ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิทยาศาสตรมหาบัณฑิต (วิทยาการคอมพิวเตอร์)

ปริญญา

วิทยาการคอมพิวเตอร์ วิทยาการคอมพิวเตอร์

สาขา **ภาควิชา**

เรื่อง การจัดสรรเครื่องเสมือนโดยใช้โปรไฟล์พลังงานของเซิร์ฟเวอร์

Virtual Machine Allocation using Server Power Profile

นามผู้วิจัย นางสาวมนพัทธ์ ลิ้มรัตนศิลป์

ได้พิจารณาเห็นชอบโดย

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก .....

(อาจารย์เสฏฐวิทย์ เกิดผล, Ph.D.)

หัวหน้าภาควิชา .....

(ผู้ช่วยศาสตราจารย์ศิริกร จันทร์นวล, M.Sc.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

.....

(รองศาสตราจารย์กัญจนา ธีระกุล, D.Agr.)

คณบดีบัณฑิตวิทยาลัย

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

วิทยานิพนธ์

เรื่อง

การจัดสรรเครื่องเสมือนโดยใช้โปรไฟล์พลังงานของเซิร์ฟเวอร์

Virtual Machine Allocation using Server Power Profile

โดย

นางสาวมนพัทธ์ ลิ้มรัตนศิลป์

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

เพื่อความสมบูรณ์แห่งปริญญาวิทยาศาสตรมหาบัณฑิต (วิทยาการคอมพิวเตอร์)

พ.ศ. 2557

มนพัทธ์ ลิ้มรัตนศิลป์ 2557: การจัดสรรเครื่องเสมือนโดยใช้โปรไฟล์พลังงานของเซิร์ฟเวอร์ ปริญญาวิทยาศาสตรมหาบัณฑิต (วิทยาการคอมพิวเตอร์) สาขาวิทยาการคอมพิวเตอร์ ภาควิชาวิทยาการคอมพิวเตอร์ อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก: อาจารย์เสฏฐวิทย์ เกิดผล, Ph.D. 78 หน้า

งานวิจัยนี้มีจุดประสงค์เพื่อลดการใช้พลังงานบน Cloud ถึงแม้ว่า Cloud computing จะมีวิธีการลดการใช้พลังงานโดยพยายามจัดสรรเครื่องเสมือนให้ทำงานบนเครื่องเซิร์ฟเวอร์จำนวนน้อยที่สุด เมื่อเรามีหลายเครื่องเซิร์ฟเวอร์ให้เลือกใช้งาน แต่ยังมีประเด็นที่ต้องพิจารณาว่า ควรเลือกเครื่องไหนบ้างมาใช้งาน เพราะเครื่องเซิร์ฟเวอร์แต่ละเครื่องที่อยู่บน Cloud ไม่ได้ผลิตมาจากบริษัทเดียวกันหรือผลิตมาในยุคเดียวกัน จึงทำให้มีประสิทธิภาพของการใช้พลังงานแตกต่างกัน มาตรฐานหนึ่งที่ใช้ในการวัดประสิทธิภาพของการใช้พลังงานของเครื่องเซิร์ฟเวอร์ก็คือมาตรฐานของ SPEC (Standard Performance Evaluation Corporation) งานวิจัยนี้ได้นำเสนออัลกอริทึม 2 วิธีคือ ESO และ FL-n ซึ่งเป็นอัลกอริทึมสำหรับการเรียงลำดับเครื่องเซิร์ฟเวอร์ โดยใช้โปรไฟล์พลังงานของเซิร์ฟเวอร์ที่สร้างจากค่าพลังงานที่วัดได้ตามมาตรฐานของ SPEC ในการคัดเลือกเพื่อให้การใช้พลังงานโดยรวมบน Cloud ลดลง ผลลัพธ์ที่ได้จากการทดลองพบว่า อัลกอริทึมแบบ ESO จะทำงานได้ประสิทธิภาพดีเมื่อเครื่องเซิร์ฟเวอร์ผลิตมาในยุคเดียวกัน ซึ่งค่าการใช้พลังงานที่ได้ไม่สูงเกินกว่า 10% เมื่อเทียบกับลำดับการใช้เครื่องเซิร์ฟเวอร์ที่ประหยัดพลังงานที่สุด (Optimal ordering) ส่วนอัลกอริทึม FL-n ทำงานได้ประสิทธิภาพดีในทุกๆ เหตุการณ์ (Scenario) แม้ว่าเครื่องเซิร์ฟเวอร์จะมาจากต่างยุคกัน ซึ่งค่าความแม่นยำได้สูงไม่เกิน 5% เมื่อเทียบกับค่า Optimal ordering

/ /

ลายมือชื่อนิสิต ลายมือชื่ออาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

Monnapat Limrattanasilp 2014: Virtual Machine Allocation using Server Power Profile. Master of Science (Computer Science), Major Field: Computer Science, Department of Computer Science. Thesis Advisor: Mr. Sethavidh Gertphol, Ph.D. 78 pages.

Cloud computing has potentials to reduce energy consumption in the data center of Cloud providers by consolidating virtual machines into as few servers as possible. The important question then becomes which physical machines should be used first, because machines from different company or different generation may not be equally efficient in energy consumption. The energy efficiency of a server is characterized by its SPEC Power benchmark. This research proposes two Greedy algorithms, ESO and FL-n, to determine the order of physical machines to be used in a Cloud environment such that the overall energy consumption is minimized. The experimental results show that ESO performed well when servers were from the same generation, using energy not more than 10% over the optimal ordering. FL-n provided good results even when servers were from different generation also, using energy not more than 5% over the optimal in all scenarios.

/ /

Student's signature Thesis Advisor's signature

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ดีด้วยความช่วยเหลืออย่างดียิ่งของ ดร. เสฏฐวิทย์ เกิดผล ประธานกรรมการที่ปรึกษา ที่กรุณาช่วยให้คำแนะนำขั้นตอนวิธีในการดำเนินงาน ชี้แนะแนวทางแก้ไขปัญหาและตรวจสอบแก้ไขข้อบกพร่องในงานวิจัยมาโดยตลอด รวมถึงที่กรุณาช่วยสั่งสอนอบรมทั้งการทำงานและเรื่องอื่นๆ มาโดยตลอด

ขอกราบขอบพระคุณ ผศ.ดร. ชวลิต ศรีสถาพรพัฒน์และผศ.ดร. สุขุมาล กิติสิน ที่ช่วยให้ข้อมูล ให้คำปรึกษาและคำแนะนำในการทำวิทยานิพนธ์ รวมถึงคำแนะนำในการนำเสนอผลงานวิจัยแบบปากเปล่าในการประชุมทางวิชาการระดับนานาชาติ

ขอกราบขอบพระคุณ คุณพ่ออติเทพ ลิ้มรัตนศิลป์และคุณแม่กานต์มณี ลิ้มรัตนศิลป์ ที่ให้กำลังใจในการเรียน ให้การสนับสนุนและให้ความช่วยเหลืออย่างที่สุดมาโดยตลอดจนสำเร็จการศึกษา

ขอขอบพระคุณคณาจารย์ทุกท่านที่ให้การอบรม สั่งสอนวิชาความรู้ต่างๆ มาจนสำเร็จการศึกษา

ขอขอบคุณบัณฑิตวิทยาลัยและภาควิชาวิทยาการคอมพิวเตอร์ ที่ได้สนับสนุนด้านอุปกรณ์ด้านเงินทุนนำเสนองานในประเทศ และสถานที่ตลอดระยะเวลาในการทำวิทยานิพนธ์

ขอขอบคุณนางสาวดารณี ฐิติประยูรวงศ์ ที่ให้ความช่วยเหลือและให้คำแนะนำการเขียนวิทยานิพนธ์ รวมถึงขอขอบคุณเพื่อนๆ ปริญญาโททุกคนที่คอยให้กำลังใจ และช่วยเหลืออำนวยความสะดวกในการสอบปากเปล่าขั้นสุดท้าย

คุณค่าและประโยชน์อันพึงมีจากวิทยานิพนธ์นี้ ขอมอบแด่ผู้มีพระคุณทุกท่าน

มนพัทธ์ ลิ้มรัตนศิลป์

กรกฎาคม 2557

สารบัญ

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

สารบัญภาพ (ต่อ)

การจัดสรรเครื่องเสมือนโดยใช้โปรไฟล์พลังงานของเซิร์ฟเวอร์

Virtual Machine Allocation using Server Power Profile

คำนำ

Cloud computing (การประมวลผลแบบกลุ่มเมฆ) คือ เทคโนโลยีรูปแบบหนึ่งสำหรับการเข้าใช้งานทรัพยากรของระบบ เช่น หน่วยประมวลผล (Compute unit), พื้นที่จัดเก็บข้อมูล (Storage) และโปรแกรม (Program) ในรูปแบบของบริการ (Services) ซึ่งสามารถเข้าถึงได้อย่างรวดเร็ว มีการดูแลควบคุมที่ง่าย โดยสามารถเข้าใช้งานได้สะดวกเพียงผ่านเครือข่ายอินเทอร์เน็ต

แนวทางการประมวลผลใหม่นี้เป็นการให้บริการด้านต่างๆ ผ่านอินเทอร์เน็ตเพียงผู้ใช้งานมีเว็บเบราว์เซอร์ (Web browser) ผู้ใช้อาจเป็นองค์กรธุรกิจที่ต้องการฝากการประมวลผลผ่านบริการบนเครือข่าย Cloud computing หรืออาจเป็นผู้ให้บริการแอพพลิเคชั่น (Application) ที่ให้บริการประชาชนในวงกว้าง ตัวอย่างเช่น บริการเว็บเซอร์วิส (Web service) เช่น flickr, gmail, facebook, Amazon ฯลฯ ซึ่งผู้ใช้ไม่จำเป็นต้องดูแลทรัพยากรเหล่านี้เอง เพียงซื้อบริการจากผู้ให้บริการ โดยคิดค่าบริการแบบ “pay per use” คือใช้เท่าไหร่จ่ายเท่าที่ใช้ อีกทั้ง Cloud ยังช่วยลดค่าใช้จ่ายในด้านการใช้พลังงานให้แก่ผู้ให้บริการอีกด้วย ผลประโยชน์นี่เองที่ทำให้ Cloud เป็นแพลตฟอร์มการประมวลผล (computing platform) ที่ได้รับความนิยมอย่างแพร่หลายในปัจจุบัน

ในปัจจุบันผู้ให้บริการ public Cloud เช่น Amazon AWS หรือ Google App Engine ได้รวมทรัพยากรบน Cloud เอาไว้ในรูปแบบของเครื่องเสมือน (Virtual machine หรือ VM) ซึ่งแต่ละ VM ประกอบไปด้วยการกำหนดค่า compute power, หน่วยความจำ (memory) และ storage เมื่อมีงานร้องขอประมวลผลขึ้นบน Cloud VM จะถูกสร้างขึ้นบนเครื่องเซิร์ฟเวอร์ โดยในหนึ่งเครื่องสามารถสร้างได้มากกว่าหนึ่ง VM และสร้างได้มากสุดเท่าที่เครื่องนั้นรองรับได้เพียงพอ

เมื่อหนึ่งเครื่องสามารถสร้างได้หลาย VM ผู้ให้บริการจึงลดค่าใช้จ่ายพลังงานรวมที่เกิดขึ้น โดยนำ VM ไปสร้างบนเครื่องเซิร์ฟเวอร์ให้เต็มทีละเครื่องไป เพื่อให้จำนวนเครื่องเซิร์ฟเวอร์ที่ถูกเปิดใช้งานมีจำนวนน้อยที่สุดและปิดเครื่องที่ไม่ใช้เสีย เพื่อการประหยัดพลังงาน

ประเด็นสำคัญที่น่าสนใจคือแล้วเครื่องเซิร์ฟเวอร์เครื่องใด ที่ควรจะถูกนำมาใช้งานก่อน เพราะเครื่องเซิร์ฟเวอร์แต่ละเครื่องไม่ได้ผลิตจากแหล่งเดียวกันหรือมาจากยุคเดียวกัน ดังนั้นจึงทำให้ประสิทธิภาพในการใช้พลังงานของเครื่องมีความแตกต่างกัน เมื่อลองสำรวจการใช้พลังงานของเครื่องเซิร์ฟเวอร์จาก SPEC (Standard Performance Evaluation Corporation) ในช่วงก่อนและหลังปี ค.ศ. 2007 พบว่าเครื่องก่อนปี ค.ศ. 2007 มีค่าการกินพลังงาน ณ สถานะว่าง (ilde state) ที่สูงมากกว่าหลังปี ค.ศ. 2007

ในงานวิจัยนี้นำเสนออัลกอริทึม 2 วิธีในการเรียงลำดับเครื่องเซิร์ฟเวอร์บนระบบ Cloud เพื่อให้ค่าการใช้พลังงานโดยรวมมีค่าน้อยที่สุด ประสิทธิภาพการใช้พลังงานของเครื่องเซิร์ฟเวอร์อ้างอิงจาก SPEC ซึ่งทำการวัดค่าพลังงานทุกๆ 10% CPU utilization

อัลกอริทึมแรกคือ Energy Slope Ordering (ESO) เรียงลำดับเครื่องเซิร์ฟเวอร์ด้วยค่าความชันของกราฟพลังงานที่ได้จากกราฟพลังงานของ SPEC ซึ่งอัลกอริทึมนี้ไม่ได้สนใจค่า idle state ส่วนอัลกอริทึมที่ 2 คือ Full-but-Last-n (FL-n) เรียงลำดับโดยขั้นแรกประเมินว่าในการประมวลผลครั้งนั้นจะใช้เครื่องเซิร์ฟเวอร์กี่เครื่อง จากนั้นสมมติว่าทุกเครื่องใช้ประสิทธิภาพของ CPU เต็มที่ (100% CPU utilization) ยกเว้นเครื่องสุดท้าย สำหรับเครื่องสุดท้ายจะเลือกโดยเปรียบเทียบกับเครื่องที่เหลืออยู่ทั้งหมดว่าเครื่องใด มีค่า n% CPU utilization น้อยที่สุดจึงเลือกเครื่องนั้นมาทำงาน ซึ่งค่า n นั้นผู้จัดสรรเครื่องสามารถกำหนดได้เองตามความเหมาะสม

ในการทดสอบประสิทธิภาพการทำงานของอัลกอริทึมที่คิดค้นขึ้น หากต้องทำการทดลองด้วยระบบ Cloud จริงทำได้ยาก เพราะการกำหนดค่าที่ต้องการในการทดลองไม่ได้ยืดหยุ่นและวัดพลังงานโดยตรงไม่ได้ จึงตัดสินใจทำการทดลองบนโปรแกรมจำลอง CloudSim ที่สามารถ

กำหนดค่าต่างๆ ไม่ว่าจะเป็น รุ่น, จำนวน, หน่วยความจำของเครื่องเซิร์ฟเวอร์ หรือกำหนด จำนวน, ขนาด, หน่วยความจำของ VM ซึ่งความยืดหยุ่นในการกำหนดค่าตัวแปรต่างๆ เหล่านี้ได้เอง ทำให้สามารถทดลองได้หลายกรณี อีกทั้ง CloudSim ยังมีการประเมินค่าพลังงานที่ระบบจะใช้งานด้วย ดังนั้นจึงเลือกโปรแกรมจำลอง ClouSim มาทดลองกับอัลกอริทึมทั้ง 2 วิธีและเปรียบเทียบประสิทธิภาพของอัลกอริทึมกับ ลำดับการใช้เครื่องเซิร์ฟเวอร์ที่ประหยัดพลังงานที่สุด (Optimal ordering) ซึ่งหาค่าได้จากการเรียงลำดับเครื่องเซิร์ฟเวอร์ทุกลำดับที่เป็นไปได้แล้วเลือกลำดับที่ทำให้ค่าผลลัพธ์ของพลังงานน้อยที่สุด จากผลการทดลองพบว่า อัลกอริทึม ESO ทำงานได้ประสิทธิภาพไม่ค่อยดีเมื่อโฮสต์มีทั้งรุ่นเก่าและรุ่นใหม่ผสมกัน ซึ่งมีการใช้พลังงานมากกว่าค่า Optimal ถึง 50% ในบางกรณี ในขณะที่อัลกอริทึม FL-n ทำงานได้ดีในทุกกรณีและมีการใช้พลังงานสูงเกินกว่าค่า Optimal เพียง 5% โดยเฉลี่ย

วัตถุประสงค์

1. สร้างวิธีการแก้ปัญหา (heuristic) ในการจัดสรรทรัพยากร (resource allocation) โดยพิจารณาจากพลังงานที่เครื่องเซิร์ฟเวอร์ใช้

2. เมื่อจัดทรัพยากรตามสมมติฐาน วัดพลังงานรวมที่เครื่องเซิร์ฟเวอร์ใช้ประมวลผล จะได้ค่าของการใช้พลังงานน้อยกว่าการจัดทรัพยากรแบบไม่ตรงตามสมมติฐาน

3. การวัดผลจะวัดผ่านทางโปรแกรมจำลอง CloudSim ซึ่งเทียบผลระหว่างวิธีการจัดสรรทรัพยากรโดยใช้อัลกอริทึมเทียบกับ ค่าของลำดับการใช้เครื่องเซิร์ฟเวอร์ที่ประหยัดพลังงานที่สุด (Optimal ordering)

4. เป็นการนำเสนอแนวคิดแบบใหม่เพื่อว่าอาจมีประโยชน์แก่นักวิจัยหรือผู้ที่สนใจซึ่งสามารถนำไปประยุกต์หรือปรับใช้ได้

ประโยชน์ที่คาดว่าจะได้รับ

เพื่อประหยัดการใช้พลังงานโดยรวมของระบบ Cloud โดยทดสอบความถูกต้องในการคิดคำนวณและประสิทธิภาพของสมมติฐาน ว่าสามารถเลือกและเรียงลำดับเครื่องเซิร์ฟเวอร์ที่เหมาะสมให้แก่การร้องขอนั้นได้อย่างถูกต้องมากน้อยเพียงใด โดยขึ้นอยู่กับปัจจัยคือ การใช้พลังงานของเครื่องเซิร์ฟเวอร์ (Server Power profile), คำร้องขอทรัพยากรที่ต้องการ, แนวคิดหรือวิธีการการเลือกทรัพยากรที่มีอยู่ให้เหมาะสมกับสภาพแวดล้อม

ขอบเขตและข้อจำกัด

จากสมมติฐานที่ตั้งไว้ กำหนดการจัดสรรทรัพยากรโดยดูกราฟพลังงานของเครื่องเซิร์ฟเวอร์เพียงอย่างเดียว ดังนั้นข้อมูลที่สำคัญที่สุดคือค่าพลังงานที่ CPU ของเครื่องเซิร์ฟเวอร์ใช้ ซึ่งไม่พิจารณาค่าจากการใช้พลังงานของ หน่วยความจำ (memory), ระบบเครือข่าย (network) และหน่วยเก็บข้อมูล (harddisk)

การนำแนวคิดที่ตั้งสมมติฐานไว้ไปทดลองลงระบบจริงของ Cloud เพื่อพิสูจน์นั้นเป็นเรื่องที่ทำได้ยากเหตุเพราะการเพิ่มหรือลดขนาดของทรัพยากรจะขึ้นกับการตกลงระหว่างผู้ใช้บริการกับ

ผู้ให้บริการ ทำให้ขนาดของทรัพยากรที่ทำการทดลองมีอยู่อย่างจำกัด ไม่ยืดหยุ่น การทดลองได้ผลลัพธ์ที่ไม่หลากหลาย จึงใช้โปรแกรมจำลอง CloudSim เพื่อทำการทดลองแทน

การตรวจเอกสาร

ทฤษฎีที่เกี่ยวข้อง

1. บทนำ

แนวคิด Cloud computing (Wikipedia, 2014) คือแนวคิดการให้บริการทรัพยากรคอมพิวเตอร์ในรูปแบบของ “สาธารณะ” ซึ่งคำว่า “สาธารณะ” หมายถึง ทรัพยากรคอมพิวเตอร์นั้นสามารถใช้งานได้โดยบุคคลทั่วไปซึ่งอาจใช้งานโดยไม่มีค่าใช้จ่ายหรือมีค่าจ่ายขึ้นอยู่กับผู้ให้บริการ แอพพลิเคชั่นต่างๆ เหล่านี้จากที่เคยอยู่บนเครื่องคอมพิวเตอร์ส่วนบุคคล หรือการประมวลผลบนเครื่องคอมพิวเตอร์เซิร์ฟเวอร์ขององค์กรหรือหน่วยงานที่เชื่อมต่อกันโดยผ่านโครงข่ายท้องถิ่น เช่น โครงข่าย LAN (Local Area Network) หรืออาจเป็นโครงข่ายขนาดใหญ่ เช่น อินเทอร์เน็ต แต่มีการจำกัดขอบเขตของการสื่อสารให้อยู่ภายในกลุ่มขององค์กรของตน หรือกลุ่มผู้ใช้งานของตน เปลี่ยนมาเป็นการประมวลผลแอพพลิเคชั่นเหล่านั้น โดยผ่านการทำงานของกลุ่มเครื่องคอมพิวเตอร์เซิร์ฟเวอร์จำนวนมากที่เชื่อมต่ออยู่ด้วยกันผ่านทางโครงข่ายอินเทอร์เน็ต ซึ่งกลุ่มเครื่องคอมพิวเตอร์เหล่านั้นได้รับการควบคุมและบริหารจัดการอย่างเป็นรูปแบบ โดยผู้ให้บริการ Cloud computing สามารถเลือกกำหนดระดับประสิทธิภาพ (Quality of Service หรือ QoS) ให้กับลูกค้าผู้ประสงค์ใช้บริการ Cloud computing ได้ตามเงื่อนไขสัญญาการใช้บริการระหว่างกัน (Service Level Agreement หรือ SLA)

ด้วยความพร้อมของเทคโนโลยีฮาร์ดแวร์และซอร์ฟแวร์ในปัจจุบัน ที่สามารถผลิตเครื่องคอมพิวเตอร์ที่มีการประมวลผลสูง รวมถึงความพร้อมของเทคโนโลยีโครงข่ายอินเทอร์เน็ตและการสื่อสารโทรคมนาคมความเร็วสูง ทำให้ผู้ใช้งานคอมพิวเตอร์มองว่าบริการทุกอย่างที่ผู้ใช้ต้องการสามารถเรียกได้ผ่านคอมพิวเตอร์ส่วนบุคคลของตนเองก็เพียงพอต่อความต้องการแล้วแต่ Cloud computing ไม่เพียงแต่รวบรวมการให้บริการในรูปแบบที่หลากหลายที่สามารถเข้าถึงและใช้งานได้โดยผ่านทางอินเทอร์เน็ตอีกทั้งยังช่วยประหยัดค่าใช้จ่ายหรือการลงทุนที่อาจเกิดขึ้นกับผู้ใช้บริการ ผู้ใช้ไม่จำเป็นต้องมีเครื่องที่มีประสิทธิภาพสูงเพื่อใช้บริการบน Cloud สามารถลดค่าใช้จ่ายในการซื้อลิขสิทธิ์ของแอพพลิเคชั่นจากต้องซื้อเท่ากับจำนวนเครื่องที่ต้องใช้ลดเหลือเพียงซื้อลิขสิทธิ์เดียว ด้วยเหตุนี้ทำให้ผู้ประกอบการ Cloud computing นำความพร้อมของเทคโนโลยี

ฮาร์ดแวร์และซอร์ฟแวร์ในปัจจุบันมาสร้างเครือข่ายการประมวลผลขนาดใหญ่ ที่ถูกควบคุมด้วยอัลกอริทึมที่มีประสิทธิภาพ พร้อมตอบสนองความต้องการของผู้ใช้บริการ ซึ่งอาจเป็นองค์กรธุรกิจที่ต้องการฝากการประมวลผลผ่านบริการบนเครือข่าย Cloud computing หรืออาจเป็นผู้ให้บริการแอพพลิเคชั่นที่ให้บริการประชาชนในวงกว้าง ตัวอย่างเช่น บริการเว็บเซอร์วิส (Web Service) เช่น flickr, gmail, facebook, Amazon ฯลฯ ทำให้เทคโนโลยี Cloud computing ถูกมองว่าเป็นแนวคิดใหม่ของการประกอบธุรกิจเสียมากกว่าทางด้านเทคโนโลยี ที่เป็นเช่นนี้เพราะการทำธุรกิจเชิงระบบสารสนเทศในปัจจุบัน ผู้ใช้บริการจำเป็นต้องมีหน่วยงานสารสนเทศอยู่ในบริษัทของตนเอง เพื่อคอยสนับสนุนการทำงานกับบริษัทที่เราซื้อซอร์ฟแวร์หรือบริการเกี่ยวกับสารสนเทศ แต่ในแนวคิดของ Cloud ผู้ใช้บริการไม่จำเป็นต้องดูแลสิ่งเหล่านี้เอง ผู้ใช้บริการเพียงซื้อบริการจากผู้ขายโดยคิดค่าบริการแบบ “pay per use” คือใช้เท่าไหร่จ่ายเท่าที่ใช้ ทำให้แนวคิดนี้เป็นประโยชน์แก่บริษัทขนาดกลางและขนาดเล็กมาก ซึ่งบริษัทเหล่านี้ไม่จำเป็นต้องเสียค่าใช้จ่ายในการซื้อเครื่องคอมพิวเตอร์เซิร์ฟเวอร์ และดูแลรักษา ทำให้ลดค่าใช้จ่ายภายในองค์กรได้อย่างมาก

2. การประมวลผลแบบกลุ่มเมฆ (Cloud computing)

2.1 แนวคิดและการทำงานของ Cloud computing ได้พัฒนาต่อเนื่องมาจากระบบดังต่อไปนี้

2.1.1 Grid Computing (การประมวลผลแบบ Grid)

เป็นรูปแบบของเทคโนโลยีการประมวลผลแบบกระจาย คำว่า Grid หมายถึง “เครือข่ายที่เชื่อมโยงและกระจายทรัพยากรให้กัน” ซึ่งในที่นี้ Grid Computing คือเครือข่ายของทรัพยากรคอมพิวเตอร์ที่เชื่อมโยงกันหมดและกระจายทรัพยากรด้านคอมพิวเตอร์ให้กัน ไม่ว่าจะเป็นสมรรถนะในด้านการประมวลผล ความจุ หรือ สมรรถนะในการถ่ายโอนข้อมูล มาสร้างระบบที่ใช้แก้ปัญหาการตัดสินใจ โดยมีการกระจายภาระการประมวลผลไปยังเครื่องต่างๆ ที่เชื่อมกันอยู่เป็นเครือข่ายเหมาะกับการประมวลผลงานที่มีขนาดใหญ่

แนวคิดของ Grid และ Cloud ได้ถูกเผยแพร่และเป็นที่รู้จักในเวลาไล่เลี่ยกัน ซึ่งแนวคิดของ Grid ได้ถูกคิดขึ้นมาก่อน ถ้าหากไม่มี Grid แนวคิดของ Cloud คงไม่เกิดขึ้น รูปแบบการทำงานของ Cloud เหมือน Grid ทุกประการ หรือพูดอีกอย่างว่า Cloud ก็คือ Grid ซึ่งถูกมองเป็นรูปแบบเสมือนทำงานอยู่ในรูปแบบแอพพลิเคชั่นโดยที่ผู้ใช้งานอยู่ในขณะนั้นไม่สามารถทราบได้

เลยว่าข้อมูลที่ได้มานั้นมาจากผู้ให้บริการคนใด แต่สิ่งที่ทำให้ Grid และ Cloud แตกต่างกันคือ รูปแบบการให้บริการและการเข้าใช้งานของผู้ใช้ Grid มักเป็นการรวบรวมทรัพยากรจากหลายองค์กรที่มีนโยบายแบ่งปันทรัพยากรระหว่างกัน โดยไม่มีค่าใช้จ่าย ในขณะที่ระบบ Cloud นั้นจะคิดค่าใช้จ่ายตามการใช้งานจริง (pay-per-use)

2.1.2 Utility computing

เป็นหลักการแชร์ทรัพยากรที่คล้ายกับ Grid Computing เพียงแต่ทรัพยากรจะถูกมองเสมือนว่าเป็นบริการสาธารณูปโภค (เช่น ไฟฟ้า น้ำประปา และโทรศัพท์) โดยบริการเหล่านี้ผู้ใช้เสียค่าใช้จ่ายเพื่อใช้งานได้ตามที่ต้องการ และจ่ายตามจำนวนหรือช่วงเวลาที่ใช้งานจริง

หลักการนี้เองที่ทำให้ Cloud computing มองการใช้งานบริการอยู่ในรูปของสาธารณะและการเสียค่าใช้จ่ายตามที่ใช้งาน (pay per use)

2.1.3 Autonomic Computing

คือ ระบบประมวลการทำงานของคอมพิวเตอร์และแก้ไขโดยอัตโนมัติ ถูกพัฒนาขึ้นโดยบริษัท IBM เมื่อปี ค.ศ. 2001 โดยระบบต้องมี 8 สิ่งสำคัญคือ

ก. ต้องสามารถบำรุงรักษาตนเองได้อย่างครอบคลุมและมีการเก็บความรู้เฉพาะเกี่ยวกับส่วนประกอบทั้งหมดของระบบ

ข. ต้องมีความสามารถในการตั้งค่าตัวเองให้เหมาะสมและหลากหลายตามเงื่อนไขที่คาดการณ์ไม่ได้ที่อาจจะเกิดขึ้น

ค. ต้องติดตามตัวเองสำหรับการทำงานดีที่สุดอย่างสม่ำเสมอ

ง. ต้องรักษาแก้ไขปัญหาได้ด้วยตัวเองและสามารถค้นหาทางเลือกในการแก้ปัญหานั้น

จ. ต้องสามารถสืบค้นการคุกคามและป้องกันตัวเองจากผู้คุกคาม

ฉ. ต้องสามารถปรับตัวตามเงื่อนไขสภาพแวดล้อม

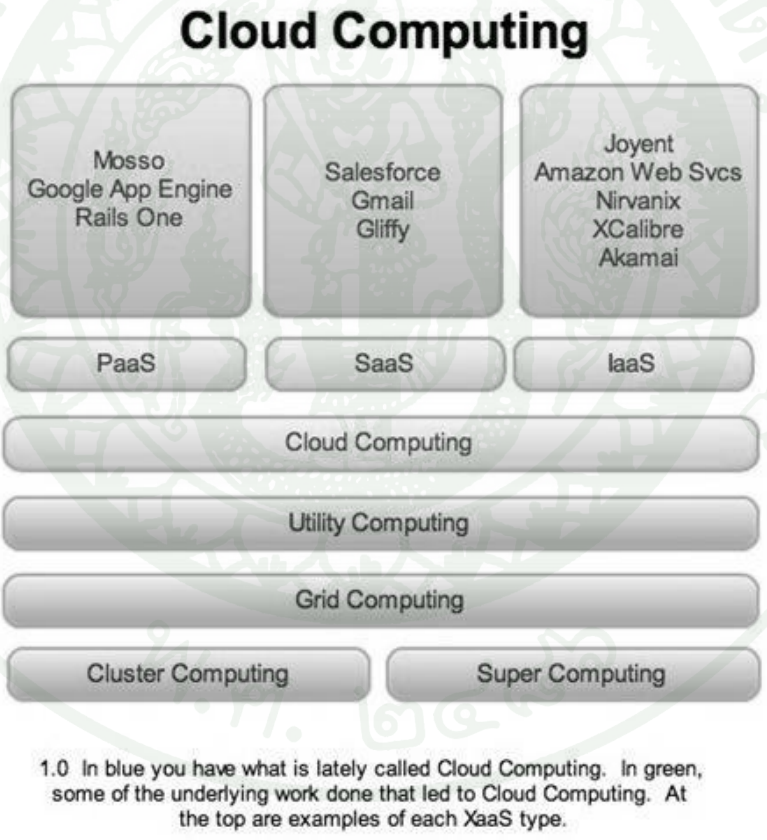
ช. ต้องอยู่บนพื้นฐานมาตรฐานเปิดแทนที่เทคโนโลยีที่มีเจ้าของ

ซ. ต้องคาดการณ์ถึงความต้องการของผู้ใช้และทำให้เกิดความมั่นใจแก่ผู้ใช้งาน

อีกองค์ประกอบหนึ่งของ Cloud computing ที่ช่วยจัดสรรแบ่งทรัพยากรในระบบให้แก่ผู้ที่เข้ามาใช้บริการบน Cloud นั่นคือนำหลักการและแนวคิดของ Autonomic Computing มาช่วยให้ระบบสามารถแบ่งจัดสรรทรัพยากรได้อย่างมีประสิทธิภาพและอัตโนมัติโดยผู้ใช้ไม่ต้องจัดสรรเอง

2.2 รูปแบบการให้บริการ Cloud

การนำ Cloud มาใช้งานร่วมกับสารสนเทศ ซึ่งการให้บริการของ Cloud บนระบบสารสนเทศทำให้การประมวลของ Cloud เกิดสถาปัตยกรรมการให้บริการขึ้น โดยมองจากรูปแบบการให้บริการสารสนเทศที่แบ่งเป็นชั้น 3 กลุ่มคือ Application, Service และ Infrastructure



ภาพที่ 1 ระดับชั้นของ Cloud computing

ที่มา: Markoff (2010)

2.2.1 Software as a Service (SaaS) หรือ On Demand Software (ODS)

เป็นความสามารถที่จัดสรรให้กับผู้ใช้งานโดยผู้ให้บริการเป็นผู้ทำการจัดหา ติดตั้งและนำออกมาแจกจ่ายให้ผู้ใช้สามารถเข้าใช้งานได้บนระบบ Cloud การใช้งานโปรแกรมสามารถเข้าถึงได้จากอุปกรณ์ไคลเอนต์ต่างๆ ผ่านทางส่วนติดต่อของไคลเอ็นต์ เช่น เว็บเบราว์เซอร์ (Web Browser) หรือโปรแกรมที่ทำมาเฉพาะ โปรแกรมเหล่านี้ไม่ถูกจัดเก็บในเครื่องคอมพิวเตอร์ของผู้ใช้หรือคือไม่มีการใช้พื้นที่บนฮาร์ดดิสก์ ผู้ใช้งานจ่ายเพียงค่าบริการการเป็นสมาชิก หรือค่าบริการตามที่ใช้งานจริง (pay per usage) ซึ่งทำให้ลดค่าใช้จ่ายลง รูปแบบของ SaaS ดีกว่าการซื้อซอฟร์แวร์ในรูปแบบเดิมคือ

ก. ลูกค้าจะได้รับบริการด้านซอฟร์แวร์และฮาร์ดแวร์ที่ดีที่สุดอยู่เสมอ โดยไม่ต้องลงทุนเองทั้งหมด เช่น ถ้าเป็นสมัยก่อนต้องซื้อซอฟร์แวร์ตัวนึงมาใช้งานในองค์กร และยังต้องซื้ออุปกรณ์ฮาร์ดแวร์ต่างๆ เช่น เครื่องเซิร์ฟเวอร์, ฮาดร์ดิสก์ รวมทั้งจ้างผู้ดูแลมาดูแลระบบ อีกทั้งยังมีค่าทำระบบสำรองข้อมูล ซึ่งทำให้มีค่าใช้จ่ายที่สูงมากในการนำซอฟร์แวร์มาใช้ โดย SaaS ลูกค้าเพียงแค่จ่ายค่าบริการเป็นรายผู้ใช้งานเท่านั้น โดยทางผู้ให้บริการจะเป็นดูแลทางด้านฮาร์ดแวร์หรือระบบผู้ดูแลให้เองทั้งหมด

ข. ช่วยให้ลูกค้าสามารถคำนวณค่าใช้จ่ายทางด้านซอฟร์แวร์ได้ชัดเจนยิ่งขึ้น ทำให้สามารถคำนวณหรือวางแผนการดำเนินงานได้ดี เพราะ SaaS จะคิดค่าบริการเป็นอัตราที่แน่นอน ถ้าหากเทียบกับสมัยก่อน อาจต้องเสียค่า Support ค่อนข้างสูง (บางทีอาจสูงกว่าค่าใช้จ่ายตอนซื้อซอฟร์แวร์ในครั้งแรกด้วยซ้ำ)

ค. การอัพเดทซอฟร์แวร์ทำได้ง่ายและใช้ระยะเวลาสั้น ปัญหาที่พบในสมัยก่อนคือ การอัพเดทซอฟร์แวร์ทำได้ยุ่งยากและเสียเวลา เช่น ต้องการดาวน์โหลดซอฟร์แวร์และจัดการลงซอฟร์แวร์บนเครื่องคอมพิวเตอร์ หากมีเครื่องคอมพิวเตอร์ที่ใช้งานประมาณ 1,000 เครื่อง จะเสียเวลามากในการอัพเดท

ตัวอย่างบริษัทที่ใช้รูปแบบ SaaS ให้บริการมีดังนี้

(1) Salesforce ให้บริการ On demand CRM software ที่ใหญ่ที่สุดในโลก

(2) Netsuite ให้บริการ ERP และ CRM software

(3) Zimbra ให้บริการ Web based groupware ที่มีผู้ใช้บริการกว่า 8 ล้านคน

(4) Zoho ให้บริการ Office suite software เช่น Word, Spreadsheets, Presentation ฯลฯ

(5) Thinkfree ให้บริการ Word, Spreadsheets แบบ online

(6) Office 365 ให้บริการ Microsoft Office ทั้ง Word, Excel, Power Point, Outlook, Onenote และ Access

(7) Google Docs ให้บริการโปรแกรมจัดการเอกสารออนไลน์ซึ่งรองรับรูปแบบไฟล์ได้แก่ DOC, XLS, ODT, RTF, CSV และ PPT

2.2.2 Platform as a Service (PaaS)

เป็นแนวคิดที่ต่อเนื่องมาจาก SaaS เมื่อการใช้งานซอฟต์แวร์ โปรแกรมและแอพพลิเคชั่นไม่ถูกยึดติดอยู่กับเครื่องคอมพิวเตอร์เครื่องใดเครื่องหนึ่งไม่ว่าเป็นเครื่องคอมพิวเตอร์, iPhone หรือ PDA Phone ทำให้เกิดกลุ่มของภาษาที่ใช้ในการพัฒนาโปรแกรมเพื่อรองรับการทำงานบนทุกๆ แพลตฟอร์มได้ ซึ่งเสมือนเป็นเครื่องมือในการพัฒนาโปรแกรม การให้บริการ Platform เพื่อพัฒนาโปรแกรมเว็บแอพพลิเคชั่นและเว็บเซอร์วิส ผู้ใช้สามารถควบคุมการใช้งาน การติดตั้งและการตั้งค่าสำหรับโปรแกรมโดยให้เหมาะสมกับสภาพแวดล้อมของผู้ให้บริการ

โดยองค์ประกอบของ PaaS ต้องประกอบด้วย

ก. การพัฒนาโปรแกรมแอพพลิเคชันภายใต้สภาพแวดล้อมเดียวกัน (Develop, Test, Deploy, Host and Maintain on the Same Integrated Environment) แม้ว่าแนวคิดของ Cloud computing นั้นสามารถให้บริการการใช้งานได้ในทุกๆ แพลตฟอร์มแต่ในความเป็นจริงทางด้านเทคนิคแม้ว่าแอพพลิเคชันโปรแกรมเป็นโปรแกรมเดียวกันก็ตาม แต่ด้วยสภาพแวดล้อมระบบปฏิบัติการ การแสดงผล ต่างกัน ดังนั้นจึงต้องมีการกำหนดแพลตฟอร์มของการใช้งานตามสภาพแวดล้อมด้วยเช่นกัน

ข. เครื่องมือในการสร้างเว็บติดต่อกับผู้ใช้ (Web Based User Interface Creation Tools) นักพัฒนาโปรแกรมควรสามารถพัฒนาโปรแกรมได้โดยไม่จำเป็นต้องเขียนโค้ดหลายหน้า เรียกใช้ไลบรารี่และฟังก์ชั่นเยอะจนไม่สามารถหาได้ว่าฟังก์ชั่นที่ทำการเรียกมาจากที่ใด ซึ่ง Cloud computing จะมีเครื่องมือเหล่านี้ช่วยให้นักพัฒนาสามารถพัฒนาโปรแกรมได้สะดวกเหมือนกับการสร้างเว็บไซต์ในปัจจุบัน

ค. สามารถใช้โปรแกรมได้พร้อมกัน (Multi-Tenant Architecture) ซึ่งต้องสามารถเรียกฟังก์ชั่นได้พร้อมกัน แม้กระทั่งการใช้ฐานข้อมูลเดียวกัน จะสามารถแก้ไขหรือปรับปรุงในเวลาเดียวกันได้

ง. การรวมเข้าด้วยกันกับเว็บเซอร์วิสและฐานข้อมูล (Integration with Web Services and Databases) เนื่องจากรูปแบบโปรแกรมหรือแอพพลิเคชันในปัจจุบันถูกออกแบบมา

ให้ทำงานในรูปแบบของเว็บเซอร์วิสเพื่อรองรับการใช้งานในอนาคตที่สามารถใช้งานจากที่แห่งใดก็ได้โดยไม่จำกัดการใช้งานภายใต้เครื่องคอมพิวเตอร์เพียงอย่างเดียว สำหรับส่วนของการพัฒนาโปรแกรม แหล่งที่มาหรือทรัพยากร (Resource) ต้องสามารถนำทรัพยากรเหล่านี้มาจากแหล่งที่มาได้หลายๆ ที่หรือคือได้มาจากหลายๆ ฐานข้อมูล ดังนั้นจึงจำเป็นต้องรวมเว็บเซอร์วิสและฐานข้อมูลเข้าด้วยกันในการให้บริการแก่ผู้ใช้

จ. สนุนการพัฒนาโปรแกรมร่วมกัน (Support for Development Team Collaboration) ด้วยแนวคิดของการเป็น PaaS ไม่ว่าจะเป็นการใช้ทรัพยากรร่วมกัน (Resource) ตั้งแต่ฐานข้อมูล เซิร์ฟเวอร์ แอพพลิเคชันโปรแกรม (SaaS) และสุดท้ายคือต้องสามารถพัฒนาโปรแกรมร่วมกันได้ ไม่ว่านักพัฒนาจะอยู่ภายใต้สภาพแวดล้อมใด ต้องสามารถทำงานในโครงการเดียวกัน แม้ว่าแต่ละคนมีหน้าที่ต่างกัน ไม่ว่าเป็นโปรแกรมเมอร์หรือนักออกแบบโปรแกรม (Designer Program) จะสามารถแบ่งงาน (Share) ทำได้ภายใต้โครงการเดียวกัน

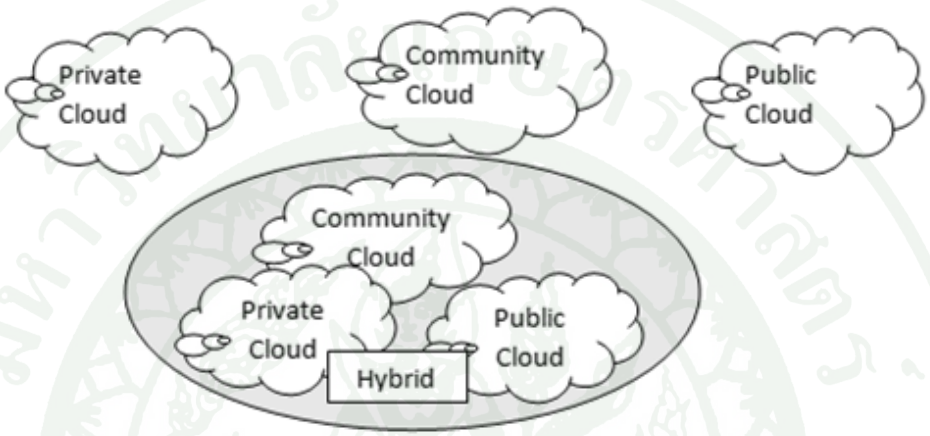
ปัจจุบันมีบริษัทที่ใช้รูปแบบ PaaS ให้บริการเช่น Amazon Web Services, Microsoft Azure Service Platform, Google Apps Engine

2.2.3 Infrastructure as a Service (IaaS)

ความสามารถที่ให้ผู้ใช้มีสิทธิ์ในการกำหนดประสิทธิภาพการประมวลผล การจัดเก็บข้อมูล เครือข่ายและข้อมูลอื่นๆ ในการใช้คอมพิวเตอร์พื้นฐาน ผู้ใช้สามารถที่จะปรับแต่งและใช้ซอฟต์แวร์ได้อย่างไม่มีข้อจำกัด ซึ่งอาจรวมถึงการแก้ไขระบบปฏิบัติการและโปรแกรม ผู้ใช้ไม่จำเป็นต้องดูแลจัดการหรือควบคุมการทำงานของ Cloud ในระดับโครงสร้าง มีเพียงการควบคุมระบบปฏิบัติการ, การจัดเก็บข้อมูลและโปรแกรมประยุกต์ที่ใช้งาน แนวคิดการให้บริการนี้ผู้ใช้บริการไม่ต้องลงทุนในการซื้อเครื่องเซิร์ฟเวอร์ อุปกรณ์จัดเก็บข้อมูลและอุปกรณ์โครงสร้างพื้นฐานด้านเนตเวิร์คและความปลอดภัย เรียกได้ว่าเป็นการเช่าใช้ ซึ่งผู้ที่รับผิดชอบในการบำรุงรักษาคือผู้ให้บริการ โดยที่ผู้ใช้บริการจะเสียค่าบริการอาจเป็นรายวัน รายเดือนหรือปี ทำให้ผู้ใช้บริการมีความคล่องตัวและมีอิสระในการเลือกใช้ Infrastructure และ Platform ต่างๆ ทั้งฮาดร์แวร์และซอฟร์แวร์เพราะทุกอย่างถูกมองเป็นการให้บริการทั้งสิ้น ผู้ที่ให้บริการในปัจจุบัน ได้แก่ Amazon Elastic Compute Cloud (EC2) หรือ GoGrid

2.3 รูปแบบการใช้งานของ Cloud

การแบ่งรูปแบบประเภทของ Cloud โดยพิจารณาจากการนำ Cloud มาให้บริการผู้ใช้บริการบน Cloud สามารถเลือกรูปแบบการให้บริการหรือจัดกลุ่มของ Cloud ขึ้นเองได้ ซึ่งรูปแบบของ Cloud มีดังนี้ (Mell and Grance, 2011)



ภาพที่ 2 แผนภาพแสดงรูปแบบการใช้งานของ Cloud ซึ่งแบ่งตามการให้บริการ

ที่มา: Wikipedia (2014)

2.3.1 Public Clouds

รูปแบบโครงสร้างที่เปิดให้ผู้ใช้งานสาธารณะ สามารถใช้ทรัพยากรที่ถูกจัดหามาให้ได้ ซึ่งโครงสร้างนี้อาจถูกจัดการ ดูแลและเป็นเจ้าของโดยหน่วยงานธุรกิจ หน่วยงานการศึกษา รัฐบาลหรือทั้งหมดดูแลร่วมกัน

2.3.2 Private Clouds

เป็นรูปแบบโครงสร้างที่ทรัพยากรที่ถูกจัดหามาถูกใช้เพียงองค์กรเดียว ซึ่งองค์กรนั้นอาจมีหลายหน่วยงาน (เช่น หน่วยงานธุรกิจ) โครงสร้างนี้ถูกจัดการและดูแลโดยองค์กร, หน่วยงานที่สามหรือของทั้งองค์กรและหน่วยงาน และอาจเปิดให้ภายนอกใช้งานด้วยหรือใช้เพียงแค่ภายใน

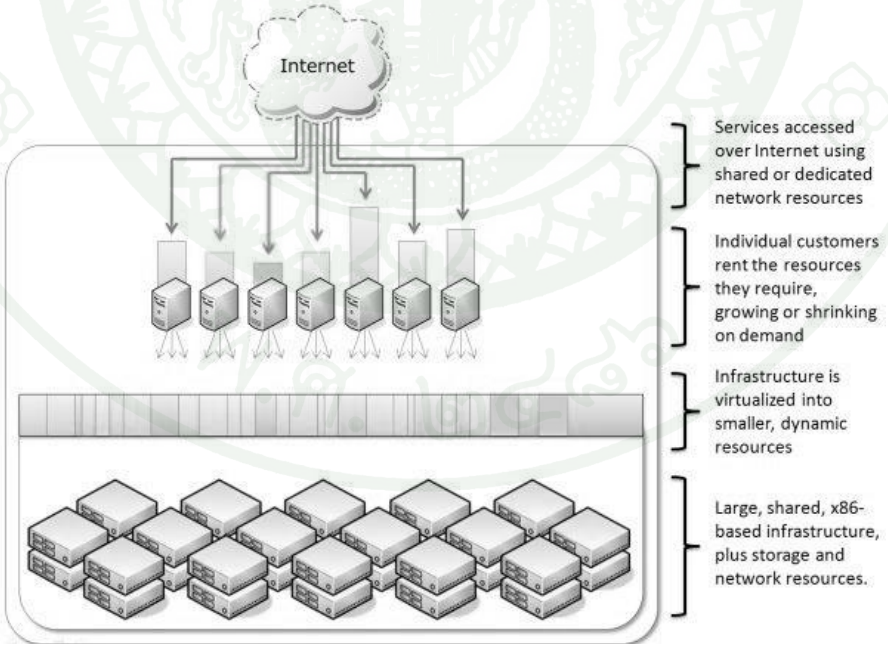
2.3.3 Hybrid Clouds

รูปแบบโครงสร้างที่ประกอบขึ้นจากรูปแบบอื่น ซึ่งรูปแบบที่ประกอบขึ้นนั้นมาจากโครงสร้างแบบ Private, Community หรือ Public และไม่ซ้ำกัน แม้จะเรียกว่าประกอบร่วมกัน แต่ยังคงความเป็นรูปแบบของตัวเองอยู่ ซึ่งการทำงานร่วมกันจะทำตามมาตรฐานที่ตกลงกันไว้หรือทำงานร่วมกันตามเทคโนโลยีที่เปิดให้ใช้ข้อมูลกับแอพพลิเคชั่น

2.3.4 Community Clouds

เป็นรูปแบบโครงสร้างที่ทรัพยากรที่ถูกจัดหามาถูกใช้กับกลุ่มองค์กรซึ่งมีความต้องการร่วมกัน (เช่น ภารกิจ, ความต้องการของระบบความปลอดภัย, นโยบายและการตัดสินใจในทางเดียวกัน) โครงสร้างนี้ถูกจัดการและดูแลโดยกลุ่มองค์กร อาจเป็นหนึ่งองค์กรหรือมากกว่าหนึ่งองค์กรในกลุ่มนั้นหรือ หน่วยงานที่สามหรือทั้งหมดดูแลร่วมกัน และอาจเปิดให้ภายนอกใช้งานหรือใช้เพียงแค่ภายใน

2.4 ส่วนประกอบของ Cloud (Cloud component)



ภาพที่ 3 ภาพรวมการทำงานบน Cloud computing

ที่มา: The Technomax Group (2014)

ถ้าหากมองภาพรวมการใช้บริการบน Cloud computing เมื่อผู้ใช้บริการเข้าใช้งานผ่านทางอินเทอร์เน็ตและไปยังเว็บไซต์ที่ต้องการใช้บริการ เมื่อมีการส่งคำร้องขอเพื่อประมวลผลข้อมูลหรือการขอใช้บริการประเภทต่างๆ ผ่านทางหน้าเว็บเบราว์เซอร์ จะมีเว็บเซอร์วิสคอยรับคำร้องขอเหล่านั้นและส่งคำร้องขอผ่านระบบโครงข่ายเสมือน (Virtualization) กล่าวถึง Virtualization เป็นการแยกการให้บริการต่างๆ ออกจากบริบทของอุปกรณ์การ Hardware ที่ทำหน้าที่สนับสนุนบริการนั้น ยกตัวอย่างเช่น ในระบบ Windows มีการทำหน่วยความจำเสมือน (Virtual Memory) เนื่องจากหน่วยความจำมีขนาดจำกัดและช่วยในการเพิ่มประสิทธิภาพของระบบอีกด้วย โครงข่ายเสมือนจึงถูกมองเสมือนว่าเครื่องเซิร์ฟเวอร์ที่เก็บข้อมูลที่อยู่ตามทุกแห่งทั่วโลกเสมือนถูกกองรวมกันไว้อยู่ ณ ที่แห่งเดียวโดยมี Virtualization Layer ช่วยในการควบคุมการทำงานและจัดสรรทรัพยากรที่มีอยู่ให้แก่คำร้องขอที่เข้ามา ดังนั้นคำร้องขอที่มาจากผู้ใช้บริการอาจถูกแบ่งกระจายไปช่วยกันประมวลผลเช่น ถ้าเราร้องขอภาพถ่ายของเมืองๆ หนึ่งจากดาวเทียม ซึ่งเครื่องเซิร์ฟเวอร์ที่เก็บภาพถ่ายนั้นอาจมีให้บริการอยู่ 4 แห่งทั่วโลก เมื่อคำร้องขอมาถึงคำร้องนั้นจะถูกส่งแบ่งให้แต่ละเครื่องช่วยกันค้นหาภาพในแต่ละส่วนของเมืองแล้วส่งภาพนั้นคืนให้ผู้ใช้บริการ

2.5 คุณลักษณะเบื้องต้นของ Cloud

ลักษณะเบื้องต้นที่เป็นปัจจัยสำคัญของ Cloud มี 5 ลักษณะดังนี้ (Mell and Grance, 2011)

2.5.1 On-demand self-service

ผู้ใช้งานสามารถกำหนดประสิทธิภาพของการประมวลผลได้ด้วยตนเอง เช่น ขนาดของหน่วยความจำในเครือข่าย, ความเร็วและจำนวนของระบบประมวลผล (CPU) หรือสามารถกำหนดความสามารถต่างๆ ให้ยืดหยุ่นโดยอัตโนมัติโดยไม่ต้องมีคนดูแล

2.5.2 Broad network access

ความสามารถในการให้บริการผ่านระบบเครือข่ายและสามารถเข้าถึงได้โดยผ่านมาตรฐานกลางไม่ว่าจะอยู่ในแพลตฟอร์ม (Platform) ใด เช่น โทรศัพท์มือถือ, โน๊ตบุ๊คหรือเครื่องพีซี

2.5.3 Resource pooling

ผู้ให้บริการจะจัดหาทรัพยากรในการประมวลผลเพื่อที่จะให้บริการกับผู้ใช้งานจำนวนมาก โดยใช้รูปแบบการให้เช่าทรัพยากรที่หลากหลาย ตามความต้องการของผู้ใช้ ผู้ใช้สามารถกำหนดขอบเขตของสมรรถนะที่ต้องการ โดยไม่ต้องมีความรู้ด้านการควบคุมบริหารจัดการหรือรู้ที่อยู่ของอุปกรณ์ประมวลผลของผู้ใช้เลย ตัวอย่างของอุปกรณ์ต่างๆ เช่น ฮาร์ดดิสก์, ซีพียู, หน่วยความจำและระบบเครือข่าย

2.5.4 Rapid elasticity

ระบบมีความสามารถของการยืดหยุ่น สามารถตรวจสอบได้และมีความรวดเร็วในการดำเนินงาน ในบางกรณีสามารถทำได้โดยอัตโนมัติเพื่อขยายขนาดการประมวลผลของระบบให้เพียงพอต่อความต้องการ ซึ่งผู้ใช้งานสามารถตกลงกับผู้ให้บริการในเรื่องของการจัดเตรียมสมรรถนะของการประมวลผลว่าเวลาใดควรทำงานอย่างไม่จำกัดและมีการควบคุมปริมาณการประมวลผลในเวลาใด

2.5.5 Measured service

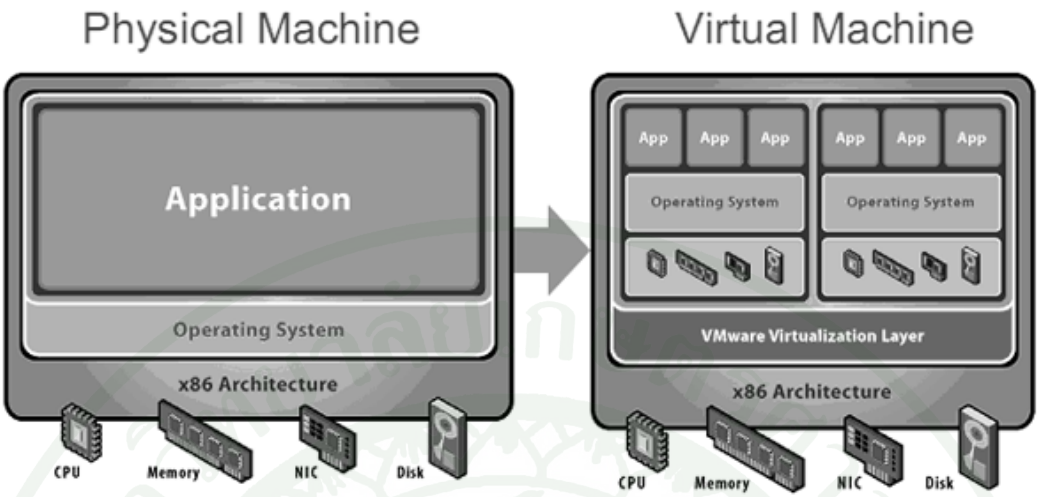
ระบบ Cloud โดยปกติแล้วจะมีการควบคุมและจัดสรรทรัพยากรแบบอัตโนมัติโดยใช้การวัดปริมาณการใช้งานของอุปกรณ์ที่ใช้งานในแต่ละประเภทของการให้บริการ เช่น ฮาร์ดดิสก์, ซีพียู, หน่วยความจำ, ระบบเครือข่ายหรือจำนวนผู้เข้ามาใช้งาน ดังนั้นทรัพยากรต่างๆ เหล่านี้จึงควรที่จะสามารถมีการควบคุมดูแลและสามารถออกรายงานได้เพื่อเป็นส่วนประกอบในการตัดสินใจของผู้ใช้งานในการพิจารณาการเพิ่มหรือลดขนาดของระบบ

งานวิจัยที่เกี่ยวข้อง

1. งานวิจัยที่เกี่ยวกับ Cloud computing

กลุ่มผู้สนใจใน Cloud computing มีหลากหลายกลุ่ม ไม่ว่าจะเป็นนักวิจัย นักลงทุนหรือผู้ดูแลระบบ ในยุคเริ่มต้นของเทคโนโลยี Cloud ในด้านการวิจัยและการพัฒนามุ่งเน้นไปในเรื่องของ Data Storage ซึ่งเป็นส่วนที่สำคัญ ดังนั้นนักพัฒนาจึงพยายามเพิ่มประสิทธิภาพการค้นหาข้อมูลหรือการคิดอัลกอริทึมในการจัดเก็บข้อมูลได้อย่างรวดเร็ว จึงได้มีการนำเทคโนโลยี Virtualization (Menasc´e, 2005) มาใช้

จากภาพที่ 4 จะเห็นว่า ทางด้านซ้ายของภาพซอฟร์แวร์ (คือ Application และ Operating System) และฮาร์ดแวร์ (คือ x86 Architecture) ถูกผูกจองไว้ด้วยกันทำให้เมื่อมีการเรียกใช้งานซอฟร์แวร์มากเกินไปจนระบบปฏิบัติการไม่สามารถรองรับได้อาจเกิดปัญหาทางเสถียรภาพ ส่วนด้านขวาของรูปคือการนำเทคโนโลยี Virtualization มาควบคุมการทำงานของฮาร์ดแวร์บนระบบโดยตรงซึ่งสามารถจำลองเครื่องคอมพิวเตอร์ดังกล่าวได้ทั้งหน่วยประมวลผล หน่วยความจำ แผงวงจรเครือข่าย ฯลฯ เป็นเหมือนเครื่องคอมพิวเตอร์อีกหลายเครื่องโดยแต่ละเครื่องสามารถติดตั้งระบบปฏิบัติการการได้แตกต่างกันภายใต้ Virtual Resource ที่ถูกสร้างขึ้นจาก Virtualization Layer จึงกล่าวได้ว่า Virtualization เป็นการใช้ขีดจำกัดของฮาร์ดแวร์อย่างเต็มประสิทธิภาพ



ภาพที่ 4 รูปแสดงความแตกต่างของเทคโนโลยีทั่วไปกับเทคโนโลยีเสมือน

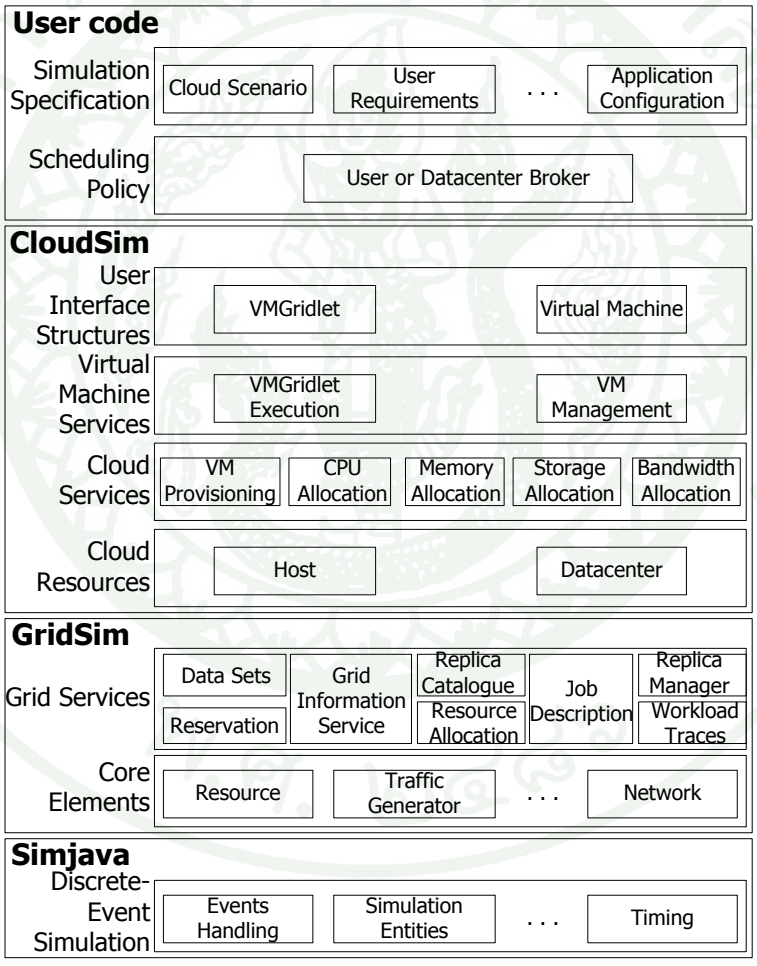
ที่มา: IT-Clever (2010)

2. CloudSim

Cloud computing จากยุคแรกสู่ยุคปัจจุบันนี้ มีกลุ่มของนักวิจัยได้สร้างโปรแกรมจำลองการทำงานของ Cloud computing ออกมาเป็น Open source ให้ผู้สนใจหรือนักวิจัยที่สนใจทดลองดาวน์โหลดมาใช้งาน

จุดประสงค์ของการสร้างโปรแกรมจำลองก็เพื่อรองรับแผนงานวิจัยที่ต้องใช้ทรัพยากรปริมาณมากแต่ไม่มีงบประมาณเพียงพอในการจัดตั้งการทดลอง จึงพัฒนาออกมาในรูปของโปรแกรมจำลองและเป็น Open source ให้นักวิจัยสามารถนำไปพัฒนาต่อยอดได้ โปรแกรมจำลองชื่อ CloudSim (Calheiros *et al*., 2010) ซึ่งถูกเขียนขึ้นและพัฒนาโดยกลุ่ม Grid Computing and Distributed Systems (GRIDS) Laboratory จากมหาวิทยาลัย Melbourne ประเทศ Australia จุดประสงค์ในการพัฒนาแอพพลิเคชั่นนี้เพื่อสนับสนุนนักวิจัยและนักพัฒนาสามารถทุ่มเทการคิดค้นเพื่อการออกแบบที่นักวิจัยเหล่านั้นต้องการพิสูจน์ข้อสมมติฐานที่ตั้งเป้าหมายไว้โดยไม่จำเป็นต้องสนใจถึงรายละเอียดการทำงานระดับล่างของ Cloud ที่เกี่ยวข้องกับ Infrastructures และ

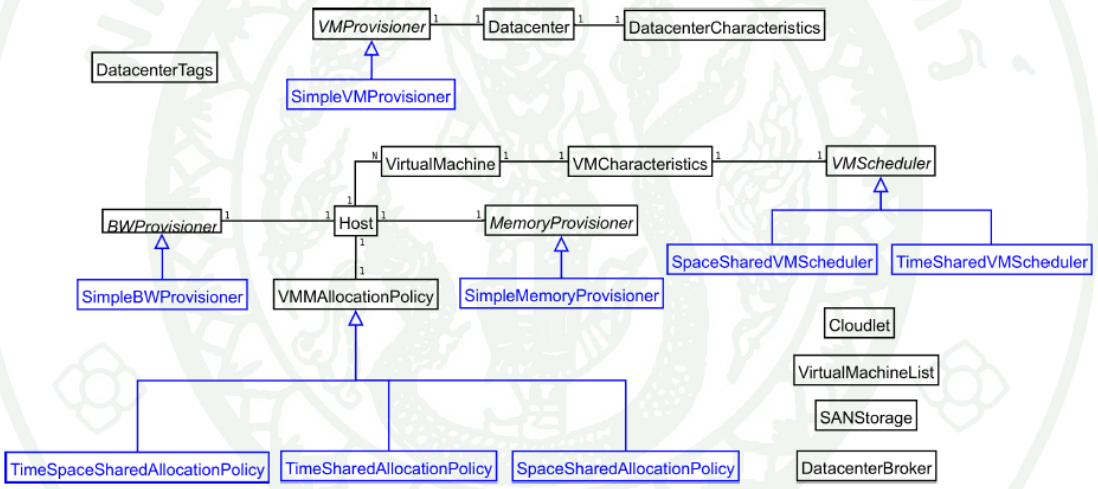
การให้บริการ อีกทั้งการนำเอาแนวคิดที่เราตั้งสมมติฐานไว้แล้วนำไปทดลองลงที่ระบบจริงของ Cloud นั้นเป็นเรื่องที่ทำได้ยาก เหตุเพราะระบบ Infrastructures ของการใช้งาน Cloud จริงจะมีจุดสิ้นสุดหรือจุดที่กำหนดไว้ว่าสามารถขยายฐานของ Infrastructures ได้เท่าไหร่ ซึ่งการกำหนดนี้จะขึ้นกับผู้ใช้บริการว่าตกลงกับผู้ให้บริการอย่างไร ทำให้การทดลองอาจได้ผลลัพธ์ที่ไม่หลากหลาย และนอกจากนี้ในการทดลองเพื่อทดสอบหลายๆ สมมติฐานในระบบจริงยากแก่การดำเนินงานมาก



ภาพที่ 5 ลำดับชั้นของสถาปัตยกรรมของ CloudSim

ที่มา: Calheiros *et al*. (2010)

แอพพลิเคชั่น CloudSim นี้สามารถสร้างระบบเสมือน (Virtual machines or VMs) ได้มากกว่า 1 ตัว ทั้งยังสามารถสร้างและจำลองศูนย์กลางการเก็บข้อมูลได้มากมายและยังสามารถกำหนดนโยบายเกี่ยวกับการย้าย VMs และทำ Auto Scaling ได้อีกด้วย ตัวแอพพลิเคชั่นถูกพัฒนาโดยภาษา Java และทางทีมพัฒนาได้แบ่งปัน Source code ให้แก่บุคคลที่สนใจสามารถนำไปพัฒนาต่อยอดได้เอง ค่าการกำหนด (Configuration) ต่างๆ ในโปรแกรมสามารถกำหนดได้เองทั้งหมด ยกตัวอย่างเช่น สามารถกำหนดรายละเอียดของ VMs ได้เองตั้งแต่ระดับเลือกว่ามี CPU ใช้กี่ Core, ใช้ Ram เท่าไหร่, MIPs ของการทำงาน CPU เป็นแบบไหน, เลือกระบบปฏิบัติการที่ใช้บน VMs ตัวนั้น, เลือกการทำงาน Scheduling ของระบบปฏิบัติการ เป็นต้น



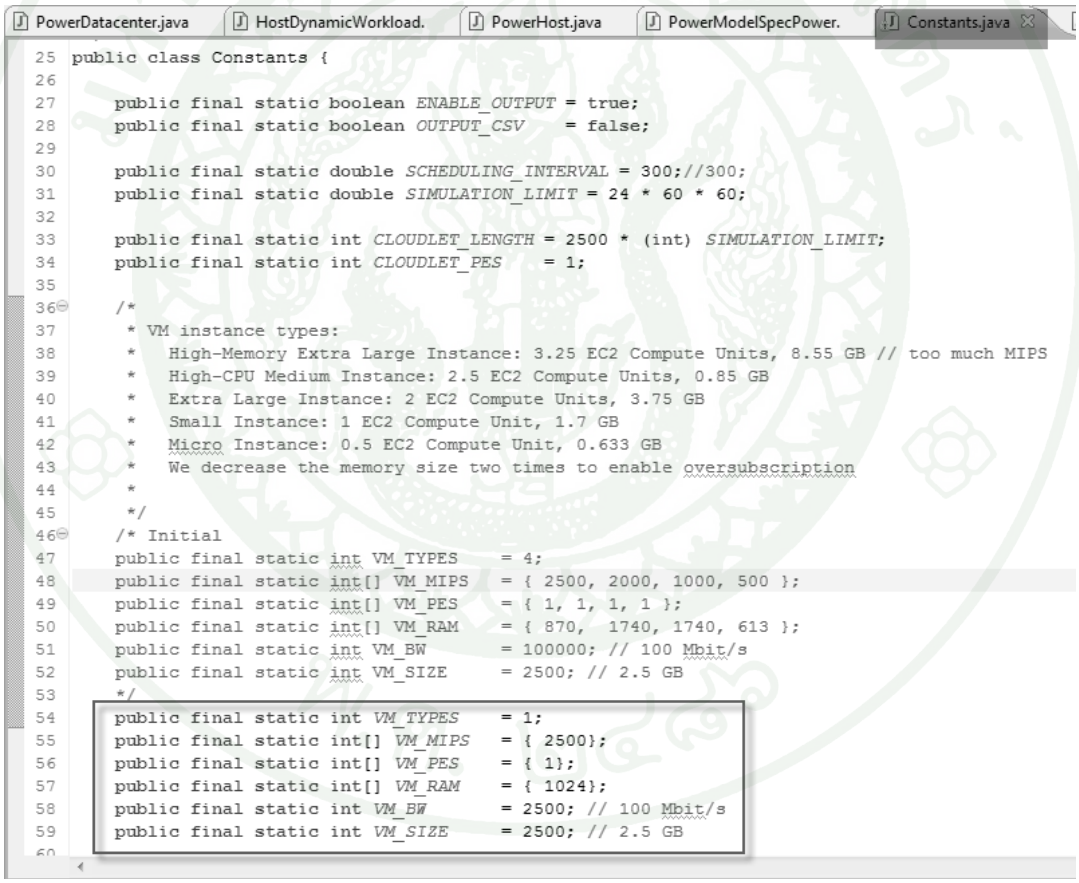
ภาพที่ 6 แสดงแผนภาพการออกแบบ Class ของ CloudSim

ที่มา: Calheiros *et al.* (2010)

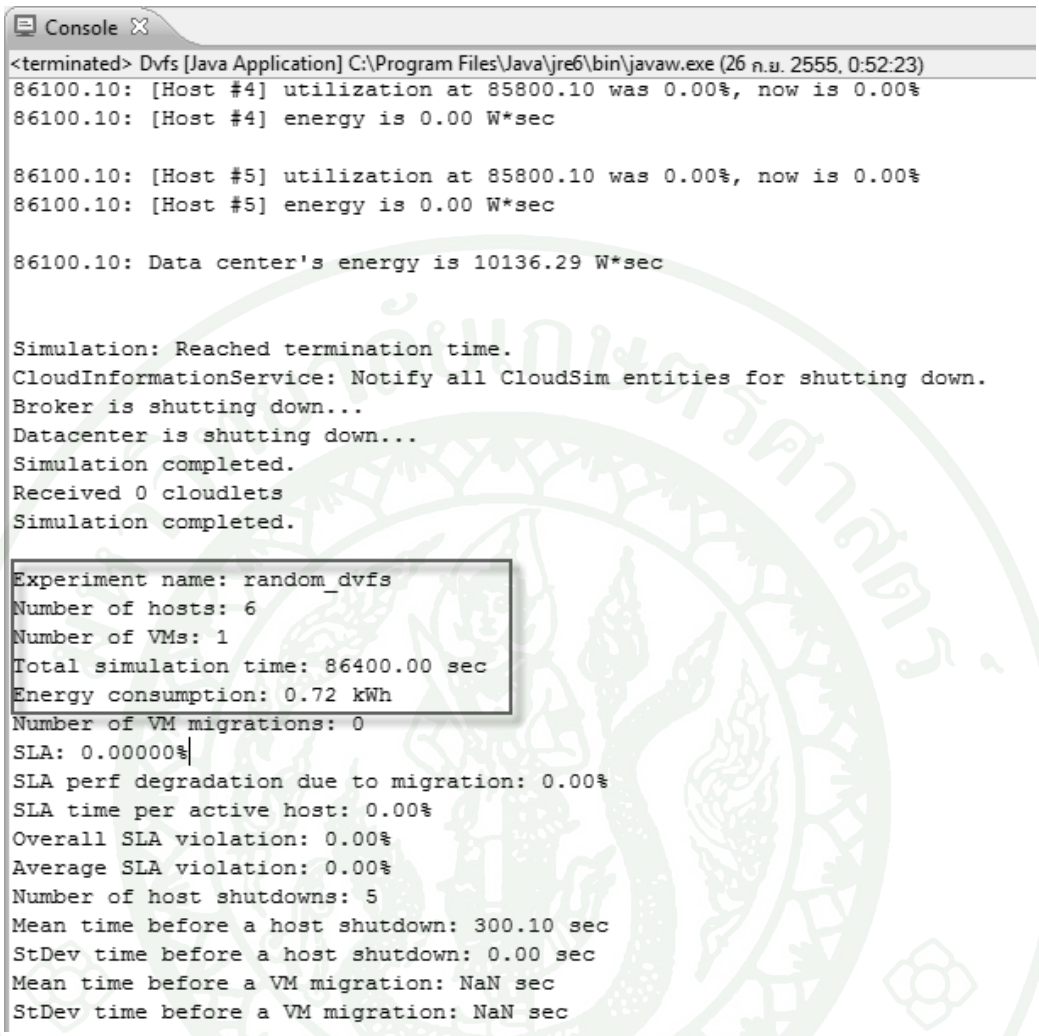
ผู้พัฒนา CloudSim ได้ออกแบบ Class การทำงานของแอพพลิเคชั่นดังภาพที่ 6 จากภาพกรอบสีฟ้าคือ Class ที่สืบทอดหรือเป็น Class ลูกจาก Class แม่ที่เป็นกรอบสีดำ ซึ่งจากภาพแสดงการออกแบบ Class ทางผู้พัฒนาหรือนักวิจัยที่ดาวน์โหลด Source code มาสามารถสร้าง Class ใหม่ขึ้นเองเพื่อการทำวิจัยโดยอาจสืบทอด Class ที่สร้างขึ้นเองนั่นจาก Class แม่ที่มีอยู่ ซึ่งทางผู้พัฒนา

CloudSim ได้สร้าง Class หลักๆ ไว้แล้วเช่น Data Center, VMs, Networking, Memory และ Class เสมือนที่เอาไว้กำหนดนโยบายทั้งในด้าน Bandwidth, Memory, VM เป็นต้น

แม้ว่าการกำหนดค่าที่ใช้ในการทดลองยังไม่มีไฟล์แยกสำหรับเก็บค่ากำหนดต่างๆ เอาไว้ ซึ่งอาจทำให้ลำบากในช่วงที่เราต้องทดลองกับข้อมูลที่มีการเปลี่ยนแปลงค่าบ่อยๆ แต่เราสามารถปรับเปลี่ยนหรือปรับปรุง Code ที่ทาง CloudSim ได้ให้มาจากการเปลี่ยนค่าโดยตรงซึ่งต้องเข้าไปเปลี่ยนใน Code โปรแกรมให้มาเป็นไฟล์ที่เก็บค่ากำหนดต่างๆ ไว้ได้เช่นกัน



ภาพที่ 7 ตัวอย่างโปรแกรมแสดงรายละเอียดการกำหนดค่า Virtual machine 1 ตัว

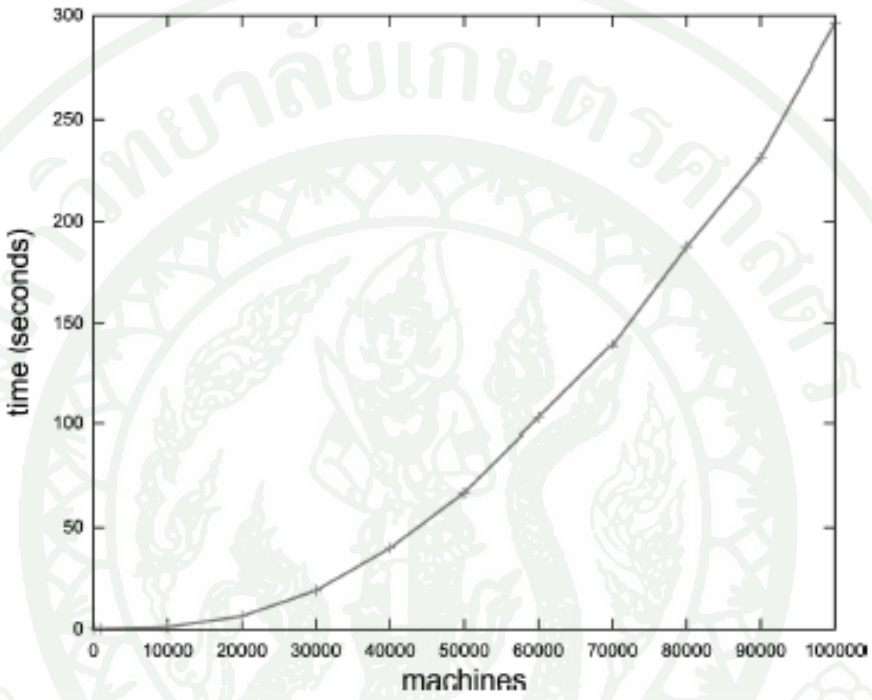


ภาพที่ 8 แสดงผลลัพธ์ของการรันโปรแกรม CloudSim

ปัจจุบัน CloudSim ได้ออกมาถึงเวอร์ชัน 3.03 เมื่อเดือนพฤษภาคม พ.ศ.2556 สามารถดาวน์โหลดโปรแกรมได้ที่ http://www.cloudbus.org/cloudsim/

อีกหนึ่งบทความที่เกี่ยวข้องกับโปรแกรมจำลอง CloudSim คือการทดลองเพื่อทดสอบประสิทธิภาพ (Buyya *et al*., 2010) ของตัวโปรแกรมว่ามีความน่าเชื่อถือหรือไม่ โดยสิ่งที่บทความนี้สนใจคือโอเวอร์เฮด (Overhead) ในการสร้างโฮสต์ที่มีจำนวนมากว่ากินระยะเวลาการสร้างมากเท่าไหร่และหน่วยความจำที่ถูกใช้ไปในการสร้างโฮสต์มีแนวโน้มเป็นอย่างไร

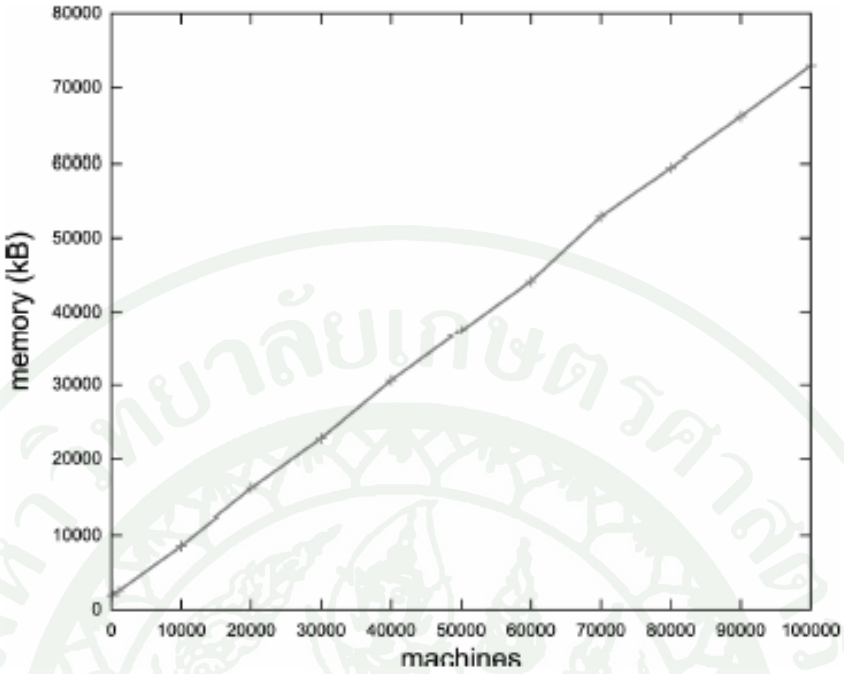
การทดลองได้ทดลองบน CPU Celeron 1.86 GHz, L2 cache 1MB แรม 1GB ทำงานด้วยระบบปฏิบัติการ Ubuntu Linux version 8.04 ทดลองโดยจำลองการสร้างโฮสต์ตั้งแต่ 100 – 100,000 ตัว



ภาพที่ 9 กราฟแสดงเวลาที่ใช้และจำนวนโฮสต์ที่ถูกสร้างขึ้นจากโปรแกรม CloudSim

ที่มา: Buyya *et al*. (2010)

จากกราฟพบว่าเวลาที่ใช้ในการสร้างโฮสต์ 100,000 ตัวใช้เวลาประมาณ 3 นาที และมีการแปรผกผันตรงของเวลากับจำนวนโฮสต์ที่สร้างแบบ Exponential



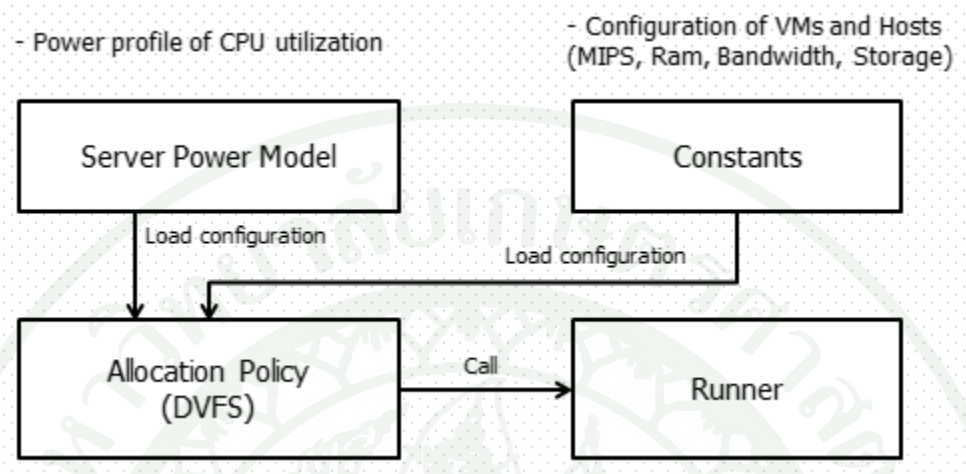
ภาพที่ 10 กราฟแสดงหน่วยความจำที่ใช้และจำนวนโฮสต์ที่ถูกสร้างขึ้นจากโปรแกรม CloudSim

ที่มา: Buyya *et al*. (2010)

ในด้านของหน่วยความจำที่เพิ่มขึ้นกับจำนวนโฮสต์ที่ถูกสร้างขึ้นนั้น ขึ้นแปรผกผันตรงแบบกราฟเส้นตรงและหน่วยความจำที่ใช้ไปประมาณ 75 MB จากผลการทดลองทั้ง 2 กราฟทำให้แน่ใจได้ว่าโปรแกรม CloudSim สามารถนำไปติดตั้งและทำงานบน Personal computer หรือ Notebook ได้ ไม่จำเป็นต้องติดตั้งบนเครื่องเซิร์ฟเวอร์ที่มีประสิทธิภาพสูง

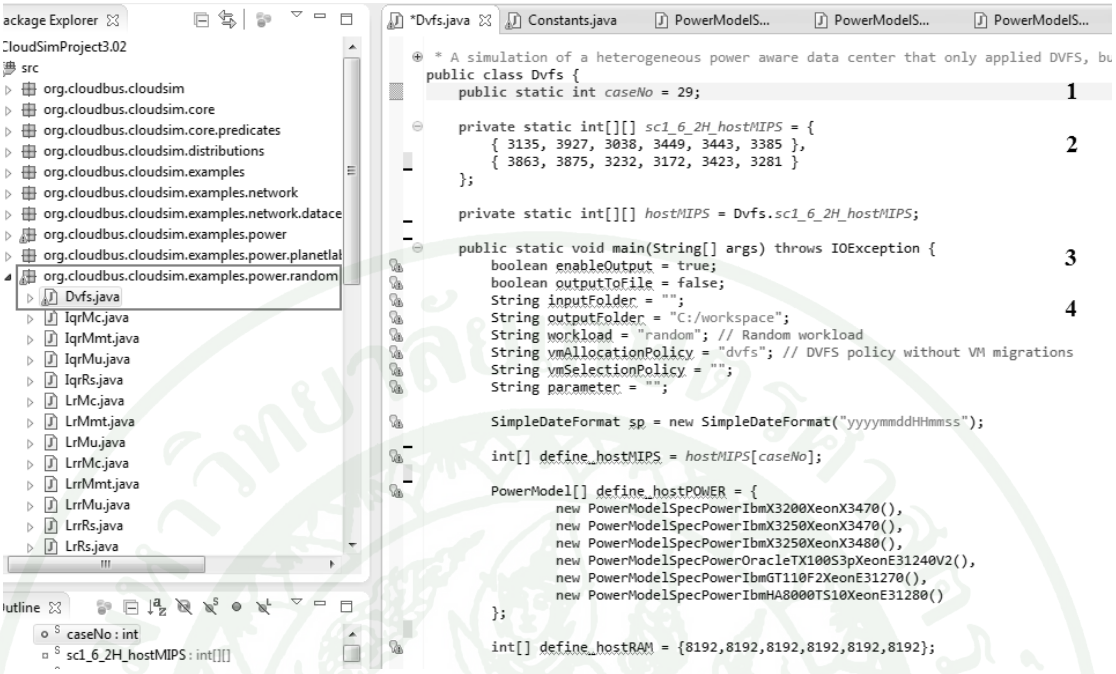
หลายงานวิจัยมีการนำโปรแกรมจำลอง CloudSim ไปใช้เพื่อทดสอบสมมุติฐานที่ตั้งไว้ เช่นงานวิจัยของ Shi *et al*. (2011) และ Li *et al*. (2013) เป็นต้น

2.1 การปรับแก้โปรแกรม CloudSim เพื่อการทดลอง



ภาพที่ 11 ภาพขั้นตอนการทำงานของ CloudSim ว่าเกี่ยวข้องกับฟังก์ชั่นในไฟล์ใดบ้าง

ใน CloudSim มี algorithm สำหรับการเลือกเครื่องเพื่อนำมาประมวลผล ซึ่งจะทำงานเลือกเครื่องเรียงตามลำดับว่าเครื่องใดถูกสร้างขึ้นในโปรแกรมก่อน ดังนั้นเราต้องเขียนปรับแก้การเลือกเครื่องให้เลือกตาม algorithm 2 แบบที่เราได้คิดขึ้น ในการทดลองนี้ผู้วิจัยใช้โมเดลการใช้พลังงานของโปรแกรม CloudSim แบบ DVFS (Dynamic Voltage and Frequency Scaling)



ภาพที่ 12 ไฟล์ DVFS.java ที่ปรับแก้สำหรับใช้ในการทดลอง

ปรับแก้ไฟล์ DVFS.java โดยมีส่วนหลักที่สำคัญตามหมายเลขที่ปรากฏในภาพที่ 12 ดังนี้

(1) หมายเลข 1

ตัวแปร caseNo เป็นตัวแปรที่กำหนดรอบที่ของการทดลอง ซึ่งทำการทดลองทั้งหมด 40 รอบ

(2) หมายเลข 2

ตัวแปร hostMIPS เป็นตัวแปรที่ไว้กำหนดค่า MIPS ของ Host แต่ละตัว

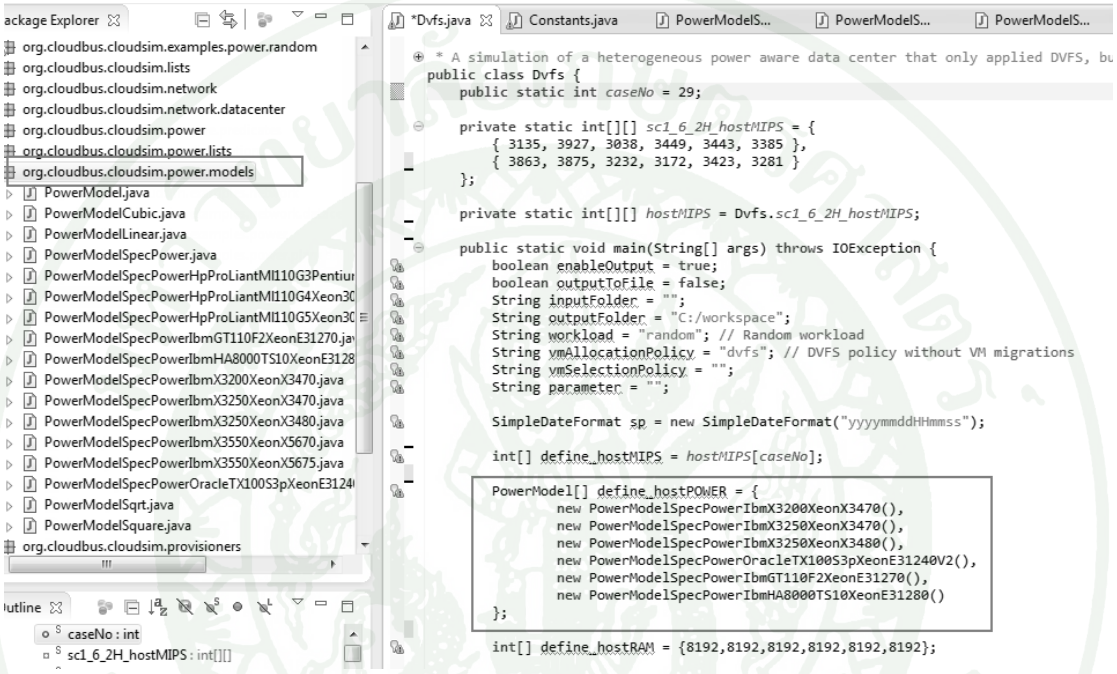
(3) หมายเลข 3

ตัวแปร enableOutput เป็นตัวแปรที่กำหนดว่าต้องการเขียนผลลัพธ์ออกเป็นไฟล์หรือไม่

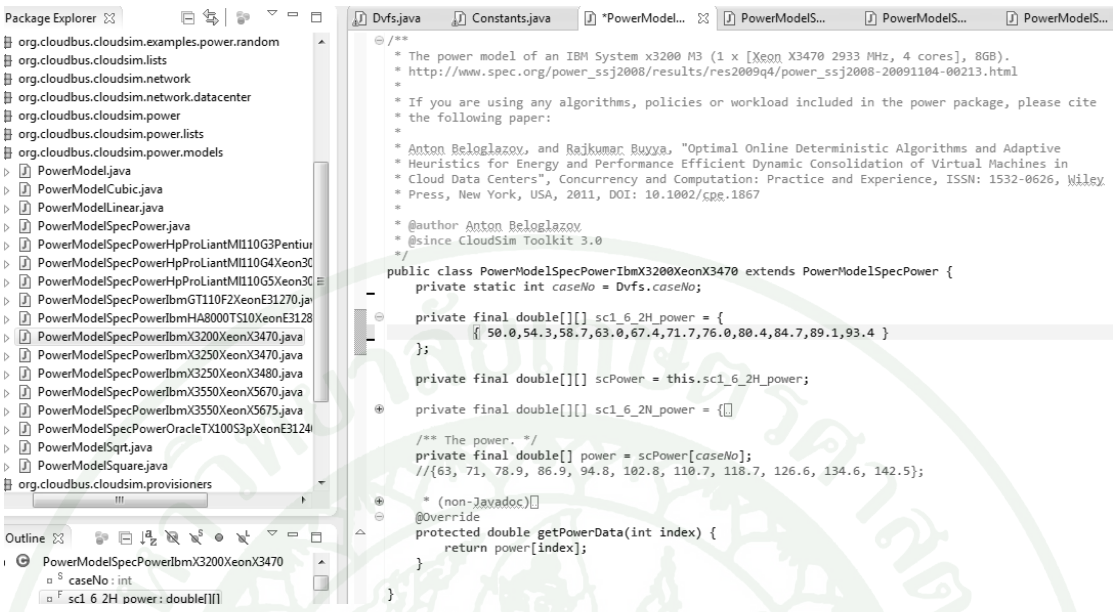
(4) หมายเลข 4

ตัวแปร outputFolder เป็นตัวแปรกำหนดที่อยู่ในการสร้างไฟล์ผลลัพธ์ หากตัวแปร enableOutput กำหนดค่าเป็น True ไฟล์ผลลัพธ์จะถูกสร้างตามที่อยู่นี้

เราสามารถสร้างโมเดลพลังงานของ Host ได้เองโดยเข้าไปสร้างไว้ที่ Package org.cloudbus.cloudsim.power.models นั่นคือเราสามารถอ้างอิง Server profile จาก SPEC เพื่อนำมาใช้ในการทดลองได้

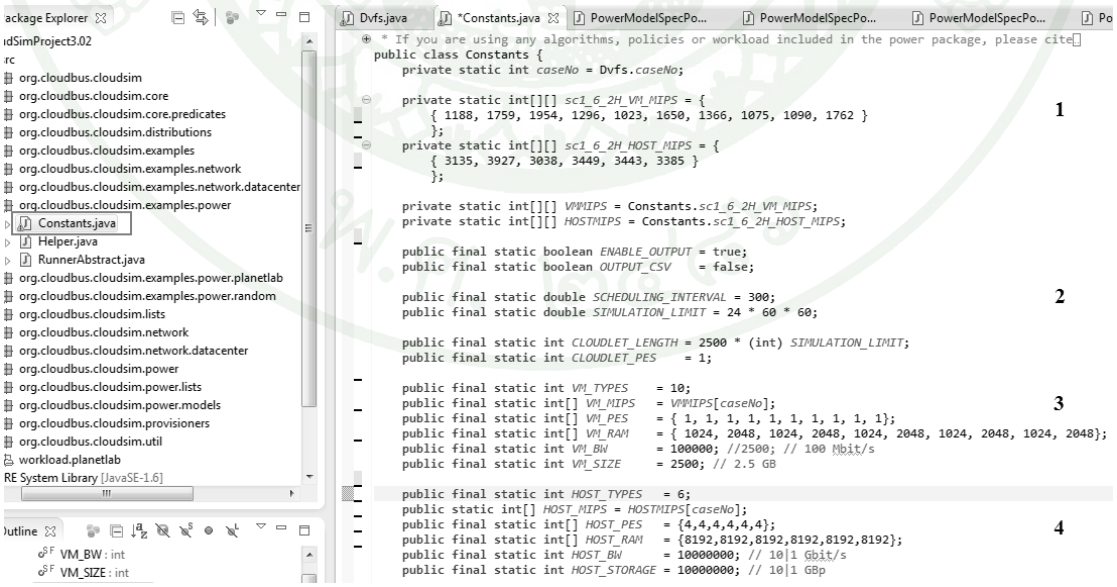


ภาพที่ 13 ส่วนกำหนดโมเดลพลังงานของ Host



ภาพที่ 14 การกำหนดค่าพลังงานที่ Host ใช้ในการประมวลผลตั้งแต่ 0% – 100% CPU utilization

ในการกำหนดค่า Configuration ของ VM และ Host ให้เข้าไปแก้ไขที่ไฟล์ Constants.java ตามภาพที่ 15 โดยส่วนสำคัญๆ ที่ปรับแก้ไขมีดังนี้



ภาพที่ 15 ไฟล์ Constants.java ที่กำหนดปรับค่า Configuration ของ Host และ VM

(1) หมายเลข 1

ตัวแปร VM_MIPS เป็นตัวแปรกำหนดค่า MIPS ให้แก่ VM แต่ละตัว ในการทดลองนี้ใช้ VM ทั้งหมด 10 ตัว

(2) หมายเลข 2

ตัวแปร SCHEDULING_INTERVAL เป็นตัวแปรที่กำหนดค่าเวลา Interval ให้แก่ Simulation time ซึ่งเมื่อถึงเวลา Interval โปรแกรม CloudSim จะมีการคำนวณพลังงานที่แต่ VM ใช้บน Host นั้นๆ มีหน่วยเป็น millisecond ของ Simulation time อีกหนึ่งค่าคือ SIMULATION_LIMIT เป็นตัวแปรที่กำหนดค่า Simulation time มีหน่วยเป็น millisecond

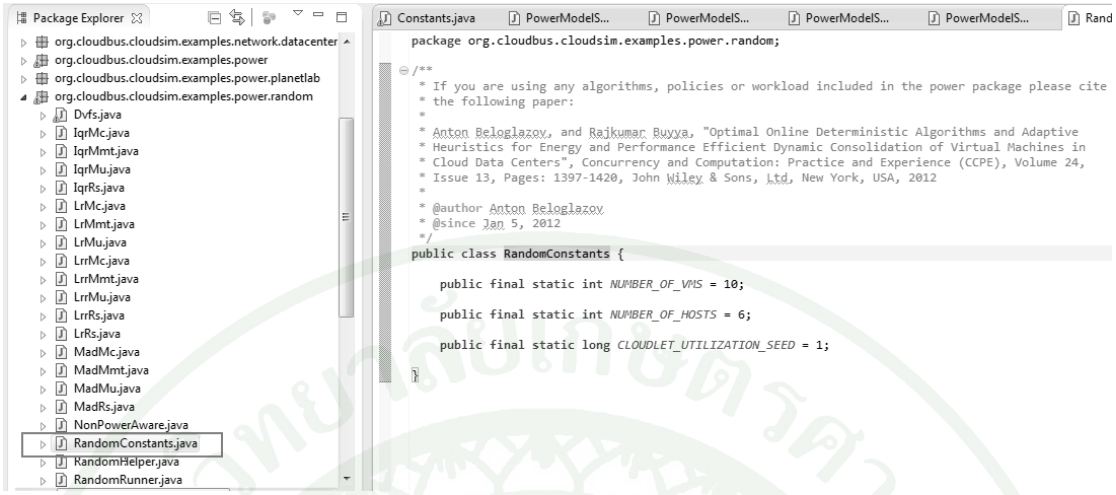
(3) หมายเลข 3

ชุดตัวแปรกำหนด Configuration ของ VM ได้แก่ VM_TYPES ชนิดของ VM, VM_MIPS ค่า MIPS, VM_PES จำนวน CPU ใน VM, VM_RAM, VM_BW และ VM_SIZE ความจุของ VM

(4) หมายเลข 4

ชุดตัวแปรกำหนด Configuration ของ Host ได้แก่ HOST_TYPES ชนิดของ Host, HOST_MIPS ค่า MIPS, HOST_PES จำนวน CPU ใน Host, HOST_RAM, HOST_BW และ HOST_STORAGE ความจุของ Host

ส่วนการกำหนดจำนวน Host และจำนวน VM ทั้งหมดที่ใช้บน Cloud เรากำหนดที่ RandomConstants.java



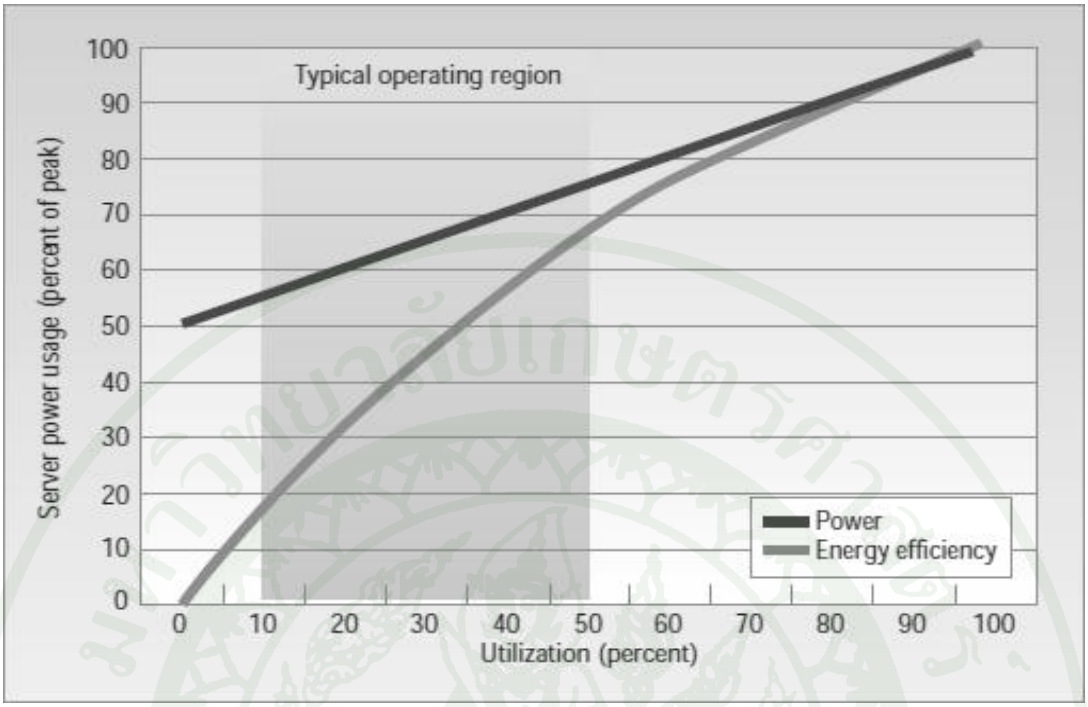
ภาพที่ 16 ไฟล์ RandomConstants.java ที่ปรับแก้จำนวน VM และ Host ที่ใช้บน Cloud

3. ประสิทธิภาพการประหยัดพลังงาน (Energy Efficiency)

อีกหนึ่งหัวข้อที่น่าศึกษาเพื่อนำมาประยุกต์ใช้ในการพัฒนา Cloud คือเรื่องของความคุ้มค่าในการประหยัดพลังงาน (Energy efficiency) (Barroso and Hölzle, 2007) ได้พูดเกี่ยวกับสัดส่วนของพลังงานที่ใช้ในการประมวลผล ว่ามีส่วนสำคัญในการประหยัดพลังงานให้กับเซิร์ฟเวอร์

ในบทความทำการสำรวจการใช้พลังงานระหว่างเซิร์ฟเวอร์กับคอมพิวเตอร์ส่วนบุคคล พบว่าเซิร์ฟเวอร์มีการทำงานอยู่ตลอดเวลา แทบจะไม่มีช่วงหยุดการทำงานเลย ถึงแม้ว่าบางช่วงของเวลามีงานเข้ามาเพื่อร้องขอในการประมวลผลจำนวนน้อย เซิร์ฟเวอร์ก็ยังคงเปิดทำงานอยู่ ไม่ได้ปิดแต่เพียงอย่างใด

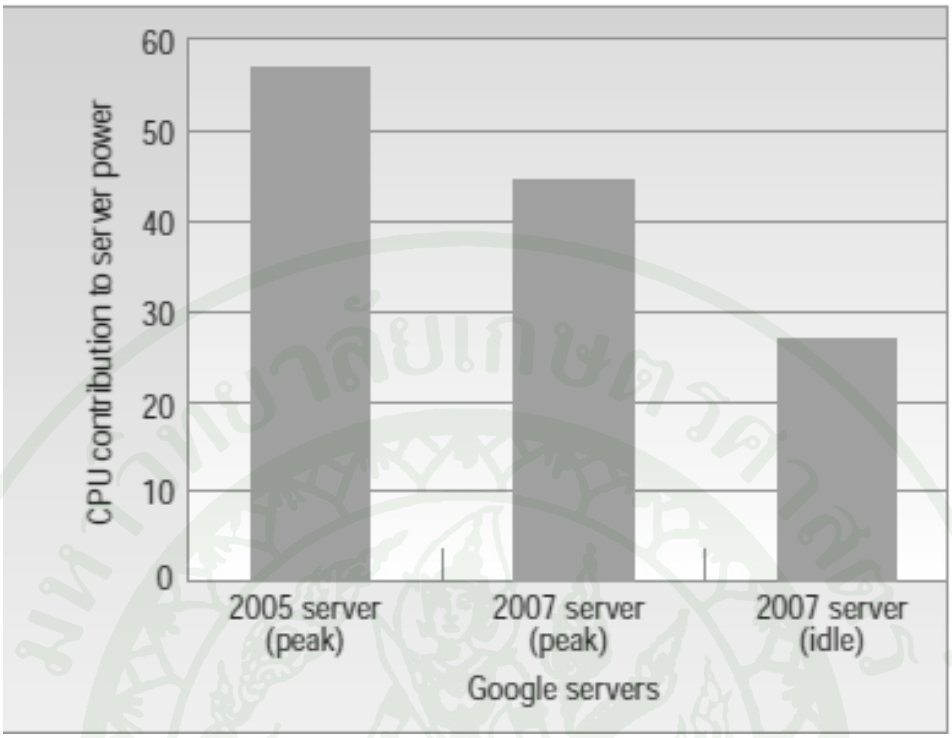
ดังนั้นเซิร์ฟเวอร์จึงเป็นอุปกรณ์หลักในการนำพลังงานมาใช้เป็นจำนวนมาก ผู้เขียนบทความจึงทำการสำรวจการใช้พลังงานของเซิร์ฟเวอร์กับ Utilization ของ CPU ว่ามีความสัมพันธ์กับพลังงานที่ถูกใช้ไปอย่างไร



ภาพที่ 17 กราฟแสดงความสัมพันธ์ระหว่างการใช้พลังงานของเซิร์ฟเวอร์และความคุ้มค่าของพลังงาน

ที่มา: Barroso and Hölzle (2007)

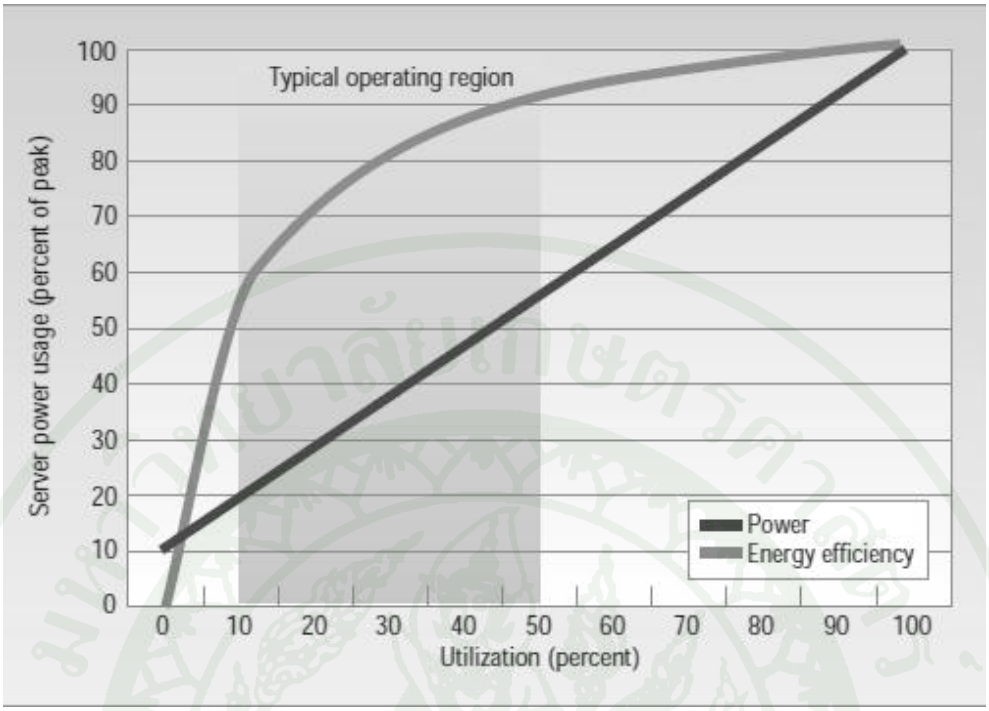
จากกราฟเมื่อ Utilization เป็น 0 ค่าพลังงานได้สูงถึง 50 เปอร์เซ็นต์ นั้นแสดงถึง แม้ว่าไม่มีงานที่ต้องประมวลผลแต่ก็ถูกใช้พลังงานไปแล้วส่วนหนึ่ง ที่เป็นเช่นนี้เพราะเมื่อเริ่มต้นเปิดเครื่องจะมีพลังงานส่วนหนึ่งถูกใช้ไปในการเปิดจนถึงเครื่องเซิร์ฟเวอร์พร้อมใช้ทำงาน ซึ่งแต่ละเซิร์ฟเวอร์ที่ถูกออกแบบมาของแต่ละบริษัท มีค่าการกินพลังงานแตกต่างกัน ตัวอย่างที่ผู้เขียนบทความนำมาอ้างอิงคือ เครื่องเซิร์ฟเวอร์รุ่นแรกของ Google



ภาพที่ 18 กราฟแสดงการใช้พลังงานของเครื่องเซิร์ฟเวอร์แต่ละรุ่นของ Google

ที่มา: Barroso and Hölzle (2007)

เครื่องเซิร์ฟเวอร์แต่ละรุ่นของ Google มีอัตราการใช้พลังงานที่ต่างกัน ยิ่งเซิร์ฟเวอร์รุ่นหลังๆ จะมีอัตราการใช้พลังงานน้อยลง นั้นแสดงให้เห็นว่าเซิร์ฟเวอร์รุ่นใหม่ๆ มีการจัดการการใช้พลังงานของเซิร์ฟเวอร์ดีขึ้นกว่ารุ่นเก่า



ภาพที่ 19 กราฟแสดงการใช้พลังงานของเครื่องเซิร์ฟเวอร์รุ่นใหม่

ที่มา: Barroso and Hölzle (2007)

บริษัทที่ผลิตเครื่องเซิร์ฟเวอร์รุ่นใหม่ๆ ได้คำนึงถึงเรื่องการกินพลังงานและพัฒนาปรับปรุงให้มีการกินพลังงานที่ลดลง เช่น จังหวะตอนเปิดเครื่องครั้งแรกจะเห็นได้จากภาพที่ 19 ว่าใช้พลังงานลดลงอย่างมาก นั้นแสดงว่ารุ่นของเครื่องเซิร์ฟเวอร์ที่ใช้ประมวลผลบน Cloud มีผลกับความมากน้อยของพลังงานที่ถูกใช้ไป และทำให้มีผลกับค่าใช้จ่ายโดยรวมที่ต้องเสียไปอีกด้วย

ในระบบ Cloud เป็นแหล่งรวมทรัพยากร เช่น เครื่องเซิร์ฟเวอร์, หน่วยความจำ, หน่วยประมวลผล, พื้นที่เก็บข้อมูล ฯลฯ ที่ไว้ใช้สำหรับการประมวลผลเมื่อมีการร้องขอ วิธีการเลือกว่าควรจะให้ทรัพยากรใดแก่การร้องขอครั้งนั้น มักดูแค่ว่าทรัพยากรที่เหลืออยู่เพียงพอกับการร้องขอนั้นหรือไม่ ไม่ได้คำนึงถึงประสิทธิภาพการใช้พลังงานของเครื่องเซิร์ฟเวอร์ ว่ามีส่วนช่วยในการประหยัดพลังงาน และมีบางงานวิจัยของ Xiao *et al*. (2013) และ Roy *et al*. (2011) ที่ไม่สนใจว่าค่าพลังงาน ณ idle state มีผลต่อการประหยัดพลังงาน แต่ไปเน้นที่วิธีการสร้าง VM ขึ้นบนโฮสต์ โดย

พยายามรวม VM ให้สร้างบนโฮสต์ให้เต็มทีละเครื่องไป และปิดเครื่องที่ไม่ได้ใช้งานลง ซึ่งจะทำให้ได้จำนวนโฮสต์ที่เปิดใช้งานมีจำนวนน้อยที่สุดแทน

ดังนั้นหากนำแนวคิดเรื่องประสิทธิภาพการใช้พลังงานของเครื่องเซิร์ฟเวอร์แต่ละรุ่น ซึ่งมีผลกับการประหยัดพลังงานมาประยุกต์ใช้ด้วย จะช่วยเพิ่มประสิทธิภาพของอัลกอริทึมในการประหยัดพลังงานของ Cloud อีกทั้งลดค่าใช้จ่ายให้แก่ทั้งผู้ให้บริการและผู้รับบริการอีกด้วย ในงานวิจัยนี้ทำการทดลองโดยใช้โปรแกรมจำลอง CloudSim ซึ่งสามารถแก้ไขอัลกอริทึมหลักของ CloudSim เปลี่ยนมาใช้อัลกอริทึมที่คิดค้นขึ้นได้ แล้วลองทดสอบเพื่อดูความคุ้มค่าของการประหยัดพลังงานที่เกิดขึ้น เมื่อเซิร์ฟเวอร์มีการประมวลผลงานที่ร้องขอเข้ามา

ในเรื่องการคำนวณและคิดค่าพลังงาน ผู้พัฒนาโปรแกรมจำลอง CloudSim ได้นำวิธีการประมาณค่าการใช้พลังงานของเครื่องเซิร์ฟเวอร์จากค่าของ CPU utilization ณ ค่าต่างๆ กัน ซึ่งอ้างอิงถึงบทความของ Beloglazov and Buyya (2011) ที่เสนออัลกอริทึมการคำนวณหาค่าพลังงานที่เหมาะสมที่สุดสำหรับเซิร์ฟเวอร์ในสถานะต่างๆ ผู้เขียนได้แบ่งการตัดสินใจเมื่อเกิดปัญหาต้องย้ายหรือปิดโฮสต์เมื่อไม่สามารถสร้าง VM บนโฮสต์ที่เกิดปัญหาได้ ซึ่งปัญหาแบ่งออกเป็น 4 ข้อคือ

(1) เมื่อโฮสต์ตัดสินใจต้องการย้าย VM เนื่องจาก VM มีการประมวลผลที่เกินขีดจำกัด (Overloaded) ที่โฮสต์สามารถรองรับได้

(2) เมื่อโฮสต์ตัดสินใจต้องการย้าย VM ทั้งหมดและโฮสต์เข้าสู่สถานะหลับ (sleep mode) เนื่องจากมีการประมวลผลที่น้อย

(3) ทำการเลือก VM จากโฮสต์ที่ VM ตัวนั้นทำให้โฮสต์เกิด Overloaded

(4) การตัดสินใจเพื่อหาพื้นที่ว่างแห่งใหม่ให้แก่ VM ที่ถูกย้ายออกมากจากโฮสต์ Overloaded หรือ Underloaded

อัลกอริทึมจะอิงปัญหา 4 ข้อข้างต้นเป็นหลักในการจัดการ allocate VM ให้กับโฮสต์ที่เหมาะสม ซึ่งในขั้นตอนก่อนจะ allocate VM ให้กับโฮสต์จะมีอัลกอริทึม Power Aware Best Fit Decreasing (PABFD) (Beloglazov and Buyya, 2011) ที่ดัดแปลงมาจากอัลกอริทึม Best Fit Decreasing (BFD) การทำงานของอัลกอริทึมนี้จะเรียงค่า CPU utilization ของ VM ทั้งหมดที่มีค่าในขณะนั้นจากน้อยไปมาก (Decresing) และนำ VM แต่ละตัวไปประมาณหาค่าพลังงานเมื่อทำงาน

บนโฮสต์แต่ละตัว จากนั้นเลือกโฮสต์ตัวที่เมื่อ VM เข้าไปทำงานแล้วได้พลังงานต่ำที่สุด allocate ให้แก่ VM ตัวนั้น

Algorithm 2: Power Aware Best Fit Decreasing (PABFD)

```
1  Input: hostList, vmList   Output: allocation of VMs
2  vmList.sortDecreasingUtilization()
3  foreach vm in vmList do
4      minPower ← MAX
5      allocatedHost ← NULL
6      foreach host in hostList do
7          if host has enough resources for vm then
8              power ← estimatePower(host, vm)
9              if power < minPower then
10                 allocatedHost ← host
11                 minPower ← power
12     if allocatedHost ≠ NULL then
13         allocation.add(vm, allocatedHost)
14 return allocation
```

ภาพที่ 20 อัลกอริทึม Power Aware Best Fit Decresing (PABFD)

ที่มา: Beloglazov and Buyya (2011)

การคำนวณค่าพลังงานที่ใช้ทั้งหมดในโปรแกรมจำลอง CloudSim ใช้สูตรการคำนวณดังสมการ (1) ซึ่งได้ผลลัพธ์ออกมาในหน่วยของ กิโลวัตต์ต่อชั่วโมง (kW/h)

$$\text{Energy} = \frac{\sum \text{Power of Datacenter}}{3600 \times 1000} \left(\text{kW/h}\right) \tag{1}$$

โดยที่

Energy คือพลังงานที่ใช้ทั้งหมดในหน่วยกิโลวัตต์ต่อชั่วโมง

$\sum$Power of Datacenter คือผลรวมพลังงานที่ระบบประมวลผลแบบกลุ่มเมฆใช้ประมวลผลในหน่วยวัตต์

ซึ่งค่าพลังงานที่ระบบประมวลผลแบบกลุ่มเมฆใช้ประมวลผล เกิดจากผลรวมพลังงานแต่ละโฮสต์ ดังสมการ (2) โดยที่แต่ละโฮสต์คิดค่าพลังงานได้จากสูตร ดังสมการ (3) หน่วยของผลลัพธ์คือ วัตต์คูณวินาที (W*sec)

$$\text{Power of Datacenter} = \sum \text{Power of Host} \quad (\text{W*sec}) \tag{2}$$

โดยที่

Power of Datacenter คือผลรวมพลังงานของโฮสต์ในหน่วยวัตต์คูณวินาที

$$\text{Power of Host} = \left(\text{fromPower} + \frac{(\text{toPower} - \text{fromPower})}{2} \right) \text{x time} \quad (\text{W*sec}) \tag{3}$$

โดยที่

Power of Host คือค่าพลังงานที่แต่ละโฮสต์ใช้ในการประมวลผลในหน่วยวัตต์คูณวินาที (W*sec)

fromPower คือค่าพลังงานของการประมวลผลในรอบที่ผ่านมา

toPower คือค่าพลังงานของการประมวลผลในรอบปัจจุบัน

time คือผลต่างของเวลาที่เริ่มประมวลผลและเวลาที่สิ้นสุดการประมวลผล

ค่าพลังงานของการประมวลผลจะคำนวณจากการนำค่า Utilization ของโฮสต์ที่ใช้ในครั้งนั้นๆ ไปเทียบกับกราฟการใช้พลังงานซึ่งกราฟได้จากเกณฑ์มาตรฐานของ Standard Performance Evaluation Corporation (SPEC) และใช้สูตรการหาพลังงานจาก Beloglazov and Buyya (2011) ได้สูตรดังสมการ (4) ในหน่วยวัตต์ (Watt)

$$\text{Power} = \text{power1} + \text{delta x} \left(\text{u} - \frac{\text{u1}}{10} \right) \text{x } 100 \quad (\text{Watt}) \tag{4}$$

โดยที่

Power คือค่าพลังงานที่โฮสต์ใช้ในการประมวลผลในหน่วยวัตต์

power1 คือค่าของพลังงานที่นำตัวแปร u1 ไปเทียบกับกราฟของ SPEC

power2 คือค่าของพลังงานที่นำตัวแปร u2 ไปเทียบกับกราฟของ SPEC

delta คือค่าเฉลี่ยในการเพิ่ม Utilization ทีละหนึ่งหน่วย โดยนำค่า (power2 – power1)/10

u คือค่า Utilization ของโฮสต์

u1 คือค่าที่นำ Utilization มาปัดเศษทิ้งแล้วคูณด้วยสิบ

u2 คือค่าที่นำ Utilization มาปัดเศษขึ้นแล้วคูณด้วยสิบ

ตัวอย่างการคำนวณหาค่าพลังงานที่แต่ละโฮสต์ใช้ในการประมวลผลในหน่วยวัตต์คูณวินาที ต้องเริ่มจากการหาค่าพลังงานของโฮสต์ที่ใช้ในหน่วยวัตต์ก่อน กำหนดให้ Utilization ของโฮสต์ ณ ขณะนั้นมีค่าเป็น 0.3187 คำนวณพลังงานตามสมการที่ (4) ดังนี้

u = 0.3187

u1 = 3

u2 = 4

power1 = 96.0

power2 = 99.5

delta = (99.5 – 96.0) / 10

จะได้ค่าพลังงาน

Power = 96.0 + 0.35 x (0.3187 – 3/10) x 100

= 96.0 + 0.6545

= ***96.6545*** Watt

เมื่อได้ค่าพลังงานของโฮสต์ในหน่วยวัตต์เรียบร้อยแล้ว นำค่าที่ได้คำนวณตามสมการที่ (3) เพื่อให้ได้ค่าพลังงานของแต่ละโฮสต์ที่ใช้ในหน่วยวัตต์คูณวินาที กำหนดให้โฮสต์ ณ ขณะนั้นมีค่าดังนี้

fromPower = 96.6545

toPower = 95.1568

time (sec) = 300

จะได้ค่าพลังงาน

Power of Host = (96.6545 + (95.1568 – 96.6545)/2) x 300
= (96.6545 + (-0.74885)) x 300
= ***28771.695*** W*sec

นอกจากนี้ งานวิจัยเกี่ยวกับอัลกอริทึมในด้านการประหยัดพลังงานมีงานนำเสนอหลายรูปแบบ อย่างบทความของ Quan *et al*. (2012) ที่พยายามลดพลังงานโดยการปิดเครื่องโฮสต์ที่ไม่ได้ใช้ ซึ่งอัลกอริทึมแบ่งการทำงานเป็น 2 ช่วงคือ

- ช่วงแรกจะทำงานโดยย้าย VM ที่ทำงานอยู่บนโฮสต์ที่มี load ต่ำ ย้ายไปรวมกับโฮสต์ที่มี load สูงเพื่อปิดเครื่องโฮสต์ที่ไม่ได้ใช้
- ช่วงที่สองจะย้าย VM ไปทำงานบนเครื่องรุ่นใหม่แทนเครื่องรุ่นเก่า

Quan *et al*. (2012) ได้ทดลองบน Cloud ที่มีสภาพแวดล้อมของเครื่องโฮสต์ทั้งเครื่องรุ่นเก่าและรุ่นใหม่ ซึ่งจากการศึกษาเครื่องรุ่นใหม่พบว่ามีประสิทธิภาพในการประหยัดพลังงานดีกว่าเครื่องรุ่นเก่า ดังนั้นอัลกอริทึมในช่วงที่สอง จึงทำการย้าย VM ไปทำงานบนเครื่องรุ่นใหม่แทนเครื่องรุ่นเก่า

การตั้งการทดลองของ Quan *et al*. (2012) ได้แบ่งเซิร์ฟเวอร์ออกเป็น 4 แบบตามจำนวน cores ของ CPU ซึ่งแบ่งเป็น single core, dule cores, quad cores และ six cores ส่วน scenarios ของทรัพยากรที่ทำการทดลองแบ่งออกเป็น 3 scenarios ดังนี้

- modern data centre เป็นกลุ่มทรัพยากรที่มีประเภทเซิร์ฟเวอร์จำนวน core ของ CPU หลาย core มากกว่าเซิร์ฟเวอร์แบบอื่นๆ
- normal data centre เป็นกลุ่มทรัพยากรที่มีประเภทเซิร์ฟเวอร์คละกันไป

- old data centre เป็นกลุ่มทรัพยากรที่มีประเภทเซิร์ฟเวอร์จำนวน core ของ CPU น้อยมีมากกว่าเซิร์เวอร์แบบอื่นๆ

Scenario	1	2	3
Round robin(MWt)	61.23	48.45	41.89
Round robin + F4G-CG (MWt)	47.64	40.83	37.21
Greedy (MWt)	50.74	40.35	35.78
Greedy + F4G-CG (MWt)	45.67	37.78	33.47
Load balance (MWt)	60.46	47.87	41.15
Load balance + F4G-CG (MWt)	46.56	39.67	36.85
F4G-CS (MWt)	46.58	36.13	32.18
F4G-CS + F4G-CG (MWt)	44.76	34.47	30.7

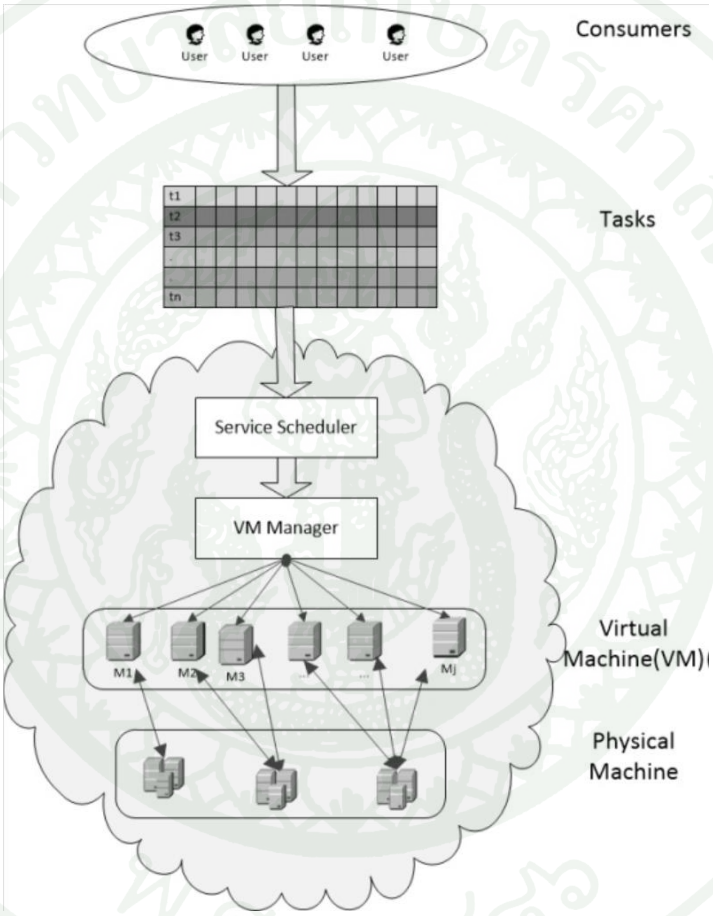
ภาพที่ 21 แสดงผลการทดลองของประสิทธิภาพการประหยัดพลังงานทั้ง 3 scenarios

ที่มา: Quan *et al*. (2012)

จากผลการทดลองที่ได้ ปรากฏว่าประสิทธิภาพการประหยัดพลังงานของ scenario ที่หนึ่งมีค่าสูงกว่าทุก scenario ซึ่ง scenario ที่หนึ่งเป็นการทดลองบนทรัพยากรที่มีเครื่องเซิร์ฟเวอร์ส่วนใหญ่มีจำนวน CPU หลาย core ซึ่งตรงกับที่ Quan *et al*. (2012) ได้กล่าวไว้ว่าเครื่องเซิร์ฟเวอร์แบบใหม่มีประสิทธิภาพการประหยัดพลังงานที่ดีกว่า ดังนั้นผลที่ได้จาก scenario ที่หนึ่งจึงดีกว่าทุก scenario

บทความของ Quan *et al*. (2012) ได้ทดลองบนสภาพแวดล้อมที่เหมือนกับงานวิจัยนี้ คือ ทดลองบนระบบ Cloud ที่มีทั้งเครื่องเซิร์ฟเวอร์รุ่นเก่าและเครื่องเซิร์ฟเวอร์รุ่นใหม่ แต่การทำงานของอัลกอริทึมจะแตกต่างกัน ในบทความพยายามย้าย VM ไปทำงานบนเครื่องรุ่นใหม่ทั้งหมด ในขณะที่งานวิจัยนี้อัลกอริทึมจะดูค่าพลังงานในการประมวลผลของเครื่องเซิร์ฟเวอร์เป็นสำคัญ ว่าเซิร์ฟเวอร์ที่ควรหยิบมาทำงานควรเป็นเซิร์ฟเวอร์ที่เมื่อทำงานงานนั้นแล้ว ได้ค่าพลังงานออกมาน้อย ไม่ได้เลือกจากประเภทว่าเครื่องซิร์ฟเวอร์เป็นเครื่องรุ่นเก่าหรือเครื่องรุ่นใหม่

อีกบทความของ Kumar and Sahoo (2014) ที่พยายามลดจำนวนเครื่องโฮสต์ในการเปิดใช้งานโดยพิจารณาจากงาน (Task) ที่เข้ามาบน Cloud ในบทความนี้อัลกอริทึมจะทำงานโดยพิจารณาจาก task ที่เข้ามาทำงานบน Cloud โดยนำ task มาเก็บไว้ในตาราง matrix ที่เก็บค่าระหว่างเวลาและ VM ว่า task ใดต้องทำงานเวลาใดบนเครื่องใด อัลกอริทึมที่ผู้เขียนบทความคิดค้นมี 3 แบบคือ



ภาพที่ 22 แผนภาพการทำงานของอัลกอริทึม

ที่มา: Kumar and Sahoo (2014)

3.1 FCFS to Random Utilized (FCFSRandomUtil)

จะจัด task ตามการเข้ามาก่อนหลัง (first come fisrt serve หรือ FCFS) และนำไป allocate บน VM แบบสุ่ม ซึ่งใช้วิธีการสุ่มแบบ uniform distribution งานที่สร้างบน VM จะถือว่า

สร้างสำเร็จเมื่อ งานนั้นไปสร้างบน VM แล้วมี threshold ของ utilization ไม่เกิน 100% ตัวอย่างเช่น การจัดงาน 20 งานบน VM จำนวน 10 ตัว ตามภาพที่ 23

Time/VM	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
1	-	[2]	[1]	-	-	[5]	[4]	[3]	-	-
2	[11]	[2]	[1]	[10]	[8]	[5]	[4]	[3]	[6]	-
3	[11,15]	[2]	[1]	[10]	[8]	[5]	[4]	[3]	[6]	-
4	[11,15]	[2]	[1]	[10,20]	[8,12]	[5]	[4,13]	[3,7]	[6]	[14]
5	[11,15]	[2]	[1]	[10,20]	[8,12]	[5]	[4,13]	[3,7]	[6]	[14]
6	[11,15]	-	[1]	[10,20]	[8,12]	[5]	[4,13]	[3,7]	[6]	[14]
7	[11,15]	[16]	[1]	[10,20]	[8,12]	[5]	[4,13]	[3,7]	[6]	[14]
8	[11,15]	[16]	[1]	[10,20]	[8,12,19]	[5]	[4,13]	[7]	[6]	[14]
9	[11,15]	[16]	[1]	[10,20]	[8,12,19]	[5]	[4,13]	[7]	[6]	[14]
10	[11]	[16]	[1]	[10,20]	[12,19]	-	[4,13]	[7]	[9]	[14]
11	[11]	[16]	[1]	[10,20]	[12,19]	-	[4,13]	[7]	[9]	[14]
12	[11]	[16]	[1]	[20]	[12,19]	-	[4,13]	[7]	[9]	[14]
13	[11]	[16]	-	[20]	[12,19]	[18]	[13]	[7]	[9]	[17]
14	[11]	[16]	-	[20]	[12]	[18]	[13]	[7]	[9]	[17]
15	[11]	[16]	-	[20]	[12]	[18]	[13]	-	[9]	[17]
16	[11]	[16]	-	[20]	[12]	[18]	[13]	-	[9]	[17]
17	[11]	[16]	-	[20]	[12]	[18]	-	-	[9]	[17]
18	[11]	[16]	-	-	[12]	[18]	-	-	[9]	[17]
19	-	-	-	-	[12]	[18]	-	-	[9]	[17]
20	-	-	-	-	[12]	[18]	-	-	-	[17]
21	-	-	-	-	-	[18]	-	-	-	[17]
22	-	-	-	-	-	[18]	-	-	-	[17]
23	-	-	-	-	-	[18]	-	-	-	[17]
24	-	-	-	-	-	-	-	-	-	[17]

ภาพที่ 23 ตาราง matrix ของการจัด task แบบ FCFSRandomUtil

ที่มา: Kumar and Sahoo (2014)

3.2 FCFS to Maximum Utilized (FCFSMaxUtil)

จะจัด task ตามการเข้ามาก่อนหลังและนำไป allocate บน VM ที่เมื่อจัด task นั้นแล้วทำให้มีค่าใกล้เคียง 100% CPU utilization ตัวอย่างเช่น การจัดงาน 20 งานบน VM จำนวน 10 ตัว ตามภาพที่ 24

Time/VM	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
1	[1,3]	[2]	[4]	[5]	-	-	-	-	-	-
2	[1,3]	[2]	[4]	[5]	[6,8]	[7]	[9]	[10]	[11]	-
3	[1,3,15]	[2]	[4,12]	[5]	[6,8]	[7,13]	[9]	[10]	[11]	[14]
4	[1,3,15]	[2]	[4,12]	[5]	[6,8]	[7,13]	[9]	[10,20]	[11,19]	[14]
5	[1,3,15]	[2]	[4,12]	[5]	[6,8]	[7,13]	[9]	[10,20]	[11,19]	[14]
6	[1,3,15]	[16]	[4,12]	[5]	[6,8]	[7,13]	[9]	[10,20]	[11,19]	[14]
7	[1,3,15]	[16]	[4,12]	[5]	[6,8]	[7,13]	[9]	[10,20]	[11,19]	[14]
8	[1,15]	[16]	[4,12]	[5]	[6,8]	[7,13]	[9]	[10,20]	[11,19]	[14]
9	[1,15]	[16]	[4,12]	[5]	[6,8]	[7,13]	[9]	[10,20]	[11,19]	[14]
10	[1]	[16]	[4,12]	[17]	[18]	[7,13]	[9]	[10,20]	[11]	[14]
11	[1]	[16]	[4,12]	[17]	[18]	[7,13]	[9]	[10,20]	[11]	[14]
12	[1]	[16]	[4,12]	[17]	[18]	[7,13]	-	[20]	[11]	-
13	-	[16]	[12]	[17]	[18]	[13]	-	[20]	[11]	-
14	-	[16]	[12]	[17]	[18]	[13]	-	[20]	[11]	-
15	-	[16]	[12]	[17]	[18]	[13]	-	[20]	[11]	-
16	-	[16]	[12]	[17]	[18]	-	-	[20]	[11]	-
17	-	[16]	[12]	[17]	[18]	-	-	[20]	[11]	-
18	-	-	[12]	[17]	[18]	-	-	-	[11]	-
19	-	-	[12]	[17]	[18]	-	-	-	-	-
20	-	-	-	[17]	[18]	-	-	-	-	-
21	-	-	-	[17]	-	-	-	-	-	-

ภาพที่ 24 ตาราง matrix ของการจัด task แบบ FCFSMaxUtil

ที่มา: Kumar and Sahoo (2014)

3.3 Maximum to Maximum Utilized (MaxMaxUtil)

จะจัด task ที่ต้องการการใช้ CPU utilization สูงสุดไป allocate บน VM ที่มีค่า CPU utilization สูงที่เพียงพอให้ task นั้นทำงานได้ ซึ่งการทำงานของอัลกอริทึมนี้จะค้นหา task ที่ต้องการการใช้ CPU utilization สูงที่สุดจาก task queue แล้วหา VM ที่มีค่า utilization สูงขณะนั้นที่เมื่อ task ข้างต้น allocate แล้วมีค่า CPU utilization ไม่เกิน 100% ตัวอย่างเช่น ภาพที่ 25 การจัดงาน 20 งานบน VM จำนวน 10 ตัว

Time/VM	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
1	[5]	[2]	[1,3]	[4]	-	-	-	-	-	-
2	[5]	[2]	[1,3]	[4]	[11]	[10,8]	[6]	[7]	[9]	-
3	[5]	[2]	[1,3,15]	[4,12]	[11]	[10,8]	[6]	[7,13]	[9]	[14]
4	[5]	[2]	[1,3,15]	[4,12]	[11,19]	[10,8]	[6]	[7,13]	[9,20]	[14]
5	[5]	[2]	[1,3,15]	[4,12]	[11,19]	[10,8]	[6]	[7,13]	[9,20]	[14]
6	[5]	[17]	[1,3,15]	[4,12]	[11,19]	[10,8]	[6]	[7,13]	[9,20]	[14]
7	[5]	[17]	[1,3,15]	[4,12]	[11,19]	[10,8]	[6]	[7,13]	[9,20]	[14]
8	[5]	[17]	[1,15]	[4,12]	[11,19]	[10,8]	[6]	[7,13]	[9,20]	[14]
9	[5]	[17]	[1,15]	[4,12]	[11,19]	[10,8]	[6]	[7,13]	[9,20]	[14]
10	[16]	[17]	[1]	[4,12]	[11]	[10]	[18]	[7,13]	[9,20]	[14]
11	[16]	[17]	[1]	[4,12]	[11]	[10]	[18]	[7,13]	[9,20]	[14]
12	[16]	[17]	[1]	[4,12]	[11]	-	[18]	[7,13]	[20]	-
13	[16]	[17]	-	[12]	[11]	-	[18]	[13]	[20]	-
14	[16]	[17]	-	[12]	[11]	-	[18]	[13]	[20]	-
15	[16]	[17]	-	[12]	[11]	-	[18]	[13]	[20]	-
16	[16]	[17]	-	[12]	[11]	-	[18]	-	[20]	-
17	[16]	[17]	-	[12]	[11]	-	[18]	-	[20]	-
18	[16]	-	-	[12]	[11]	-	[18]	-	-	-
19	[16]	-	-	[12]	-	-	[18]	-	-	-
20	[16]	-	-	-	-	-	[18]	-	-	-
21	[16]	-	-	-	-	-	-	-	-	-

ภาพที่ 25 ตาราง matrix ของการจัด task แบบ MaxMaxUtil

ที่มา: Kumar and Sahoo (2014)

ผลการทดลองปรากฏว่าการทำงานแบบ Maximum to Maximum Utilized ได้ประสิทธิภาพการประหยัดพลังงานดีที่สุด เพราะการแบ่ง task และจัดเรียงให้ทำงานเต็มประสิทธิภาพของเครื่อง VM เป็นเครื่องๆ ไปจะช่วยลดจำนวนเครื่องการทำงานลงเป็นผลให้ประสิทธิภาพของการใช้พลังงานโดยรวมลดลง

ซึ่งอัลกอริทึมในการจัดเรียง task ที่ต้องการ CPU utilization สูงไป allocate บนเครื่อง VM ที่มี CPU utilization สูงเช่นกัน เสมือนพยายามทำให้ VM ทำงานเต็มประสิทธิภาพ 100% ของ CPU utilization จะคล้ายกับการทำงานในการเรียงลำดับของอัลกอริทึมในงานวิจัยนี้ แต่ต่างที่ในงานวิจัยจะเรียงลำดับตามค่าพลังงานจากน้อยไปมากของการใช้พลังงาน ณ 100% CPU utilization

อีกแนวคิดการประหยัดพลังงานคือการกำหนด Policy ให้แก่งาน (Task) ที่เข้ามา ในบทความของ Luo *et al*. (2012) ได้คิดค้นอัลกอริทึมโดยพิจารณาจาก task ที่เข้ามาทำงานบน Cloud ว่าเป็น task ประเภทใด จากนั้นแบ่งแยก task แต่ละอย่างไปตามแต่ละ Policy ได้แก่ CPU, Memory, Storage และ Network

Type of Job	Dependence	Type of policy	Policy template
Compute intensive	★★★	Network Policy+ Storage Policy	Slow down network, turn off the extra module Reduce the disk speed, memory status change
Storage intensive	★★★	CPU Policy	DVFS，VOVO
I/O intensive	★★★	CPU Policy	DVFS，VOVO
Compute Storage Intensive	★★ ★	Network Policy	Slow down network, turn off the extra module
Compute I/O Intensive	★★ ★	Storage Policy	Reduce the disk speed, memory status change
Storage I/O Intensive	★★ ★	CPU Policy	DVFS，VOVO

ภาพที่ 26 ตารางตัวอย่างการแบ่ง task ตามแต่ละประเภท Policy

ที่มา: Luo *et al*. (2012)

ในบทความทำการทดลอง 2 ครั้งคือครั้งแรกจัด task แบบไม่มี Policy และครั้งที่สองแบบมี Policy จากนั้นนำมาเปรียบเทียบกัน ผลปรากฏว่าการจัด task แบบมี Policy ทำให้ประสิทธิภาพการใช้พลังงานโดยรวมลดลง เพราะเกิดจากการแบ่ง task ให้ทำงานบนเครื่องที่มีประสิทธิภาพตรงตามความต้องการในการประมวลผลจึงส่งผลให้ได้ประสิทธิภาพของการใช้พลังงานโดยรวมดี

ซึ่งในงานวิจัยนี้ไม่ได้แบ่งงานตาม Policy และไม่ได้แบ่ง task แต่ทดลองโดยการประมวลผลงานที่รับเข้ามาเป็นช่วงเวลาแทน แล้วใช้การเรียงลำดับพลังงานที่เครื่องโฮสต์ใช้ในการตัดสินใจว่าเครื่องใดควรถูกใช้ในการประมวลผล ซึ่งสมมุติว่าเครื่องโฮสต์ทุกเครื่องทำงานเต็มประสิทธิภาพ (100% CPU utilization) ยกเว้นเครื่องสุดท้าย

อุปกรณ์และวิธีการ

อุปกรณ์

1. เครื่องคอมพิวเตอร์ หน่วยประมวลผล Intel Core2 Duo processor T7300 2.00 GHz แรม 3 GB, ฮาร์ดดิสก์ 120 GB
2. โปรแกรมจำลองระบบประมวลผลแบบกลุ่มเมฆ CloudSim version 3-0-0
3. ระบบปฏิบัติการ Microsoft Windows 7 Professional

วิธีการ

1. Power Model

จากงานวิจัยของ Barroso and Hölzle (2007) รูปแบบการใช้พลังงานของเครื่องเซิร์ฟเวอร์แบบเก่าและแบบใหม่มีการใช้พลังงานแตกต่างกัน โดยเฉพาะช่วง idle state ที่ไม่มี workload ในการประมวลผล พบว่าเครื่องเซิร์ฟเวอร์แบบเก่ามีการใช้พลังงานที่สูงกว่าเครื่องเซิร์ฟเวอร์แบบใหม่ เมื่อเราลองนำค่าพลังงานจาก SPEC ของแต่ละเครื่องเซิร์ฟเวอร์มาวาดกราฟพบว่า ค่าความชันของกราฟไม่ได้แตกต่างกันมาก นั่นแสดงให้เห็นว่าประสิทธิภาพการใช้พลังงานของเครื่องเซิร์ฟเวอร์แบบเก่าไม่ได้ขึ้นอยู่กับการใช้พลังงานมากใน idle state

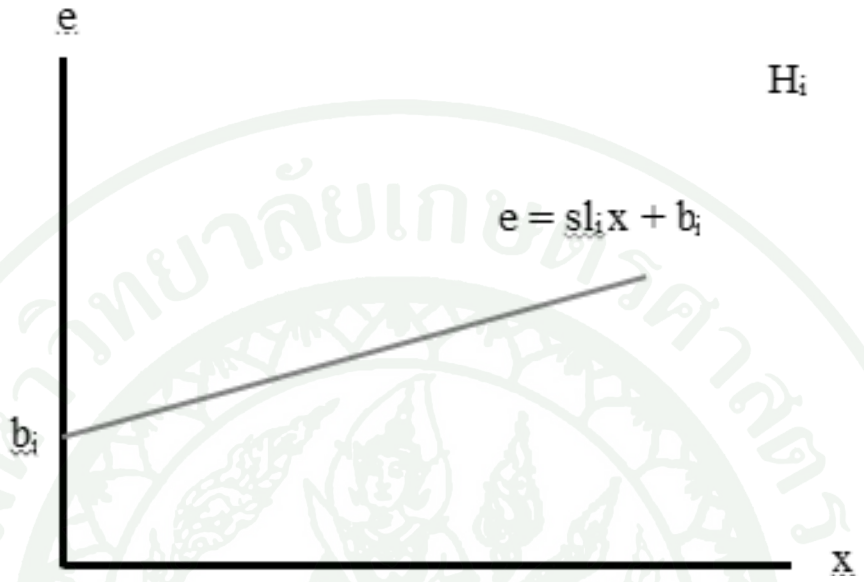
สมการสำหรับรูปแบบการใช้พลังงานของเซิร์ฟเวอร์ H_i ซึ่ง H_i คือเซิร์ฟเวอร์หรือโฮสต์บน Cloud ในภาพที่ 27 แกนตั้งแสดงด้วยค่า e และแกนนอนแสดงด้วยค่า x ได้สมการ (5)

$$e = sl_i x + b_i \quad (5)$$

โดยที่

e	คือค่าพลังงานเฉลี่ยในหน่วยวัตต์ ซึ่งได้มาจาก SPEC
sl_i	คือค่าความชันของกราฟพลังงาน
x	คือค่า CPU utilization ของโฮสต์ ซึ่งมีค่าระหว่าง 0% ถึง 100%

b_i คือค่าพลังงานเฉลี่ยในหน่วยวัตต์ขณะ idle state



ภาพที่ 27 กราฟสมการรูปแบบการใช้พลังงาน

เมื่อลองสำรวจการใช้พลังงานของเครื่องเซิร์ฟเวอร์จาก SPEC (Standard Performance Evaluation Corporation, 1995) ในช่วงก่อนและหลังปี ค.ศ. 2007 พบว่าความชันของกราฟพลังงานมีค่าอยู่ระหว่าง 0.4 – 0.8 ดังแสดงในตารางที่ 1

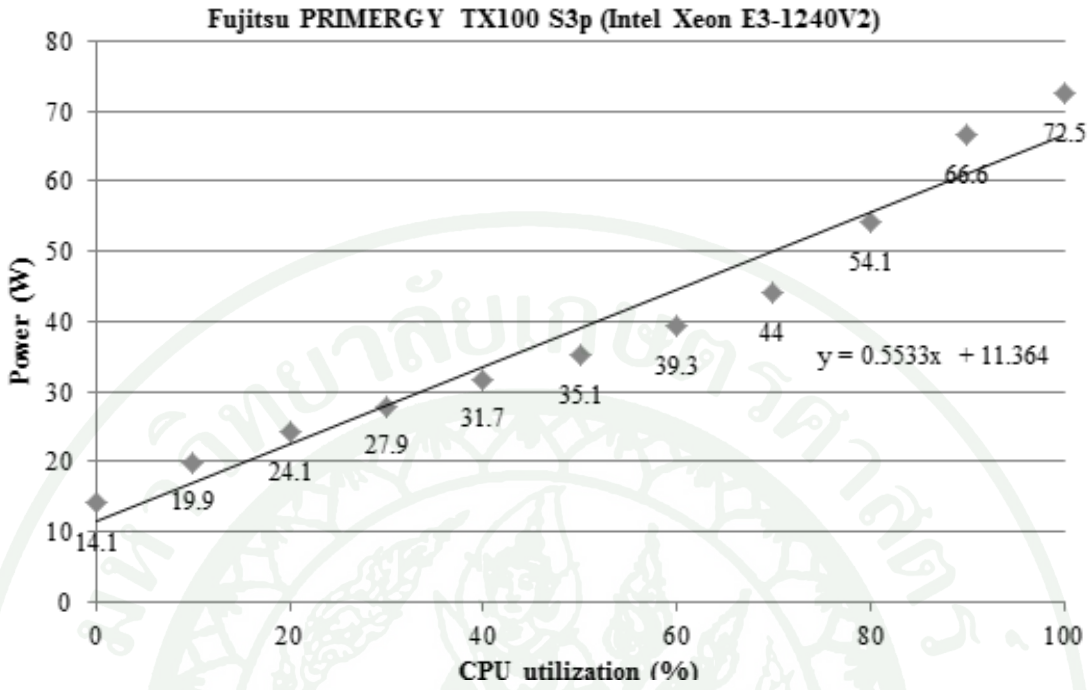
ตารางที่ 1 ตารางแสดงค่าความชันกราฟพลังงานของการสุ่มโฮสต์ในช่วงก่อนและหลังปี ค.ศ. 2007

ก่อน/หลัง ค.ศ. 2007	**รุ่น**	**สมการ**
ก่อนปี ค.ศ. 2007	Fujitsu Siemens Computers PRIMERGY RX300 S3 (Intel Xeon L5335)	0.7309x + 191.64
	Fujitsu Siemens Computers PRIMERGY TX120 (Intel Xeon 3070)	0.4483x + 65.214
	SuperMicro, Inc. Supermicro 6025B-TR+ (Intel Xeon processor L5335)	0.7891x + 197.73

ตารางที่ 1 (ต่อ)

ก่อน/หลัง ค.ศ. 2007	รุ่น	สมการ
	Fujitsu Siemens Computers PRIMERGY TX150 S6 (Intel Xeon X3220)	0.5469x + 80.664
	Fujitsu Siemens Computers PRIMERGY TX150 S5 (Intel Xeon 3070)	0.4231x + 89.9
	Colfax International CX2266-N2	0.9291x + 194.09
หลังปี ค.ศ. 2007	IBM System x3200 M3	0.7082x + 41.509
	IBM System x3250 M3	0.7369x + 38.127
	IBM Corporation System x3250 M3	0.7340x + 36.336
	Fujitsu PRIMERGY TX100 S3p (Intel Xeon E3-1240V2)	0.5533x + 11.364
	Acer Incorporated Gateway GT110 F2 (Intel Xeon E3-1270)	0.8015x + 20.473
	Hitachi, Ltd. HA8000/TS10 (DL)	0.7509x + 20.773

หากลองวาดกราฟแสดงการใช้พลังงานของเครื่องเซิร์ฟเวอร์จะได้กราฟดังภาพที่ 28 แต่ละจุดบนกราฟคือค่าพลังงาน ณ แต่ละค่าของ CPU utilization ซึ่งความชันของกราฟหาได้จากการใช้ฟังก์ชั่น Linear Regreesion (Hester, 2006)



ภาพที่ 28 กราฟการใช้พลังงานของเครื่องเซิร์ฟเวอร์ Fujitsu PRIMERGY TX100 S3p (Intel Xeon E3-1240V2) จาก SPEC

2. System Model

การกำหนดค่าตัวแปรที่ใช้ในอัลกอริทึมแบ่งเป็น 4 ส่วนดังนี้

2.1 โฮสต์ (Host)

กำหนดตัวแปรแสดงโฮสต์ทั้งหมดบน Cloud แทนด้วย H ซึ่งแต่ละโฮสต์แสดงด้วย H_i โดยที่ 1<= i <= H และการประมวลผลแทนด้วยตัวแปร $MIPS_i$ จำนวน Memory ของโฮสต์แทนด้วย RAM_i แบนด์วิชแทนด้วย BW_i ฮาดร์ดิสแทนด้วย ST_i นอกจากนี้ยังแทนตัวแปร ความชันของกราฟพลังงานแต่ละโฮสต์ด้วย sl_i และค่าพลังงาน ณ idle state ด้วย b_i

2.2 เครื่องเสมือน (Virtual Machine)

กำหนดตัวแปรแสดง VM แทนด้วย M แต่ละ VM แสดงด้วย V_j โดยที่ 1<= j <= M และการประมวลผลแทนด้วยตัวแปร $MIPS_j$ จำนวน Memory ของ VM แทนด้วย RAM_j แบนด์วิชแทนด้วย BW_j ฮาดร์ดิสแทนด้วย ST_j

2.3 Workload

ใน CloudSim กำหนดค่า workload ในรูปแบบของช่วงเวลา เช่น 1 วัน, 2 วัน หรือ 10 ชั่วโมง ทุกๆ 300 วินาทีของเวลาในการจำลอง CloudSim จะคำนวณการใช้พลังงานของแต่ละโฮสต์ ซึ่งใช้ค่า CPU utilization ไปคิดหาค่าพลังงานในแต่ละโฮสต์ ค่าของ CPU utilization คำนวณจากค่า $MIPS_j$ ของ VM ที่ทำงานบนโฮสต์นั้น หารด้วย $MIPS_i$ ของโฮสต์นั้น เช่น โฮสต์มี VM ทำงานอยู่ 3 เครื่อง มีขนาดของ MIPS 100, 100 และ 200 โฮสต์มีขนาดของ MIPS 500 คำนวณค่า CPU utilization ได้

$$(100 + 100 + 200) / 500 = 80\%$$

จากนั้น CloudSim จะนำค่า CPU utilization ที่คำนวณได้ไปหาค่าพลังงานของโฮสต์ที่ทำงาน ณ ขณะเวลานั้น

2.4 Allocation

การจัด VM เข้าใช้งานบนโฮสต์ CloudSim จะจัดแบบเรียงลำดับไปเรื่อยๆ คือนำ VM เรียงบนโฮสต์ตัวแรกให้เต็มก่อนแล้วจึงไปเรียงบนโฮสต์ตัวถัดไป ซึ่งพิจารณาจากค่า MIPS, RAM, BW และ ST ระหว่าง VM กับโฮสต์ว่าเพียงพอให้ VM นั้นสามารถสร้างบนโฮสต์ได้หรือไม่

3. Algorithm

จุดประสงค์หลักของอัลกอริทึมคือเพื่อลดค่าใช้จ่ายของพลังงานที่ใช้ในการประมวลผล เนื่องจากโฮสต์ที่มีอยู่บน Cloud มีหลากหลายรุ่น ถ้าเราเลือกโฮสต์ที่มีความคุ้มค่าในการประหยัดพลังงานมาทำงาน จะมีคุณประโยชน์อย่างมากในการช่วยลดต้นทุนของค่าใช้จ่ายเกี่ยวกับพลังงานที่สูญเสียไป

ได้คิดค้นและสร้างอัลกอริทึมขึ้นมา 2 อัลกอริทึม

3.1 Simple Energy Slope Ordering (ESO)

ซึ่งทำงานโดยการเรียงโฮสต์ตามค่าของความชันของกราฟพลังงานของโฮสต์จากน้อยไปมาก

ตารางที่ 2 ตารางตัวอย่างแสดงลำดับการเรียงโฮสต์โดยใช้อัลกอริทึม ESO

หมายเลข	รุ่น	สมการ	ลำดับการเรียงโดย ESO
1	Host A	0.4195x + 75	**1**
2	Host B	0.7546x + 63	**4**
3	Host C	0.6269x + 20	2
4	Host D	0.7443x + 34	3

3.2 Full-but-Last-n (FL-n)

ซึ่งทำงานโดยเรียงโฮสต์ตามค่าพลังงาน ณ ขณะที่ CPU utilization ทำงานเต็ม 100% โดยเรียงจากค่าน้อยไปมาก ส่วนโฮสต์ตัวสุดท้ายจะนำมาพิจารณากับโฮสต์ที่เหลืออยู่ในระบบ ว่าพลังงาน ณ ขณะที่ CPU utilization เท่ากับค่า n เช่น n คือค่า CPU utilization 30% หรือ 60% ว่ามีค่าน้อยที่สุดหรือไม่ ถ้าไม่น้อยที่สุดจะสลับนำโฮสต์ที่มีค่าน้อยที่สุดมาแทนที่

ตารางที่ 3 ตารางแสดง Psudocode ของอัลกอริทึม FL-n

ขั้นที่	การทำงาน
1.	Order Hosts by ascending sl_i and save in HOST_LIST
2.	Calculate ALL_MIPS_VM = $\sum_j$VM_MIPS$_j$, 1 <= j <= M
3.	Find X_MIPS where X_MIPS is the minimum integer that satisfies $\min_i \sum_i$H_MIPS$_i$ >= ALL_MIPS_VM, where 1 <= X_MIPS <= H. Thus, X_MIPS is the minimum number of hosts to match the sum of MIPS of all VMs.
4.	Repeat step 2 and 3 to find X_RAM, X_BW, and X_ST
5.	Calculate X = MAX(X_MIPS, X_RAM, X_BW, X_ST). X is the minimum number of hosts to serve all VMs.
6.	Order hosts by ascending energy consumption at 100% CPU utilization, select the first X-1 hosts and save them into NEW_HOST_LIST.
7.	Order the rest of the hosts by ascending energy consumption at n% and append them to NEW_HOST_LIST
8.	Return NEW_HOST_LIST

ตัวอย่างแสดงการเรียงลำดับโฮสต์โดยใช้อัลกอริทึม FL-n กำหนดให้มีโฮสต์ว่างบน Cloud 4 โฮสต์ จากตารางที่ 2 ทำให้ได้รายการโฮสต์คือ {A, C, D, B} ซึ่งมี MIPS 500, 1000, 1500, 2000 และมี VM ที่ต้องการใช้งาน 5 เครื่อง มี MIPS 100, 300, 500, 700, 1000 ดังนั้นรายการโฮสต์ใหม่เมื่อเรียงตาม CPU utilization 100% คือ {C, D, A, B} เมื่อคำนวณหาค่า X ตามอัลกอริทึมจะได้ค่า X เป็น 3 แต่ก่อนที่จะนำโฮสต์ 3 ตัวเข้า NEW_HOST_LIST เราต้องนำโฮสต์ตัวสุดท้ายที่จะหยิบเข้ารายการใหม่ไปเทียบกับโฮสต์ที่เหลือ โดยเทียบหาค่าน้อยที่สุดที่ CPU utilization n% กำหนดให้ n เป็น 30 เมื่อเทียบค่าจากตารางที่ 4 จะได้รายการของโฮสต์ใหม่ที่เรียงลำดับตามอัลกอริทึม FL-n คือ {C, D, B}

ตารางที่ 4 ตารางตัวอย่างแสดงค่าพลังงานของแต่ละโฮสต์ของแต่ละช่วง CPU utilization

โฮสต์	พลังงาน (Watt)										
	Idle	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
A	75	79.2	83.4	87.6	91.8	96	100.2	104.4	108.6	112.8	117
B	63	70.5	78.1	85.6	93.2	100.7	108.3	115.8	123.4	130.9	138.5
C	20	26.3	32.5	38.8	45.1	51.3	57.6	63.9	70.2	76.4	82.7
D	34	41.4	48.9	56.3	63.8	71.2	78.7	86.1	93.5	101	108.4

ถ้าลองแทนค่า n ด้วยค่า 30, 60 และ 100 จะได้การเรียงลำดับการเลือกเครื่องเซิร์ฟเวอร์ดังตารางที่ 5

ตารางที่ 5 ตารางแสดงลำดับการเรียงโฮสต์โดยใช้อัลกอริทึม FL-n อ้างอิงข้อมูลการเรียงจากตารางที่ 4 ซึ่ง n มีค่า 30, 60, 100

โฮสต์	FL-n		
	30	**60**	**100**
A	**4**	3	3
B	3	**4**	**4**
C	**1**	**1**	**1**
D	2	2	2

4. วิธีการทดลอง

4.1 วิธีการ

การประเมินวัดค่าผลที่ได้จากการทดลองจะนำค่าที่ได้มาเปรียบเทียบกับ ลำดับการใช้เครื่องเซิร์ฟเวอร์ที่ประหยัดพลังงานที่สุด (Optimal Ordering) ซึ่งเป็นการเรียงลำดับที่ทำให้ค่าการใช้พลังงานโดยรวมของการทดลองครั้งนั้นมีค่าการใช้พลังงานที่น้อยที่สุดของการประมวลผลบน

CloudSim การหาค่า Optimal ordering หาได้จากการเรียงลำดับทุกลำดับที่เป็นไปได้ ตัวอย่างเช่นมีโฮสต์ 4 เครื่อง ดังนั้นทุกลำดับที่สามารถเกิดขึ้นได้ดังแสดงในตารางที่ 6

ตารางที่ 6 ตารางแสดงลำดับการเรียงโฮสต์ที่สามารถเกิดขึ้นได้

ลำดับการเรียงที่เป็นไปได้ทั้งหมด	
[1, 2, 3, 4]	[3, 1, 2, 4]
[1, 2, 4, 3]	[3, 1, 4, 2]
[1, 3, 2, 4]	[3, 2, 1, 4]
[1, 3, 4, 2]	[3, 2, 4, 1]
[1, 4, 2, 3]	[3, 4, 1, 2]
[1, 4, 3, 2]	[3, 4, 2, 1]
[2, 1, 3, 4]	[4, 1, 2, 3]
[2, 1, 4, 3]	[4, 1, 3, 2]
[2, 3, 1, 4]	[4, 2, 1, 3]
[2, 3, 4, 1]	[4, 2, 3, 1]
[2, 4, 1, 3]	[4, 3, 1, 2]
[2, 4, 3, 1]	[4, 3, 2, 1]

ดังนั้นเมื่อทำการทดลองทุกการเรียงลำดับที่เป็นไปได้จะสามารถหาค่า Optimal ordering ได้ จากนั้นนำค่าที่ได้จากผลการทดลองมาเทียบกับค่าที่ได้จาก Optimal ordering หาค่าออกมาในรูปของเปอร์เซ็นต์

การทดลองจะแบ่งตามเหตุการณ์ (Scenario) แต่ละ scenario จะทดลองทั้งหมด 40 ครั้ง แต่ละครั้งของการทดลองมีการกำหนดและสุ่มค่าดังนี้

4.1.1 VM

กำหนด VM 10 ตัว แต่ละตัวสุ่มค่า MIPS ระหว่าง 1,000 – 2,000

4.1.2 Host

กำหนดให้โฮสต์แต่ละตัวสุ่มค่า MIPS ระหว่าง 3,000 – 4,000 โฮสต์ที่ทำการทดลองแบ่งชนิดออกเป็นโฮสต์รุ่นใหม่และโฮสต์รุ่นเก่า โฮสต์แต่ละรุ่นกำหนดให้สุ่มค่าความชันของกราฟพลังงานและค่า idle state ดังตารางที่ 7

ตารางที่ 7 ตารางแสดงการกำหนดค่าสุ่มของโฮสต์ที่ใช้ในการทดลอง

ชนิดของโฮสต์	**ค่าความชันของกราฟพลังงาน (sli)**	**ค่า idle state (bi)**
โฮสต์รุ่นเก่า	0.4 – 0.8	50 - 80
โฮสต์รุ่นใหม่	0.4 – 0.8	20 - 40

ในการทดลองจะเพิ่มโฮสต์จาก 4, 5 และ 6 โฮสต์ตามลำดับ ในจำนวนโฮสต์ที่เท่ากันแบ่งการทดลองอีก 3 ครั้งโดยแบ่งตามชนิดของโฮสต์คือ ทดลองด้วยโฮสต์รุ่นเก่าทั้งหมด, โฮสต์รุ่นใหม่ทั้งหมดและโฮสต์แบบผสม (โฮสต์รุ่นเก่าและโฮสต์รุ่นใหม่)

ดังนั้น scenario ที่เกิดขึ้นทั้งหมดสำหรับการทดลองสรุปได้ดังตารางที่ 8

ตารางที่ 8 ตารางสรุป scenario ที่ใช้ในการทดลอง

จำนวนโฮสต์	**โฮสต์ scenario**	**จำนวนเครื่องรุ่นเก่า**	**จำนวนเครื่องรุ่นใหม่**	**จำนวน VMs**
4	รุ่นเก่า	4	0	10
	รุ่นใหม่	0	4	
	ผสม	2	2	

ตารางที่ 8 (ต่อ)

จำนวนโฮสต์	โฮสต์ scenario	จำนวนเครื่องรุ่นเก่า	จำนวนเครื่องรุ่นใหม่	จำนวน VMs
5	รุ่นเก่า	5	0	10
	รุ่นใหม่	0	5	
	ผสม	2	3	
6	รุ่นเก่า	6	0	10
	รุ่นใหม่	0	6	
	ผสม	3	3	

ผลและวิจารณ์

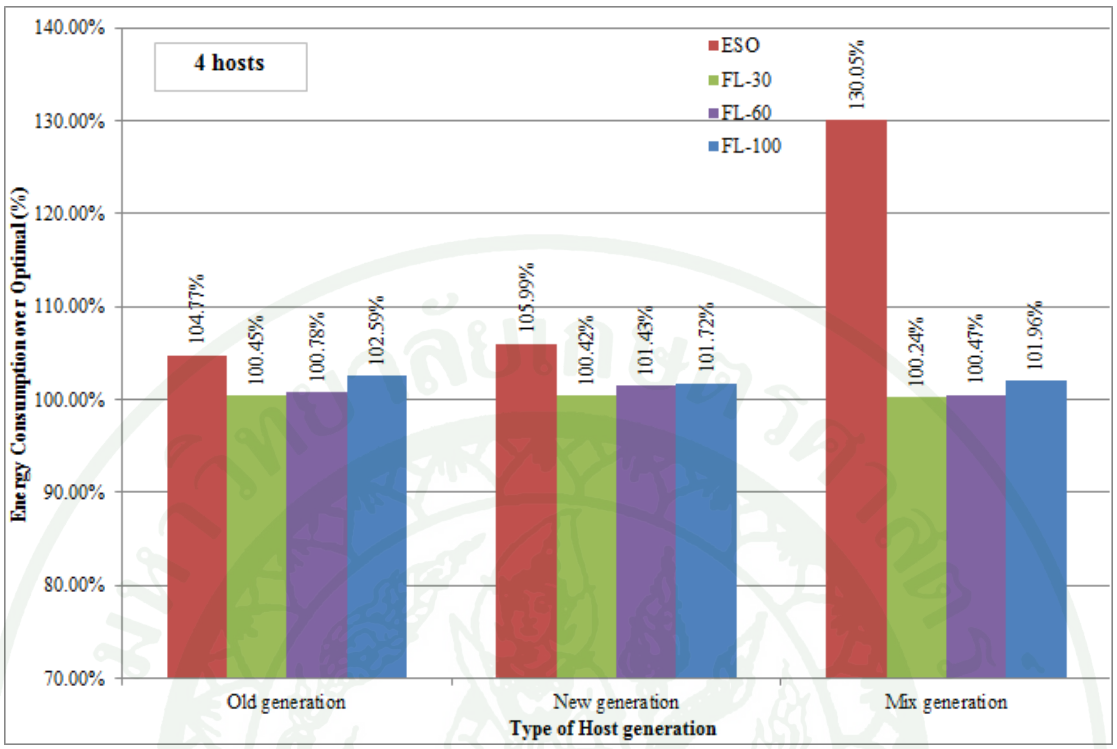
ผล

ผลการทดลองที่ได้จากอัลกอริทึม ESO, FL-100, FL-30 และ FL-60 สรุปผลการทดลองตามจำนวนโฮสต์ที่ใช้ในการทดลองในรูปแบบของกราฟ โดยแกนตั้งของกราฟแสดง ค่าเฉลี่ยการใช้พลังงานของแต่ละอัลกอริทึมเทียบกับ optimal ordering ที่ทำการทดลองจำนวน 40 ครั้ง มีหน่วยเป็นเปอร์เซ็นต์ (%) แกนนอนของกราฟแสดงโฮสต์ scenario และกราฟสีแดงแสดงผลการทดลองของอัลกอริทึม ESO กราฟสีน้ำเงินแสดงผลการทดลองของอัลกอริทึม FL-100 กราฟสีเขียวแสดงผลการทดลองของอัลกอริทึม FL-30 และกราฟสีม่วงแสดงผลการทดลองของอัลกอริทึม FL-60

ผลการทดลองที่ได้มีดังต่อไปนี้

1. ผลการทดลองโฮสต์ 4 เครื่อง

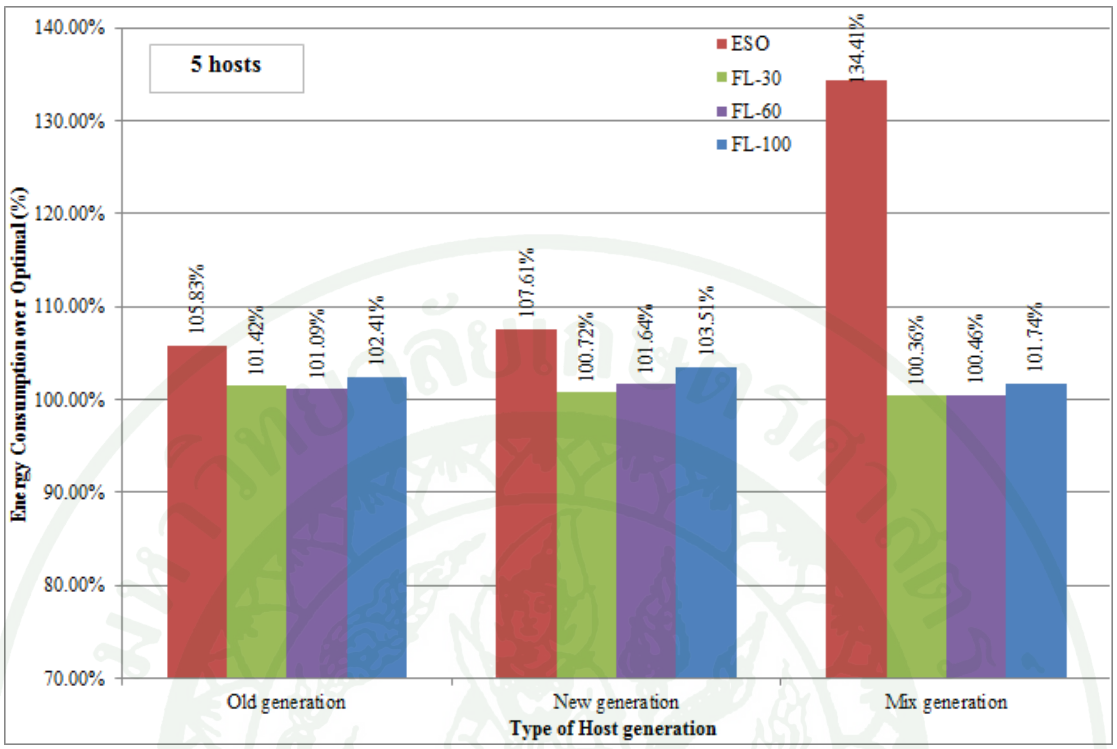
ผลลัพธ์การทดลอง เมื่อทดลองโดยโฮสต์ 4 เครื่อง สรุปได้ดังภาพที่ 29



ภาพที่ 29 กราฟแสดงค่าเฉลี่ยการใช้พลังงานของแต่ละอัลกอริทึมโดยทดลองกับโฮสต์ 4 เครื่อง

2. ผลการทดลองโฮสต์ 5 เครื่อง

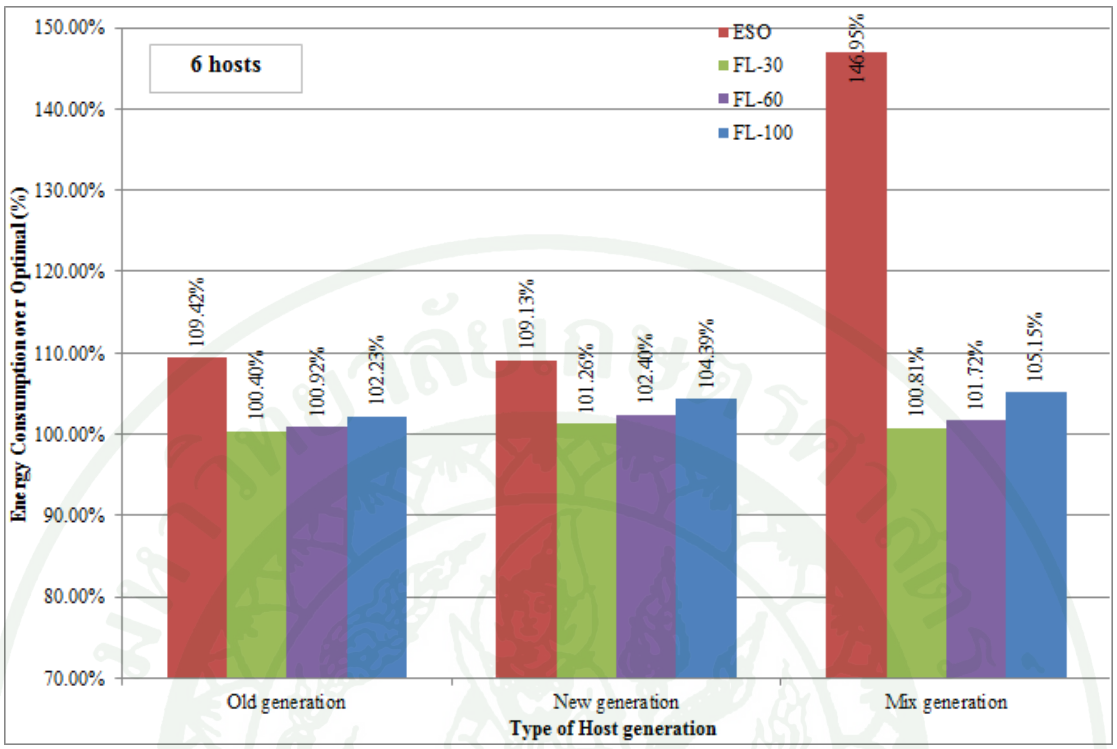
ผลลัพธ์การทดลอง เมื่อทดลองโดยโฮสต์ 5 เครื่อง สรุปได้ดังภาพที่ 30



ภาพที่ 30 กราฟแสดงค่าเฉลี่ยการใช้พลังงานของแต่ละอัลกอริทึมโดยทดลองกับโฮสต์ 5 เครื่อง

3. ผลการทดลองโฮสต์ 6 เครื่อง

ผลลัพธ์การทดลอง เมื่อทดลองโดยโฮสต์ 6 เครื่อง สรุปได้ดังภาพที่ 31



ภาพที่ 31 กราฟแสดงค่าเฉลี่ยการใช้พลังงานของแต่ละอัลกอริทึมโดยทดลองกับโฮสต์ 6 เครื่อง

วิจารณ์

ค่าเฉลี่ยที่ได้จากการทดลองเทียบกับ optimal ordering สรุปได้ดังตารางที่ 9

ตารางที่ 9 ตารางสรุปผลการทดลองของ scenario ที่ใช้ในการทดลอง

อัลกอริทึม	ค่าเฉลี่ยการใช้พลังงานของอัลกอริทึมเทียบกับ optimal ordering (%)								
	4 โฮสต์			5 โฮสต์			6 โฮสต์		
	รุ่นใหม่	รุ่นเก่า	ผสม	รุ่นใหม่	รุ่นเก่า	ผสม	รุ่นใหม่	รุ่นเก่า	ผสม
ESO	105.99	104.77	130.05	107.61	105.83	134.41	109.13	109.42	146.95
FL-30	100.42	100.45	100.24	100.72	101.42	100.36	101.26	100.40	100.81
FL-60	101.43	100.78	100.47	101.64	101.09	100.46	102.40	100.92	101.72
FL-100	101.72	102.59	101.96	103.51	102.41	101.74	104.39	102.23	105.15

1. วิจารณ์ผล ESO อัลกอริทึม

เมื่อพิจารณา ESO อัลกอริทึม ผลที่ได้จากการทดลองพบว่า Old generation hosts และ New generation hosts ได้ค่าเฉลี่ยออกมาใกล้เคียงกัน ซึ่งแตกต่างกันไม่เกิน 2% เมื่อลองเพิ่มจำนวนโฮสต์ในการทดลอง แนวโน้มของค่าเฉลี่ยโฮสต์แต่ละรุ่นมีค่าสูงขึ้น ใน New generation hosts ค่าแนวโน้มสูงขึ้นไล่เลี่ยกัน แต่ Old generation hosts กับ Mix generation hosts ระหว่างโฮสต์ 5 เครื่องกับโฮสต์ 6 เครื่อง มีค่ากระโดดจาก 105.83 เป็น 109.42 และ 134.41 เป็น 146.95

ผลการทดลองเมื่อจำนวนโฮสต์ที่เท่ากัน Mix generation hosts มีค่าเฉลี่ยเมื่อเทียบกับ optimal ordering สูงที่สุดและสูงถึงเกือบ 50% เมื่อทดลองกับโฮสต์จำนวน 6 เครื่อง

2. วิจารณ์ผล FL-n อัลกอริทึม

ผลการทดลองพบว่าทั้ง 3 generations มีค่าผลลัพธ์ที่ได้ใกล้เคียงกัน ซึ่งค่าแตกต่างกันเมื่อเทียบกับ optimal ordering ไม่เกิน 3% ซึ่งเป็นการบอกถึงประสิทธิภาพของอัลกอริทึมว่าทำงานได้ประสิทธิภาพที่ใกล้เคียงกัน ไม่ว่าโฮสต์จะรุ่นเก่า, รุ่นใหม่ หรือรุ่นผสม และพบว่าอัลกอริทึมแบบ FL-30 เมื่อเทียบค่าผลลัพธ์แต่ละรุ่นของโฮสต์ต่างกันไม่ถึง 1%

นอกจากนี้แนวโน้มผลการทดลองของ New generation hosts เมื่อ n มีค่า 30, 60 และ 100 มีแนวโน้มเพิ่มขึ้นเมื่อเพิ่มจำนวนเครื่องในการทดลอง ผลการทดลองของ Old generation hosts เมื่อ n มีค่า 100 มีแนวโน้มของผลลัพธ์ลดลง ส่วนผลการทดลองของ Mix generation hosts เมื่อ n มีค่า 30, 60, 100 และ Old generation hosts เมื่อ n มีค่า 30 และ 60 ได้ผลลัพธ์ขึ้นๆ ลงๆ ไม่คงที่ พิจารณาจาก Mix generation hosts เมื่อ n มีค่า 60 และ 100 จากเครื่องโฮสต์ 4 เครื่องไป 5 เครื่อง มีแนวโน้มลดลงและสูงขึ้นจากเครื่องโฮสต์ 5 เครื่องไป 6 เครื่อง และ Mix generation hosts เมื่อ n มีค่า 30 ได้ผลลัพธ์ที่มีแนวโน้มสูงขึ้น

สรุปและข้อเสนอแนะ

สรุป

วัตถุประสงค์ของงานวิจัยคือ นำเสนออัลกอริทึมในการเรียงลำดับการจัดสรร VMs บนเครื่องโฮสต์บน Cloud computing ซึ่งช่วยทำให้ผลรวมของพลังงานที่ใช้ในการประมวลผลมีค่าน้อยที่สุด ในงานวิจัยนี้นำเสนอ 2 อัลกอริทึม คือ ESO และ FL-n โฮสต์แต่ละตัวอ้างอิงข้อมูลพลังงานจาก SPEC การเรียงลำดับแบบ ESO จะพิจารณาลำดับการเรียงจากค่าความชันของกราฟพลังงานของแต่ละโฮสต์ ส่วนอัลกอริทึม FL-n พิจารณาการเรียงลำดับจากค่าพลังงาน ณ idle state

จากผลการทดลองพบว่า ESO อัลกอริทึมทำงานได้ประสิทธิภาพที่ต่ำมาก เมื่อทดลองในโฮสต์รุ่นผสมซึ่งมีค่าพลังงาน ณ idle state ที่แตกต่างกัน ในขณะที่ FL-n ทำงานได้ประสิทธิภาพที่ดีกว่าในทุกๆ scenario

เมื่อพิจารณาอัลกอริทึม ESO จะทำงานได้ดีเมื่อโฮสต์ที่ใช้ประมวลผลอยู่ในรุ่นเดียวกัน ซึ่งได้ค่าเฉลี่ยเมื่อเทียบกับ optimal ordering ไม่เกิน 10% และถ้าหากเทียบผลลัพธ์ระหว่างรุ่นเก่าและรุ่นใหม่จะมีค่าเฉลี่ยเทียบกับ optimal ordering ไม่มากเกิน 2% แต่ผลลัพธ์ที่ได้เมื่อโฮสต์ที่ประมวลผลผสมกันระหว่างรุ่นเก่าและรุ่นใหม่ ค่าที่ได้มีค่า 30% และสูงถึง 50% เทียบกับ optimal ordering เมื่อโฮสต์ที่ใช้ในการทดลองเพิ่มมากขึ้น ที่เป็นเช่นนี้เพราะอัลกอริทึมแบบ ESO ใช้ความชันของกราฟพลังงานในการเรียงลำดับโฮสต์ ซึ่งค่าความชันนี้ถ้าพิจารณาจากตารางที่ 1 เรื่องค่าความชันระหว่างเครื่องเซิร์ฟเวอร์เก่าและเครื่องเซิร์ฟเวอร์ใหม่ จะเห็นว่าค่าความชันนั้นมีค่าอยู่ระหว่างช่วงเดียวกัน แต่ค่าพลังงาน ณ idle state มีค่าไม่เท่ากัน ในการทดลองอาจเป็นไปได้ว่า ESO เลือกเครื่องผสมระหว่างเครื่องเก่าและเครื่องใหม่ที่มีค่าความชันใกล้เคียงกัน แต่ค่าพลังงานต่างกันมาก พอผลลัพธ์ที่ได้ออกมาค่าของ ESO จึงต่างจาก FL-n มาก

ถ้ายิ่งเพิ่มจำนวนโฮสต์มากขึ้นผลลัพธ์ที่ได้จะแย่ลง จะเห็นได้ชัดจากผลลัพธ์ของโฮสต์แบบผสม แต่จะไม่แตกต่างมากเมื่อโฮสต์เป็นรุ่นเดียวกันทั้งหมด

ผลการทดลองเห็นได้อย่างชัดเจนว่าอัลกอริทึมแบบ FL-n ทำงานได้ดีกว่า ซึ่งค่าเฉลี่ยเทียบกับ optimal ordering สูงไม่เกิน 103% ถึงแม้จะเพิ่มจำนวนโฮสต์ในการทดลองก็ไม่สามารถบอกได้ว่าทำให้การทำงานของอัลกอริทึมแย่ลงหรือไม่

เมื่อลองเปลี่ยนค่า n จาก 30 เป็น 60 ผลที่ได้จากการทดลองไม่ได้แตกต่างกันมากแต่อย่างใด เพื่อทดสอบว่า n มีผลต่อความเปลี่ยนแปลงของค่าเฉลี่ยหรือไม่ จึงทดลองค่า n ที่ 100 โดยอ้างอิงว่าโฮสต์ทุกตัวทำงานเต็มประสิทธิภาพ 100% ทุกตัว ผลการทดลองที่ได้ของ FL-100 มีค่าแตกต่างจากผลลัพธ์ของ FL-30 อยู่ที่ระหว่าง 1-5%

ตารางที่ 10 ตารางสรุปความแม่นยำของแต่ละอัลกอริทึมเมื่อเทียบกับ optimal ordering

อัลกอริทึม	**ความแม่นยำของอัลกอริทึมเทียบกับ optimal ordering (%)**								
	4 โฮสต์			**5 โฮสต์**			**6 โฮสต์**		
	รุ่นใหม่	**รุ่นเก่า**	**ผสม**	**รุ่นใหม่**	**รุ่นเก่า**	**ผสม**	**รุ่นใหม่**	**รุ่นเก่า**	**ผสม**
ESO	25.0	42.5	20.0	17.5	10.0	12.5	15.0	5.0	5.0
FL-30	75.0	70.0	80.0	67.5	50.0	70.0	50.0	75.0	65.0
FL-60	62.5	67.5	77.5	55.0	42.5	70.0	37.5	52.5	47.5
FL-100	62.5	45.0	70.0	42.5	30.0	60.0	25.0	40.0	40.0

จากตารางที่ 10 แสดงค่าความแม่นยำของแต่ละอัลกอริทึม ซึ่งค่าความแม่นยำนี้ก็คือค่าผลลัพธ์ที่ได้นั้นมีค่าเท่ากับค่า optimal ordering ซึ่งแสดงผลลัพธ์เป็นเปอร์เซ็นต์ ว่ามีความแม่นยำมากน้อยเพียงใด ผลปรากฏว่า อัลกอริทึม ESO ได้ค่าความแม่นยำต่ำมาก ค่าที่ได้น้อยสุดถึง 20% ในการหาความแม่นยำของเครื่องเซิร์ฟเวอร์รุ่นผสม ส่วนอัลกอริทึมแบบ FL-n หาค่าความแม่นยำได้ดีและค่าความแม่นยำสูงถึง 80% ในผลลัพธ์การทดลองรุ่นผสมของเครื่องโฮสต์ 4 เครื่อง จุดที่

น่าสนใจคือผลการทดลองเครื่องโฮสต์รุ่นผสมของอัลกอริทึมแบบ FL-n จะได้ผลลัพธ์ค่าความแม่นยำสูงกว่า เครื่องโฮสต์รุ่นเก่าและรุ่นใหม่

เมื่อนำข้อมูลผลการทดลองที่ได้มาวิเคราะห์ข้อมูลแบบ box and whisker plot (Stapel, 2014) จะได้ดังตารางที่ 11 จากค่าในตารางพบว่า ข้อมูลของอัลกอริทึมแบบ FL-n มีค่าเบนไปทางซ้าย นั่นหมายความว่ากลุ่มข้อมูลส่วนใหญ่มีค่าตรงกับค่า optimal ordering คือ 100% ซึ่งต่างจากอัลกอริทึมแบบ ESO ที่เมื่อลองพิจารณาค่าผลการทดลองของเครื่องเซิร์ฟเวอร์รุ่นใหม่และเครื่องเซิร์ฟเวอร์รุ่นเก่า ข้อมูลที่ได้ไม่เอียงไปทางใดทางหนึ่ง นั่นหมายความว่าค่าที่ได้จากการทดลองไม่ค่อยมีข้อมูลซ้ำกันหรือข้อมูลเป็นกลุ่ม แต่เมื่อพิจารณาค่าผลการทดลองแบบผสมของ ESO พบว่า ค่าที่ได้จากการทดลองส่วนใหญ่เบนไปด้านขวา

ตารางที่ 11 ตารางวิเคราะห์ข้อมูลแบบ box and whisker plot

อัลกอริทึม		ค่าเฉลี่ยการใช้พลังงานเทียบกับ optimal ordering (%)								
		4 โฮสต์			5 โฮสต์			6 โฮสต์		
		รุ่นใหม่	รุ่นเก่า	ผสม	รุ่นใหม่	รุ่นเก่า	ผสม	รุ่นใหม่	รุ่นเก่า	ผสม
ESO	Q1	100.14	100.00	109.08	101.68	101.39	109.40	101.54	101.39	128.08
	Med.	104.48	101.37	134.49	106.84	103.82	141.02	109.36	107.81	147.71
	Q3	107.85	107.85	147.09	112.40	109.86	150.38	116.55	114.96	161.83
FL-30	Q1	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
	Med.	100.00	100.00	100.00	100.00	100.16	100.00	100.25	100.00	100.00
	Q3	100.43	100.29	100.00	100.59	101.39	100.47	101.60	100.20	100.86
FL-60	Q1	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
	Med.	100.00	100.00	100.00	100.00	100.44	100.00	101.04	100.00	100.48
	Q3	101.31	100.32	100.00	103.13	101.51	100.47	104.10	101.28	102.49

ตารางที่ 11 (ต่อ)

อัลกอริทึม		ค่าเฉลี่ยการใช้พลังงานเทียบกับ optimal ordering (%)								
		4 โฮสต์			5 โฮสต์			6 โฮสต์		
		รุ่นใหม่	รุ่นเก่า	ผสม	รุ่นใหม่	รุ่นเก่า	ผสม	รุ่นใหม่	รุ่นเก่า	ผสม
FL-100	Q1	100.00	100.00	100.00	100.00	100.00	100.00	100.14	100.00	100.00
	Med.	100.00	100.29	100.00	100.58	100.86	100.00	102.75	100.54	101.87
	Q3	105.27	105.27	100.81	106.25	103.57	100.92	107.51	103.66	107.49

ซึ่งตารางวิเคราะห์นี้ทำให้เห็นถึงประสิทธิภาพการทำงานของอัลกอริทึมว่าแบบ FL-n ทำงานได้อย่างมีประสิทธิภาพดีเยี่ยม ที่ทำให้มีกลุ่มของข้อมูลที่เท่ากับค่า optimal ordering ออกมามาก ซึ่งต่างจากของ ESO ที่ผลลัพธ์ที่ได้ค่อนข้างกระจาย ไม่มีข้อมูลแบบเป็นกลุ่มเลย

จากสรุปผลการทดลองของทั้ง 2 อัลกอริทึม ถ้าหากในระบบ Cloud มีเพิ่มอัลกอริทึมเกี่ยวกับการเลือกเครื่องเซิร์ฟเวอร์โดยดูจาก Power profile มาช่วยในขั้นตอนการจัดสรรทรัพยากรจะช่วยลดค่าใช้จ่ายของพลังงานโดยรวมที่สูญเสียไป ถึงแม้ว่าทรัพยากรที่มีอยู่ในระบบนั้น จะมีทั้งเครื่องรุ่นใหม่และเครื่องรุ่นเก่าก็ไม่มีผลกับประสิทธิภาพในการจัดสรรทรัพยากรของอัลกอริทึมแต่อย่างใด

ข้อเสนอแนะ

อัลกอริทึมแบบ FL-n มีจุดที่น่าสนใจคือค่าของ n ว่ามีผลต่อประสิทธิภาพการทำงานของอัลกอริทึมหรือไม่ ค่าของ n ที่กำหนดขึ้นจากการทดลองนั้น หามาจากการประมาณโดยคิดว่าทุกเครื่องเซิร์ฟเวอร์จะทำงานเต็มประสิทธิภาพคือ 100% CPU utilization ส่วนเครื่องสุดท้ายนั้นไม่น่าจะทำงานได้ถึงครึ่งนึงคือ 50% CPU utilization ดังนั้นจึงกำหนดว่าเครื่องสุดท้ายน่าจะทำงาน

ประมาณ 30% CPU utilization หลังจากดูผลการทดลองพบว่าได้ผลที่ดีเยี่ยม แต่เพื่อความมั่นใจในประสิทธิภาพของอัลกอริทึมจึงเพิ่มการทดลองที่ n เป็น 60 และ 100 ตามมา

เมื่อพิจารณาจากผลการทดลองของ n ทั้ง 3 ค่า พบว่า ผลที่ได้ไม่แตกต่างกันมากเมื่อมีการทดลองกับค่า n ที่ 30, 60 หรือ 100 แต่ถึงกระนั้นก็ไม่สามารถพิสูจน์ได้ว่าค่า n นั้นไม่มีผลต่อประสิทธิภาพการทำงานของอัลกอริทึมจริง เพราะผลการทดลองที่ออกมาอาจเกิดจากการยังไม่ได้ปรับเปลี่ยนค่าอีกหลายปัจจัย เช่น ค่า MIPS ของ host หรือ VMs, จำนวน workload ที่คงที่เสมอ หรือ วิธีการสุ่มค่าของ MIPS ให้แต่ละ VM และแต่ละ host อาจยังไม่ดีพอ

ดังนั้นจึงควรที่จะทำการทดลองด้วยค่า n ที่หลากหลายและปรับเปลี่ยนตัวแปรอื่นๆ เพื่อดูผลของอัลกอริทึมว่ายังคงทำงานได้อย่างมีประสิทธิภาพอยู่หรือไม่ และควรลองเพิ่ม scenario ในการทดลอง หรือ จำนวนเครื่องทั้ง VM และ host ให้มากขึ้น อาจทำให้สรุปผลที่ได้ชัดเจนยิ่งขึ้น

เอกสารและสิ่งอ้างอิง

Wikipedia. 2014. **Cloud Computing**. Available Source: http://en.wikipedia.org/wiki/Cloud_computing, 8 July 2014.

Markoff, John. 2010. **Cloud Computing**. Available Source: http://programminggeeks.com/Cloud-computing/, 8 July 2014.

The Technomax Group. 2014. **A Cloud Infrastructure**. Available Source: http://www.technomax.com/thecloud.html, 8 July 2014.

Peter Mell, Timothy Grance. 2011. **The NIST Definition of Cloud Computing**. NIST Special Publication: 800-145.

Menascé, Daniel A. 2005. **Virtualization: Concepts, Application, and Performance Modeling**. Int. CMG Conference.

IT-Clever. 2010. **Virtualization**. Available Source: http://www.it-clever.com/2010/11/, 8 July 2014.

Rodrigo N. Calheiros, Rajiv Ranjan, César A. F. De Rose, Rajkumar Buyya. 2010. **CloudSim : A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services**. Pontifical Catholic University of Rio Grande do Sul Porto Alegre, Brazil.

Rajkumar Buyya, Rajiv Ranjan, Rodrigo N. Calheiros. 2010. **Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities**. Pontifical Catholic University of Rio Grande do Sul Porto Alegre, Brazil.

Shi, Y., Jiang, X., & Ye, K. September, 2011. **An energy-efficient scheme for Cloud resource provisioning based on Cloudsim**. In Cluster Computing (CLUSTER), 2011 IEEE International Conference on (pp. 595-599). IEEE.

Li, X., Jiang, X., Ye, K., & Huang, P. June, 2013. **DartCSim+: Enhanced CloudSim with the Power and Network Models Integrated**. In Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on (pp. 644-651). IEEE.

Luiz André Barroso, Urs Hölzle. 2007. **The Case for Energy-Proportional Computing**. the IEEE Computer Society: 0018-9162/07.

Xiao, Z., Song, W., & Chen, Q. 2013. **Dynamic resource allocation using virtual machines for Cloud computing environment**. Parallel and Distributed Systems, IEEE Transactions on, 24(6), 1107-1117.

Roy, N., Dubey, A., & Gokhale, A. July, 2011. **Efficient autoscaling in the Cloud using predictive models for workload forecasting**. In Cloud Computing (CLOUD), 2011 IEEE International Conference on (pp. 500-507). IEEE.

Quan, Dang Minh, et al. 2012. **Energy efficient resource allocation strategy for cloud data centres**. Computer and Information Sciences II. Springer London, 2012. 133-141.

Anton Beloglazov, Rajkumar Buyya. 2011**. Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers**. CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE Concurrency Computat.: Pract. Exper. 2011; 00:1–24.

Kumar, Dilip, and Bibhudatta Sahoo. 2014. **Energy Efficient Heuristic Resource Allocation for Cloud Computing.** International Journal (2014).

Luo, Liang, et al. 2012. **A resource scheduling algorithm of cloud computing based on energy efficient optimization methods.** Green Computing Conference (IGCC), 2012 International. IEEE, 2012.

Standard Performance Evaluation Corporation. 1995. **SPECpower_ssj2008 Results**. Available Source: http://www.spec.org/power_ssj2008/results/, 8 July 2014.

Jerry Hester. 2006. **Physics Tutorial: Linear Regression**. Available Source: http://www.clemson.edu/ces/phoenix/tutorials/regression/index.html, 8 July 2014. Clemson University.

Stapel, Elizabeth. 2014. **Box-and-Whisker Plots: Interquartile Ranges and Outliers.** Purplemath. Available Source: http://www.purplemath.com/modules/boxwhisk3.htm, 8 July 2014.

ภาคผนวก

ผลงานตีพิมพ์

1. Virtual Machine Allocation using Server Power Profile, The 11th International Joint Conference on Computer Science and Software Engineering (JCSSE 2014), May 14-16, 2014. pp. 154-159 Pattaya city, Thailand

2014 11th International Joint Conference on Computer Science and Software Engineering (JCSSE)

Virtual Machine Allocation using Server Power Profile

Monnapat Limrattanasilp
Department of Computer Science
Kasetsart University
Bangkok, Thailand
l.monnapat@gmail.com

Sethavidh Gertphol
Department of Computer Science
Kasetsart University
Bangkok, Thailand
fscisvg@ku.ac.th

***Abstract*— Cloud computing has potentials to reduce energy consumption in the data center of Cloud providers by consolidating virtual machines into as few servers as possible. The important question then becomes which physical machines should be used first, because machines from different company or different generation may not be equally efficient in energy consumption. The energy efficiency of a server is characterized by its SPEC Power benchmark. This paper proposes two Greedy algorithms, ESO and FL-n, to determine the order of physical machines to be used in a Cloud environment such that the overall energy consumption is minimized. The experimental results show that ESO performed well when servers were from the same generation, using energy not more than 10% over the optimal ordering. FL-n provided good results even when servers were from different generation also, using energy not more than 5% over the optimal in all scenarios.**

Keywords—Cloud computing; Graph of energy; Energy efficiency; CloudSim; Power profile; VM allocation

I. INTRODUCTION

Cloud computing is a new and interesting platform which allows users to access resources, such as compute power, storage, and software, in term of services. It provides easy and fast configuration and setup through the Internet using only web browser. The users of Cloud computing do not have to procure and maintain the resources, but instead pay as they use. The Cloud also has potentials to reduce cost and energy usage for the Cloud providers. These benefits result in the Cloud becoming a popular computing platform right now.

Many public cloud providers such as Amazon AWS or Google App Engine aggregate Cloud resources into a virtual machine (VM) unit, where one VM has specific configuration consisting of a compute power, memory, and storage. When a VM is instantiated, it is then allocated onto a physical machine. Cloud providers can potentially reduce the overall energy consumption by consolidating VMs onto as few physical machines as possible and turning off the rest.

The important question then becomes which physical machines should be used first, because machines from different company may not be equally efficient in energy consumption. Furthermore, since Cloud computing is a hot topic right now and demands for Cloud resources keep increasing, Cloud providers also need to expand their capacity by adding new servers into their system. The new and the old generation of servers may also be different in energy efficiency.

This paper proposes two Greedy algorithms to determine the order of physical machines to be used in a Cloud environment such that the overall energy consumption is minimized. The power efficiency of a server is characterized by its SPEC Power benchmark result, which measures energy consumption of that server at every 10% CPU utilization interval. The first algorithm, called Energy Slope Ordering (ESO) simply arranges the servers according to the slope of the SPEC benchmark result graph. However, this method overlooks the idle power consumption of the servers. Thus, the second algorithms, called Full-but-Last-n (FL-n), first tries to estimate the number of physical machines that is sufficient for accommodating all VMs and assumes that all machines are driven to full capacity (100% CPU utilization) except the last one which operates at n% CPU utilization. Then the algorithm finds the best machines with that assumption.

To test the performance of our algorithms, we simulated the energy consumption of the allocations that are the results of the two algorithms in CloudSim. We also determined the optimal ordering of the machines to be used by exhaustively simulated all possible ordering and selected the one with the lowest consumption. The results from our algorithm were compared with the optimal solution. The experiment results showed that The experiment results showed that the ESO algorithm performed poorly when the hosts were a mixed of an old and new generation machines, giving an allocation that used almost 50% more power than the optimal ordering in the worst case. On the other hand, the FL-n performed well, giving an allocation that used at most 5% more energy than the optimal.

This paper is divided into several sections as follows. The second section describes works related to this paper. The third section explains the energy model and system model of our algorithms, while the fourth section outlines the algorithms themselves. The fifth and sixth sections summarize the experimental setup and results respectively. Finally the last section discusses the work and offers the direction for future work.

978-1-4799-5822-1/14/$31.00 ©2014 IEEE

2014 11th International Joint Conference on Computer Science and Software Engineering (JCSSE)

II. RELATIVE WORKS

A. CloudSim

CloudSim is a Cloud simulator developed by Grid Computing and Distributed Systems (GRIDS) Laboratory from Melbourn University, Australia. It is written in Java and is easy to declare and manage cloud resources such as hosts, VMs, and data centers. It also comes with basic host/VM migration and auto-scaling algorithms. Researchers can extend CloudSim easily by replacing its standard algorithms with new ones. There are many works that use CloudSim as a testbed for their algorithms, such as [3] and [4].

B. Server Power Model

In [2], the authors compared the power consumption of servers manufactured in the year 2005 and 2007 and found that the servers built in 2005 consumed much more power at idle. Thus, the idle power contributed a large factor into energy cost of the whole data center and should be considered when trying to save energy. However, many works such as [10], [11] ignore the idle power and aim to consolidate VMs into as few hosts as possible. This method may lead to sub-optimal allocations if the data center has a mix of old and new generation machines with different power profile.

III. MODEL

A. Power Model

In [3], power model of old and new generation servers were different. During an idle state with no workload, an old generation servers used higher energy consumption than a new generation servers. Furthermore, by arbitrarily selecting an old and new generation servers from SPEC power benchmark and plotted the power consumption graph, we found that the slope of old generation servers do not differ much from new generation servers. It means that energy efficiency of old generation servers are not as good only because they use a lot of energy at idle.

The equation of the power model for a server Hi, where H_i is server or host in cloud, is shown in Fig 1. The vertical axis is e and horizontal axis is x. The equation is $e = sl_i x + b$ that is:

e is the average active power in Watt (W), which is reported in SPEC

sl_i is the slope of energy graph of the server H_i

x is a CPU utilization of server in percent (%), the value is between 0% (Idle state) to 100%

b_i is average active power in Watt (W) at idle state (CPU utilization 0%)

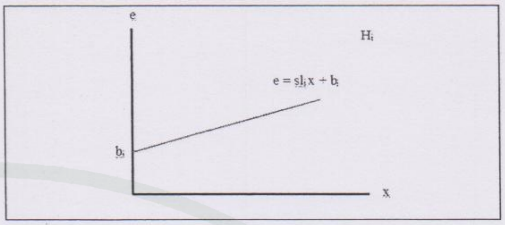


Fig. 1. Graph equation of power model

We survey the energy of the server from SPEC, in the years before and after 2007 of Host and found that the slope of the graph has the energy range between 0.4 - 0.8 as shown in Table I.

TABLE I. THE EQUATION OF POWER MODEL OF HOST FROM SPEC

Host before 2007	Host after 2007
Fujitsu Siemens Computers PRIMERGY RX300 S3 (Intel Xeon L5335)	IBM System x3200 M3
0.7309x + 191.64	0.7082x + 41.509
Fujitsu Siemens Computers PRIMERGY TX120 (Intel Xeon 3070)	IBM System x3250 M3
0.4483x + 65.214	0.7369x + 38.127
SuperMicro, Inc. Supermicro 6025B-TR+ (Intel Xeon processor L5335)	IBM Corporation System x3250 M3
0.7891x + 197.73	0.7340x + 36.336
Fujitsu Siemens Computers PRIMERGY TX150 S6 (Intel Xeon X3220)	Fujitsu PRIMERGY TX100 S3p (Intel Xeon E3-1240V2)
0.5469x + 80.664	0.5533x + 11.364
Fujitsu Siemens Computers PRIMERGY TX150 S5 (Intel Xeon 3070)	Acer Incorporated Gateway GT110 F2 (Intel Xeon E3-1270)
0.4231x + 89.9	0.8015x + 20.473
Colfax International CX2266-N2	Hitachi, Ltd. HA8000/TS10 (DL)
0.9291x + 194.09	0.7509x + 20.773

For example, the power usage in 2008 SPEC benchmark of "Fujitsu PRIMERGY TX100 S3p (Intel Xeon E3-1240V2)" [6] are plotted as in Fig 2.

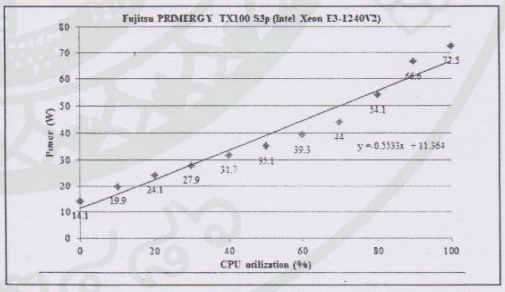


Fig. 2. Energy graph of Fujitsu Primergy TX100S3p (Intel Xeon E3-1240V2)

Each dot is the average power at each CPU utilization level. To find the slope of energy graph, we compared the linear regression line by using a function in Excel [9] that calculated the slope and the value of y-intercept.

B. System Model

1) Hosts

In this model, the number of hosts in the Cloud is given as H. Each host is denoted by H_i, where $1 <= i <= H$. The host H_i is capable of processing $MIPS_i$ million instruction per second and has RAM_i amount of memory, BW_i amount of bandwidth and ST_i amount of Storage. It also has the energy graph with slope sl_i and b_i intercept as in power model.

2) Virtual Machine (VM)

In this model, the number of VMs in the Cloud is given as M. Each VM is denoted by V_j, where $1 <= j <= M$. The VM V_j is capable of processing $MIPS_j$ million instruction per second and has RAM_j amount of memory, BW_j amount of bandwidth and ST_j amount of Storage.

3) Workload

Workload in CloudSim is declared in term of period (1 day, 2 days or 15 hours) which is the same value of simulation time. Every 300 simulated seconds, CloudSim calculates energy spent to process workload on each host. The CPU utilization on each host is calculated from the total of $MIPS_j$ of VM running on that host. For example, if the cloud has 3 VMs with 100, 100 and 200 $MIPS_j$ running on one host with $MIPS_i$ equals 500, the CPU utilization of that host will be (100 + 100 + 200)/500 = 80% and then calculated energy by using this CPU utilization in [4] to calculate power of host.

4) Allocation

Before CloudSim starts simulation, it allocates VMs onto Hosts sequentially using a simple Greedy heuristic. Firstly, the VMs and Hosts are ordered arbitrarily. Each VM is allocated to the first available Host with enough MIPS, RAM, BW, and ST to accommodate that VM. When a VM is allocated to a Host, the resources of that Host are deducted accordingly. If a VM cannot be allocated to the first host on the list, the host is removed from the list and the next one is considered. The process is repeated until there is no more VM left. Note that we may not use all Hosts in the Cloud.

IV. ALGORITHM

The purpose of our allocation is to decrease the total cost of energy consumption by all Hosts in processing the workload. Because there are many type of machines with different energy profile, if we order the Hosts to be used appropriately, an energy efficient host will be used before an inefficient one and thus should result in a better energy consumption. We developed two algorithms to order the hosts. The first one is a simple Energy Slope Ordering (ESO) Algorithm which sorts hosts according to increasing energy slopes. This algorithm allows hosts with less steep slopes, and thus better energy consumption with regards to CPU utilization, to be used first.

The second algorithm is called Full-but-Last-n (FL-n). In this algorithm, we first estimate x, the number of hosts that would be used in the allocation, by adding up the capacity of hosts until the sum is greater than the sum of all VM capacity. We assume that all hosts but the last one would be used to a full capacity (resulting in 100% CPU utilization, while the last host used n% utilization. Consequently, the sorting step will select x-1 hosts with increasing energy consumption at 100% capacity. We then sort the rest of the hosts with increasing the energy consumption at n% capacity. The pseudo-code of FL-n algorithm is shown in Table II.

Comparing the two algorithms, we can see that the Energy Slope Ordering is simpler, but it does not take into account the energy consumption of host at idle (which is the y-intercept of the energy graph). The Full-but-Last-n does consider the energy consumption at idle, but it may give sub-optimal allocation if it mis-predicts the number of hosts used. The value of n is arbitrary set, and may also impact the quality of the solution provided FL-n too.

TABLE II. PROCESS OF ALGORITHM FL-N

BEGIN

1. Order Hosts by ascending sl_i and save in *HOST_LIST*
2. Calculate *ALL_MIPS_VM* = $\sum_j VM\_MIPS_j$, $1 <= j <= M$
3. Find *X_MIPS* where *X_MIPS* is the minimum integer that satisfies $\min_i \sum_i H\_MIPS_i >= ALL\_MIPS\_VM$, where $1 <= X\_MIPS <= H$. Thus, *X_MIPS* is the minimum number of hosts to match the sum of MIPS of all VMs.
4. Repeat step 2 and 3 to find *X_RAM*, *X_BW*, and *X_ST*
5. Calculate X = MAX(*X_MIPS*, *X_RAM*, *X_BW*, *X_ST*). X is the minimum number of hosts to serve all VMs.
6. Order hosts by ascending energy consumption at 100% select the first X-1 hosts and save them into *NEW_HOST_LIST*.
7. Order the rest of the hosts by ascending energy consumption at n% and append them to *NEW_HOST_LIST*
8. Return *NEW_HOST_LIST*

END

V. EXPERIMENT

A. The Optimal Ordering

To evaluate our algorithms, we compare the total energy used by both our orderings with an optimal ordering. The optimal ordering is the one with the lowest total energy used when simulated in CloudSim. The optimal ordering is found by an exhaustive search; testing the energy used of every possible ordering. For example, if we have 4 hosts, all possible ordering in Table III are tested to find the one with the lowest energy consumption. The energy used by our orderings is then compared with the optimal ordering as percentage over the optimal.

TABLE III. ALL POSSIBLE ORDERING OF 4 HOSTS

[1, 2, 3, 4]	[3, 1, 2, 4]
[1, 2, 4, 3]	[3, 1, 4, 2]
[1, 3, 2, 4]	[3, 2, 1, 4]
[1, 3, 4, 2]	[3, 2, 4, 1]
[1, 4, 2, 3]	[3, 4, 1, 2]
[1, 4, 3, 2]	[3, 4, 2, 1]
[2, 1, 3, 4]	[4, 1, 2, 3]
[2, 1, 4, 3]	[4, 1, 3, 2]
[2, 3, 1, 4]	[4, 2, 1, 3]
[2, 3, 4, 1]	[4, 2, 3, 1]
[2, 4, 1, 3]	[4, 3, 1, 2]
[2, 4, 3, 1]	[4, 3, 2, 1]

B. Experiment Setup

The slope and intercept of the energy profile of each host is randomly selected. We differentiate between two types of host, a new and an old generation host. The slope of a new generation host is randomly determined to be between 0.4 to 0.8, while its intercept is randomly determined to be between 0 to 40. On the other hand, the slope of an old generation host is randomly selected between 0.4 to 0.8, while the intercept is randomly selected between 40 to 80. According to our quick check of the SPEC energy benchmark, the energy slope of the old and new hosts is roughly the same while the power consumption at idle is vastly different between the old and new hosts. Table IV summarizes the randomly selected parameters of energy profile.

TABLE IV. SUMMARY THE RANDOM PARAMETERS OF HOST

Host Type	sl_i	b
Old Generation	0.4 – 0.8	50 – 80
New Generation	0.4 – 0.8	20 – 40

Each VM is randomly selected to use between 1000 to 2000 MIPS while the MIPS of a host is randomly selected to be between 3000 to 4000. There is no difference between the MIPS of old and new generation machines. Furthermore, we do not consider RAM, BW, and ST of VMs and Hosts. Since these parameters are just additional dimensions in our algorithms (in addition to MIPS), leaving them out simplify the experiment without causing effects on the result of the algorithms.

The experiment is set up with 10 VMs and 4, 5 and 6 Hosts respectively. We have 3 types of host mix: all new generation hosts, all old generation hosts, and mix generation hosts. The number of old and new hosts in each host mix for 4, 5, and 6 hosts are shown in Table V. A scenario is a particular combination of VM and host mix, for example, 10 VMs, 6 hosts, all new generation (a scenario corresponds to a row in Table V). One sample in a scenario is an experiment setup with randomly selected parameters according to that scenario. Consequently, two samples belonging to the same scenario have the same number of VMs and hosts, but each host and VM may have different MIPS and energy profile.

For each scenario, we randomly selected 40 samples. With each sample, the energy used by optimal ordering is found by exhaustive search and simulation. In the same sample we also simulate to find the energy consumption of ESO, FL-30 and FL-60 algorithms. The energy used by our orderings is then compared with the optimal ordering as percentage over the optimal. The results of 40 samples are then averaged to determine the effectiveness of our algorithms in that scenario. We then change the scenario and repeat the process until all scenarios are examined.

VI. RESULTS

In this experiment, we set on Intel Core 2 Duo processor T7300 2.00 GHz, RAM 3 GB, HDD 120 GB and OS Microsoft Windows 7 Professional. One iteration of testing, we random MIPS of 10 VMs, random MIPS of all Hosts, random slope graph power of all Hosts and idle power of all Hosts. In this iteration, we run 2 times - to find a result from algorithm and to find an optimum result. The details of experiment are in Table V.

TABLE V. THE DETAILS OF SET UP EXPERIMENT

Number of Hosts for testing	Number of random slope graph power of Hosts			Total VMs
	Host Mix	Old	New	
4	Old	4	0	10
	New	0	4	
	Mix	2	2	
5	Old	5	0	10
	New	0	5	
	Mix	2	3	
6	Old	6	0	10
	New	0	6	
	Mix	3	3	

In Fig 3, 4 and 5 shows the results of the Energy Slope Ordering algorithm. The vertical axis of the graphs show the average percentage (over 40 samples) of the energy used by each algorithm over the optimal ordering. The horizontal axis represent each scenario in the experiment setup. The Energy Slope Ordering gave good results when hosts in data centers are from similar generation, providing solutions with energy consumption no more than 10% on the average over the optimal ordering. Furthermore, the ESO algorithm was equally effective in both data centers with all old generation machines and all new generation machines, providing allocations with less than 2% difference on the average.

However, when servers are from different generations, the ESO algorithm fared pretty badly, giving out orderings that used 30% more energy than the optimal ordering on the average and up to almost 50% more in the case of 10 VM - 6 hosts. This is because new servers and old servers have different idle power, which the ESO algorithm does not take into account. In addition, when the number of hosts were increased, the ESO algorithm worsen. This effect did not materialize when hosts were from the same generation.

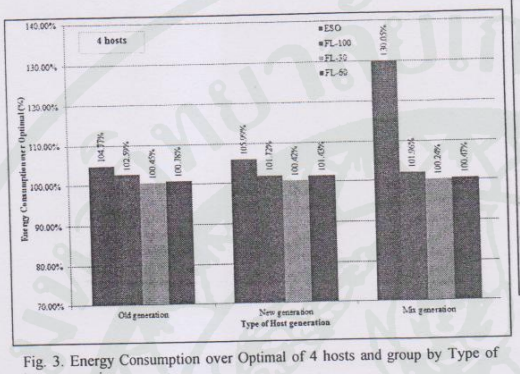


Fig. 3. Energy Consumption over Optimal of 4 hosts and group by Type of host generation

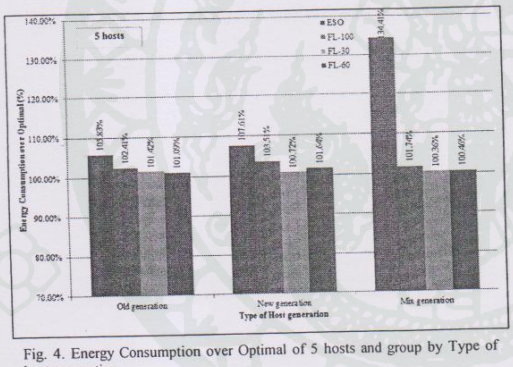


Fig. 4. Energy Consumption over Optimal of 5 hosts and group by Type of host generation

Fig 3, 4 and 5 show the result of Full-but-Last-n where n equals 100%, 30% and 60% respectively. It is clear from the Figures that FL-n performed very well, giving orderings that consume energy no more than 103% over the optimal ordering on average. Furthermore, there is no evidence of poor performance from the algorithm when the number of hosts increase. In addition, there is not much performance difference when we changed the value n from 30% to 60% too.

To evaluate the effect of the value n in FL-n algorithm, we also conducted the same experiment where n = 100% (FL-100). In FL-100 algorithm, we assume that hosts used for allocation will use all their capacity at 100% and thus using maximum power. The results in Fig 3, 4 and 5 show that FL-100 had a slight performance decrease of around 1-5% from FL-30 due to this over-estimation.

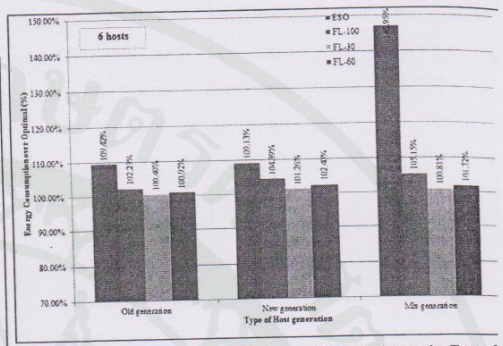


Fig. 5. Energy Consumption over Optimal of 6 hosts and group by Type of host generation

Table VI, VII, VIII and IX show the accuracy of the ESO, FL-100, FL-30 and FL-60 to find the optimal ordering. The accuracy is defined as the percentage that an algorithm produces the same ordering as the optimal ordering. The ESO algorithm did very badly, finding the optimal solution less than 20% of the time in mix generation servers. The FL-n was much better and found the optimal ordering unto 80% of the time in 10 VMs - 4 mixed hosts scenarios. Interestingly, the FL-n algorithm was better at producing the optimal solution when the data center has mixed generation hosts. Furthermore, FL-30 was better than FL-100 and FL-60 at finding the optimal ordering also.

TABLE VI. ACCURACY OF ESO ALGORITHM

The average of Energy Consumption over Optimal (%)								
New	*Old*	*Mix*	*New*	*Old*	*Mix*	*New*	*Old*	*Mix*
105.99	104.77	130.05	107.61	105.83	134.41	109.13	109.42	146.9
The standard deviation of Energy Consumption over Optimal (%)								
5.89	6.03	21.47	6.32	5.12	23.62	7.32	7.35	28.83
The algorithm accuracy (%)								
25.0	42.5	20.0	17.50	10.00	12.50	15.00	5.00	5.0
4 Hosts			**5 Hosts**			**6 Hosts**		

TABLE VII. ACCURACY OF FL-N ALGORITHM AT N IS 100%

The average of Energy Consumption over Optimal (%)								
New	*Old*	*Mix*	*New*	*Old*	*Mix*	*New*	*Old*	*Mix*
101.72	102.59	101.96	103.51	102.41	101.74	104.39	102.23	105.15
The standard deviation of Energy Consumption over Optimal (%)								
3.18	3.99	5.61	4.53	3.53	3.93	5.06	3.25	7.32
The algorithm accuracy (%)								
62.5	45.0	70.0	42.50	30.00	60.00	25.00	40.00	40.00
4 Hosts			**5 Hosts**			**6 Hosts**		

TABLE VIII. ACCURACY OF FL-N ALGORITHM AT N IS 30%

The average of Energy Consumption over Optimal (%)								
New	*Old*	*Mix*	*New*	*Old*	*Mix*	*New*	*Old*	*Mix*
100.42	100.45	100.24	100.72	101.42	100.36	101.26	100.40	100.81
The standard deviation of Energy Consumption over Optimal (%)								
0.90	1.04	0.60	1.46	2.98	1.04	2.02	1.24	1.81
The algorithm accuracy (%)								
75.0	70.0	80.0	67.50	50.00	70.00	50.00	75.00	65.00
4 Hosts			**5 Hosts**			**6 Hosts**		

TABLE IX. ACCURACY OF FL-N ALGORITHM AT N IS 60%

The average of Energy Consumption over Optimal (%)								
New	*Old*	*Mix*	*New*	*Old*	*Mix*	*New*	*Old*	*Mix*
101.43	100.78	100.47	101.64	101.09	100.46	102.40	100.92	101.72
The standard deviation of Energy Consumption over Optimal (%)								
2.83	1.79	1.50	2.58	1.58	1.25	3.23	1.54	2.45
The algorithm accuracy (%)								
62.5	67.5	77.5	55.0	42.50	70.00	37.5	52.50	47.50
4 Hosts			**5 Hosts**			**6 Hosts**		

VII. DISCUSSIONS

This paper proposed two algorithms to select the order of hosts to be used for accommodating virtual machines in the Cloud computing environment such that the overall energy consumption is lowest. Each host has an energy profile in form of a graph which is obtainable using SPEC Power benchmark. The Energy Slope Ordering algorithm orders hosts according to its energy slope graph, while the Full-but-Last-n considers the idle power of hosts too when ordering. The experiment results shows that the ESO algorithm performed poorly when hosts are from mixed generations with different idle power, while FL-n performed very well in all situations.

One interesting question is whether the estimated CPU utilization of the last host (n) has any effects on the quality of solution of FL-n or not. From the experiment, there seems to be not much different, but it should be verified with more values of n and more scenarios. Furthermore, the optimal ordering found may not be the allocation with lowest power consumption because there might be a better allocation that uses more hosts but with low utilization on each host. If that is the case, a new algorithm that does not use Greedy Heuristic should be developed.

REFERENCES

[1] Rajkumar Buyya, Rajiv Ranjan, Rodrigo N. Calheiros. 2010. Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities. Pontifical Catholic University of Rio Grande do Sul Porto Alegre, Brazil.

[2] Luiz André Barroso, Urs Hölzle. 2007. The Case for Energy-Proportional Computing. the IEEE Computer Society: 0018-9162/07.

[3] Shi, Y., Jiang, X., & Ye, K. (2011, September). An energy-efficient scheme for cloud resource provisioning based on cloudsim. In Cluster Computing (CLUSTER), 2011 IEEE International Conference on (pp. 595-599). IEEE.

[4] Li, X., Jiang, X., Ye, K., & Huang, P. (2013, June). DartCSim+: Enhanced CloudSim with the Power and Network Models Integrated. In Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on (pp. 644-651). IEEE.

[5] Anton Beloglazov, Rajkumar Buyya. 2011. Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers. CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE Concurrency Computat.: Pract. Exper. 2011; 00:1–24.

[6] Peter Mell, Timothy Grance. 2011. The NIST Definition of Cloud Computing. NIST Special Publication: 800-145.

[7] Standard Performance Evaluation Corporation. 1995. SPECpower_ssj2008 Results. http://www.spec.org/power_ssj2008/results/, 1995 - 2012 Standard Performance Evaluation Corporation.

[8] Anton Beloglazov, Rajkumar Buyya. 2010. Energy Efficient Resource Management in Virtualized Cloud Data Centers. 978-0-7695-4039-9/10, 2010, 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing.

[9] Physics Tutorial: Linear Regression. 2006. Jerry Hester. http://www.clemson.edu/ces/phoenix/tutorials/regression/index.html, 2006 Clemson University. All Rights Reserved. Photo's Courtesy Corel Draw.

[10] Xiao, Z., Song, W., & Chen, Q. 2013. Dynamic resource allocation using virtual machines for cloud computing environment. Parallel and Distributed Systems, IEEE Transactions on, 24(6), 1107-1117.

[11] Roy, N., Dubey, A., & Gokhale, A. (2011, July). Efficient autoscaling in the cloud using predictive models for workload forecasting. In Cloud Computing (CLOUD), 2011 IEEE International Conference on (pp. 500-507). IEEE.

ประวัติการศึกษา และการทำงาน

ชื่อ-นามสกุล	นางสาวมนพัทธ์ ลิ้มรัตนศิลป์
วัน เดือน ปีที่เกิด	11 ตุลาคม 2528
สถานที่เกิด	เขตป้อมปราบศัตรูพ่าย จังหวัดกรุงเทพมหานคร
ประวัติการศึกษา	วท.บ. (วิทยาการคอมพิวเตอร์) มหาวิทยาลัยเกษตรศาสตร์
ตำแหน่งหน้าที่การงานปัจจุบัน	ซอฟต์แวร์ เอ็นจิเนียริ่ง
สถานที่ทำงานปัจจุบัน	บริษัท แอคเซนเจอร์ โซลูชั่น จำกัด
ทุนการศึกษาที่ได้รับ	ทุนบัณฑิตศึกษา คณะวิทยาศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ (พ.ศ. 2552 – 2554)