# บทที่ 2

# ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

บทนี้จะกล่าวถึง ทฤษฎีและงานวิจัยที่เกี่ยวข้องในวิทยานิพนธ์ฉบับนี้ส่วนของทฤษฎีที่นำมาใช้ในการดำเนินการวิจัย ประกอบไปด้วย ทฤษฎีในของเวอร์ชวลไลเซชั่น และงานวิจัยที่เกี่ยวข้อง

ส่วนของงานวิจัยที่นำมาเป็นแนวทางการศึกษาเพื่อทำวิจัยนั้น ประกอบไปด้วยงานวิจัยและบทความที่มีการประยุกต์การปรับปรุงการทำเช็คพอยต์ งานวิจัยของเซนที่เป็นเวอร์ชวลไลเซชั่นในอีกแนวทาง รวมถึงงานวิจัยของโปรแกรมคีมูที่เป็นพื้นฐานในการพัฒนาต่อยอดของงานวิจัยนี้ ดังจะแสดงรายละเอียดของทฤษฎี หลักการ และแนวทางการศึกษาเพื่อทำงานวิจัยชิ้นนี้ ดังต่อไปนี้

## 2.1 ทฤษฎีที่เกี่ยวข้อง

### 2.1.1 เทคโนโลยีของเวอร์ชวลไลเซชั่น

เทคโนโลยีของเวอร์ชวลไลเซชั่น (Jones, 2006) เป็นเทคโนโลยีที่ช่วยเพิ่มความสามารถให้กับฮาร์ดแวร์ ซึ่งจะทำให้สามารถใช้งานระบบปฏิบัติการหลาย ๆ ระบบได้พร้อม ๆ กันบนเครื่องคอมพิวเตอร์เดียวกันได้ เทคโนโลยีนี้เป็นการสร้างสภาพแวดล้อมจำลองของเครื่องคอมพิวเตอร์ขึ้นมา โดยที่สภาพแวดล้อมที่ถูกจำลองขึ้นจะทำงานอยู่ภายใต้ฮาร์ดแวร์นั้น โดยมีซอฟต์แวร์เวอร์ชวลไลเซชั่น ที่จะช่วยจำลองและสร้างส่วนควบคุมที่เชื่อมต่อกับฮาร์ดแวร์เพื่อทำหน้าที่เสมือนเครื่องคอมพิวเตอร์อีกเครื่องหนึ่ง ซึ่งระบบการทำงานของเครื่องเสมือนที่มีระบบปฏิบัติการทำงานอยู่แต่ละเครื่องนั้น จะถูกเรียกว่า เวอร์ชวลแมชชีน (Virtual Machine) โดยมีการจำลองเพื่อสร้างการทำงานในลักษณะของการจัดสรรทรัพยากรต่าง ๆ บนเครื่องคอมพิวเตอร์ให้กับเวอร์ชวลแมชชีนนั้น เช่น การจำลองหน่วยประมวลผล, หน่วยความจำหลัก, การติดต่อผ่านเครือข่าย และอุปกรณ์อินพุต/เอาต์พุตต่าง ๆ ในส่วนซอฟต์แวร์ เวอร์ชวลไลเซชั่นจะรวมถึงระบบปฏิบัติการ และโปรแกรมต่างๆ ซึ่งจะทำงานอยู่ในส่วนบนสุดของเวอร์ชวลแมชชีน ซึ่งการ

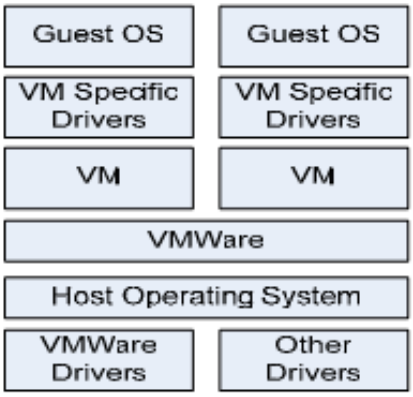
ทำงานของซอฟต์แวร์เวอร์ชวลแมชชีนต่าง ๆ สามารถที่จะทำงานพร้อมกันได้ในเครื่องเดียวกันโดยจะขอการเข้าใช้ทรัพยากรบนเครื่องคอมพิวเตอร์ร่วมกัน ซึ่งจะมีโปรแกรมที่ช่วยในการจัดการและควบคุมการทำงานของเวอร์ชวลแมชชีน ซึ่งแบ่งออกเป็นเทคโนโลยีที่นิยมนำไปใช้หลัก ๆ ได้แก่

2.1.1.1 **เนทีฟเวอร์ชวลไลเซชั่น**

เป็นการจำลองเครื่องโดยสร้างระบบเตรียมไว้สำหรับที่จะสามารถนำเอาระบบปฏิบัติการอื่น ๆ มาติดตั้งและสามารถที่จะทำงานอยู่ภายใต้เวอร์ชวลไลเซชั่นบนเครื่องคอมพิวเตอร์เดียวกันได้ ซึ่งเราจะเรียกว่าระบบปฏิบัติที่มาติดตั้งเพิ่มเติมนี้ว่า เกสโอเอส (Guest OS) โดยที่ระบบปฏิบัติการที่ได้นำเอามาทำงานนั้นไม่จำเป็นต้องเปลี่ยนแปลงหรือแก้ไขการทำงานของระบบปฏิบัติการนั้นภายใน ซึ่งคำสั่งงานที่มาจากระบบปฏิบัติการ (เกสโอเอส) จะถูกซอฟต์แวร์เวอร์ชวลไลเซชั่นจัดการในการจัดการข้อมูลให้ รวมถึงการทำงานส่วนต่าง ๆ เสมือนระบบปฏิบัติการทำงานบนสภาวะแวดล้อมเสมือนจริง วิธีการมีข้อดีเนื่องจากการเชื่อมต่อของซอฟต์แวร์และระบบปฏิบัติการจะทำงานผ่านเวอร์ชวลไลเซชั่น โดยจะไม่ขึ้นกับฮาร์ดแวร์ จึงสามารถนำไปดำเนินงานยังเครื่องคอมพิวเตอร์เครื่องอื่นที่มีฮาร์ดแวร์แตกต่างกันได้ รวมถึงการนำเอาระบบปฏิบัติการที่มีคุณสมบัติที่แตกต่างกันมาทำงานร่วมกัน โดยไม่เกิดความขัดแย้งในการทำงานของเวอร์ชวลแมชชีนได้ ตัวอย่างของซอฟต์แวร์เวอร์ชวลไลเซชั่น ที่ใช้เทคนิคนี้ก็คือ Microsoft Virtual Server และ VMware ESX Server แต่เนื่องจากซอฟต์แวร์จะต้องมีการเตรียมข้อมูลตัวรับรู้ หน่วยความจำหลัก และอุปกรณ์ต่อพ่วงอื่น ๆ จึงอาจพบความล่าช้า เนื่องจากซอฟต์แวร์เวอร์ชวลไลเซชั่นจะต้องมีการแปลงระหว่างคำสั่งงานไปมาระหว่างเครื่องคอมพิวเตอร์และเวอร์ชวลแมชชีน และซอฟต์แวร์เวอรชวลไลเซชั่นจะต้องสร้างและคอยจัดการโครงสร้างข้อมูลต่าง ๆ อีกด้วย

ภาพที่ 1

สถาปัตยกรรมของเวอร์ชวลไลเซชั่นของ VMWare



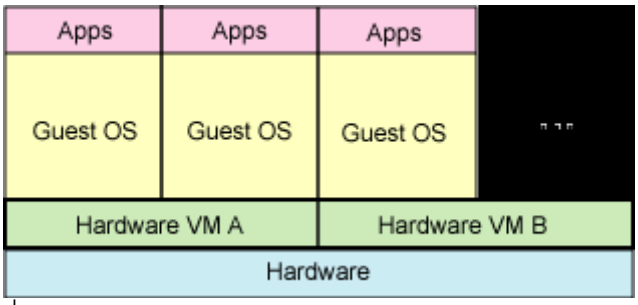

ที่มา: http://science.rbru.ac.th/paper/index.php?topic=4

จากภาพที่ 1 จะแสดงให้เห็นถึงสถาปัตยกรรมของวีเอ็มแวร์ที่เวอร์ชวลแมชชีนที่มีเกสโอเอสจะทำงานอยู่บนโปรแกรมขับอุปกรณ์ของเวอร์ชวลแมชชีนนั้น โดยที่ระบบปฏิบัติการของเครื่องแม่งานจะมีโปรแกรมขับอุปกรณ์ของวีเอ็มแวร์ด้วยเช่นเดียวกัน

2.1.1.2 **อีมูเลชั่นเวอร์ชวลไลเซชั่น**

เป็นการทำงานบนเวอร์ชวลแมชชีน โดยซอฟต์แวร์เวอร์ชวลไลเซชั่น ที่จำลองการทำงานของเครื่องคอมพิวเตอร์ใด ๆ ขึ้นมา ทำให้สามารถใช้งานระบบปฏิบัติการนั้นบนหน่วยประมวลผลอีกตัวหนึ่งได้ โดยส่วนใหญ่เป็นวิธีการที่ใช้กันมายาวนาน หลายครั้งถูกนำมาใช้เพื่อทำให้สามารถใส่ซอฟต์แวร์ตัวใหม่เข้าไปในหน่วยประมวลผลตัวใหม่ได้ ก่อนที่จะทำงานอื่น ๆ ต่อไป โดยจะมีการทดลองและทดสอบการทำงานของระบบปฏิบัติการนั้น

ภาพที่ 2
โครงสร้างของเวอร์ชวลไลเซชั่นแบบอีมูเลชั่น




ที่มา: http://www.ibm.com/developerworks/library/l-linuxvirt/

จากภาพที่ 2 แสดงถึงโครงสร้างของเวอร์ชวลไลเซชั่นในรูปแบบอีมูเลชั่นที่ซอฟต์แวร์ จะมีการจำลองฮาร์ดแวร์ของเครื่องคอมพิวเตอร์บนเวอร์ชวลแมชชีนของ A และ B โดยจะมีระบบปฏิบัติการ หรือเกส โอเอสทำงานผ่านการทำงานเวอร์ชวลแมชชีน A และ B เพื่อติดต่อกับฮาร์ดแวร์ของเครื่องคอมพิวเตอร์โดยตรง

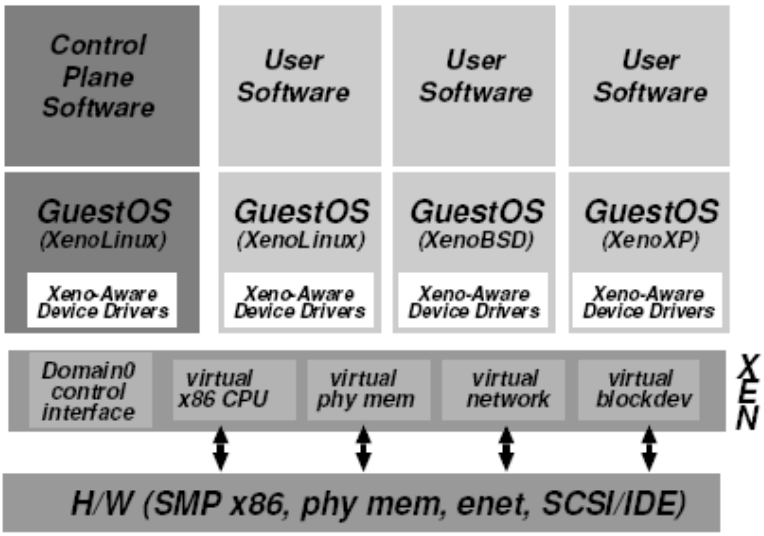
**2.1.1.3 พาราเวอร์ชวลไลเซชั่น**

เป็นอีกเทคนิคหนึ่งในการทำเวอร์ชวลไลเซชั่น มีลักษณะคล้ายคลึงกับเทคนิคในรูปแบบเนทีฟเวอร์ชวลไลเซชั่น โดยมีสิ่งที่แตกต่างกันจะเป็นในส่วนที่เวอร์ชวลแมชชีนนั้นสามารถระบุไปถึงการทำงานของฮาร์ดแวร์ได้ในทันที ซึ่งจะทำให้การทำงานของเวอร์ชวลแมชชีนมีประสิทธิภาพที่ได้จะใกล้เคียงกับประสิทธิภาพของระบบปฏิบัติการที่ทำงานในรูปแบบปกติ

Xen (Barham, Dragovic, Fraser, Hand, Harris, Ho, Neugebauery, Pratt and War_eld, 2003) เป็นเวอร์ชวลไลเซชั่นซอฟต์แวร์ ซึ่งเป็นโอเพ่นซอร์ส (Opensource) พัฒนาอยู่บนพื้นฐานเทคนิคแบบพาราเวอร์ชวลไลเซชั่น โดยมีสถาปัตยกรรมตามภาพที่ 3 โดยจะมีส่วนของเซนไฮเปอร์ไวเซอร์ (Hypervisor), เกสโอเอส (Guest OS) ที่เป็นคนละประเภท ซึ่งรวมถึงโดเมน 0 ที่มีซอฟต์แวร์ที่ทำหน้าที่จัดการของเซน ผ่านเซโนลินิกซ์ (Xeno Linux)

ภาพที่ 3

โครงสร้างของเซน และส่วนต่าง ๆ




ที่มา: http://www.cl.cam.ac.uk/research/srg/netos/papers/2003-**xen**sosp.pdf

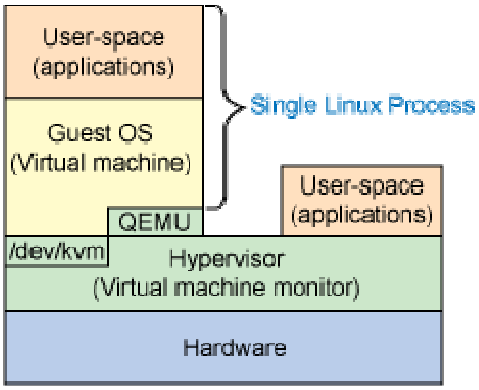
ภาพที่ 3 แสดงถึงโครงสร้างของเวอร์ชวลไซชั่นซอฟต์แวร์เซน ที่มีการแบ่งการทำงานในส่วนของการแก้ไขเกสโอเอส ในส่วนของโปรแกรมขับอุปกรณ์ให้สามารถทำงานผ่านเซนซึ่งจะมีโดเมนศูนย์เป็นส่วนที่ควบคุม และมีหน่วยประมวลผล หน่วยความจำ เครือข่าย รวมถึงหน่วยเก็บข้อมูลจำลองต่างๆ ซึ่งจะทำการสั่งงานไปยังฮาร์ดแวร์เครื่องคอมพิวเตอร์ โดยเกสโอเอสมีทั้ง ลินุกซ์ รวมถึงวินโดว์ด้วย

คีมู (Qemu) (Bellard and Jones, 2006) เป็นเวอร์ชวลไลเซชั่นซอฟต์แวร์ ที่เป็นลักษณะของการจำลองการทำงานของเครื่องคอมพิวเตอร์ ซึ่งจะทำให้การทำงานของเกสโอเอสนั้นไม่จำเป็นต้องมีการแก้ไขเปลี่ยนแปลงการทำงานหรือคำสั่งภายใน โดยคีมูมีความสามารถในการแปลงคำสั่งของเกสโอเอสกับฮาร์ดแวร์ โดยการแปลงชุดคำสั่งเพื่อทำให้เกสโอเอสและฮาร์ดแวร์สามารถทำงานร่วมกันได้ ดังนั้นจึงทำให้สามารถนำเอาเกสโอเอสไปทำงานบนเครื่องอีกประเภทได้ เช่น เราสามารถนำเกสโอเอสวินโดว์ ไปดำเนินงานยังเครื่องที่ใช้หน่วยประมวลผลพาวเวอร์พีซี ได้

เควีเอ็ม (KVM) (Jones, 2007) เป็นเวอร์ชวลไลเซชั่นซอฟต์แวร์ที่เป็นมีลักษณะการทำงานเป็นแบบพาราเวอร์ชวลไลเซชั่น โดยที่มีการนำเอาคีมูเข้ามาใช้ในการจัดการอินพุต/เอาต์พุต (Input/Output) และมีการจัดการหน่วยความจำให้ โดยที่แต่ละเกสโอเอสจะมีการหน่วยความจำจัดสรรไว้

ภาพที่ 4

โครงสร้างของเควีเอ็ม ที่มีคีมูทำงานเป็นส่วนหนึ่ง




ที่มา: http://www.ibm.com/developerworks/linux/library/l-linux-kvm/

จากภาพที่ 4 แสดงในส่วนของโครงสร้างของเควีเอ็ม ซึ่งมีการพัฒนาขึ้นมาจากคีมู โดยในส่วนของเวอร์ชวลแมชชีนนั้นจะมีในส่วนของการติดต่อกับฮาร์ดแวร์ (Hypervisor) และ มีส่วนของการติดต่อผ่านฮาร์ดแวร์โดยตรง (/dev/kvm) ซึ่งเกสโอเอสจะทำงานอยู่บนเวอร์ชวลแมชชีนนั้น

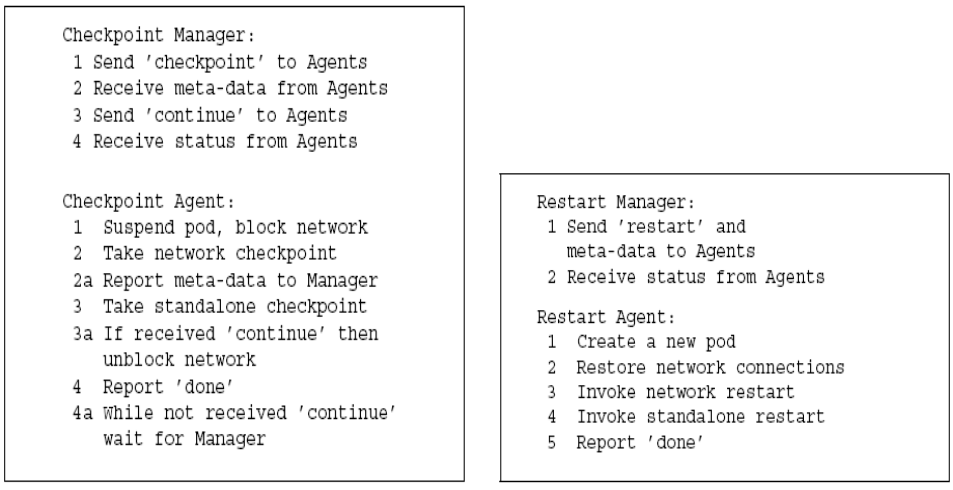
## 2.2 งานวิจัยที่เกี่ยวข้อง

ในส่วนของงานวิจัยและบทความที่เกี่ยวข้องกับวิทยานิพนธ์ฉบับนี้ ประกอบไปด้วยงานวิจัยและบทความที่เกี่ยวข้องกับการทำงานของเช็คพอยต์ของเวอร์ชวลไลเซชั่น

### 2.2.1 งานวิจัยเรื่องการซ่อนการทำเช็คพอยต์และรีสตาร์ตของโปรแกรมประยุกต์ผ่านกลุ่ม

จากงานวิจัยเรื่อง Transparent Checkpoint-Restart of Distributed Applications on Commodity Clusters (Laadan and Nieh, 2007) ได้มีการแสดงการทำงานของ ZapC โดยใช้เวอร์ชวลเซชั่นซอฟต์แวร์พอด (POD) ซึ่งจะทำการแบ่งการทำงานในการทำเช็คพอยต์ออกเป็น 3 ส่วน ซึ่งได้แก่ การทำเช็คพอยต์ในระดับเครื่องเดี่ยว การทำเช็คพอยต์ในระดับเครือข่าย ในแต่ละโหนดของโปรแกรมประยุกต์ และส่วนของการจัดการการทำงานร่วมกันในการทำเช็คพอยต์

ภาพที่ 5
ขั้นตอนวิธีของการทำงานของเช็คพอยต์ของตัวแทนที่ทำหน้าที่ควบคุมในการทำการเช็คพอยต์



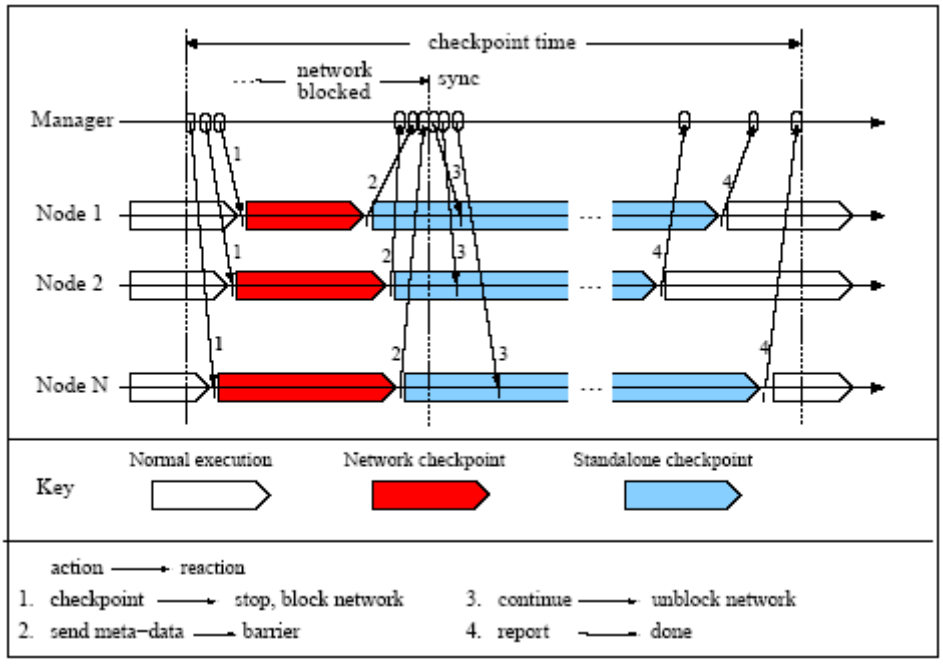

ที่มา: http://www.ncl.cs.columbia.edu/publications/cluster2005_fordist.pdf

จากภาพที่ 5 แสดงถึงขั้นตอนการทำงานของทำเช็คพอยต์ในระบบคลัสเตอร์โดยเครื่องที่สั่งเช็คพอยต์จะสั่งงานผ่านตัวแทน ซึ่งตัวแทนจะทำการหยุดการทำงานของพอด (POD) และ

หยุดการติดต่อผ่านระบบเครือข่าย จากนั้นจึงเช็คพอยต์ระบบเครือข่าย แล้วจึงส่งข้อมูลให้กับเครื่องที่สั่งงาน จากนั้นก็ทำการเช็คพอยต์แต่ละเครื่องในระบบคลัสเตอร์ เมื่อเครื่องที่สั่งงานได้รับผลการเช็คพอยต์แล้วจะสั่งให้ตัวแทนทำงานต่อ ซึ่งเมื่อได้รับคำสั่งแล้วก็จะทำการเริ่มการติดต่อผ่านระบบเครือข่าย แล้วรายงานผลว่าเสร็จเรียบร้อย ขณะเดียวกันเครื่องที่สั่งงานก็จะได้รายงานสถานะจากตัวแทน และในขั้นตอนของการรีสตาร์ต เครื่องที่สั่งงานจะส่งข้อมูลและคำสั่งรีสตาร์ตให้แก่ตัวแทน จากนั้นตัวแทนจะสร้างพอดขึ้นมาใหม่ แล้วทำการสร้างการเชื่อมต่อผ่านเครือข่ายขึ้นมาจากข้อมูลเดิม แล้วจึงรีสตาร์ตระบบเครือข่ายจากนั้นจึงรีสตาร์ตเครื่องคอมพิวเตอร์แต่ละเครื่องแล้วจึงรายงานผล ซึ่งจะมีเส้นเวลาจากที่ได้อธิบายในเบื้องต้นในภาพที่ 6

ภาพที่ 6

เส้นเวลาของโคออนิเนตเช็คพอยต์ในการทำงาน



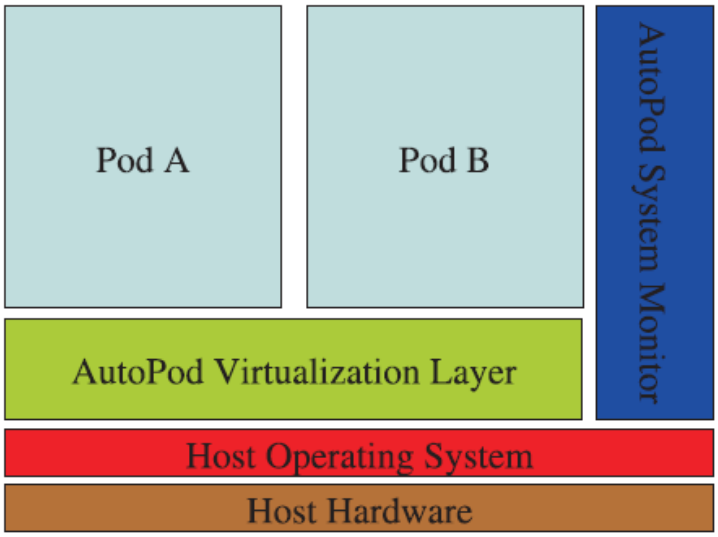

ที่มา: http://www.ncl.cs.columbia.edu/publications/cluster2005_fordist.pdf

### 2.2.2 งานวิจัยเรื่องการลดระยะเวลาของการหยุดทำงานของระบบเพื่อใช้ในการปรับปรุงดูแลรักษา

จากงานวิจัยเรื่อง Reducing Downtime due to System Maintenance and Upgrades มีการสร้างระบบ AutoPod ซึ่งเป็นซอฟต์แวร์เวอร์ชวลไลเซชั่นที่พัฒนาโดยพื้นฐานหลักจาก POD (Potter and Nieh, 2005)

ภาพที่ 7

แบบจำลองของ AutoPod




ที่มา: www.ncl.cs.columbia.edu/publications/lisa2005_fordist.pdf

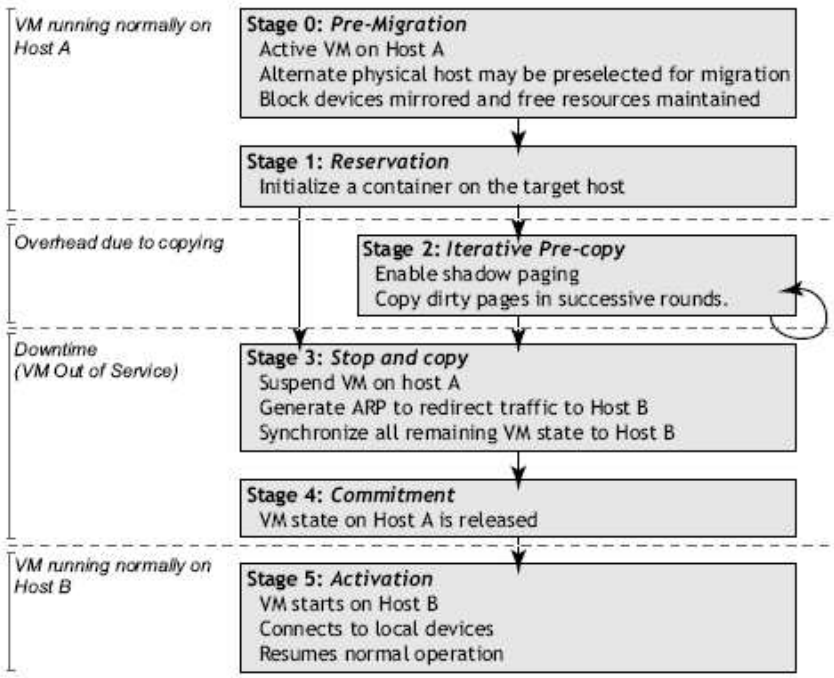
โดยการรวมเอากระบวนการต่าง ๆ ให้ทำงานอยู่ภายใน POD (Process Domain) ซึ่งแต่ละพอดจะมีชื่อเสมือนกำหนดไว้ การทำงานในการไมเกรชั่นจาก AutoPod ไปยังเครื่องหนึ่งนั้นจะมีการควบคุมการทำงานภายใน Pod ให้สามารถทำงานต่อไปได้ ดังภาพที่ 7 นั้นแสดงถึงแบบจำลองของ AutoPod ที่จะมีเลเยอร์ของ AutoPod เพื่อรองรับการทำงานของ Pod A และ Pod B ที่จะมีระบบมอนิเตอร์ของ AutoPod คอยควบคุมและสั่งงาน ซึ่งเนื่องจากเป็นการกำหนดการทำงานของโปรเซสภายใต้ Pod ทำให้สามารถนำเอาโปรแกรมประยุกต์ไปทำงานต่อยังเครื่องคอมพิวเตอร์เครื่องอื่นได้ จึงทำให้เครื่องคอมพิวเตอร์เครื่องเดิมที่เคยทำงานอยู่สามารถทำการปรับปรุงดูแลรักษาได้ โดยไม่กระทบต่อการให้บริการของระบบมากนัก

### 2.2.3 งานวิจัยเรื่องไลฟ์ไมเกรชั่นบนเวอร์ชวลแมชชีน

จากงานวิจัยเรื่อง Live Migration of Virtual Machines (Clark, Fraser, Hand, Hansen, Jul, Limpach, Pratt and Warfield, 2005) ซึ่งเซนได้นำเสนอวิธีการทำไลฟ์ไมเกรชั่นที่จะช่วยลดระยะของการหยุดให้บริการของเครื่องได้ โดยมีการแบ่งขั้นตอนของทำงานของไลฟ์ไมเกรชั่นในการส่งข้อมูลออกเป็นชุดย่อย ซึ่งจะมีทั้งหมด 5 ขั้นตอน โดยจะมีการทำงานดังภาพที่ 8 ที่จะมีการแบ่งการทำงานในช่วงแรกจะทำการเตรียมข้อมูลของเวอร์ชวลแมชชีนก่อนที่จะทำการส่งข้อมูลไปยังเครื่องปลายทาง จากนั้นจึงเริ่มทำการส่งข้อมูลที่เลือกในส่วนที่ไม่มีการเปลี่ยนแปลงบ่อยๆ จากนั้นจึงเข้าสู่ช่วงที่หยุดการทำงานของเวอร์ชวลแมชชีนต้นทางเพื่อทำการส่งข้อมูลในส่วนที่เหลือ เมื่อเครื่องปลายทางได้รับทั้งหมดแล้วจึงทำงานต่อเนื่องไป ซึ่งวิธีการนี้จะช่วยสามารถช่วยลดระยะเวลาที่ใช้ในการส่งข้อมูลขนาดใหญ่ในครั้งเดียว จึงทำให้เวอร์ชวลแมชชีนเสียระยะเวลาในการทำไมเกรชั่นไปยังเครื่องปลายทางลดลง

ภาพที่ 8

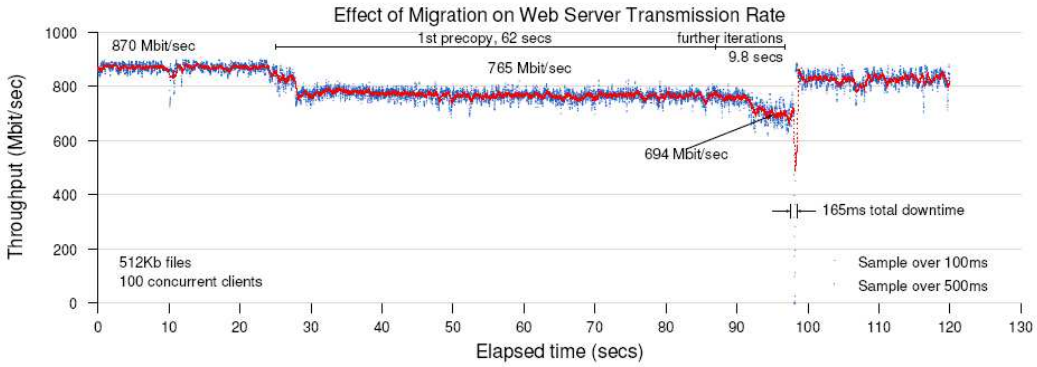
ขั้นตอนของการทำงานของการทำไลฟ์ไมเกรชั่นบนเซน




ที่มา: http://www.cl.cam.ac.uk/research/srg/netos/papers/2005-migration-nsdi-pre.pdf

ภาพที่ 9

แสดงระยะเวลาที่เครื่องหยุดให้บริการ ขณะมีการทำไลฟ์ไมเกรชั่น




ที่มา: http://www.cl.cam.ac.uk/research/srg/netos/papers/2005-migration-nsdi-pre.pdf

จากภาพที่ 9 จะแสดงให้เห็นว่า ในช่วงของการเตรียมข้อมูล แล้วส่งข้อมูลที่ไม่มีการเปลี่ยนแปลงบ่อย จะทำให้อัตราการทำงานของเวอร์ชวลแมชชีนลดลง ซึ่งจะเสียระยะเวลาในการหยุดให้บริการเพียง 185 มิลิวินาที จากนั้นจึงทำงานต่อได้อย่างต่อเนื่อง ซึ่งจะเห็นได้ว่าไลฟ์ไมเกรชั่น จะช่วยให้เวอร์ชวลแมชชีนสามารถกลับเข้าสู่การทำงานได้อย่างรวดเร็ว