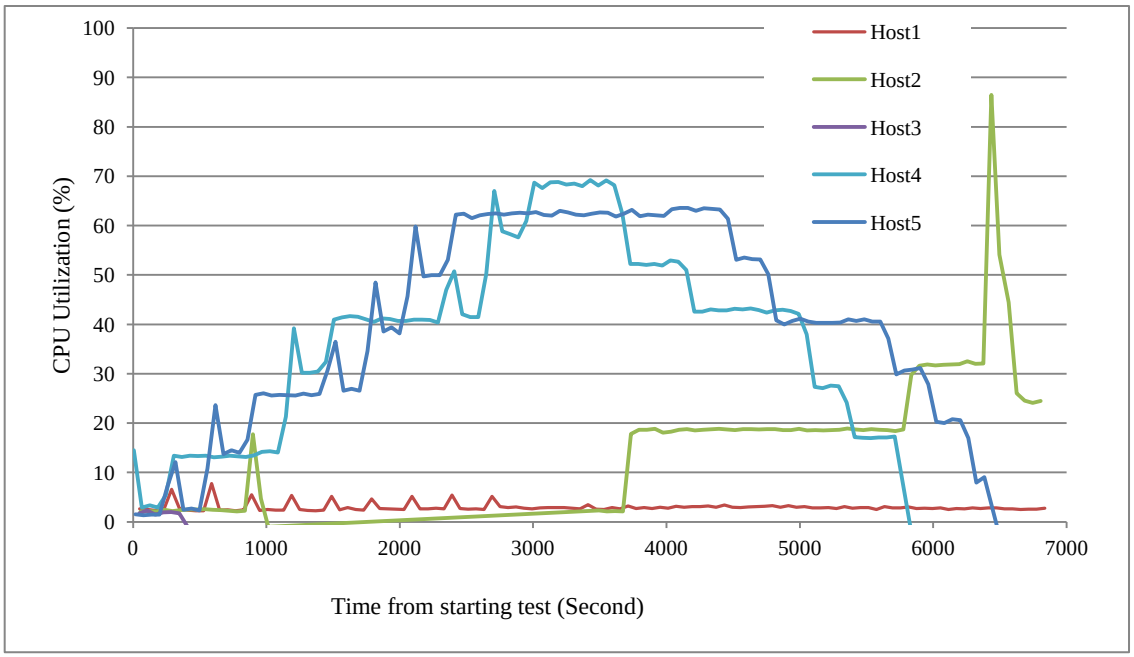

รูปที่ 11 เวลาที่ใช้ในการประมวลผลเมื่อเปิดระบบอัตโนมัติ ตลอดการทดลอง

### 5.1.3. ภาระการประมวลผล

จากรูปที่ 13 และ 14 ภาระการประมวลผลของเครื่องคอมพิวเตอร์เครื่องที่ 1 (เส้นสีแดง) จะไม่ถูกสนใจ เนื่องจากเป็นเครื่องที่ถูกควบคุมไม่ให้รับเครื่องคอมพิวเตอร์แบบเสมือน เมื่อพิจารณาโฮสต์ที่เหลือ จะเห็นว่าในกรณีที่ปิดระบบอัตโนมัติ จะเห็นการใช้งานโฮสต์ทุกเครื่องในระดับที่ใกล้เคียงกันอยู่ที่ประมาณ 25% ถึง 40% แต่ในกรณีที่เปิดระบบอัตโนมัติ จะเห็นว่าโฮสต์ที่เปิดอยู่จะถูกใช้งานได้อย่างเต็มที่มากกว่า แต่ไม่ถึงระดับที่ถูกใช้งานมากจนต้องแย่งทรัพยากรกันระหว่างเครื่องคอมพิวเตอร์แบบเสมือน ซึ่งเป็นผลจากขั้นตอนวิธีที่ได้เสนอขึ้น




รูปที่ 12 ภาระการประมวลผลเมื่อปิดระบบอัตโนมัติ ตลอดการทดลอง




รูปที่ 13 ภาระการประมวลผลเมื่อเปิดระบบอัตโนมัติ ตลอดการทดลอง

## 5.2. การวิจารณ์ผล

### 5.2.1. การเพิ่มขึ้นของเวลาที่ใช้ประมวลผล

จากการพิจารณาการกระจายตัวของเวลาที่ใช้ประมวลผล จากรูปที่ 11 และ 12 จะเห็นว่ามีการเกาะกลุ่มกันของเวลาที่ใช้ประมวลผลอย่างมีนัยสำคัญ ซึ่งการเกาะกลุ่มจะอยู่ที่เวลาประมาณ 0.6, 1.2, 1.8 และ 2.4 วินาที ซึ่งสัมพันธ์กับข้อมูลจากรูปที่ 10 โดยเลขเหล่านี้เป็นจำนวนเท่าของ 0.6 ซึ่งเป็นเวลาที่ใช้ในการประมวลผลต่ำสุด เมื่อรวมผลที่เห็นได้ชัดนี้เข้ากับข้อมูลการรับภาระงานจากรูปที่ 13 และ 14 จะสามารถอธิบายได้ว่าสาเหตุของการเพิ่มขึ้นของเวลาที่ใช้ในการประมวลผล ในสภาพที่ภาระการประมวลผลของโฮสต์ยังต่ำอยู่นั้น เกิดจากอะไร

โดยการเพ่งเล็งไปที่ช่วงเวลาระหว่างวินาทีที่ 3,000 ถึงวินาทีที่ 4,000 ซึ่งเป็นช่วงเวลาที่ระบบมีภาระงานสูงสุด เราสามารถประมาณอย่างง่ายๆ ได้ว่าจำนวนของคอมพิวเตอร์เสมือนบนโฮสต์เครื่องหนึ่งเป็นเท่าไหร่ โดยการหารจำนวนเครื่องเสมือนทั้งหมดด้วยเครื่องโฮสต์ที่เปิดอยู่ จะเห็นว่าในการทดลองที่ปิดระบบอัตโนมัติ โฮสต์เครื่องหนึ่งจะรับ 2 ถึง 3 เครื่องเสมือน แต่บนระบบอัตโนมัติที่มีโฮสต์เปิดอยู่เพียงสองเครื่อง ณ ขณะนั้น โฮสต์เครื่องหนึ่งต้องรับเครื่องเสมือนถึง 5 เครื่อง ซึ่งเมื่อเครื่องเสมือน *n* เครื่อง ที่อยู่บนโฮสต์เดียวกัน เริ่มทำงานแบบเดียวกัน ณ เวลาเดียวกัน CPU Time ของโฮสต์นั้น จะถูกแบ่งเท่าๆ กัน ระหว่างเครื่องเสมือน *n* เครื่องนั้น ผลก็คือ งานที่ถูกทำอยู่บนเครื่องเสมือนทั้ง *n* เครื่องนั้น ก็จะทำงานนานขึ้นเป็น *n* เท่า จากเวลาที่ใช้คำนวณปกติ

จากการอธิบายเช่น นี้จึงสามารถอธิบายได้ว่าทำไมเวลาที่ใช้ประมวลผลถึงเกาะกลุ่มอยู่ ณ ช่วงที่เป็นจำนวนเท่าของเวลาประมวลผลต่ำสุด รวมถึงว่า ทำไมในกรณีที่ไม่เปิดระบบอัตโนมัติถึงมีเวลาที่ใช้ประมวลผลสูงสุดอยู่ที่ประมาณ 1.8 วินาที และอีกกรณีหนึ่งที่ประมาณ 3 วินาที อย่างไรก็ตาม เราไม่สามารถที่จะป้องกันการเกิดการทำงานพร้อมๆ กันในสภาวะการใช้งานจริงได้ แต่สิ่งที่จะสามารถทำได้ก็คือ การลดความน่าจะเป็นที่จะเกิดการทำงานพร้อมกันให้ได้มากที่สุด

แต่นี่ก็ไม่ใช่ปัญหาใหญ่สำหรับคุณภาพการให้บริการ (Quality of Service) ตราบใดที่งานทุกงานยังสามารถทำเสร็จได้ในระยะเวลาอันสั้น แต่ในทางกลับกัน หากภาระการประมวลผลของโฮสต์เครื่องใดเครื่องหนึ่งสูงขึ้นจนถึง 100% เป็นเวลานาน คุณภาพการให้บริการจะถูกกระทบโดยตรงจากการเพิ่มขึ้นของเวลาที่ใช้ในการประมวลผล ไปสู่เวลาที่ไม่สามารถยอมรับได้ ดังนั้น สภาวะนี้จึงเป็นสภาวะที่ขั้นตอนวิธีควรที่จะคำนึงถึง และพยายามหลีกเลี่ยงไม่ให้เกิดขึ้น เพื่อรักษาคุณภาพในการให้บริการเอาไว้

### 5.2.2. ความเร็วในการปรับตัวของระบบ

จากการที่ระบบทำงานสลับกันทุกๆ 5 นาที ระหว่าง Scaling Phase กับ Balancing Phase ในสภาวะที่ภาระการประมวลผลมีการเปลี่ยนแปลงอยู่ตลอดเวลา เมื่อภาระการประมวลผลของโฮสต์เครื่องหนึ่งเพิ่มขึ้นอย่างฉับพลันจนทำให้เกิดความไม่สมดุลขึ้น จะต้องใช้การทำงานใน Balancing Phase เพื่อแก้ปัญหานี้ ดังนั้น ช่วงเวลาที่นานที่สุดที่จะต้องรอจนปัญหานี้ถูกแก้ไขคือ 10 นาที ส่วนในอีกกรณีหนึ่ง เมื่อระบบต้องการเปิดโฮสต์เครื่องใหม่มาช่วยแบ่งเบาภาระของระบบ การแก้ปัญหาจะต้องเริ่มด้วยการทำ Scaling Phase 1 รอบ เพื่อให้เครื่องใหม่ถูกเปิดขึ้นมา จากนั้นต้องทำการโยกย้ายภาระการประมวลผลด้วย Balancing Phase อีก 1 รอบ ดังนั้น ช่วงเวลาที่นานที่สุดที่ต้องรอเพื่อแก้ปัญหานี้คือ 15 นาที

ดังนั้น จึงสรุปได้ว่า ในสภาวะที่มีการเปลี่ยนแปลงภาระการประมวลผลของระบบไม่มากนัก ระบบของเราจะสามารถปรับตัวให้เข้ากับการเปลี่ยนแปลงได้ภายใน 15 นาที แต่ในสภาวะที่มีการเปลี่ยนแปลงภาระการประมวลผลของระบบอย่างมากและถี่ การปรับตัวใน Scaling Phase ครั้งละ 1 โฮสต์ ทุกๆ 10 นาที อาจจะไม่เร็วพอที่จะสามารถปรับตัวให้ทันกับการเปลี่ยนแปลงภาระงานของระบบได้

### 5.2.3. ข้อจำกัดของการควบคุมการใช้หน่วยประมวลผล

ในสภาวะของการทดลอง เราสามารถควบคุม CPU Affinity ได้ทั้งจากระดับ Hypervisor และระดับโปรแกรมประยุกต์ (Application) ดังนั้น ภาระการประมวลผลทั้งหมดบนโฮสต์จึงสามารถแบ่งการใช้งานได้อย่างเท่าเทียมระหว่างหน่วยประมวลผลทุกหน่วยบนโฮสต์นั้น แต่ในสภาวะของการใช้งานจริงของระบบที่ให้บริการโครงสร้างพื้นฐาน (Infrastructure as a Service) เราไม่สามารถควบคุมความสมดุลในการใช้งานหน่วยประมวลผลเสมือน (VCPU) บนเครื่องคอมพิวเตอร์แบบเสมือนแต่ละเครื่องได้ ความไม่แน่นอนนี้เป็นอุปสรรคอย่างมากต่อการจัดการทรัพยากรในระบบให้สมดุล ดังนั้น การควบคุม CPU Affinity ที่ฉลาดขึ้นในระดับ Hypervisor จะก่อให้เกิดการใช้งานหน่วยประมวลผลของโฮสต์ในระบบได้อย่างเต็มที่มากขึ้น ซึ่งจะช่วยพัฒนาขั้นตอนวิธีในการทำให้ระบบสมดุลที่มีอยู่ในปัจจุบัน ให้ดีขึ้นไปอีกได้โดยอัตโนมัติ แต่ประเด็นนี้อยู่นอกเหนือขอบเขตของโครงการนี้

# 6 สรุปผลการดำเนินงานและข้อเสนอแนะ

## 6.1. สรุปผล

โครงการนี้ได้สร้างมิดเดิลแวร์สำหรับระบบการประมวลผลแบบกลุ่มเมฆ “เมฆินทร์” ที่มีความสามารถเพียงพอต่อการใช้งานในระดับพื้นฐาน สามารถเชื่อมต่อเข้ากับเครื่องมือสำหรับการจัดการ และประกอบเข้ากับระบบปฏิบัติการได้เป็นอย่างดี และจากการทดสอบระบบเคลื่อนย้ายคอมพิวเตอร์เสมือนแบบอัตโนมัติ ในเชิงประสิทธิภาพ พบว่าสามารถประหยัดพลังงานได้ตามแนวคิดข้างต้นที่เสนอไว้ โดยไม่กระทบต่อการการให้บริการผู้ใช้งาน

## 6.2. ปัญหาและอุปสรรค

1) การเพิ่มตัวแบบใหม่ของคอมพิวเตอร์เสมือนเข้าในระบบไม่สามารถทำได้อย่างอิสระ แต่ต้องสร้างจากไฟล์อิมเมจที่ถูกปรับแต่งมาอย่างรัดกุม ถูกต้องตามคู่มือการใช้งานเท่านั้น ซึ่งทำให้ความสะดวกในการใช้งานในส่วนนี้ลดลง
2) การใช้ Network File System (NFS) สำหรับสร้าง Shared Storage Service อาจเป็นปัญหาต่อการขยายตัวของระบบ ไปสู่ขนาดที่มีเครื่องโฮสต์เป็นจำนวนมาก
3) การใช้งานหลายส่วนในระบบยังคงถูกควบคุมด้วยนโยบาย แทนที่จะเป็นตัวระบบเอง จึงเป็นการเพิ่มภาระในการบังคับใช้นโยบายให้กับผู้ดูแลระบบ เช่น นโยบายการตั้งรหัสผ่านของคอมพิวเตอร์แบบเสมือน หรือนโยบายในการบังคับเครื่องคอมพิวเตอร์เสมือนให้รับไอพีแอดเดรสจาก DHCP Server เท่านั้น เป็นต้น ซึ่งถ้าสามารถบังคับใช้ได้ด้วยตัวระบบเอง ก็น่าจะช่วยลดภาระของผู้ดูแลระบบได้

## 6.3. แนวทางการพัฒนาต่อ

1) พัฒนาความมั่นคงปลอดภัยให้กับระบบ ทั้งส่วนของ Cloud API Service และส่วนของการติดต่อกันระหว่างเครื่องด้วย Socket
2) พัฒนาความสามารถด้าน High Avalilability ให้ระบบสามารถทำงานต่อไปได้ แม้ว่าเครื่องบางเครื่องในระบบจะมีปัญหา
3) พัฒนาความสามารถในการเชื่อมต่อเมฆินทร์สองระบบเข้าด้วยกัน และทำงานร่วมกันในลักษณะ Intercloud
4) พัฒนาระบบ Cloud Storage เพื่อที่จะเสริมการให้บริการด้านพื้นที่เก็บข้อมูล ให้กับระบบบริการทรัพยากรการประมวลผลที่มี

# 7 บรรณานุกรม


[1] P. T. Endo, G. Goncalves, and J. Kelner, "A survey on open-source cloud computing solutions," in *VIII Workshop em Clouds, Grids e Aplicaçõ*es, 2010, pp.3-16

[2] N. Leavitt, "Is cloud computing really ready for prime time?," Computer, vol. 42, no. 1, pp. 15-20, Jan 2009.

[3] National Institute of Standards and Technology. (2011). The NIST Definition of Cloud Computing [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[4] T. C. Chieu, A. Mohindra, A. A. Karve and A. Segal, "Dynamic scaling of web applications in a virtualized cloud computing environment," in Proc. *IEEE Int. Conf. on e-Business Engineering (ICEBE 2009)*, Macau, China, 2009, pp. 281-286.

[5] N. Noppakuat, J. Seangrat, and P. Uthayopas, "Effective workload management strategies for a cloud of virtual machine," in *14th Int. Annu. Symp. on Computational Science and Engineering(ANSCSE)*, 2010, pp.442-447

[6] H. Jinhua, G. Jianhua, S. Guofei, and Z. Tianhai, "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment," in *Parallel Architectures, Algorithms and Programming (PAAP), 2010 3rd Int. Symp. on*, 2010, pp.89-96.

[7] J. M. Galloway, K. L. Smith, and S. S. Vrbsky, "Power aware load balancing for cloud computing," in *Proc. World Congr. on Engineering and Computer Science*, San Francisco, 2011, pp.127-132

[8] Y. Chao-Tung, W. Kuan-Chieh, C. Hsiang-Yao, K. Cheng-Ta, and H. Ching-Hsien. "Implementation of a green power management algorithm for virtual machines on cloud computing," in Ubiquitous Intelligence and Computing, Germany, Springer Berlin - Heidelberg, 2011, pp.280-249.

[9] M. Fenn, M. Murphy, J. Martin and S. Goasguen. "An evaluation of KVM for use in cloud computing," in *Proc. 2nd Int. Conf. on the Virtual Computing Initiative*, RTP, NC, USA, 2008.

[10] C. Clark *et al.*, "Live Migration of Virtual Machines," in Proc. *Symp. on Networked Systems Design and Implementation*, 2005, pp.273-286.

[11] M. Armbrust *et al*., "Above the clouds: A berkeley view of cloud computing," University of California, Berkeley, Tech. Rep. USB-EECS-2009-28, Feb 2009.

[12] D. Chantry. (2009). Mapping Applications to the Cloud [Online]. Available: http://msdn.microsoft.com/en-us/library/dd430340.aspx.

[13] D. Coulson. (2010). libvirt 0.7.5 Application Development Guide [Online]. Available: http://libvirt.org/guide/pdf/Application_Development_Guide.pdf

# 8 ภาคผนวก

**ภาคผนวก ก. ขั้นตอนการสั่งให้มิดเดิลแวร์เริ่มทำงาน**

หลังจากทำการติดตั้งระบบปฏิบัติการบนเครื่องคอมพิวเตอร์ทุกเครื่องเรียบร้อยแล้ว (สามารถศึกษาขั้นตอนได้จากโครงการ “การพัฒนาดิสทริบิวชั่นสำหรับระบบการประมวลผลแบบกลุ่มเมฆ”) เครื่องที่ใช้เป็นเครื่องหลักในการติดตั้งไปยังเครื่องอื่นๆ จะมีไฟล์ต้นแบบที่ถูกสร้างขึ้นโดยอัตโนมัติ อยู่ที่ /maekin/var/startup.xml ให้ผู้ติดตั้ง เข้าสู่ระบบด้วยสิทธิของ root แล้ว ทำการแก้ไขไฟล์นี้ให้เป็นไปตามที่ผู้ใช้ต้องการ (เป็นการตั้งค่าต่างๆ ที่สำคัญของระบบเมฆินทร์) ด้วยโปรแกรม vi

```
[root@mainHost ~]# vi /maekin/var/startup.xml
```

หลังจากทำการแก้ไขไฟล์เรียบร้อยแล้ว ให้พิมพ์คำสั่ง ดังนี้

```
[root@mainHost ~]# cd /maekin/lib/middleware/
[root@mainHost middleware]# python cloudStartup.py /maekin/var/startup.xml
```

ระบบจะเริ่มทำการติดต่อไปยังเครื่องคอมพิวเตอร์ทุกๆ เครื่อง และประสานงานจนเกิดเป็นระบบประมวลผลแบบกลุ่มเมฆขึ้น ซึ่งในตอนนี้ ก็สามารถใช้งานระบบเมฆินทร์ได้ ผ่านทาง Command Line

**ภาคผนวก ข. การใช้งานระบบเมฆินทร์ผ่านทาง Command Line**

หลังจากติดตั้งระบบเสร็จสิ้น ผู้ใช้สามารถควบคุมระบบเมฆินทร์ได้ด้วยคำสั่ง mksh ที่เครื่องคอมพิวเตอร์ใดๆ ก็ได้ในระบบ ดังนี้

```
[root@anyHost ~]# mksh
```

แล้วจะเห็น Shell อันใหม่ขึ้นมาในระบบ เช่นนี้

```
mksh>_
```

ผู้ใช้สามารถใช้คำสั่ง help ในการดูรายละเอียดของคำสั่งทั้งหมดที่สามารถทำได้ผ่านทาง mksh ซึ่งจะเห็นสิ่งที่สามารถทำได้ตามตารางที่ 2

ตารางที่ 2 คำสั่งสำหรับเครื่องมือควบคุมเมฆินทร์ผ่าน COMMAND LINE

| **คำสั่ง** | คำอธิบาย |
|---|---|
| changecloud | เปลี่ยนการเชื่อมต่อไปยังเมฆินทร์อื่นๆ ที่อยู่นอกระบบที่กำลังใช้อยู่ |
| cloudinfo | ดูตัวแปรและสถานะของระบบ |
| guestcreate | สร้างคอมพิวเตอร์แบบเสมือน |
| guestdestroy | ทำลายคอมพิวเตอร์แบบเสมือน |

| | |
|---|---|
| guestduplicate | คัดลอกคอมพิวเตอร์แบบเสมือน |
| guestforceoff | บังคับปิดคอมพิวเตอร์แบบเสมือน |
| guestinfo | ดูข้อมูลของคอมพิวเตอร์แบบเสมือน |
| guestlist | แสดงรายชื่อคอมพิวเตอร์แบบเสมือนที่มีทั้งหมด |
| guestmigrate | เคลื่อนย้ายคอมพิวเตอร์แบบเสมือน |
| guestmonitor | ดูสถานะการใช้งานปัจจุบันของคอมพิวเตอร์แบบเสมือน |
| guestpause | หยุดคอมพิวเตอร์แบบเสมือนชั่วคราว |
| guestrestore | สั่งให้คอมพิวเตอร์เสมือนที่ save กลับมาทำงานต่อ |
| guestresume | สั่งให้คอมพิวเตอร์เสมือนที่หยุดไปชั่วคราว ทำงานต่อ |
| guestsave | หยุดการทำงานของคอมพิวเตอร์เสมือนแล้วบันทึกลงหน่วยความจำสำรอง |
| gueststart | สั่งให้คอมพิวเตอร์เสมือนเริ่มทำงาน |
| help | แสดงรายการคำสั่งที่ใช้งานได้ทั้งหมด |
| help [command] | แสดงรายละเอียดการใช้งานของคำสั่งที่ระบุ |
| hostadd | เพิ่มคอมพิวเตอร์เข้ามาในระบบ |
| hosthibernate | hibernate เครื่องคอมพิวเตอร์ |
| hostinfo | แสดงข้อมูลของเครื่องคอมพิวเตอร์ |
| hostlist | แสดงรายการของเครื่องคอมพิวเตอร์ในระบบทั้งหมด |
| hostmonitor | ดูสถานะการใช้งานปัจจุบันของคอมพิวเตอร์ |
| hostremove | เอาเครื่องคอมพิวเตอร์ที่ระบุ ออกไปจากระบบ |
| hostshutdown | ปิดเครื่องคอมพิวเตอร์ที่ระบุ |
| hoststandby | standby เครื่องคอมพิวเตอร์ที่ระบุ |
| hostwakeup | เปิดเครื่องคอมพิวเตอร์ที่อยู่ในระบบขึ้นมาใช้งาน |
| servicemigrate | ย้ายส่วนประกอบที่สำคัญในระบบไปยังเครื่องคอมพิวเตอร์ที่ระบุ |
| setautomode | เปลี่ยนรูปแบบการทำงานในการจัดการทรัพยากรอัตโนมัติ |
| setishost | ปรับเครื่องคอมพิวเตอร์ที่ระบุ ว่าจะอนุญาตให้รับคอมพิวเตอร์แบบเสมือนหรือไม่ |
| templateadd | เพิ่มตัวแบบของคอมพิวเตอร์แบบเสมือนเข้าไปในระบบ |
| templatecreatefromguest | สร้างตัวแบบของคอมพิวเตอร์แบบเสมือนจากคอมพิวเตอร์แบบเสมือนที่ระบุ |
| templateinfo | ดูรายละเอียดของตัวแบบของคอมพิวเตอร์แบบเสมือน |
| templatelist | แสดงรายการของตัวแบบของคอมพิวเตอร์แบบเสมือนในระบบทั้งหมด |
| templateremove | ทำลายตัวแบบของคอมพิวเตอร์แบบเสมือนที่ระบุ |
| where | ระบุว่า mksh กำลังเชื่อมต่ออยู่กับ Global Controller ที่อยู่ที่ไหน |
| quit, exit | ออกจาก mksh |

**ภาคผนวก ค. คู่มือการใช้งาน API**

Cloud API Service จะให้บริการโดยผู้ที่จะใช้บริการจะต้องส่ง HTTP request มาที่ URL ที่เป็นคำสั่งแต่ละคำสั่ง และใส่ตัวแปรต่างๆ ให้ถูกต้อง เช่น ต้องการดูข้อมูลของ guest ที่มีค่า guestID เป็น 3 จาก Cloud API Service ที่เปิดอยู่ที่ไอพีแอดเดรส 158.108.34.111 พอร์ต 8080 ก็จะต้องส่ง request ไปที่ URL ต่อไปนี้

http://158.108.34.111:8080/guest/getInfo?guestID=3

และ Server ก็จะตอบกลับมาในรูปแบบของ XML ซึ่งคำสั่งต่างๆ จะถูกแบ่งเป็น 5 กลุ่ม อธิบายการใช้งานได้ดังต่อไปนี้

**ค.1. API ที่เกี่ยวกับระบบ task**

ใน Cloud API Service มีคำสั่งอยู่จำนวนหนึ่ง ที่ไม่สามารถทำให้เสร็จสิ้นได้ทันทีหลังจากที่สั่ง แต่เป็นคำสั่งที่ต้องใช้เวลานานในการทำงาน ดังนั้น API ที่อยู่ในประเภทนี้จะนำงานไปเข้าคิว แล้วส่งค่า taskID กลับไป เพื่อให้ต้นทางสามารถใช้ค่า taskID นี้ในการถามสถานะของงานว่าทำไปถึงไหนแล้ว

❖ **/task/poll?taskID=X**

เป็นงานที่ทำได้ทันที ใช้ในการถามสถานะของงานที่ระบุว่ากำลังเข้าคิว กำลังทำ หรือทำเสร็จแล้ว รวมถึงรายละเอียดของงานนั้น

**ค.2. API ที่เกี่ยวกับระบบ cloud**

❖ **/cloud/getStorageInfo**

เป็นงานที่ทำได้ทันที ใช้แสดงข้อมูลการใช้งานหน่วยความจำสำรองบนเครื่องที่รับหน้าที่เป็น Shared Storage Service

❖ **/cloud/getInfo**

เป็นงานที่ทำได้ทันที ใช้แสดงข้อมูลทั่วไปเกี่ยวกับระบบ ได้แก่ Cloud UUID, ชื่อของ Cloud, ข้อมูลเกี่ยวกับระบบเน็ตเวิร์ค, กลุ่มของไอพีแอดเดรสที่สามารถใช้ได้ รวมถึงรูปแบบการทำงานแบบอัตโนมัติที่ใช้อยู่

❖ **/cloud/migrateGlobalController?targetHostID=X**

เป็นงานที่ต้องเข้าคิว ใช้ในการเคลื่อนย้าย Global Controller และ DHCP Controller จากเครื่องปัจจุบันไปยังเครื่องอื่น

❖ **/cloud/migrateInformationService?targetHostID=X**

เป็นงานที่ต้องเข้าคิว ใช้ในการเคลื่อนย้าย Database Service จากเครื่องปัจจุบันไปยังเครื่องอื่น

❖ **/cloud/migrateCA?targetHostID=X**

เป็นงานที่ต้องเข้าคิว ใช้ในการเคลื่อนย้าย Certificate Authority จากเครื่องปัจจุบันไปยังเครื่องอื่น

❖ **/cloud/migrateNFS?targetHostID=X**

เป็นงานที่ต้องเข้าคิว ใช้ในการเคลื่อนย้าย Shared Storage Service จากเครื่องปัจจุบันไปยังเครื่องอื่น โดยงานนี้เป็นงานที่จำเป็นต้องปิดการให้บริการระบบก่อนที่จะมีการเคลื่อนย้าย รวมถึงผู้ย้าย จะต้องทำการคัดลอกไฟล์ใน /maekin/var/storage จากเครื่องต้นทาง ไปยังเครื่องปลายทางไว้ก่อนด้วย เนื่องจากระบบไม่สามารถคัดลอกให้ได้โดยอัตโนมัติ

❖ **/cloud/setAutoMode?mode={0,1,2}**

เป็นงานที่ทำได้ทันที ใช้ในการปรับเปลี่ยนรูปแบบในการจัดการทรัพยากรแบบอัตโนมัติ โดย 0 หมายถึงไม่เปิดใช้งานระบบนี้ 1 หมายถึงเปิดใช้งานเฉพาะการโยกย้ายคอมพิวเตอร์เสมือนแบบอัตโนมัติ (เพื่อประสิทธิภาพ) และ 2 คือเพิ่มการควบคุมการเปิดปิดโฮสต์แบบอัตโนมัติด้วย (เพื่อประหยัดพลังงาน)

**ค.3. API ที่เกี่ยวกับระบบ host**

❖ **/host/getInfo[?hostID=X]**

เป็นงานที่ทำได้ทันที ใช้แสดงข้อมูลทั่วไปของคอมพิวเตอร์เครื่องที่ระบุ หรือหากไม่ระบุก็จะแสดงทั้งหมด

❖ **/host/getCurrentCPUInfo?hostID=X**

เป็นงานที่ทำได้ทันที ใช้แสดงภาระการใช้งานหน่วยประมวลผล ณ ขณะนั้นของเครื่องคอมพิวเตอร์ที่ระบุ

❖ **/host/getCurrentMemoryInfo?hostID=X**

เป็นงานที่ทำได้ทันที ใช้แสดงปริมาณการใช้งานหน่วยความจำหลัก ณ ขณะนั้น ของเครื่องคอมพิวเตอร์ที่ระบุ

❖ **/host/getCurrentNetworkInfo?hostID=X**

เป็นงานที่ทำได้ทันที ใช้แสดงอัตราการรับและส่งข้อมูล ณ ขณะนั้นของเครื่องคอมพิวเตอร์ที่ระบุ

❖ **/host/getCurrentStorageInfo?hostID=X**

เป็นงานที่ทำได้ทันที ใช้แสดงปริมาณการใช้งานหน่วยความจำสำรอง ณ ขณะนั้น ของเครื่องคอมพิวเตอร์ที่ระบุ

❖ **/host/getAllCurrentInfo?[hostID=X]**

เป็นงานที่ทำได้ทันที ใช้แสดงข้อมูลสถานะปัจจุบัน ทั้งหน่วยประมวลผล หน่วยความจำหลัก หน่วยความจำสำรอง และอัตราการรับส่งข้อมูล

❖ **/host/close?hostID=X&mode={standby,hibernate,shutdown}**

เป็นงานที่ต้องเข้าคิว ใช้ในการปิดคอมพิวเตอร์ที่ระบุ โดยสามารถเลือกได้ว่าจะทำการปิดในรูปแบบไหน ซึ่งก่อนการปิดในรูปแบบใดก็ตาม Service ต่างๆ รวมถึงเครื่องคอมพิวเตอร์เสมือน ที่อยู่บนเครื่องคอมพิวเตอร์นี้ทั้งหมดจะถูกย้ายออกไปสู่คอมพิวเตอร์เครื่องอื่นโดยอัตโนมัติ

❖ **/host/wakeup?hostID=X**

เป็นงานที่ต้องเข้าคิว ใช้ในการเปิดเครื่องคอมพิวเตอร์ที่ปิดอยู่ Standby อยู่ หรือ hibernate อยู่ ให้กลับขึ้นมาทำงานได้อีกครั้ง

❖ **/host/add?hostName=ABC&MACAddress=00:12:F0:D4:8F:35&IPAddress=158.108.34.116**

เป็นงานที่ต้องเข้าคิว ใช้ในการเพิ่มเครื่องคอมพิวเตอร์เครื่องใหม่เข้ามาในระบบ ซึ่งเครื่องคอมพิวเตอร์นี้จะต้องถูกติดตั้งระบบปฏิบัติการมาจนเสร็จสมบูรณ์แล้ว จึงจะสามารถเพิ่มเข้ามาได้

❖ **/host/remove?hostID=X**

เป็นงานที่ต้องเข้าคิว ใช้ในการถอดเครื่องคอมพิวเตอร์ที่ระบุออกไปจากระบบ โดยก่อนที่จะถอดออกนั้น Service ต่างๆ รวมถึงเครื่องคอมพิวเตอร์เสมือน ที่อยู่บนเครื่องคอมพิวเตอร์นี้ทั้งหมด จะถูกย้ายออกไปสู่คอมพิวเตอร์เครื่องอื่นโดยอัตโนมัติ

❖ **/host/setIsHost?hostID=X&isHost={0,1}**

เป็นงานที่ทำได้ทันที ใช้ในการปรับค่าว่าจะอนุญาตให้เครื่องคอมพิวเตอร์ที่ระบุ สามารถรองรับเครื่องคอมพิวเตอร์แบบเสมือนได้หรือไม่

**ค.4. API ที่เกี่ยวกับระบบ guest**

❖ **/guest/create?guestName=X&templateID=X&memory=X&vCPU=X[&inbound=X][&outbound=X]**

เป็นงานที่ต้องเข้าคิว ใช้ในการสร้างคอมพิวเตอร์เสมือนเครื่องใหม่ขึ้นมา โดยที่ยังไม่เปิด การกำหนดหน่วยความจำหลัก (Memory) ให้กำหนดเป็นเมกะไบต์ (MB) ส่วนการจำกัดการใช้งานแบนด์วิธ ให้กำหนดในหน่วยกิโลไบต์ต่อวินาที (KB/sec)

❖ **/guest/rename?guestID=X&guestName=newName**

เป็นงานที่ทำได้ทันที ใช้ในการเปลี่ยนชื่อของคอมพิวเตอร์แบบเสมือนที่ระบุ

❖ **/guest/start?guestID=X[&targetHostID=X]**

เป็นงานที่ต้องเข้าคิว ใช้ในการเปิดเครื่องคอมพิวเตอร์เสมือนที่ระบุ โดยสามารถระบุว่าจะให้เปิดที่คอมพิวเตอร์เครื่องที่ระบุได้หากต้องการ แต่ถ้าไม่ระบุ คอมพิวเตอร์เสมือนเครื่องนี้จะถูกเปิดบนเครื่องคอมพิวเตอร์ที่มีภาระงานต่ำที่สุดในตอนนี้

❖ **/guest/suspend?guestID=X**

เป็นงานที่ต้องเข้าคิว ใช้ในการหยุดคอมพิวเตอร์เสมือนชั่วคราว

❖ **/guest/resume?guestID=X**

เป็นงานที่ต้องเข้าคิว ใช้ในการสั่งให้คอมพิวเตอร์เสมือนที่ถูก Suspend ทำงานต่อ

❖ **/guest/save?guestID=X**

เป็นงานที่ต้องเข้าคิว ใช้ในการหยุดคอมพิวเตอร์เสมือนชั่วคราวและเก็บลงบนหน่วยความจำสำรอง

❖ **/guest/restore?guestID=X[&targetHostID=X]**

เป็นงานที่ต้องเข้าคิว ใช้ในการสั่งให้คอมพิวเตอร์เสมือนที่ถูก save ทำงานต่อ

❖ **/guest/forceOff?guestID=X**

เป็นงานที่ต้องเข้าคิว ใช้ในการบังคับปิดคอมพิวเตอร์เสมือนที่ระบุ

❖ **/guest/destroy?guestID=X**

เป็นงานที่ต้องเข้าคิว ใช้ทำลายคอมพิวเตอร์เสมือนที่ระบุ ออกจากระบบ

❖ **/guest/getInfo[?guestID=X]**

เป็นงานที่ทำได้ทันที ใช้แสดงข้อมูลทั่วไป เช่น ชื่อเครื่อง, ไอพีแอดเดรส, แมคแอดเดรส, ทรัพยากรที่ได้รับ ของคอมพิวเตอร์เสมือนเครื่องที่ระบุ หรือถ้าหากไม่ระบุ ก็จะแสดงข้อมูลของเครื่องคอมพิวเตอร์เสมือนทุกเครื่องในระบบ

❖ **/guest/getState?[guestID=X]**

เป็นงานที่ทำได้ทันที ใช้แสดงสถานะปัจจุบันของคอมพิวเตอร์เสมือนเครื่องที่ระบุ หรือถ้าหากไม่ระบุ ก็จะแสดงข้อมูลของเครื่องคอมพิวเตอร์เสมือนทุกเครื่องในระบบ

❖ **/guest/getCurrentCPUInfo?guestID=X**

เป็นงานที่ทำได้ทันที ใช้แสดงร้อยละของภาระการประมวลผลที่ถูกใช้โดยคอมพิวเตอร์เสมือนเครื่องนี้ ณ เวลาปัจจุบัน โดยเทียบกับเครื่องคอมพิวเตอร์จริงที่ใช้

❖ **/guest/getCurrentMemoryInfo?guestID=X**

เป็นงานที่ทำได้ทันที ใช้แสดงปริมาณการใช้งานหน่วยความจำหลัก ของคอมพิวเตอร์เสมือนที่ระบุ ณ ขณะนั้น

❖ **/guest/getCurrentNetworkInfo?guestID=X**

เป็นงานที่ทำได้ทันที ใช้แสดงอัตราการรับส่งข้อมูลของคอมพิวเตอร์เสมือนเครื่องที่ระบุ ณ ขณะนั้น

❖ **/guest/getCurrentIOInfo?guestID=X**

เป็นงานที่ทำได้ทันที ใช้แสดงอัตราการอ่านเขียนข้อมูล ลงบนหน่วยความจำสำรอง ณ ขณะนั้น

❖ **/guest/getCustomizedInfo?cpu={0,1}&memory={0,1}&network={0,1}&io={0,1}&guestIDs=X[,X[,X ... ]]**

เป็นงานที่ทำได้ทันที ใช้แสดงข้อมูลที่ถูกระบุให้แสดง (0 คือไม่แสดง, 1 คือแสดง) โดยสามารถระบุคอมพิวเตอร์เสมือนที่ต้องการข้อมูลได้มากกว่า 1 เครื่อง

❖ **/guest/scaleMemory?guestID=X&memory=X**

เป็นงานที่ต้องเข้าคิว ใช้ในการปรับขนาดของหน่วยความจำหลักที่คอมพิวเตอร์เสมือนที่ระบุจะได้รับ มีหน่วยเป็นเมกะไบต์ โดยการเปลี่ยนแปลงนี้จะเกิดผลเมื่อเครื่องถูกปิด และเปิดขึ้นมาใหม่เท่านั้น

❖ **/guest/scaleCPU?guestID=X&vCPU=X**

เป็นงานที่ต้องเข้าคิว ใช้ในการปรับจำนวนของหน่วยประมวลผลจำลองที่คอมพิวเตอร์เสมือนที่ระบุจะ โดยการเปลี่ยนแปลงนี้จะเกิดผลเมื่อเครื่องถูกปิด และเปิดขึ้นมาใหม่เท่านั้น

❖ **/guest/scaleBandwidth?guestID=X[&inbound=X][&outbound=X]**

เป็นงานที่ต้องเข้าคิว ใช้ในการปรับขนาดของแบนด์วิดท์ที่คอมพิวเตอร์เสมือนที่ระบุจะได้รับ มีหน่วยเป็นกิโลไบต์ต่อวินาที โดยการเปลี่ยนแปลงนี้จะเกิดผลเมื่อเครื่องถูกปิด และเปิดขึ้นมาใหม่เท่านั้น

❖ **/guest/migrate?guestID=X&targetHostID=X**

เป็นงานที่ต้องเข้าคิว ใช้ในการเคลื่อนย้ายคอมพิวเตอร์เสมือนไปยังเครื่องคอมพิวเตอร์ที่ต้องการ

❖ **/guest/duplicate?guestName=X&sourceGuestID=X[&memory=X][&vCPU=X][&inbound=X][&outbound=X]**

เป็นงานที่ต้องเข้าคิว ใช้ในการสร้างคอมพิวเตอร์เสมือนเครื่องใหม่ที่มีลักษณะเหมือนกับคอมพิวเตอร์เสมือนที่ใช้เป็นต้นแบบ โดยสามารถที่จะเปลี่ยนแปลงปริมาณทรัพยากรที่จะให้กับคอมพิวเตอร์เสมือนเครื่องใหม่ได้ด้วย หากต้องการ

**ค.5. API ที่เกี่ยวกับระบบ template**

❖ **/template/getInfo[?templateID=X]**

เป็นงานที่ทำได้ทันที ใช้แสดงข้อมูลของตัวแบบของคอมพิวเตอร์เสมือนที่ระบุ แต่ถ้าไม่มีการระบุ ก็จะนำตัวแบบทั้งหมดออกมาแสดง

❖ **/template/add?fileName=X&OS=X&description=X&minimumMemory=X&maximumMemory=X**

เป็นงานที่ทำได้ทันที ใช้ในการเพิ่มตัวแบบของคอมพิวเตอร์เสมือนอันใหม่เข้าไปในระบบ โดยก่อนที่จะเพิ่ม ผู้เพิ่มจำเป็นต้องนำอิมเมจไฟล์ของตัวแบบ ไปใส่ไว้ใน /maekin/var/storage/templates ของเครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็น Shared Storage Server ก่อน และใช้ชื่อไฟล์นั้นใส่เป็นตัวแปร fileName

❖ **/template/remove?templateID=X**

เป็นงานที่ต้องเข้าคิว ใช้ในการทำลายตัวแบบของคอมพิวเตอร์เสมือนที่ระบุ ออกจากระบบ

❖ **/template/createFromGuest?sourceGuestID=X&description=X**

เป็นงานที่ต้องเข้าคิว เพิ่มตัวแบบของคอมพิวเตอร์เสมือนอันใหม่เข้าไปในระบบ โดยใช้คอมพิวเตอร์เสมือนที่ระบุ เป็นตัวแบบ

# ประวัตินิสิต

1. ชื่อ-นามสกุล ประยุกต์ เจตสิกทัต เลขประจำตัวนิสิต 51052090

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์
ที่อยู่ปัจจุบัน 31/4 ถนนอุดมศิริ ตำบลหน้าเมือง อำเภอเมือง จังหวัดราชบุรี รหัสไปรษณีย์ 70000
โทรศัพท์บ้าน 032314063 โทรศัพท์เคลื่อนที่ 0858620711
E-mail koon.prayook@gmail.com

ระดับการศึกษา:

| คุณวุฒิการศึกษา | จากโรงเรียน/สถาบัน | ปีการศึกษาที่จบ |
|---|---|---|
| มัธยมศึกษาตอนปลาย | โรงเรียนมหิดลวิทยานุสรณ์ | 2550 |
| มัธยมศึกษาตอนต้น | โรงเรียนเบญจมราชูทิศ ราชบุรี | 2547 |