พันโท ปรัชญา เฉลิมวัฒน์

อาจารย์ กองวิชาวิศวกรรมไฟฟ้า
ส่วนการศึกษา โรงเรียนนายร้อยพระจุลจอมเกล้า

# การปรับแต่งค่าประสิทธิภาพของลินุกซ์คลัสเตอร์ด้วยโปรแกรม HPL (High Performance Linpack)

# Linux Cluster Performance Tuning using HPL

## บทคัดย่อ

โปรแกรม **HPL (High Performance Linpack)** เป็นโปรแกรมทดสอบประสิทธิภาพของการประมวลผลแบบขนานสำหรับคลัสเตอร์และซูเปอร์คอมพิวเตอร์ซึ่งต้องมีการปรับแต่งค่าตัวแปรต่าง ๆ เพื่อให้สามารถทราบค่าประสิทธิภาพสูงสุดได้ บทความนี้รายงานผลการทดสอบค่าประสิทธิภาพของคลัสเตอร์ด้วย **HPL** รวมถึงขั้นตอนในการปรับแต่งค่าประสิทธิภาพเพื่อให้ได้ค่าสูงสุด ผลการทดสอบพบว่าลักษณะการปรับแต่งค่าเป็นปัญหาแบบ **Combinatorial** ซึ่งน่าจะสามารถนำวิธีการของ “ทฤษฎีการคำนวณทางพันธุกรรม” มาใช้เพื่อให้สามารถหาค่าที่เหมาะสมได้อย่างอัตโนมัติ

**คำหลัก** ลินุกซ์คลัสเตอร์, การประมวลผลแบบขนาน, ค่าประสิทธิภาพ, โปรแกรม HPL

## Abstract

Linux clusters have been potential alternative to real supercomputer recently. Obtaining good performance from the cluster is a difficult issue because there are several factors to be adjusted to obtain peak performance. HPL (High Performance Linpack) is a benchmark program for testing parallel performance of clusters and supercomputers. This paper describes HPL performance results and details of procedures for performance tuning of CRMA Linux cluster. The result of this study shows that manually tuning of HPL is difficult and not definite. The problem is considered combinatorial that, potentially, can be automatically solved using Genetic Algorithms (GAs).

**Keywords:** Linux Cluster, parallel processing, performance tuning, Linpack, HPL

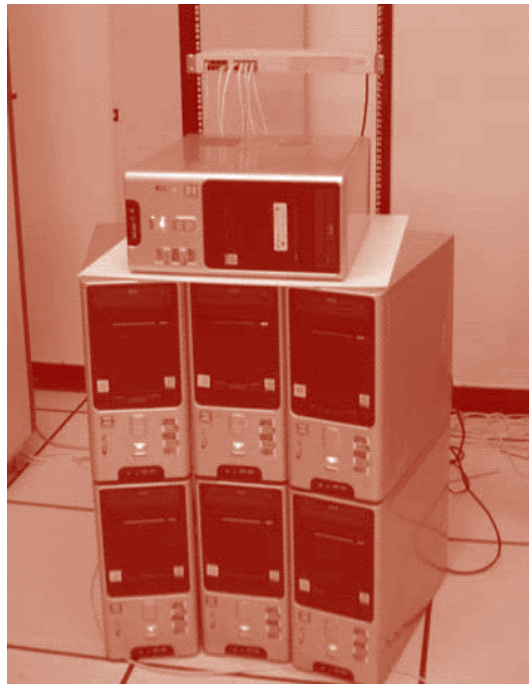
## 1. บทนำ

ลินุกซ์คลัสเตอร์ได้รับการยอมรับให้เป็นทางเลือกในการคำนวณที่ต้องการสมรรถนะของคอมพิวเตอร์สูงเทียบเท่าหรือดีกว่าซูเปอร์คอมพิวเตอร์ อย่างไรก็ตามการที่จะสามารถใช้งานคลัสเตอร์ให้เต็มประสิทธิภาพได้นั้นเป็นเรื่องยาก ต้องใช้ปัจจัยในการพิจารณาและการทดลองหลายอย่าง บทความนี้รายงานผลการทดสอบประสิทธิภาพของลินุกซ์คลัสเตอร์ของโรงเรียนนายร้อยพระจุลจอมเกล้า (CRMA Cluster)(1) เพื่อค้นหาค่าประสิทธิภาพที่เหมาะสมเพื่อใช้เป็นข้อมูลที่สามารถทำให้ปรับแต่งค่าของโปรแกรมประยุกต์ให้สามารถใช้งานคลัสเตอร์ได้อย่างเต็มประสิทธิภาพ

การประยุกต์ใช้งานคลัสเตอร์โดยทั่วไปจะใช้ในการประมวลผลข้อมูลขนาดใหญ่ หรือกรณีที่ต้องการสมรรถนะการคำนวณที่เกินขีดความสามารถของซีพียูเดียวได้ ตัวอย่างโครงการที่ใช้การคำนวณแบบขนานสามารถดูรายละเอียดเพิ่มเติมได้ใน(5)(6)

### 1.1 ข้อมูลด้านฮาร์ดแวร์ของ CRMA Cluster

รูปที่ 1 แสดงภาพของลินุกซ์คลัสเตอร์ ซึ่งติดตั้งอยู่ ณ โรงเรียนนายร้อยพระจุลจอมเกล้า ประกอบด้วยโหนดฟรอนท์เอ็นด์ จำนวน 1 โหนด และโหนดคำนวณจำนวน 6 โหนด โหนดฟรอนท์เอ็นด์เชื่อมต่อกับระบบเครือข่ายภายใน และเชื่อมต่อกับโหนดคำนวณด้วยเครือข่าย Gigabit Ethernet ซึ่งเป็นเครือข่ายเฉพาะงานคำนวณของคลัสเตอร์เท่านั้น



**รูปที่ 1** คลัสเตอร์ของโรงเรียนนายร้อยพระจุลจอมเกล้า (CRMA Cluster)

ข้อมูลฮาร์ดแวร์ของโหนดต่างๆ ประกอบด้วยซีพียู Pentium® 4 ความเร็ว 3.0 GHz หน่วยความจำ 512 Mbytes และมีฮาร์ดดิสก์ขนาด 80 GBytes ทุกโหนดมี NIC (network interface card) 2 แบบคือ Gigabit Ethernet และ Fast Ethernet ความเร็วสูงสุดทางทฤษฎีเท่ากับ 1000 Mbps และ 100 Mbps ตามลำดับ (Mbps = mega bits per second) ค่าความเร็วการสื่อสารข้อมูลระหว่างโหนดวัดด้วยโปรแกรม Netpipe ได้เท่ากับ 328 Mbps ซึ่งจะทำให้มีค่าความเร็วโดยรวม (aggregated bandwidth) อยู่ที่ 1.968 Gbps หน่วยความจำรวม 3 Gbytes และพื้นที่ฮาร์ดดิสก์รวมเท่ากับ 480 Gbytes

ความเร็วในการประมวลผลสูงสุดของ 1 ซีพียู (วัดด้วยโปรแกรม HPL) มีค่าเท่ากับ 4 Gflops ดังนั้นในทางทฤษฎีแล้วหากไม่มีการสูญเสียเวลาในการสื่อสารระหว่างโหนดจะทำให้มีค่าประสิทธิภาพ

สูงสุดทางทฤษฎี (Theoretical Peak Performance) เท่ากับ 24 Gflops เมื่อใช้การคำนวณด้วย 6 โหนด คลัสเตอร์ที่เร็วที่สุดในโลก ณ ปัจจุบัน คือ IBM eServer Blue Gene Solution[1] มีค่าประสิทธิภาพเท่ากับ 280,600 Gflops (วัดด้วยโปรแกรม HPL) มีจำนวนหน่วยคำนวณ 131,072 โพรเซสเซอร์ คิดเป็นค่าประสิทธิภาพต่อโพรเซสเซอร์ได้เท่ากับ 2.14 Gflops และผลการทดสอบประสิทธิภาพของ Itanium 2 Cluster ของ NECTEC ที่ 32 โพรเซสเซอร์มีค่าเท่ากับ 102.2 Gflops หรือประมาณ 3.19 Gflops ต่อโพรเซสเซอร์[2]

**ตารางที่ 1** สมรรถนะของโพรเซสเซอร์แบบต่างๆ

| โพรเซสเซอร์ | GFLOPS |
|---|---|
| 1.5 GHz Itanium2 | 6 |
| 3 GHz Xeon | 4.5 |
| 3 GHz P4* | 4 |
| 1.6 GHz Pentium M | 0.2 |

*หมายเหตุ โพรเซสเซอร์ของ CRMA Cluster

### 1.2 ข้อมูลด้านซอฟท์แวร์

CRMA Cluster ใช้ซอฟท์แวร์รหัสเปิด (open source software) ทั้งในส่วนที่เป็นระบบปฏิบัติการ คอมไพล์เลอร์ และโปรแกรมประยุกต์ ที่ฟรอนท์เอ็นด์โหนดจะใช้ระบบปฏิบัติการ Ubuntu Linux 5.10 เพื่อความง่ายในการจัดการและใช้เป็นที่เก็บไฟล์หลักในการคำนวณผ่านระบบไฟล์ NFS (network file system) ไปยังโหนดคำนวณ ซึ่งใช้ระบบปฏิบัติ-การ Gentoo Linux 3.3.5 เพื่อให้ได้ประสิทธิภาพสูงสุดในการคำนวณ ลินุกซ์ทั้งสองแบบใช้เคอร์เนิ่ลเวอร์ชั่น 2.6.12 ใช้ไลบรารี่สำหรับการสื่อสารข้อมูล mpich 1.2 โปรแกรม HPL (high performance computing linpack benchmark) เวอร์ชั่น 1.0a และ GotoBLAS เวอร์ชั่น 1.03

## 2. งานวิจัยที่เกี่ยวข้อง

HPL (High Performance Linpack)(2) เป็นโปรแกรม benchmark สำหรับคอมพิวเตอร์แบบขนาน/คอมพิวเตอร์สมรรถนะสูง ที่มีสถาปัตย-กรรมแบบกระจายหน่วยความจำ (distributed memory architecture) โดยนำเอา benchmark ที่ทดสอบประสิทธิภาพของการคำนวณคือ LINPACK[3] ให้ทำงานร่วมกับไลบรารี่ MPI (message passing interface)[4] เพื่อให้โหนดคำนวณในคลัสเตอร์ หรือซูเปอร์คอมพิวเตอร์สามารถสื่อสารข้อมูลกันได้

HPL เป็นซอฟท์แวร์ที่ใช้ในการแก้ปัญหา dense linear system ด้วยความละเอียดของการคำนวณทางคณิตศาสตร์ระดับ double precision (64 bits) โดยเฉพาะสำหรับเครื่องคอมพิวเตอร์แบบขนานที่ใช้หน่วยความจำแบบกระจาย HPL เป็นโปรแกรมที่สามารถนำไปใช้ได้กับคอมพิวเตอร์หลายประเภททำให้มี portability ที่สูง สามารถดาวน์โหลดได้ฟรีจาก www.netlib.org เพื่อให้ผู้ใช้สามารถนำไปทดสอบประสิทธิภาพของเครื่องคอมพิวเตอร์ของตนเอง นอกจากนั้น HPL ยังใช้เป็นมาตรฐานการจัดอันดับซูเปอร์

[1] http://www.top500.org/system/7747
[2] http://www.hpcc.nectec.or.th/wiki/index.php/Itanium2
[3] http://www.netlib.org/benchmark/hpl/
[4] MPICH. http://www-unix.mcs.anl.gov/mpi/mpich/

คอมพิวเตอร์ โดยจะมีการจัดอันดับคอมพิวเตอร์ที่เร็วที่สุดในโลก 500 อันดับแรก(3) โดยจะวัดเป็นหน่วย GFLOPS (Giga Floating Point Operations Per Second) หรือเพื่อหาค่า R_max ซึ่งเป็นค่า GFLOPS สูงสุดที่สามารถคำนวณ Linpack ได้

อัลกอริธึ่มใน HPL จัดเป็นกลุ่มดังนี้:

- Two-dimensional block-cyclic data distribution
- Right-looking variant of the LU factorization with row partial pivoting featuring multiple look-ahead depths
- Recursive panel factorization with pivot search and column broadcast combined
- Various virtual panel broadcast topologies
- bandwidth reducing swap-broadcast algorithm
- backward substitution with look-ahead of depth 1

โปรแกรม HPL สามารถใช้ในการทดสอบเวลาในการคำนวณของโปรแกรมเพื่อที่จะหาความถูกต้องอย่างละเอียดของผลลัพธ์ รวมถึงเวลาที่ใช้ในการคำนวณ ค่าประสิทธิภาพที่ดีที่สุดสำหรับแต่ละเครื่องนั้นขึ้นอยู่กับปัจจัยหลายอย่าง หากไม่คำนึงถึงประสิทธิภาพการเชื่อมต่อระหว่างโหนดแล้ว HPL อัลกอริธึ่มมีขีดความสามารถในการขยาย (scalability) เป็นอย่างดี เนื่องจากค่า parallel efficiency นั้นคำนวณจากการใช้งานของหน่วยความจำต่อโพรเซสเซอร์ (per processor memory usage) ซอฟท์แวร์แพ็คเกจของ HPL ต้องใช้ไลบรารี่อื่นร่วมกันคือ:

- MPI (Message Passing Interface MPI 1.1 compliant)
- Basic Linear Algebra Subprograms BLAS โดยที่ BLAS จะใช้งานไลบรารี่ของ Linpack
- หรือ Vector Signal Image Processing Library VSIPL

### 2.1 ปัญหาของการใช้โปรแกรม HPL

ถึงแม้ว่าโปรแกรม HPL จะสามารถใช้เป็นเครื่องมือในการวัดค่าประสิทธิภาพของการคำนวณแบบขนานของซูเปอร์คอมพิวเตอร์ได้เป็นอย่างดี โดยมีการวัดหน่วยคำนวณเป็น GFLOPS การปรับแต่งค่าในการวัดนั้นต้องอาศัยความชำนาญของผู้ปรับแต่งเนื่องจากมีตัวแปรที่เกี่ยวข้องเป็นจำนวนมาก งานวิจัยในบทความนี้เป็นจุดเริ่มต้นในการค้นหาวิธีการปรับแต่งค่าให้เป็นไปแบบอัตโนมัติ โดยการประยุกต์ใช้ทฤษฎีการคำนวณทางพันธุกรรม (Genetic Algorithms) เพื่อค้นหาค่าของตัวแปรที่เหมาะสม ซึ่งผู้เขียนเชื่อว่าน่าจะเป็นงานวิจัยในอนาคตที่มีประโยชน์(6)

### 2.2 การใช้ไลบรารี่ของ GotoBLAS

โปรแกรม HPL เรียกใช้ไลบรารี่ Basic Linear Algebra System (BLAS) ซึ่งพัฒนาโดยกลุ่ม NetLib อย่างไรก็ตามประสิทธิภาพของ BLAS นั้นมีข้อบกพร่องในการคำนวณซึ่ง Goto(4) เป็นผู้ปรับปรุงประสิทธิภาพของ BLAS ให้ดีขึ้นและตั้งชื่อ GotoBLAS ผลการทดลองในบทความนี้ใช้ไลบรารี่ของ GotoBLAS เป็นหลักในการคำนวณร่วมกับ HPL โดยที่ Goto ได้ทำการปรับปรุงการทำงานของฟังก์ชั่นบางส่วนของ BLAS ให้ทำงานได้อย่างมีประสิทธิภาพมากขึ้น

## 3. การทดสอบประสิทธิภาพของคลัสเตอร์ด้วย HPL

รายละเอียดการติดตั้งโปรแกรม HPL สามารถดูได้จากเว็บไซต์ http://research.crma.ac.th กล่าวโดยย่อแล้ว หลังจากที่คอมไพล์ซอฟท์แวร์ของ HPL จะได้ไฟล์ชื่อ xhpl ซึ่งใช้สำหรับทำการทดสอบด้วยการเรียกผ่านโปรแกรม mpirun เพื่อสั่งให้แต่ละโหนดคำนวณเริ่มทำงานประมวลผลแบบขนานไปพร้อมๆ กัน

### 3.1 การปรับแต่งค่าตัวแปรของ HPL.dat

โปรแกรม HPL จะสามารถใช้เป็นเครื่องมือในการวัดประสิทธิภาพ ผู้ใช้งานต้องทำการปรับค่าสำหรับการใช้งานที่เหมาะสมกับขนาดและคุณลักษณะของคลัสเตอร์ การปรับแต่งค่าตัวแปรของโปรแกรม HPL ต้องกำหนดค่าต่างๆ ที่ต้องการให้โปรแกรมทดสอบในไฟล์ชื่อ HPL.dat ค่าตัวแปรที่สำคัญในไฟล์นี้แสดงในตารางที่ 2 จากตัวแปรทั้งหมด 27 ตัวแปร โดยมีตัวแปรจำนวน 9 ตัวที่สามารถกำหนดมิติของตัวแปรย่อยเพิ่มได้อีก

**ตารางที่ 2** ค่าตัวแปรสำคัญในการปรับแต่ง

| ตัวแปร | คำอธิบาย |
|---|---|
| N | จำนวนของปัญหา |
| Ns | ขนาดของปัญหา (แม็ตตริกซ์) |
| #NBs | จำนวนขนาดของบล็อกข้อมูล |
| NBs | ขนาดของบล็อกข้อมูล |
| #Grids | จำนวนกริดของการใช้ซีพียู |
| Ps, Qs | ขนาดของซีพียูกริด (row,column) |

ตัวแปรอื่นๆ ได้แก่ threshold, PFACT, RFACT, NBMINs, NDIVs, BCASTs, DEPTHs, swapping threshold, L1, U, Equilibration เป็นต้น ซึ่งจะไม่กล่าวรายละเอียดสำหรับค่าของตัวแปรเหล่านี้

นอกจากนั้นยังสามารถใช้ "กฎหัวแม่มือ" (Rules of Thumb)(7) เพื่อหาค่า N ที่เหมาะสม ดังนี้

N = Sqrt ((No._of_Nodes*Memory_per_node)/8)*0.82

= Sqrt ((6*512*1024)/8)*1024*0.82

= 16454

สำหรับ *NB* นั้นไม่มีกฎที่แน่ชัดต้องใช้การทดลอง ในที่นี้พบว่า *NB* ที่ดีที่สุดคือ 80

### 3.2 การเพิ่มประสิทธิภาพโปรแกรมด้วยการเปลี่ยนคอมไพล์เลอร์

นอกเหนือจากการปรับแต่งค่าตัวแปรในไฟล์ HPL.dat แล้วการเลือกใช้คอมไพล์เลอร์ยังมีส่วนสำคัญในการทำให้ได้ค่าประสิทธิภาพที่สูงขึ้นด้วย ผลลัพธ์ของการทดลองในบทความนี้ใช้ฟอร์แทรนคอมไพล์เลอร์ ifort ของบริษัทอินเทล[1] แทนการใช้คอมไพล์เลอร์ g77 เวอร์ชั่น 3.4.5 ที่ให้มากับระบบลินุกซ์ เมื่อคอมไพล์ GotoBLAS ด้วย ifort แล้วคอมไพล์โปรแกรม HPL ใหม่ทำให้ค่าประสิทธิภาพของคลัสเตอร์เพิ่มขึ้นตามรายละเอียดในผลการทดสอบประสิทธิภาพในบทต่อไป

## 4. ผลการทดสอบประสิทธิภาพ

ขั้นตอนการทดสอบประสิทธิภาพแบ่งเป็นการทดสอบประสิทธิภาพของ GotoBLAS การ

[1] Intel Fortran Compiler for Linux, http://www.intel.com/cd/software/products/asmo-na/eng/compilers/flin/219857.htm

ทดสอบเบื้องต้นด้วยข้อมูลขนาดเล็กทั้งนี้เพื่อค้นหาข้อมูลของค่าตัวแปรที่ต้องการเป็นจุดเริ่มต้นในการปรับแต่ง จากนั้นการทดสอบจะดำเนินการต่อด้วยการปรับให้ตัวแปรอื่นๆ คงที่ แล้วปรับแต่งค่าตัวแปรที่เป็นปัจจัยสำคัญ คือ ขนาดของปัญหา (N) และขนาดของบล็อกข้อมูล (NBs)

### 4.1 ผลการทดสอบประสิทธิภาพของ GotoBLAS

การทดลองนี้เป็นการเปรียบเทียบประสิทธิภาพของ BLAS แบบดั้งเดิมกับ GotoBLAS(4) ที่ได้รับการพัฒนาขึ้นมาใหม่ โดยไม่มีการปรับค่าตัวแปรที่ให้มากับโปรแกรม HPL ตารางที่ 3 แสดงให้เห็นค่าประสิทธิภาพที่เพิ่มขึ้นเมื่อใช้ GotoBLAS แทนที่ไลบรารี่ของ BLAS ที่ดาวน์โหลดมาจาก NETLIB ด้วยการใช้โหนดคำนวณจำนวน 4 โหนด (4 CPUs) และเปลี่ยนขนาดของข้อมูลเป็น 3 ขนาด คือ A (N=30-35), B (N=100-800), และ C (N=200-1600)

**ตารางที่ 3** เปรียบเทียบประสิทธิภาพของ BLAS และ GotoBLAS (หน่วยเป็น GFLOPS)

| Data set | BLAS | GotoBLAS | % เพิ่ม |
|---|---|---|---|
| *A* | 0.019 | 0.019 | 0% |
| *B* | 0.607 | 0.942 | 55% |
| *C* | 1.164 | 2.448 | 110% |

ตารางที่ 3 แสดงให้เห็นผลลัพธ์ที่ได้จากการทดลอง โดยที่ประสิทธิภาพของการทำงานของซีพียูที่เพิ่มขึ้นประมาณสองเท่าเมื่อใช้ GotoBLAS ในการคอมไพล์เพื่อสร้างไบนารี่ของโปรแกรม HPL

### 4.2 ผลการทดสอบเบื้องต้นด้วยข้อมูลขนาดเล็ก

ผลการทดสอบขั้นต้นให้ผลการทดสอบค่าประสิทธิภาพที่ค่อนข้างต่ำ ด้วยการเลือกใช้เพียง 4 ซีพียูในการทดลอง

**ตารางที่ 4** ค่าประสิทธิภาพของ HPL เบื้องต้นด้วย 4 โพรเซสเซอร์ (หน่วยเป็น GFLOPS)

| | Block Size *(NBs)* | | | |
|---|---|---|---|---|
| ***N*** | **4** | **10** | **20** | **40** |
| 4,000 | 2.83 | 5.38 | 7.05 | 7.5 |
| 6,000 | 3.6 | 6.92 | 8.64 | 9.38 |
| 10,000 | 4.20 | 7.98 | 8.78 | 9.47 |

ตารางที่ 4 แสดงค่าประสิทธิภาพของ HPL ทำงานแบบขนานโดยใช้ 4 โพรเซสเซอร์ ผลลัพธ์ที่ได้แสดงให้เห็นถึงแนวโน้มในการเพิ่มขึ้นของ GFLOPS เมื่อมีการเพิ่มขนาดของปัญหา *(N)* รวมถึงการเพิ่มขนาดของบล็อกข้อมูล *(NBs)* อย่างไรก็ตามค่าประสิทธิภาพเมื่อใช้ *N* = 10,000 และ *NBs* = 40 นั้นมีค่าประสิทธิภาพเท่ากับ 9.47/4 = 2.37 GFLOPS ต่อโพรเซสเซอร์

### 4.3 ผลการทดสอบด้วยการปรับแต่งค่า NBs

จากการทดลองที่ผ่านแสดงให้เห็นว่าปัจจัยสำคัญอีกประการในการเพิ่มค่าประสิทธิภาพคือขนาดของบล็อกข้อมูล ซึ่งกำหนดโดยค่า *NBs* ในไฟล์ HPL.dat ในการทดลองนี้จะเพิ่มค่า *N* ไว้สูงสุดที่ 14,000

**ตารางที่ 5** ค่าประสิทธิภาพของ HPL ทดลองเปลี่ยนค่าของ *NBs* ใช้ 4 โพรเซสเซอร์ (หน่วยเป็น GFLOPS)

| | Block Size *(NBs)* | | |
|---|---|---|---|
| ***N*** | **10** | **40** | **80** |
| 4,000 | 5.38 | 5.64 | 7.45 |
| 10,000 | 7.98 | 9.47 | 11.7 |
| 14,000 | 8.54 | 11.72 | 13.09 |

ตารางที่ 5 แสดงค่าประสิทธิภาพของการทำงานแบบขนานของโปรแกรม HPL เมื่อใช้ 4 โพรเซสเซอร์ และเพิ่มขนาดของบล็อกข้อมูลเป็น 80 โดยได้ค่าประสิทธิภาพสูงสุดเท่ากับ 13.09 เมื่อใช้ *N* = 14,000 คิดเป็นต่อโพรเซสเซอร์ได้เท่ากับ 13.09/4 = 3.27 GFLOPS/Processor หรือประมาณ 81% ของค่าสูงสุดที่ซีพียูจะทำได้

### 4.4 ผลการทดสอบด้วยการเพิ่มจำนวนซีพียู

การทดลองที่ผ่านมาใช้ 4 ซีพียูในการคำนวณทั้งนี้เพื่อความง่ายในการวิเคราะห์และเกิดความสมดุลในการทำงานกับแม็ตริก การเพิ่มจำนวนซีพียูเป็นอีกปัจจัยสำคัญในการทำให้ค่าประสิทธิภาพเพิ่มขึ้นได้ เนื่องจากคลัสเตอร์ของ รร.จปร. มีโหนดคำนวณเพียง 6 โหนด ขอบเขตของการทดลองนี้จึงถูกจำกัดอยู่ที่จำนวนซีพียูเท่ากับ 6 ตารางที่ 5 แสดงให้เห็นค่าประสิทธิภาพของ HPL เมื่อใช้ 6 ซีพียูร่วมกันประมวลผลแบบขนาน

**ตารางที่ 6** ค่าประสิทธิภาพของ HPL เมื่อใช้ 6 โพรเซสเซอร์ (หน่วยเป็น GFLOPS)

| | Block Size *(NBs)* | | |
|---|---|---|---|
| ***N*** | **40** | **80** | **120** |
| 4,000 | 6.60 | 8.46 | 6.82 |
| 10,000 | 10.2 | 12.68 | 10.25 |
| 14,000 | 13.47 | 15.53 | 13.05 |

จากตารางที่ 6 ค่าประสิทธิภาพสูงสุดเท่ากับ 15.53 GFLOPS เมื่อใช้ *NBs* = 80 และ *N* = 14,000 คิดเป็น 15.53/6 = 2.58 GFLOPS ต่อโพรเซสเซอร์ จากการเพิ่มขนาดของบล็อกข้อมูลให้เป็น 120 แต่ค่าประสิทธิภาพกลับลดลงแสดงให้เห็นว่าค่า *NBs* ที่ประมาณ 80 นั้นจะให้ค่าประสิทธิภาพสูงสุดซึ่งตรงกับคำแนะนำของ Rules of Thumb(7) อย่างไรก็ตามการเพิ่มจำนวนซีพียู (จาก 4 เป็น 6) นั้นไม่สามารถประกันได้ว่าประสิทธิภาพการคำนวณแบบขนานจะดีขึ้น เนื่องจากเวลาในการสื่อสารนั้นจะเพิ่มตามขึ้นด้วย

## 5. สรุป

### 5.1 สรุปผลการทดสอบ

จากการทดสอบค่าประสิทธิภาพด้วยวิธีการแบบต่างๆ นั้นทำให้ทราบถึงปัจจัยที่มีผลสำคัญต่อการคำนวณคือ "ขนาดของข้อมูล" และ "การจัดแบ่งกระจายข้อมูล" ไปตามโหนดคำนวณต่างๆ ได้อย่างลงตัว นอกจากนั้นการเลือกใช้ไลบรารี่ที่เหมาะสม และคอมไพล์เลอร์ที่มีประสิทธิภาพยังมีส่วนสำคัญในการทำให้โปรแกรมแบบขนานนั้นทำงานได้อย่างมีประสิทธิภาพบนคลัสเตอร์ ซึ่งโดยสรุปสามารถทำให้ได้ค่าประสิทธิภาพต่อโพรเซสเซอร์ถึง 2.5 GFLOPS จากค่าสูงสุดทาง

ทฤษฎี 4 GFLOPS และได้ค่า R_max เท่ากับ 15.53 GFLOPS

### 5.2 งานวิจัยในอนาคต

ผลการทดสอบค่าประสิทธิภาพของคลัสเตอร์แสดงให้เห็นถึงความเป็นไปได้ในการนำคลัสเตอร์มาใช้งานจริงกับโปรแกรมประยุกต์ ถึงแม้ว่าการปรับแต่งด้วยมือจะให้ค่าประสิทธิภาพที่ประมาณร้อยละ 62 ของค่าประสิทธิภาพสูงสุดในทางทฤษฎีการปรับแต่งค่าในการทดลองนี้ยังไม่สามารถประกันได้ว่าเป็นการปรับแต่งที่ดีที่สุด ทั้งนี้เนื่องจากค่าตัวแปรในการปรับแต่ง HPL.dat นั้นมีจำนวนมากและเกินกว่าการคาดเดาค่าที่เหมาะสมด้วยการทดลอง ซึ่งเข้าลักษณะของปัญหาแบบ Combinatorial การขยายผลของงานวิจัยในอนาคตสามารถทำได้ด้วยการประยุกต์ใช้โปรแกรม "การคำนวณด้วยทฤษฎีทางพันธุกรรม" หรือ Genetic Algorithms(6) มาใช้หาค่าที่เหมาะสมที่ดีที่สุดสำหรับคลัสเตอร์ต่อไป

## กิตติกรรมประกาศ

งานวิจัยนี้ได้รับการสนับสนุนจากศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) และเป็นส่วนหนึ่งของ "โครงการความร่วมมือทางวิชาการระหว่าง สวทช. และ รร.จปร." ผู้เขียนขอขอบคุณ ร.อ.เสกสิทธิ์ ศิริพละ ที่ให้ความช่วยเหลือในการดำเนินงานโครงการ ดร.ศรเทพ วรรณรัตน์ และคณะจาก NECTEC ที่ให้การสนับสนุนและให้คำแนะนำต่างๆ ในการทดสอบค่าประสิทธิภาพเป็นอย่างดี

## เอกสารอ้างอิง

(1) ปรัชญา เฉลิมวัฒน์, พันโท "การทดสอบประสิทธิภาพของ 64-bit Linux Cluster" วารสารทางวิชาการ, สภาอาจารย์ ส่วนการศึกษา โรงเรียนนายร้อยพระจุลจอมเกล้า, 2548 หน้า 111-117.

(2) A. Petitet, R. C. Whaley, J. Dongarra, A. Cleary, "HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers", http://www.netlib.org/benchmark/hpl, Jan.20, 2004.

(3) TOP500 Supercomputer Sites, http://www.top500.org

(4) GotoBLAS, ไลบรารี่สำหรับสมการเชิงเส้น http://www.tacc.utexas.edu/resources/software

(5) P. Chalermwat, T. El-Ghazawi, and J. LeMoigne, "Image Registration by Parts", in proceedings of Image Registration Workshop (IRW97), NASA Goddard Space Flight Center, MD, Nov. 1997.

(6) P. Chalermwat, T. El-Ghazawi, and J. LeMoigne, "2-Phase GA-based Image Registration on Parallel Clusters", International Journal of Future Generation Computing Systems, Vol. 17, issue 4, Jan. 2001.

(7) S. Ram, "HPL Rules of Thumb", http://www.vpac.org/~sram/top500/hpl.html