แอพพลิเคชั่น Sort แบบขนานบน Cluster ของโทรศัพท์สมาร์ทโฟน MultiCore-ARM

PARALLEL-SORTING APPLICATION ON CLUSTER OF MULTICORE-ARM SMART PHONES

ชนะชัย จุมพล

ภาณุพงศ์ ถนัดค้า

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2559

แอพพลิเคชั่น Sort แบบขนานบน Cluster ของโทรศัพท์สมาร์ทโฟน MultiCore-ARM

PARALLEL-SORTING APPLICATION ON CLUSTER OF MULTICORE-ARM SMART PHONES

ชนะชัย จุมพล

ภาณุพงศ์ ฉนัดค้า

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2559

ปริญญานิพนธ์ปีการศึกษา 2559

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง แอพพลิเคชั่น Sort แบบขนานบน Cluster ของโทรศัพท์สมาร์ทโฟน MultiCore-ARM

PARALLEL-SORTING APPLICATION ON CLUSTER OF MULTICORE-ARM SMART PHONES

ผู้จัดทำ

1. นายชนะชัย จุมพล รหัสนักศึกษา 56010246

2. นายภาณุพงศ์ ถนัดค้า รหัสนักศึกษา 56010927

_____อาจารย์ที่ปรึกษา

(ผศ.ดร.สุรินทร์ กิตติธรกุล)

แอพพลิเคชั่น Sort แบบขนานบน Cluster ของโทรศัพท์มือถือ MultiCore-ARM

นายชนะชัย จุมพล 56010246
นายภาณุพงศ์ ถนัดค้า 56010927
ผศ.ดร.สุรินทร์ กิตติธรกุล อาจารย์ที่ปรึกษา
ปีการศึกษา 2559

บทคัดย่อ

การจัดเรียงข้อมูลเป็นวิธีการจัดการข้อมูลที่สำคัญทางวิทยาการคอมพิวเตอร์ เพราะข้อมูลที่ถูกจัดเรียงจะนำไปประมวลผลได้ง่ายขึ้นในปัจจุบัน มีวิธีการมากมายสำหรับการจัดเรียงข้อมูล แต่ในการจัดเรียงข้อมูลที่เป็น Big data นั้นไม่เพียงแค่ใช้อัลกอริทึ่มที่ดีในการจัดเรียงแล้ว ควรใช้คอมพิวเตอร์ที่มีประสิทธิภาพสูงในการรับมือด้วย โครงการนี้เป็นการทดลองนำโทรศัพท์เคลื่อนที่สมาร์ทโฟนหลายๆ เครื่องมาใช้งานเป็นเครื่องคอมพิวเตอร์ประสิทธิภาพสูง ซึ่งสามารถจัดเรียงข้อมูลที่เป็นตัวเลข 32 บิต จำนวนมากได้

โครงการนี้จะใช้โทรศัพท์เคลื่อนที่สมาร์ทโฟนจำนวนมากมาทำงานกันแบบขนานในการจัดเรียงข้อมูลตัวเลขไม่เกิน 32 บิตให้ถึง 100 ล้านตัว โดยจะมีโทรศัพท์เครื่องหนึ่งทำหน้าที่เป็นมาสเตอร์ (Master) คอยส่งข้อมูลไปยังโทรศัพท์ทั้งหมดรวมถึงเครื่องที่เป็นมาสเตอร์ด้วย จากนั้นแต่ละเครื่องจะทำการจัดเรียงข้อมูลที่มี และรวบรวมการจัดเรียงข้อมูลกับเครื่องอื่นๆ สุดท้ายข้อมูลทั้งหมดจะมารวมที่เครื่องที่เป็นมาสเตอร์ โดยข้อมูลทั้งหมดถูกจัดเรียงแล้ว

โครงการนี้จะเริ่มจากการทดลองโดยใช้คอมพิวเตอร์ตั้งโต๊ะ/คอมพิวเตอร์โน๊ตบุ๊คก่อน โดยเริ่มจากการจำลองเครือข่ายในคอมพิวเตอร์ ต่อมาจึงเพิ่มเครื่องคอมพิวเตอร์อีก 9 เครื่อง เพื่อสร้างเครือข่ายจากคอมพิวเตอร์หลายๆเครื่อง และสุดท้ายก็นำ Java application มาใช้ในโทรศัพท์สมาร์ทโฟน

Parallel-sorting Application on Cluster of Multicore-ARM Smart Phones

Mr.Chanachai Jumpon 56010246

Mr.Phanupong Thanutka 56010927

Asst.Prof.Dr. Surin Kittitornkun Advisor

Academic Year 2016

Abstract

Sorting algorithms are important in Computer Science because sorted data can be processed further. In the present day, there are many sorting algorithms. But to sort big data, one should use not only good algorithm but also high performance computer to handle. This project is an experiment to apply a mobile application on smartphones as high performance computer for sorting a large number 32-bit Integer data.

This project can utilize a number of smartphones for sorting up to 100 million of 32-bit Integer data in parallel. One of them acts as "Master". The Master will divide data and send to other smartphones including itself, Every smartphone will sort data and merge sorted data with one another. Finally, all data will be gathered by the Master.

First, this project will use two desktop/notebook computers to test the algorithm in the local network. Next, we will increase up to 10 computers to test the algorithm again. Finally, we will port this Java application to test on a number smartphones.

กิตติกรรมประกาศ

ปริญญานิพนธ์แอพพลิเคชั่น Sort แบบขนานบน Cluster ของสมาร์ทโฟน MultiCore-ARM ฉบับบนี้ ได้รับการสนับสนุน ให้คำปรึกษาและอนุเคราะห์ด้านเครื่องคอมพิวเตอร์ โดยเฉพาะสถานที่ทำการทดลองซึ่งเป็นที่จำเป็นสำหรับโครงงานนี้จาก ผศ.ดร.สุรินทร์ กิตติธรกุล อาจารย์ที่ปรึกษาจนโครงงานนี้ประสบผลสำเร็จสำเร็จ คณะผู้จัดทำขอขอบคุณท่านมา ณ โอกาสนี้

ชนะชัย จุมพล

ภาณุพงศ์ ถนัดค้า

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญรูป

รูป หน้า

สารบัญรูป (ต่อ)

รูป หน้า

สารบัญรูป (ต่อ)

รูป หน้า

สารบัญรูป (ต่อ)

รูป หน้า

สารบัญตาราง

ตาราง **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นของมาปัญหา

การจัดเรียงข้อมูลเป็นวิธีการจัดการข้อมูลที่สำคัญทางวิทยาการคอมพิวเตอร์ เพราะข้อมูลที่ถูกจัดเรียงจะนำไปใช้ได้ง่ายขึ้น ในปัจจุบัน มีวิธีการมากมายสำหรับการจัดเรียงข้อมูล แต่ในการจัดเรียงข้อมูลที่เป็น Big data นั้น นอกจากอัลกอริทึ่มที่ดีในการจัดเรียงแล้ว ควรใช้คอมพิวเตอร์ที่มีประสิทธิภาพสูงในการทำงานด้วย เพื่อรองรับ Big data ที่ต้องการประมวลผลด้วย

โครงการนี้เป็นการทดลองนำโทรศัพท์เคลื่อนที่สมาร์ทโฟนหลายๆเครื่องมาใช้งานเป็นเครื่องคอมพิวเตอร์ประสิทธิภาพสูง ที่ทำงานผ่านแอปพลิเคชั่นที่สร้างขึ้น ในการเพิ่มประสิทธิภาพในการจัดเรียงข้อมูล (Sorting) จำนวนมาก เพื่อรองรับ Big data ที่มีแนวโน้มจะนำมาใช้ในหลากหลายเทคโนโลยีในอนาคต

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อทำการทดลองการสร้างเครื่องคอมพิวเตอร์ระดับสูงจากโทรศัพท์เคลื่อนที่สมาร์ทโฟนผ่านแอปพลิเคชั่นที่สร้างขึ้น
2. เพื่อค้นหาวิธีที่จะเพิ่มประสิทธิภาพของการจัดเรียงข้อมูล

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้รับความรู้เรื่องการสร้างแอปพลิเคชั่นบนระบบปฏิบัติการแอนดรอยด์
2. ได้รับความรู้เรื่องอัลกอริทึ่มของการจัดเรียงผ่านภาษา Java

1.4 ขอบเขตของโครงงาน

1. สามารถจัดเรียงข้อมูลที่เป็นตัวเลขไม่เกิน 32 บิตได้
2. สามารถใช้กับโทรศัพท์ที่มีระบบปฏิบัติการแอนดรอยด์รุ่นใดก็ได้

1.5 วิธีการดำเนินงาน

1. ศึกษาภาษา Java เบื้องต้น
2. ค้นหา Sorting program เบื้องต้นมาเพื่อใช้งาน
3. เริ่มพัฒนาโปรแกรม เพื่อทดลองใช้ในคอมพิวเตอร์ตั้งโต๊ะเครื่องเดียวก่อน
4. พัฒนาโปรแกรมเพิ่มเติม เพื่อทดลองใช้กับคอมพิวเตอร์หลายเครื่อง
5. พัฒนา Sorting Algorithm และการส่งข้อมูลผ่าน Network ของโปรแกรม
6. เริ่มพัฒนา Application ที่ใช้บน Android

1.6 ส่วนประกอบของรายงาน

รายงานเล่มนี้ประกอบด้วยส่วนประกอบ 5 ส่วน คือ

บทที่ 1 บทนำ กล่าวถึง ความเป็นมา วัตถุประสงค์ ขอบเขตการดำเนินงานของโครงการ วิธีการดำเนินงาน ผลประโยชน์ที่คาดว่าจะได้รับ และ ส่วนประกอบของรายงาน

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง กล่าวถึง ทฤษฎีพื้นฐานที่ใช้ในการทำโครงงานประกอบด้วย Sorting algorithm, Multithreading และ Java

บทที่ 3 การออกแบบและพัฒนา กล่าวถึง รายละเอียดของโปรแกรมที่พัฒนา และ การทำงานของโปรแกรม

บทที่ 4 การทดลองและผลการทดลอง กล่าวถึง รายละเอียดการทดลองของโปรแกรม ผลการทดลองประสิทธิภาพของโปรแกรม และ วิเคราะห์ผลการทดลอง

บทที่ 5 บทสรุป กล่าวถึง บทสรุปของโครงงาน แนวทางการดำเนินงานและแผนการดำเนินงานในภาคเรียนที่ 2

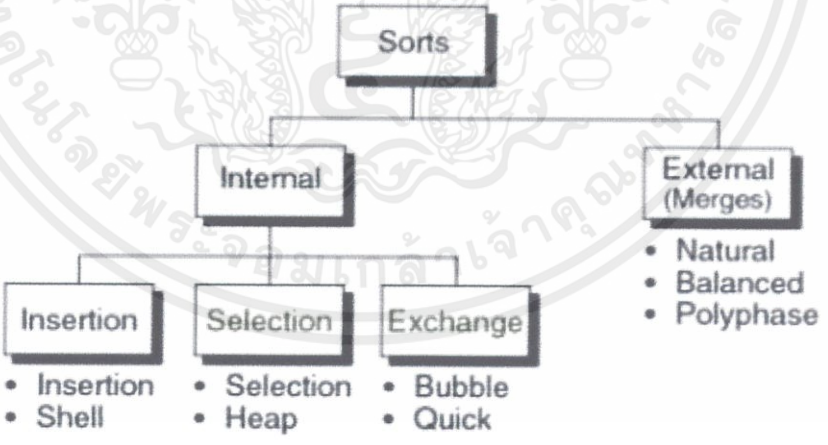
บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 Sorting Algorithm

การเรียงลำดับ (Sorting) เป็นการจัดลำดับของข้อมูลให้เป็นระเบียบมีแบบแผน ช่วยให้การค้นหาสิ่งของหรือข้อมูลรวดเร็วและมีประสิทธิภาพมากขึ้น เช่น การค้นหาความหมายของคำในพจนานุกรม ทำได้ค่อนข้างง่ายและรวดเร็วเนื่องจากมีการเรียงลำดับคำตามตัวอักษรไว้อย่างมีระบบและเป็นระเบียบ หรือ การค้นหาหมายเลขโทรศัพท์ในสมุดโทรศัพท์ ซึ่งมีการเรียงลำดับ ตามชื่อและชื่อสกุลของเจ้าของโทรศัพท์ไว้ ทำให้สามารถค้นหา หมายเลขโทรศัพท์ของคนที่ต้องการได้อย่างรวดเร็ว เป็นต้น

2.1.1 ประเภทของ Sorting Algorithm

เนื่องจากมีวิธีการมากมายที่สามารถใช้ในการเรียงลำดับข้อมูลได้ บางวิธีก็มีขั้นตอนการจัดเรียงเป็นแบบง่ายๆ ตรงไปตรงมา แต่ใช้เวลาในการจัดเรียงลำดับนาน และบางวิธีก็มีขั้นตอนในการจัดเรียงลำดับแบบซับซ้อนยุ่งยากแต่ใช้เวลาในการจัดเรียงไม่นานนัก ดังนั้นจึงควรศึกษาวิธีการจัดเรียงลำดับด้วยวิธีการต่าง ๆ เพื่อเลือกใช้วิธีการที่ดีและเหมาะสมกับระบบงานนั้นที่สุด วิธีการเรียงลำดับสามารถแบ่งออกเป็น 2 ประเภทดังรูปที่ 2.1



รูปที่ 2.1 ประเภทของการเรียงลำดับข้อมูล

2.1.1.1 การเรียงลำดับแบบภายใน (Internal sorting)

เป็นการเรียงลำดับที่ข้อมูลทั้งหมดต้องอยู่ในหน่วยความจำหลักเวลาที่ใช้ในการเรียงลำดับจะคำนึงถึงเวลาที่ใช้ในการเปรียบเทียบและเลื่อนข้อมูลภายในความจำหลัก

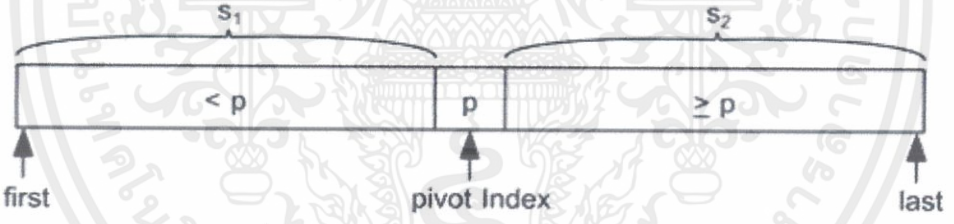
2.1.1.2 การเรียงลำดับแบบภายนอก (External sorting)

เป็นการเรียงลำดับข้อมูลที่เก็บอยู่ในหน่วยความจำสำรอง ซึ่งเป็นการเรียงลำดับข้อมูลในแฟ้มข้อมูล (File) เวลาที่ใช้ในเรียงลำดับต้องคำนึงถึงเวลาที่เสียไประหว่างการถ่ายเทข้อมูลจากหน่วยความจำหลักและหน่วยความจำสำรองนอกเหนือจากเวลาที่ใช้ในการเรียงลำดับข้อมูลแบบภายใน

2.1.2 Sorting Algorithm ที่นำมาใช้

2.1.2.1 ควิกซอร์ต (Quick sort)

การจัดเรียงข้อมูลแบบ Quick sort มีโครงสร้างในการแบ่งข้อมูลดังแสดงในรูปที่ 2.2 โดยตำแหน่งของ pivotIndex คือตำแหน่งที่ใช้เป็นข้อมูลหลักในการเปรียบเทียบเพื่อจัดเรียงข้อมูล

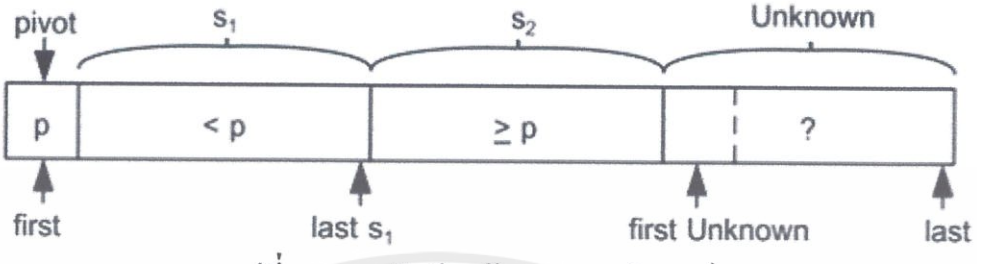


รูปที่ 2.2 ตำแหน่งของ Pivot Index ในควิกซอร์ต

ในการจัดเรียงข้อมูลจะนำข้อมูลที่จะจัดเรียงมาเปรียบเทียบกับข้อมูลค่าไพวอท p และข้อมูลในส่วน s_1 จะมีข้อมูลอยู่ในช่วง s_1 = theArray[first … pivotIndex-1] ซึ่งข้อมูลทั้งหมดในช่วงนี้จะมีค่าน้อยกว่าข้อมูลค่าไพวอท p และข้อมูลที่อยู่ในช่วง s_2= theArray[pivotIndex+1… last] เป็นข้อมูลที่มีค่ามากกว่าหรือเท่ากับค่าไพวอท p

ในการจัดเรียงข้อมูลแบบ Quick sort ได้มีหลักการและโครงสร้างของข้อมูลดังแสดงในรูปที่ 2.3 โดยเริ่มจากกำหนดให้ตำแหน่งแรกในอาร์เรย์เป็นตำแหน่งของ pivot เพื่อใช้เป็นค่าสำหรับเปรียบเทียบในการจัดเรียงข้อมูล ซึ่งข้อมูลที่น้อยกว่า pivot จะเก็บข้อมูลต่อจากตำแหน่งของ pivot คือข้อมูลในส่วน s_1 และข้อมูลที่มากกว่า pivot คือส่วน s_2 ซึ่งจะเก็บต่อจากข้อมูลส่วน s_1 และข้อมูลส่วน

Unknown คือที่ยังไม่ได้จัดเรียงข้อมูล และได้แสดงตัวอย่างในการจัดเรียงข้อมูลแบบ Quick sort ได้ในรูป 2.3 ด้านล่าง



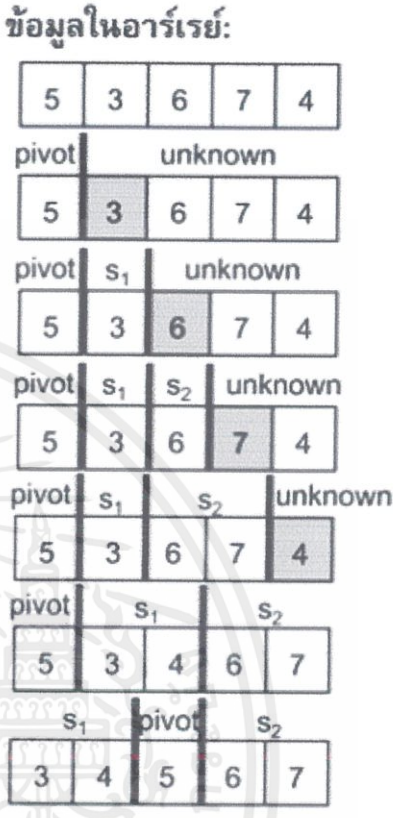
รูปที่ 2.3 การจัดเรียงข้อมูลของควิกซอร์ด

นี่คือ Pseudo code ของ Quick sort ดังรูปที่ 2.4

```
1   +quickSort(inout theArray[]:ItemArray,in first:integer,in last:integer):ItemArray
2     if(first < last){
3       Choose a pivot item p from theArray[first..last]
4       Partition the items of theArray[first..last] about p
5       quickSort(theArray,first,pivotIndex-1)
6       quickSort(theArray,pivatInde+1,last)
7       return theArray
8   +partition(in theArray[]:DataType,in first:integer,in last:integer):integer
9     Choose the pivot and swap it with theArray[first]
10    p = theArray[fist]
11    lastS1 = first
12    firstUnknown = first+1
13    while(firstUnknown <= last)
14      if(theArray[fistUnknown] < p){
15        Move theArray[fistUnknown] into S1
16      }else{
17        Move theArray[fistUnknown] into S2
18      }
19    }
20    Swap theArray[first] with theArray[lastS1]
21    return lastS1
```

รูปที่ 2.4 Pseudo code ของ Quick sort

ในการวัดประสิทธิภาพของ Quick sort นั้น ให้ดูจากรูปที่ 2.5 และ 2.6



รูปที่ 2.5 การวัดประสิทธิภาพของ Quick sort worst-case (ซ้าย)

รูปที่ 2.6 การวัดประสิทธิภาพของ Quick sort average-case (ขวา)

วิเคราะห์ประสิทธิภาพรูปที่ 2.5

ข้อมูลของ pivot มีค่าน้อยที่สุดในอาร์เรย์ ซึ่งในกรณีนี้ต้องเปรียบเทียบข้อมูลทั้งหมด n-1 ครั้ง จากข้อมูลทั้งหมด n ข้อมูล และในรอบต่อไปในการเรียกใช้ฟังก์ชัน/เมธอด quickSort ซ้ำจะเปรียบเทียบข้อมูลเท่ากับ n-2 ครั้ง ทำจนกระทั่งข้อมูลในการจัดเรียงเหลือ 1 ข้อมูล ซึ่งเท่ากับ

$$1 + 2 + ... + (n-1) = n \times \left(\frac{n-1}{2}\right) = \frac{1}{2}(n^2 - n)$$

Worst case ของ Quick sort คือ n^2

วิเคราะห์ประสิทธิภาพรูปที่ 2.6

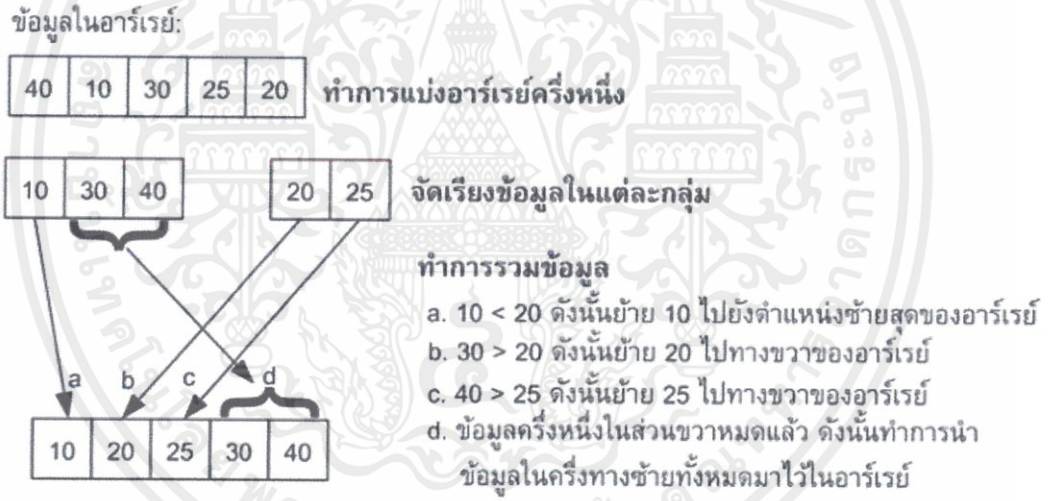
มีการวนลูปตามจำนวนทั้งหมด n ข้อมูล และจัดเรียงข้อมูลเท่ากับ $\log_2 n$ ดังนั้นทำให้ big O ของ Average case เท่ากับ $n\log_2 n$

การจัดเรียงข้อมูลที่มีทั้งส่วน s_1 และ s_2 เมื่อพิจารณาค่า average-case คือมีจำนวนข้อมูลภายใน s_1 และ s_2 ที่มีค่าที่ใกล้เคียงกัน ในกรณีนี้ค่า big-O จะเท่ากับ $O(\log_2 n)$

2.1.2.2 Merge Sort

การจัดเรียงข้อมูลแบบ Merge sort เป็นการจัดเรียงข้อมูลที่มีประสิทธิภาพ โดยจะใช้หลักการการเรียกซ้ำ (Recursive) ของอัลกอริทึมที่ทำหน้าที่ในการจัดเรียงข้อมูลกลับมาใช้ใหม่ตลอด

การจัดเรียงข้อมูลแบบ Merge sort จะเริ่มจากการแบ่งข้อมูลในอาร์เรย์ออกเป็นส่วนย่อยแล้วนำข้อมูลในส่วนย่อยนี้ไปจัดเรียงข้อมูลภายในส่วนย่อยนั้นๆ เมื่อจัดเรียงข้อมูลในส่วนย่อยเสร็จเรียบร้อยแล้ว ก็จะนำข้อมูลในส่วนย่อยกลับมารวมกัน (Merge) ใหม่อีกครั้งหนึ่ง



รูปที่ 2.7 การจัดเรียงข้อมูลแบบ Merge Sort

นี่คือ Pseudo code ของ Merge sort ดังรูปที่ 2.8

```
+mergeSort(inout theArray[]:ItemArray,in first:integer,in last:integer): ItemArray
  if(first < last){
    mid = (first+last)/2
    mergeSort(theArray,first,mid)
    mergeSort(theArray,mid+1,last)
    merge(theArray,first,mid,last)
  }
  retrun theArray
+merge((in theArray[]:DataType,in first:integer,in mid:integer,in last:integer)
  tempArray = create new array
  first1 = first
  last1 = mid
  first2 = mid+1
  last2 = last
  index = first
  while(first1 <= last1)&&(first2 <= last2){
   if(theArray[first1] < theArray[first2]){
     tempArray[index] = theArray[first1]
     first = first1+1
   }else{
     tempArray[index] = theArray[first2]
     first2 = first2+1
   }
   index = index+1
  }
  while(first1 <= last1){
    tempArray[index] = theArray[first1]
    first1 = first1+1
    index = index+1
  }
  while(first2 <= last2){
    tempArray[index] = theArray[first2];
    first2= first2+1
    index = index+1
  }
  for(index = first;index <= last;index++){
     theArray[index] = tempArray[index];
  }
```

รูปที่ 2.8 Pseudo code ของ Merge sort

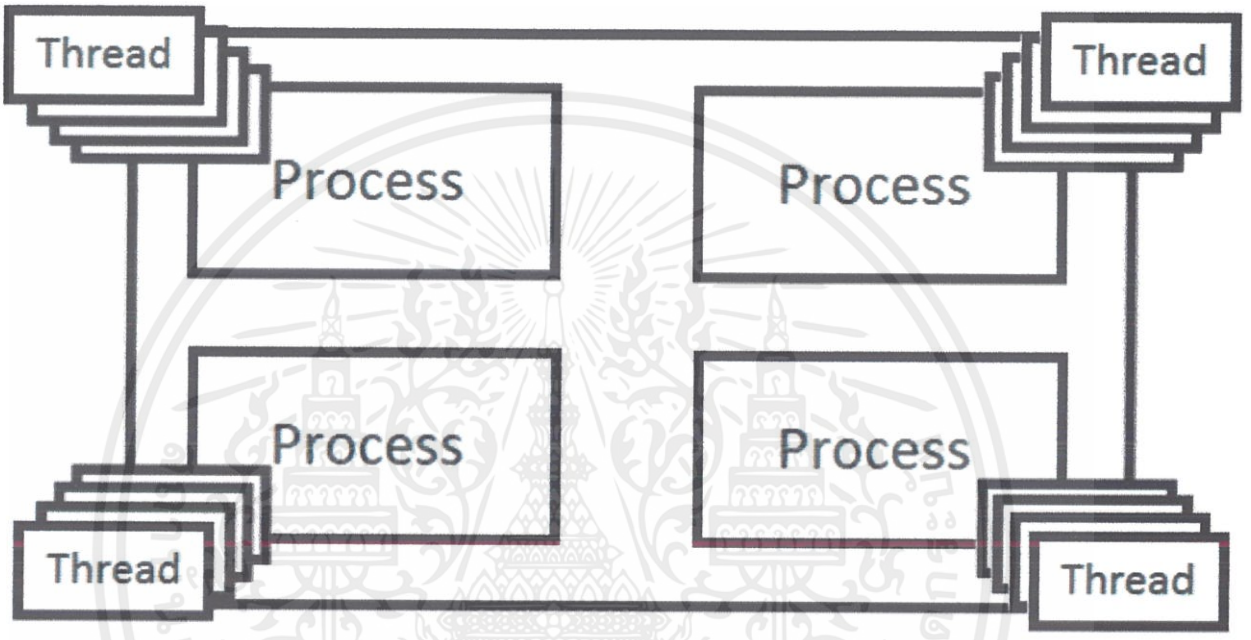
ส่วนประสิทธิภาพของ Merge sort นั้น การเรียกใช้ฟังก์ชัน/เมธอด mergesort ในครั้งที่ 1 แบ่งข้อมูลออกเป็นสองส่วน และเมื่อเรียกฟังก์ชัน/เมธอด mergesort ในครั้งต่อไป จะแบ่งข้อมูลทั้งหมดออกเป็นสี่ส่วน ทำจนกระทั่งแยกจำนวนกลุ่มข้อมูลเท่ากับจำนวนข้อมูลในอาร์เรย์คือ n ข้อมูล

ดังนั้นในการแบ่งข้อมูลเพื่อใช้ในการจัดเรียงข้อมูลเหมือนกับการหารสองของขนาดกลุ่มข้อมูลในการจัดเรียงข้อมูลซึ่งมีค่าเป็น $f(n) = \log_2 n$ และมีรอบในการจัดเรียงข้อมูลทั้งหมด n ข้อมูล ดังนั้น

ในกรณีของ worst-case การจัดเรียงข้อมูลแบบ Merge sort มีค่า big O เท่ากับ O ($n\log_2 n$)

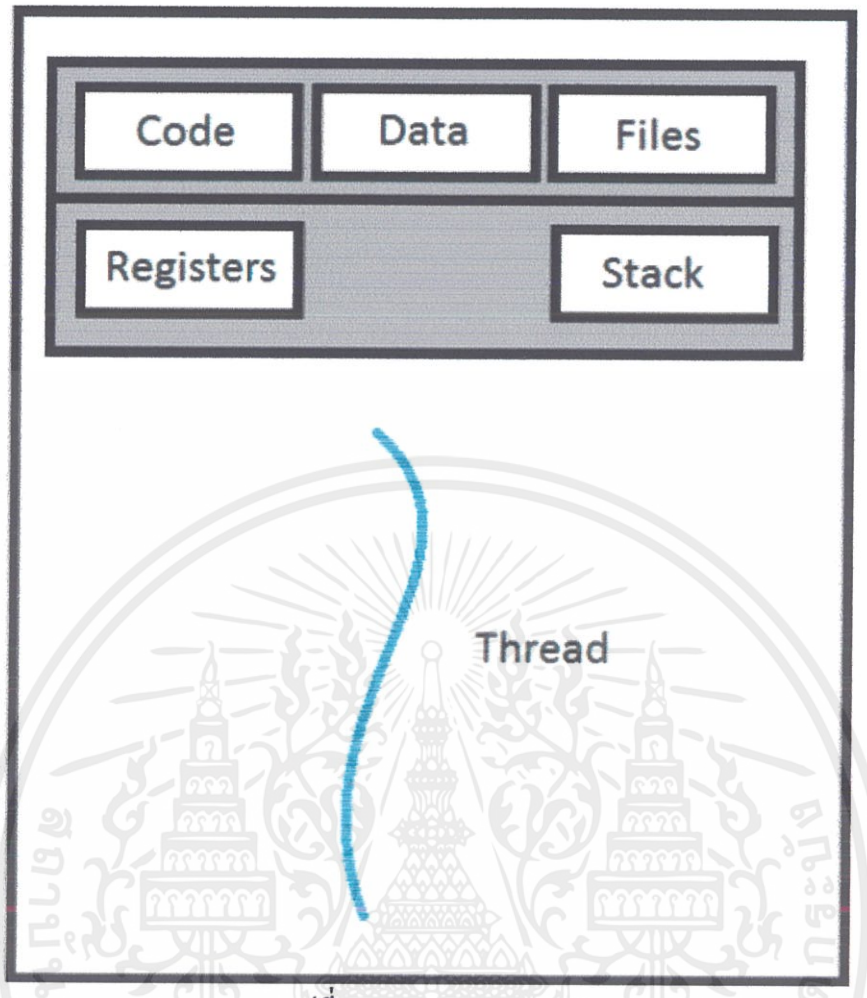
2.2 Multithread

Thread คือหน่วยการทำงานย่อยที่อยู่ใน Process โดยมีการแบ่งทรัพยากรต่างๆให้ โดยปกติแล้ว 1 Process จะมี 1 Thread ซึ่งถ้าหากใน 1 Process นั้นมีหลาย Thread จะเรียก Multithread โดย Multithread นั้นจะช่วยในการตอบสนองการทำงาน เช่น การรองรับผู้ใช้หลายคนพร้อมกัน การใช้ทรัพยากรร่วมกันภายใน Process เดียวกัน



รูปที่ 2.9 Process และ Thread

ในบางเหตุการณ์การใช้งานแบบ Single Thread อาจต้องการการทำงานพร้อมกันหลายๆงาน เช่น Web service ที่ต้องติดต่อกับ Client หลายตัว ถ้าหากเราใช้ Single Thread เราก็จะต้องคิดวิธีการจัดการเมื่อ Client หลายๆ Client เข้ามาใช้ Server พร้อมๆกัน เช่น การนำ Queue เข้ามาช่วยในการทำงานทีละตัว

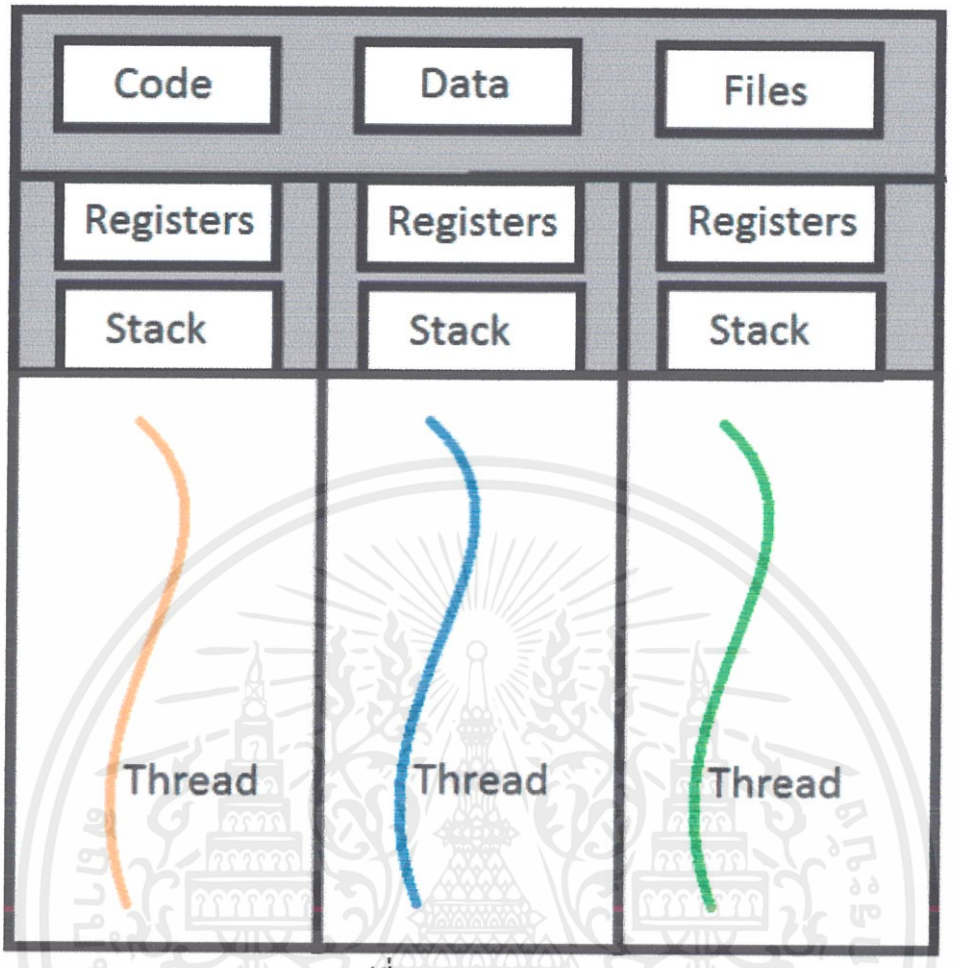


รูปที่ 2.10 Single Thread

แต่ถ้าหากเราใช้ Multithread เข้ามาช่วยในการรองรับ Client เราสามารถรองรับ Client ได้หลายๆ Client พร้อมกัน

แต่ละ Thread ประกอบไปด้วย

1. Thread ID หมายเลข Thread ใน Processor
2. Program counter ใช้นับคำสั่งที่ประมาณผล
3. Register set ใช้เก็บค่าที่ทำงานอยู่
4. Stack ใช้เก็บประวัติการประมวลผล

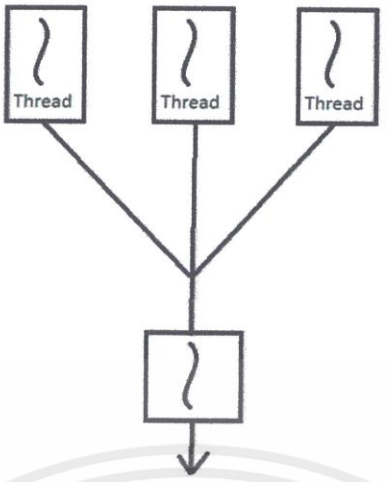


รูปที่ 2.11 Multithread

Multithreading Models

1. Many-to-one model

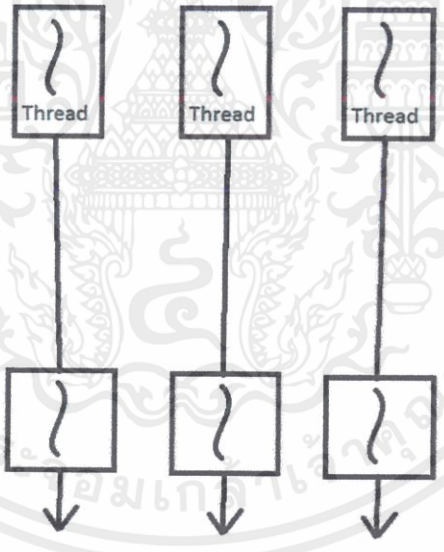
คือ Kernel Thread 1 ตัว กับ User Thread หลายตัว เป็นการออกแบบที่ยอมให้เพียง Thread เดียวเท่านั้นที่เข้าถึง Kernel Thread โดยโมเดลนี้ยอมให้สร้าง User Thread ได้หลายตัวตามต้องการ แต่ไม่สามารถประมวลผลพร้อมกัน เพราะยอมให้เข้าใช้ Kernel Thread ได้เพียงครั้งละ 1 Thread เท่านั้น



รูปที่ 2.12 การทำงานของ Multithread แบบ Many-to-one model

2. One-to-one model

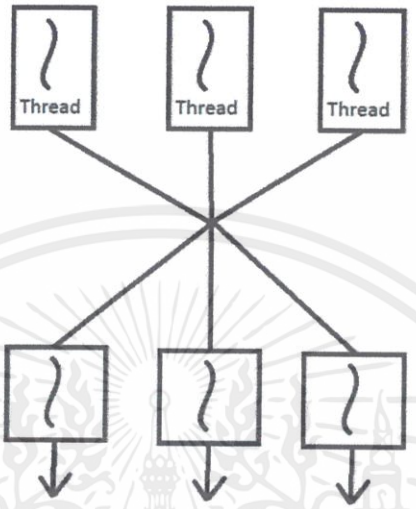
หมายถึง Kernel Thread 1 ตัว กับ User Thread 1 ตัว ซึ่งจะสามารถประมวลผลแบบขนานได้



รูปที่ 2.13 การทำงานของ Multithread แบบ One-to-one model

3. Many-to-many model

สามารถสร้าง User Thread หลายตัวและ Kernel Thread หลายตัวซึ่งจะประมวลผลแบบขนานบนการทำงานแบบ Multiprocessor



รูปที่ 2.14 การทำงานของ Multithread แบบ Many-to-many model

Fork and Join

ในการทำงานโดยใช้ Thread นั่นจะเห็นว่ามีการทำงานคือ

1.สร้าง Thread จำนวนหนึ่งขึ้นมา มอบหมายเลขงานให้แต่ละ Thread

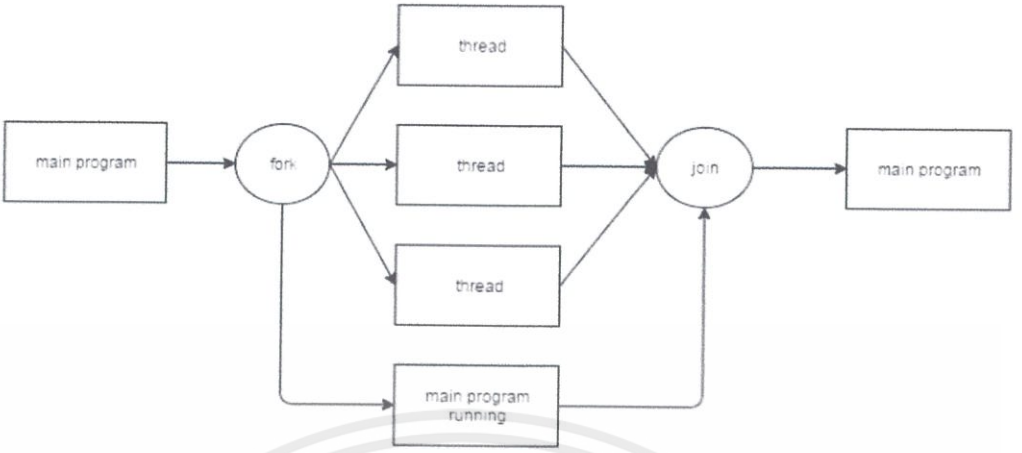
2.ให้ Thread นั่นทำงาน

3.รวมสรุปผล

ในขั้นตอนที่ 1 จะเห็นว่ามีลักษณะที่เหมือนส้อมเป็นด้ามตรงแล้วแยกออกมาซึ่งลักษณะแบบนี้เรียก Fork ซึ่งคือการที่ Process นั้นๆสร้าง Process ย่อยออกมา

ขั้นตอนต่อมาคือการปล่อยให้ Process ย่อยนั้นทำงาน

ขั้นตอนสุดท้ายคือการกลับมารวมกันซึ่งก็คือการ Join ซึ่งการ Join คือการทำให้ Thread ปัจจุบันรอจนกว่า Thread ย่อยนั้นทำงานเสร็จ จึงจะทำ Thread ปัจจุบันต่อ



รูปที่ 2.15 การทำงานแบบ Multithread ด้วยวิธี Fork และ Join

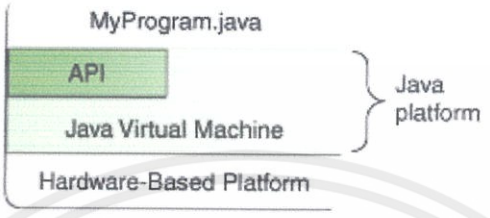
2.3 Java

รูปที่ 2.16 โลโก้ของภาษา Java

Java เป็นภาษาโปรแกรมที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุ (OOP : Object Oriented Programming) ที่พัฒนาโดยเจมส์ กอสลิงและวิศวกรคนอื่นๆที่บริษัท Sun microsystems ซึ่งถูกพัฒนาขึ้นในปี พ.ศ. 2534 (ค.ศ. 1991) โดยเป็นส่วนหนึ่งของ the Green Project และสำเร็จในปี พ.ศ. 2538 (ค.ศ. 1995) แต่เดิมเรียกภาษา Oak แต่มีปัญหาทางลิขสิทธิ์จึงเปลี่ยนไปใช้ชื่อ Java ซึ่งเป็นชื่อของกาแฟแทน สามารถเขียนโดยไม่ขึ้นกับแพลตฟอร์ม(สถาปัตยกรรม และ ระบบปฏิบัติการ) แต่ก็ต้องทำการติดตั้ง JVM (Java Virtual Machine) เพื่อให้ Java นั้นสามารถทำงานได้

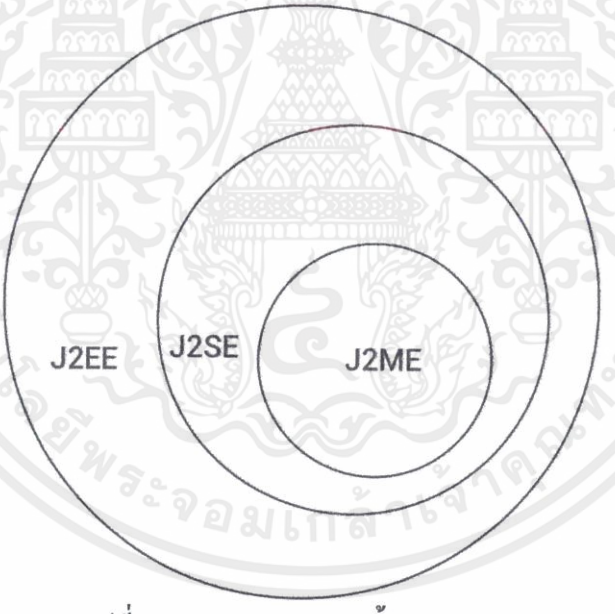
Java platform

ใน Java นั้นโปรแกรมจะต้องสามารถรันได้ทุก Platform เหมือนกันโดยไม่ขึ้นกับ Hardware และ Operating System ทำให้มีสิ่งที่เข้ามาช่วยคือ Java platform คือสภาพแวดล้อมสำหรับการรันโปรแกรม Java โดยจะมี Java Virtual machine และ Library มาตรฐานของ Java



รูปที่ 2.17 Java platform

โดยตัว Java platform นั้นจะมี 3 Edition ด้วยกัน



รูปที่ 2.18 Java platform ทั้ง 3 Edition

1. Java ME (Java Platform Micro Edition)
   เหมาะสำหรับการใช้พัฒนา Application บน Mobile device , PDA หรืออุกรณ์ขนาดเล็กในสมัยก่อน

2. Java SE (Java Platform Standard Edition)

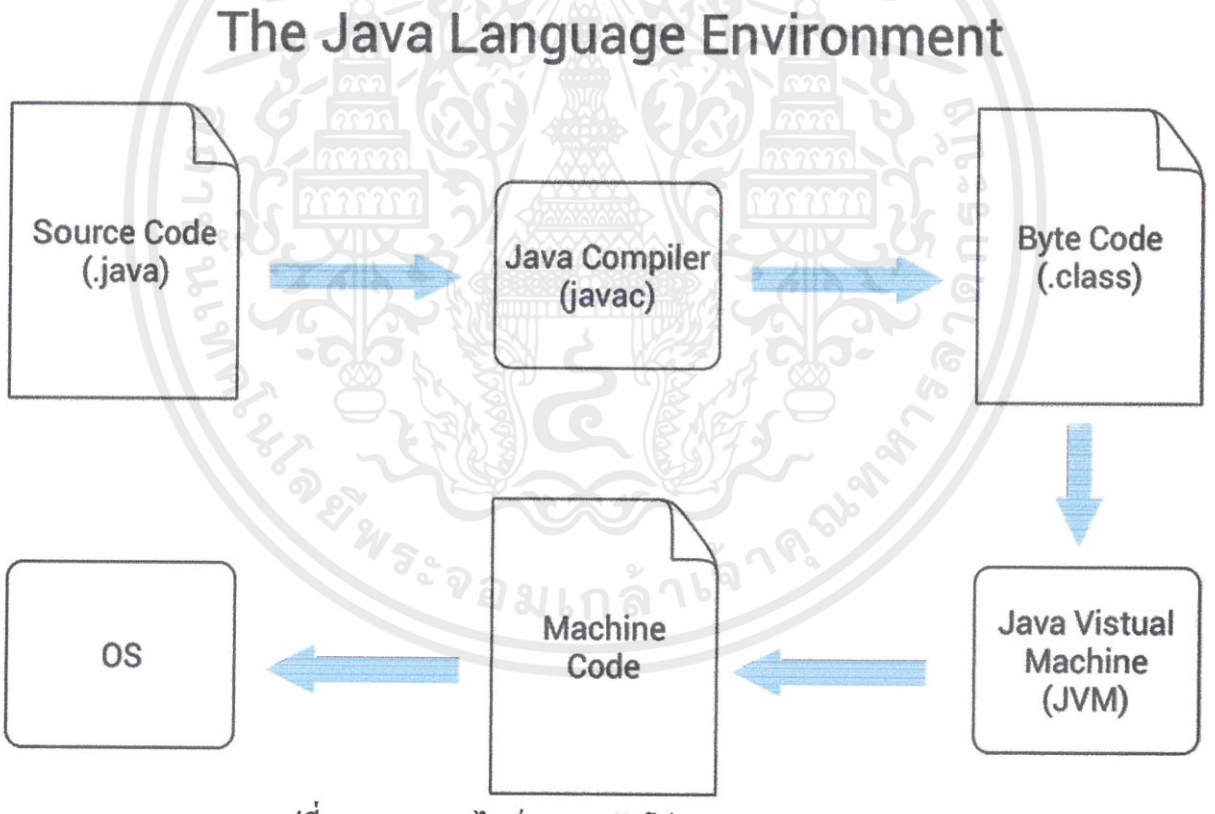
   จะมี API ที่จำเป็นสำหรับการเขียนโปรแกรมแบบ High Level เช่น การเชื่อมต่อ Database , GUI เหมาะสำหรับ Desktop Application

3. Java EE (Java Platform Enterprise Edition)

   จะมีทุกอย่างที่ Java SE มี แต่จะเหนือกว่าคือเทคโนโลยีที่ใช้พัฒนา Application บนฝั่ง Server เช่น JSP , Java Servlet

Java Compile และรันได้อย่างไร

เริ่มจาก Source code ภาษา Java นำไป Compile โดยผ่าน Javac ก็จะแปลง Source code.java นั้นเป็น .class ที่เป็น byte code แล้วก็จะเอา byte code นั้นไปรันบนเครื่องใดก็ได้ที่มี Java Platform โดยจะส่งลงไปใน JVM (Java Virtual Machine) เพื่อที่จะรันนั่นเอง



รูปที่ 2.19 การคอมไพล์และการรันโปรแกรมภาษา Java

ข้อดีข้อเสียของภาษา Java

ข้อดี

1.สามารถทำงานได้หลาย Platform

2.เป็นภาษาเชิงวัตถุ (OOP) เหมาะสำหรับการพัฒนาโปรแกรมที่มีความซับซ้อน

3.เป็นภาษาที่ตรวจสอบข้อผิดพลาด Compile time และ Run time ช่วยให้ Debug โปรแกรมง่ายขึ้น

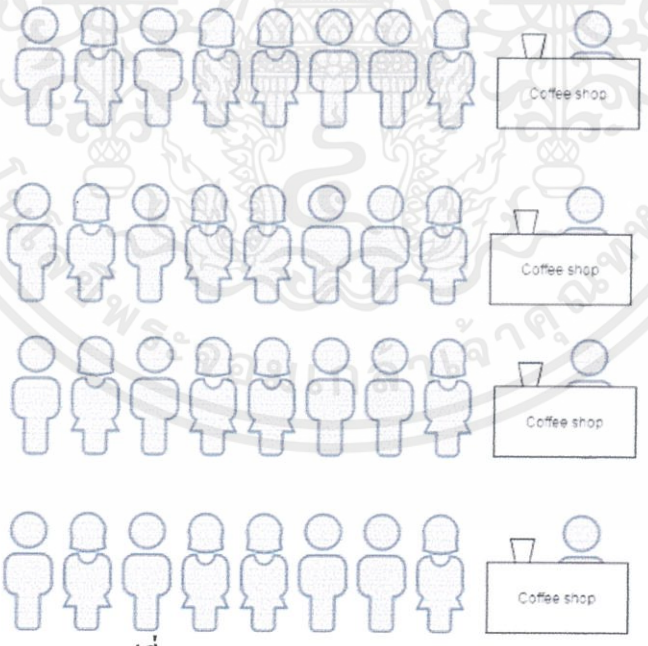
4.มี IDE, Library ต่างๆมากมาย

ข้อเสีย

ทำงานได้ช้ากว่า Native code (โปรแกรมที่ Compile ให้อยู่ในรูปของภาษาเครื่อง) เพราะต้องถูกเปลี่ยนเป็นภาษากลางก่อนจึงจะเป็นภาษาเครื่องอีกทีหนึ่ง

2.3.1 Parallelism

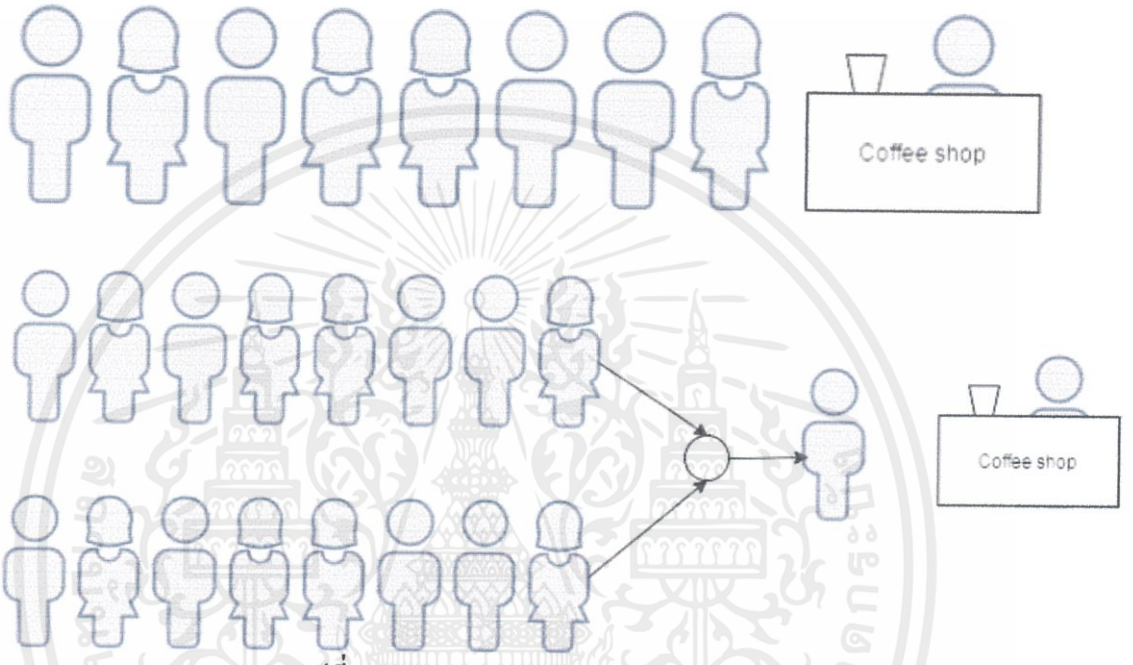
Parallelism คือการทำงานหลายๆอย่างในเวลาเดียวกันแบบขนาน ยกตัวอย่างเช่น มีร้านกาแฟ 2 ร้าน แต่ละร้านมีแถวร้านละ 1 แถว ทั้งสองนั้นสามารถซื้อกาแฟพร้อมกันได้โดยไม่ต้องรออีกแถว



รูปที่ 2.20 การทำงานแบบ Parallelism

2.3.2 Concurrency

Concurrency คือการทำงานหลายๆอย่างราวกับว่าเกิดขึ้นในเวลาเดียวกัน ยกตัวอย่างเช่น การเข้าแถวเพื่อซื้อกาแฟ ทุกคนสามารถอยากกินกาแฟพร้อมๆกันได้ แต่ไม่สามารถซื้อกาแฟได้พร้อมกัน ซึ่งจะแตกต่างกับ Parallel ตรงที่หากมีหลายแถวแต่มีร้านกาแฟเพียงร้านเดียวก็ต้องซื้อกาแฟได้เพียงทีละคนเท่านั่น



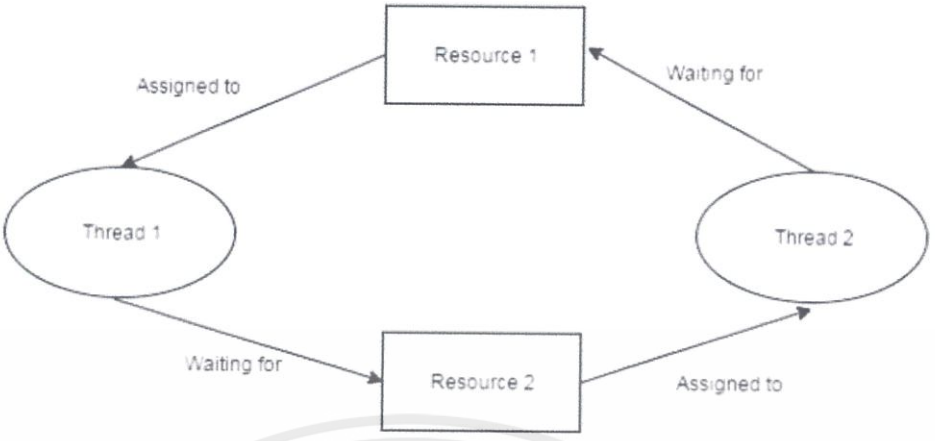
รูปที่ 2.21 การทำงานแบบ Concurrency

ซึ่ง Concurrency นั่นเราจะต้องบริหารจัดการทรัพยากรให้ดีไม่เช่นนั้นจะเกิดปัญหาที่ตามมา เช่น

Thread Interference เป็นเรื่องของการที่ Thread อัพเดทค่าที่ตัวแปรเดียวกันพร้อมกัน ทำให้ค่าที่ได้นั้นผิดไป

Memory Consistency Errors เป็นการดึงข้อมูลที่ผิดพลาด เช่น Thread 1 ต้องการดึงค่าที่ตัวเองอัพเดท แต่ว่าโดน Thread 2 ไปอัพเดททับ ทำให้ Thread 1 ได้รับข้อมูลที่ Thread 2 อัพเดทแทน

Deadlock คือการที่ Thread ต้องการใช้ทรัพยากรและร้องขอทรัพยากรจากระบบ แต่ทรัพยากรนั้นยังไม่วางเพราะว่ามี Thread อื่นกำลังใช้งานอยู่ และ Thread อื่นนั้นก็กำลังรอให้ทรัพยากรอื่นว่างเช่นกัน ทำให้เกิดการรอคอยที่ไม่สิ้นสุด



รูปที่ 2.22 การเกิด Deadlock

2.3.3 Programming

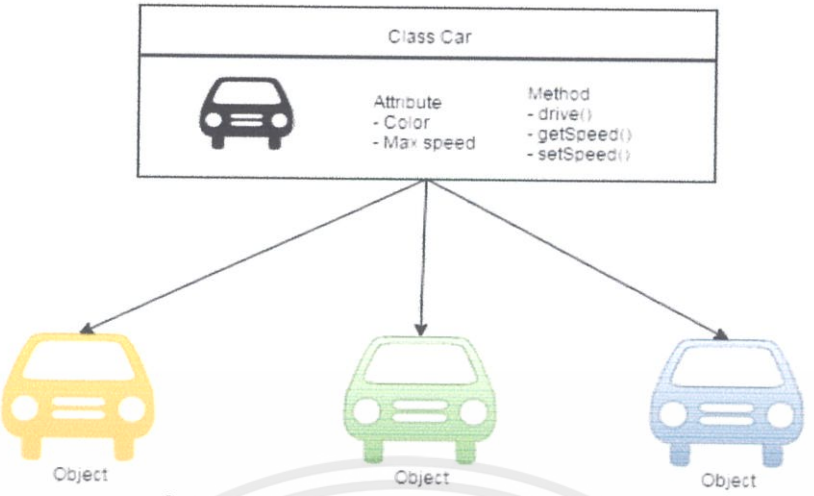
การเขียนโปรแกรม คือ การเขียนหรือสร้างคำสั่งให้คอมพิวเตอร์ทำงานได้ตามที่ต้องการ ด้วยภาษาที่คอมพิวเตอร์เข้าใจ โดยที่ผู้เขียนจะต้องเข้าใจถึงปัญหาและวิธีการแก้ปัญหา และรวมไปถึงคำศัพท์และไวยากรณ์ของภาษาตลอดจนกฎเกณฑ์ข้อบังคับต่างๆ

โดยภาษาที่ใช้เขียนโปรแกรมนั้นมีมากมายหลายภาษา เช่น C , C++ , Java , Python , Java script , php , R , Go เป็นต้น ซึ่งแต่ละภาษานั้นก็มีข้อดีข้อเสีย การนำไปใช้ที่แตกต่างดังนั้นควรเลือกภาษาให้เหมาะสมกับสิ่งที่จะด้วย

2.3.4 Object Oriented Programming

Object Oriented Programming (OOP) หรือ การเขียนโปแกรมเชิงวัตถุ เป็นรูปแบบหนึ่งในการเขียนโปรแกรมคอมพิวเตอร์ ที่ให้ความสำคัญกับ Object (วัตถุ) ซึ่งสามารถนำมาประกอบกันและนำมาทำงานร่วมกันได้ โดยการแลกเปลี่ยนข้อมูลเพื่อนำมาประมวลผลและส่งข้อมูลไปให้ Object อื่นที่เกี่ยวข้องเพื่อให้ทำงานต่อไป โดยแต่ละ Object จะประกอบด้วย

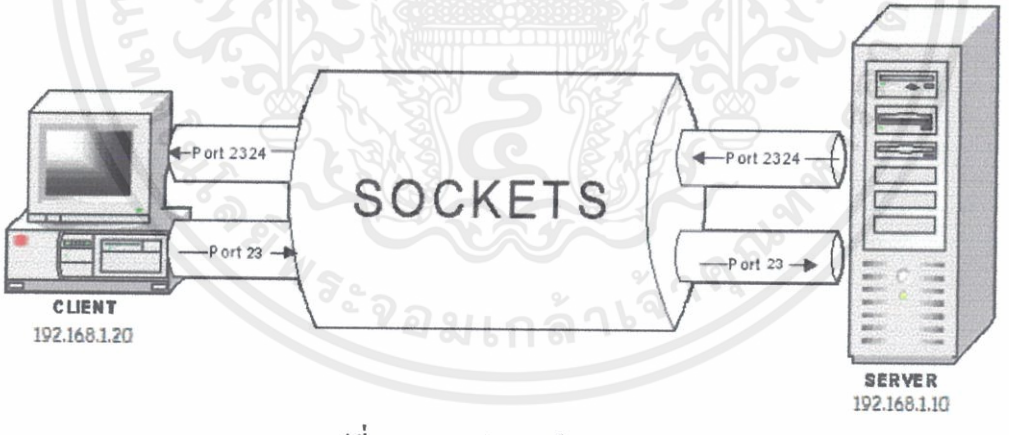
1. Attribute เป็นส่วนที่ใช้ในการเก็บข้อมูลต่างๆของ Object เช่น คน มี Attribute คือ ชื่อ สีผม เป็นต้น
2. Method เป็นสิ่งที่ Object นั้นสามารถทำได้ เช่น รถสามารถ วิ่ง เบรค เพิ่มความเร็ว เป็นต้น
3. Class คือ กลุ่มของ Object ที่มีโครงสร้างพื้นฐานพฤติกรรมเดียวกัน ดังนั้น Object ที่มีคุณสมบัติเดียวกันจะถูกรวมอยู่ใน Class เดียวกัน หรือก็คือ Class คือต้นแบบที่มีไว้สร้าง Object



รูปที่ 2.23 ความสัมพันธ์ระหว่าง Class และ Object

2.4 Socket

Socket เป็น Framework ที่ใช้ในการสื่อสารระหว่างโปรแกรมโดยผ่าน Network โดยในการสื่อสารนั้นเราจะต้องทำการระบุ 2 อย่างคือ IP และ Port ที่ต้องระบุ Port เพราะบนเครื่องคอมพิวเตอร์หนึ่งอาจจะทำงานที่ต้องใช้การสื่อสารผ่าน Network มากกว่า 1 โปรแกรม เหมือนกับการส่งจดหมายไปหาเพื่อนที่หอพักซึ่งเราต้องระบุที่อยู่ของหอพักและต้องระบุห้องที่เพื่อนของเรานั้นอยู่ด้วย ซึ่งที่อยู่ของหอพักนั้นเปรียบได้กับ IP และแต่ละห้องของหอพักเปรียบได้กับ Port นั้นเอง



รูปที่ 2.24 การทำงานผ่าน Socket

2.5 Virtual Machine

รูปที่ 2.25 ลักษณะการทำงานของ Virtual Machine

Virtual Machine คือการจำลองคอมพิวเตอร์ภายในคอมพิวเตอร์ เสมือนว่ามีคอมพิวเตอร์หลายเครื่องในเครื่องเดียว ประโยชน์ในการใช้งานลักษณะนี้เช่น การทดลองโปรแกรมหลายระบบปฏิบัติการ เมื่อเกิดความผิดพลาดใดๆก็จะไม่กระทบกับคอมพิวเตอร์ที่เป็นเครื่องหลัก

2.5.1 Virtual Box

รูปที่ 2.26โลโก้ Virtual Box

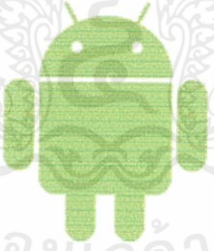
Virtual Box เป็นโปรแกรมประเภท Virtual Machine คือใช้พื้นที่ส่วนหนึ่งในฮาร์ดดิสก์จำลองเป็นเครื่องคอมพิวเตอร์หรือสมาร์ทโฟนขึ้นมา โดยจะมีการจัดสรรทรัพยากรต่างเช่น Ram CPU Network ร่วมกันกับเครื่องคอมพิวเตอร์ที่สร้างตัวมันเองขึ้นมาประโยชน์คือสามารถที่จะทำการจำลองระบบปฏิบัติการต่างๆ ทดลองการทำงานต่างๆโดยไม่ต้องหาเครื่องเหล่านั้นมาติดตั้งเพิ่มเติม

2.5.2 Genymotion

รูปที่ 2.27 โลโก้ของ Genymotion

Genymotion เป็น Android Emulator ที่มาพร้อมกับ Android system image หลายตัวทำให้ผู้พัฒนาไม่ต้องกังวลเรื่องการตั้งค่าต่างๆ อีกทั้งยังมีเครื่องมือที่เข้ามาช่วยในการพัฒนาหลายอย่างเช่น การเชื่อมต่อเข้ากับกล้อง, GPS เป็นต้น โดยตัว Genymotion จะใช้ Virtual Box มาช่วยในการสร้าง Virtual Machine บนเครื่องคอมพิวเตอร์ ทำให้เราสามารถทดสอบ Application บนสมาร์ทโฟน Android รุ่นต่างๆ ได้โดยไม่จำเป็นต้องมีสมาร์ทโฟนรุ่นนั้นจริงๆ

2.6 Android



รูปที่ 2.28 โลโก้ของ Android

Android เป็นระบบปฏิบัติการแบบ Open source มีพื้นฐานอยู่บน Linux ซึ่งถูกพัฒนาโดยบริษัท Android, Inc. ซึ่ง Google ได้ทำการซื้อบริษัทในปี พ.ศ. 2548ในอดีตถูกนำมาใช้กับอุปการณ์สมาร์ทโฟน, แท็บแล็ต ปัจจุบันถูกนำมาใช้ในอุปกรณ์อื่นๆอย่างแพร่หลาย

Android OS ถูกคิดค้นและพัฒนาขึ้นมาจากบริษัท Android Inc. โดยมี Andy Rubin กับเพื่อนร่วมกันก่อตั้งบริษัทขึ้นมาและในปี 2005 และต่อมา Google ได้เข้าซื้อกิจการของบริษัทและแต่งตั้งให้เป็นบริษัทในเครือของ Google ในปี 2007 Google ได้ทำการเปิดตัว Android เป็นครั้งแรกพร้อมกับการจัดตั้งกลุ่ม

ความร่วมมือทางธุรกิจโดยมีบริษัทยักษ์ใหญ่ด้านผู้ผลิตฮาร์ดแวร์, ซอฟต์แวร์, ฯลฯ เข้าร่วมกลุ่มมากมาย และในปีถัดมา Google ได้ทำการเปิดตัวสมาร์ทโฟน HTC Dream (HTC G1) ซึ่งเป็นสมาร์ทโฟนเครื่องแรกที่รันบนระบบปฏิบัติการแอนดรอยด์และนับเป็นจุดเริ่มต้นของแอนดรอยด์ที่เข้าสู่ตลาดสมาร์ทโฟน

เวอร์ชั่นของ Android นั้นมีผลต่อการใช้งานด้วยเช่นกัน เนื่องจากในแต่ละเวอร์ชั่นจะมีการปรับปรุงแก้ไขให้การทำงานดีขึ้นจากเวอร์ชั่นก่อนอยู่เสมอ ดังนั้นจึงนับเป็นปัจจัยนึงในการเลือกซื้อสมาร์ทโฟน Android มาใช้งานด้วยเช่นกัน และชื่อในแต่ละเวอร์ชั่นก็จะถูกตั้งชื่อเป็นชื่อของขนมต่างๆ และจะมีการเรียงลำดับตัวอักษรไปเรื่อยๆ (A-Z)

รูปที่ 2.29 สัญลักษณ์ของ Android เวอร์ชั่นต่างๆ

2.6.1 Android Studio

รูปที่ 2.30 โลโก้ของ Android Studio

Android Studio เป็น IDE Tool จาก Google ที่ใช้สำหรับการพัฒนา Application บนอุปกรณ์ที่ใช้ระบบปฏิบัติการ Android โดยพัฒนาจากแนวคิดพื้นฐานมาจาก InteliJ IDEA คล้าย ๆ กับการทำงานของ Eclipse และ Android ADT Plugin โดยวัตถุประสงค์ของ Android Studio คือต้องการพัฒนาเครื่องมือ IDE ที่สามารถพัฒนา App บน Android ให้มีประสิทธิภาพมากขึ้น ทั้งด้านการออกแบบ GUI และสร้างความสะดวกให้แก่ผู้ใช้งาน โดยในการใช้ Android Studio นี้จะใช้ภาษา Java ในการพัฒนาเป็นหลัก

บทที่ 3
การออกแบบและพัฒนา

3.1 ขอบเขตของโปรแกรมที่พัฒนา

- โปรแกรมสามารถ Sort แบบขนานโดยใช้โทรศัพท์หลายๆเครื่องหรือเครื่องเดียวได้
- โปรแกรมสามารถรองรับไฟล์ขนาดใหญ่ได้
- โทรศัพท์ที่เป็นตัวหลัก (Server) สามารถกระจายบอกโทรศัพท์ที่ไม่ใช่ตัวหลัก (Client) ได้

3.2 ข้อจำกัดของโปรแกรมที่พัฒนา

- โปรแกรมสามารถจัดเรียงข้อมูลได้เฉพาะตัวเลขขนาด 32 บิต

3.3 เครื่องมือที่ใช้พัฒนา

3.3.1 ซอฟแวร์ที่ใช้

Eclipse : ใช้สำหรับเขียนและรัน Java application

Android Studio : ใช้สำหรับเขียนโค้ดสำหรับ Android Application

Genymotion : ใช้สำหรับรัน Android Emulator

Virtual Box: ใช้สำหรับเก็บ Emulator ที่ Genymotion สร้างขึ้น

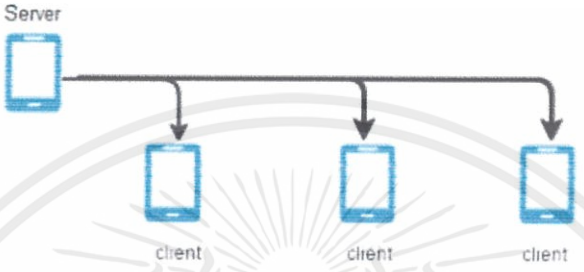
3.3.2 ภาษาที่ใช้

Java : เนื่องจากสามารถพัฒนาอัลกอริทึ่มให้ทำงานได้ทั้งบนโทรศัพท์และคอมพิวเตอร์ทั่วไป

3.4 รูปแบบโครงสร้างของโปรแกรม

3.4.1 หลักการออกแบบของโปรแกรม

Cluster จะใช้ลักษณะการทำงานแบบ “Client-Server” โดยจะให้โทรศัพท์หรือเครื่องคอมพิวเตอร์เครื่อง 1 ทำงานเป็น Server แจกข้อมูลไปยัง Client เครื่องต่างๆ โดยข้อมูลที่ต้องการนำมา Sort จะถูกเก็บอยู่ที่ Server ดังรูป



รูปที่ 3.1 การทำงานของโปรแกรม

3.4.2 การจัด Schedule

3.4.2.1 แนวคิดโครงสร้างของโปรแกรม

แนวคิดของโปรแกรมมาจาก Multithreading merge-sort เพราะวิธีนี้ใช้การแบ่งไฟล์ออกเป็นส่วนๆ ก่อนที่จะส่งไปยัง Client เพื่อลดเวลาในการจัดการไฟล์ลง และสามารถใช้กับจำนวนเครื่องที่ชัดเจนได้ เพราะในการทำ Multithread merge-sort แค่แบ่ง Array เป็นส่วนๆ ได้โดยง่าย

3.4.2.2 ลักษณะการทำงานของโปรแกรม

1. Server จะ Broadcast ให้ Client ทราบ IP และ Port ที่ใช้ติดต่อกัน
2. Client เชื่อมต่อเข้ามา และส่งข้อมูลของเครื่องตัวเองมายัง Server
3. Server คำนวณจำนวน Client และคำนวณหน้าที่ของ Client ว่าจะส่งหรือรับข้อมูลต้องทำกี่ครั้งลำดับเป็นอย่างไรโดยอ้างอิงจากข้อมูลประสิทธิภาพของแต่ละเครื่อง

4.1 Server ส่งข้อมูลแต่ละส่วน หน้าที่ และตารางที่ระบุ IP และ Port ของแต่ละเครื่องให้ Client แต่ละส่วน

4.2 Server ทำการ Sort ข้อมูลส่วนของตัวเอง

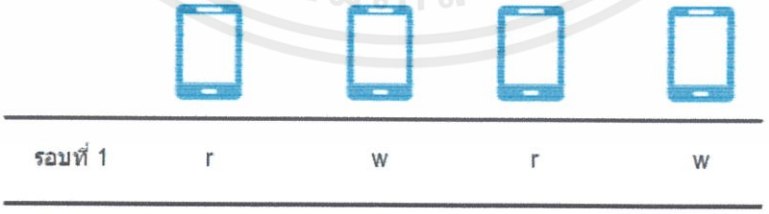
5. Client รับข้อมูลและทำการ Sort ข้อมูล ถ้า Client นั้นต้องรอรับข้อมูลด้วยก็ทำทั้ง Sort และรอรับข้อมูลพร้อม Sort ไปด้วยพร้อมกันในลักษณะ Multithread
6. Client ส่งข้อมูลไปหา Client หรือ Server ที่รอรับข้อมูลอยู่
7. Client รับข้อมูลเพื่อทำการ Merge ถ้า Client หรือ Server ต้องรับข้อมูลต่อไปอีก ก็ทำทั้ง Merge และรอรับข้อมูล

8. ทำข้อ 6 และข้อ 7 ซ้ำจนข้อมูลที่ถูก Sort แล้วทั้งหมดมาถึง Server
9. Server Merge ข้อมูลทั้งหมดแล้วเขียนลงไฟล์
10. ทำการตรวจสอบความถูกต้อง

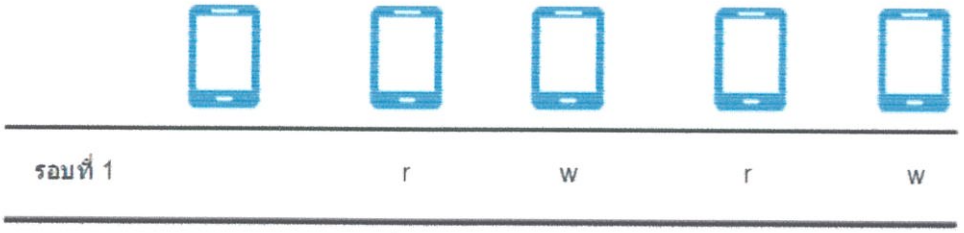
3.4.2.3 แนวคิดการจัด Schedule ของโปรแกรม

หลังจากที่ Server รู้จำนวน Client ที่เข้ามาเชื่อมต่อแล้ว หน้าที่ต่อไปของ Server คือจัดหน้าที่การทำงานของแต่ละเครื่องว่าจะ Read socket(r) หรือจะ Write socket(w) โดยจะมีการจัดลำดับดังนี้

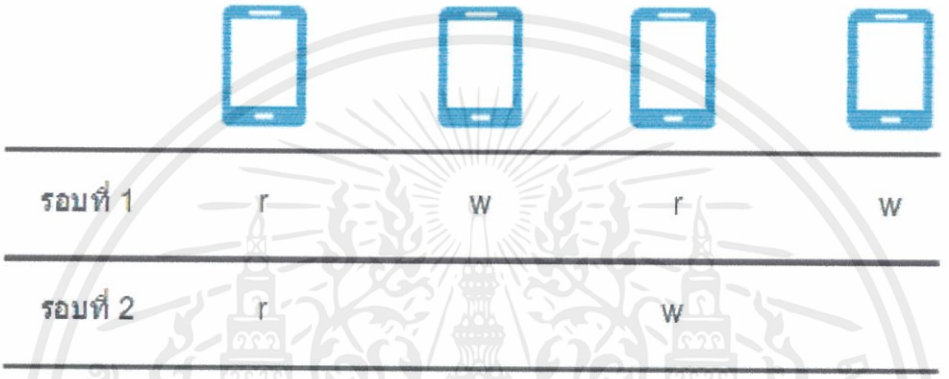
1. รอบแรกจะเริ่มที่ตัวสุดท้ายจะทำหน้าที่ Write socket และตัวถัดมาจากตัวสุดท้ายจะทำหน้าที่ Read socket
2. เขียนสลับ Write socket และ Read socket สลับกันจนมาจึงตัวก่อน Server
3. ถ้าตัวก่อน Server เป็น Read socket ตัว Server ไม่ต้องทำอะไร แต่ถ้าเป็น Write socket ตัว Server จะได้หน้าที่ Read socket ดังรูปที่ 3.2 และ รูปที่ 3.3
4. รอบต่อมาเริ่มที่ตัวสุดท้ายถ้าตัวปัจจุบันทำหน้าที่ Write socket ไปแล้วให้เลื่อนไปตัวถัดไปถ้าไม่ใช่ให้เพิ่มหน้าที่เป็น Write socket
5. เขียนสลับ Write socket และ Read socket สลับกันโดยที่ตัวที่จะใส่หน้าที่นั้นต้องยังไม่ได้ทำหน้าที่ Write socket สลับกันจนมาจึงตัวก่อน Server
6. ถ้าตัวก่อน Server เป็น Read socket ตัว Server ไม่ต้องทำอะไร แต่ถ้าเป็น Write socket ตัว Server จะได้หน้าที่ Read socket
7. รอบต่อๆไปทำข้อ 4 – 6 ซ้ำจนทุกเครื่องได้ทำหน้าที่ Write socket ยกเว้น Server ดังรูปที่ 3.4 และรูปที่ 3.5



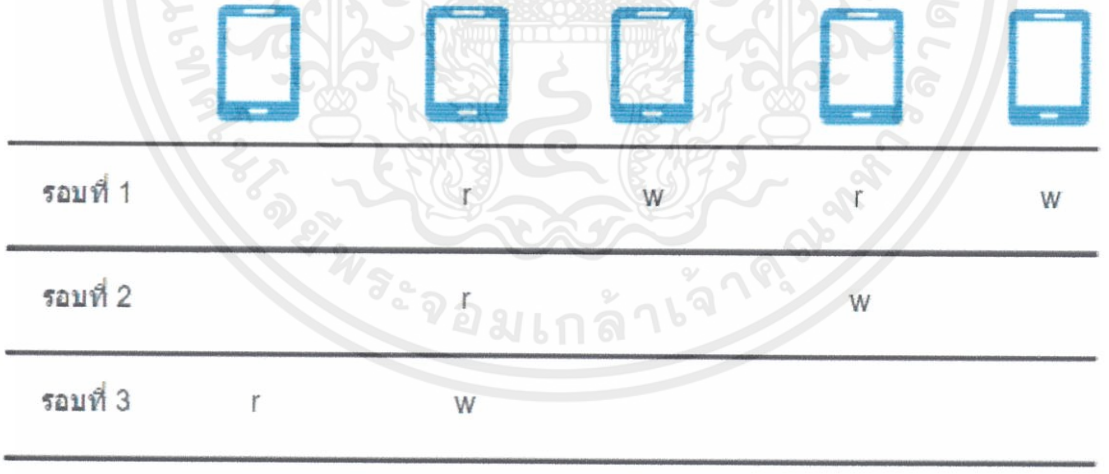
รูปที่ 3.2 การจัด Schedule รอบที่ 1 ในกรณี 4 เครื่อง



รูปที่ 3.3 การจัด Schedule รอบที่ 1 ในกรณี 5 เครื่อง



รูปที่ 3.4 การจัด Schedule ในกรณี 4 เครื่อง



รูปที่ 3.5 การจัด Schedule ในกรณี 5 เครื่อง

3.4.2.4 วิธีการแบ่งไฟล์

วิธีการแบ่งไฟล์ของ Cluster จะแบ่งเป็นส่วนๆตามที่ได้กำหนดไว้ และเมื่อแบ่งเสร็จส่วนใดส่วนหนึ่ง จะทำการส่งไฟล์ส่วนนั้นและทำการแบ่งไฟล์ส่วนอื่นในเวลาเดียวกัน

3.4.2.5 สัดส่วนการแบ่งไฟล์

สัดส่วนการแบ่งไฟล์มี 2 รูปแบบ

3.4.2.4.1 การแบ่งที่เท่ากันหมด

โดยไฟล์จะถูกแบ่งให้แต่ละ Device จะเท่ากัน

3.4.2.4.2 การแบ่งที่ตามประสิทธิภาพของเครื่อง

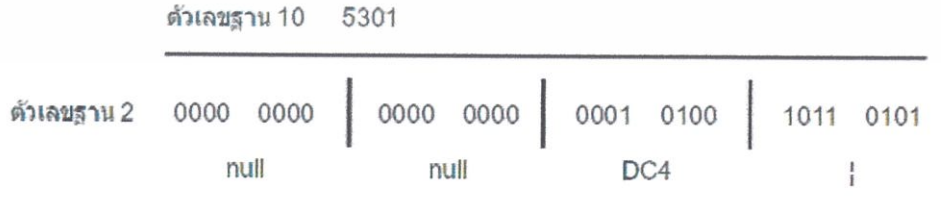
ไฟล์ที่ถูกแบ่งให้แต่ละเครื่องจะไม่เท่ากัน ขึ้นอยู่กับว่าจะจัดสัดส่วนแบบไหน

3.4.3 การจัดการภายในเครื่อง

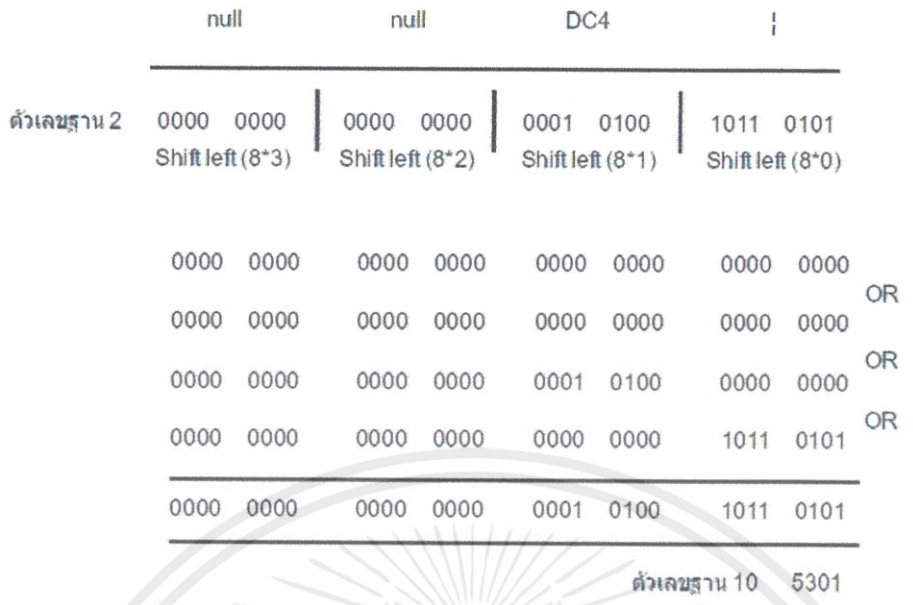
3.4.3.1 วิธีการอ่านไฟล์

ไฟล์หลักที่ใช้จะอยู่ในรูปแบบของ [ตัวเลข,ตัวเลข,] และจะทำการแบ่งไฟล์ตามจำนวนเครื่องที่มีอยู่หลังจากนั้นทำการนำเอาตัวเลขมาแปลงเป็น byte โดยการตัดแต่ละ 8 บิตของตัวเลขแบบไม่คิดเครื่องหมายทำให้ 1 ตัวเลขจะใช้พื้นที่ 4 byte (หรือ 32 บิต) ดังรูปที่ 3.6 และเขียนลงไฟล์แล้วจึงนำไฟล์นั้นส่งให้ Client แต่ละตัว ซึ่งในการแปลงตัวเลขให้เป็น 4 byte นั้นจะช่วยในการทำให้ฝั่งที่อ่านไฟล์รู้จำนวนข้อมูลโดยการหารขนาดไฟล์ด้วย 4 อีกทั้งการเขียนไฟล์เป็น byte นั้นยังเร็วกว่าการเขียนไฟล์ที่เป็น Integer

ในการอ่านไฟล์ที่ส่งผ่านแต่ละเครื่องไฟล์ก็ถูกเก็บอยู่ในรูปแบบของ Byte คือเลข 1 ตัว (32 บิต)จะใช้พื้นที่ 4 Byte ในการเก็บ ซึ่งแต่ละเครื่องก็จะอ่านขึ้นมาทีละ 4 byte และนำทั้ง 4 byte นั้นประกอบกลับกลายเป็นเลข 32 บิต ตามเดิมโดยการนำแต่ละ byte มา Shift left แล้วนำมา OR กันดังรูปที่ 3.7



รูปที่ 3.6 การแปลงเลข 32 บิตให้อยู่ในรูปของ 4 byte เพื่อเขียนลงไฟล์



รูปที่ 3.7 การแปลง Ascii 4 byte ให้เป็นเลข 32 บิต

3.4.3.2 การรันแบบ Single Thread

ในการรันแบบ Single Thread จะเป็นลักษณะที่นำไฟล์ที่ได้รับมาอ่านค่า Integer มาใส่ Array และทำการ Sort array นั้น และรอส่ง หรือรับข้อมูลมาเพื่อ Merge ตามจำนวนที่ได้รับมา

3.4.3.3 การรันแบบ Multithreaded

ในการรันแบบ Multithreaded จะใช้ลักษณะเดียวกับโปรแกรมโดยรวม นั่นคือ Multithread merge-sort โดยจะแบ่งไฟล์ที่ได้รับให้เท่ากับจำนวน Threads และทำงานตามรูปแบบเดียวกัน คือให้แต่ละ Thread มีหน้าที่ต่างๆ เช่น Thread ที่ทำการจัดเรียงข้อมูลที่ได้รับอย่างเดียวและ Thread ที่ทำหน้าที่ Merge ข้อมูลด้วย

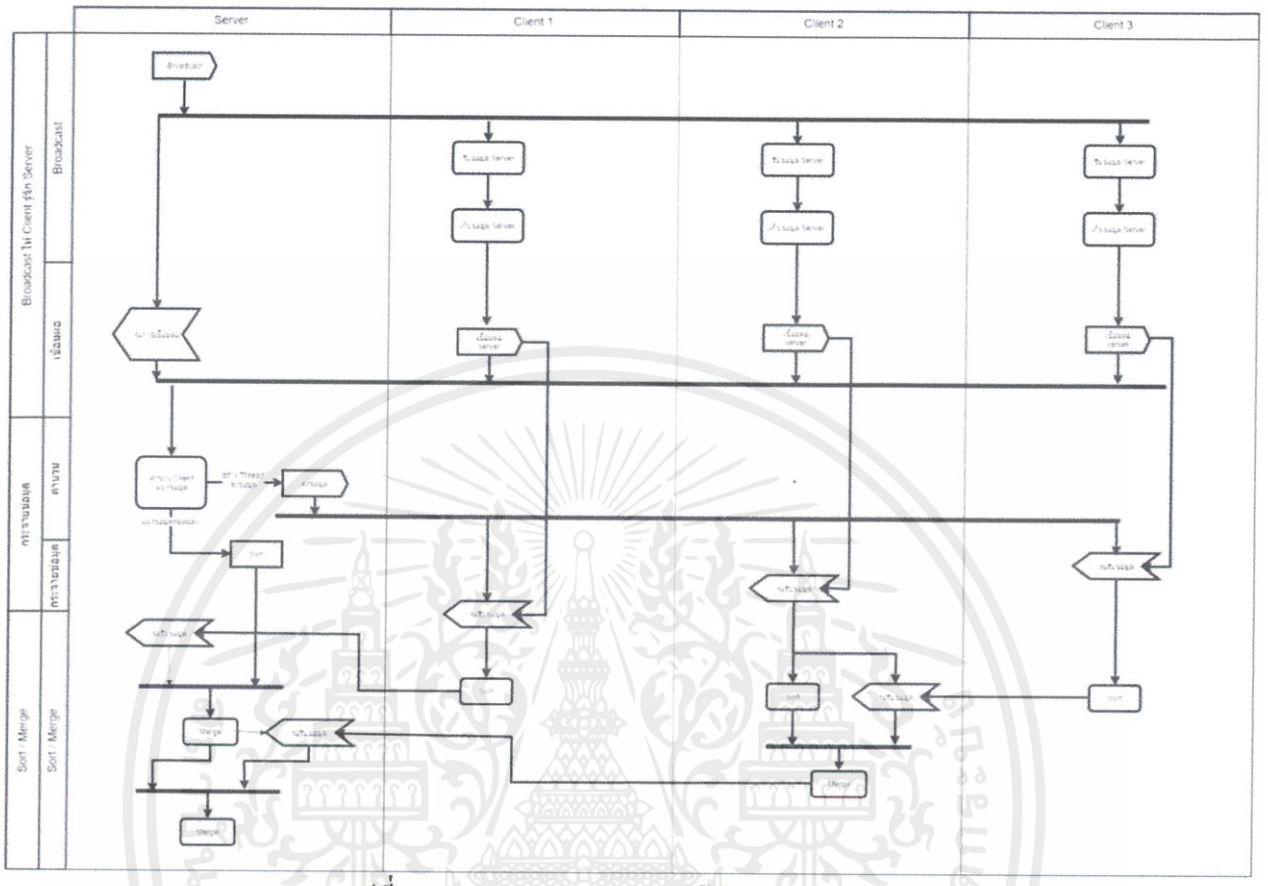
เมื่อข้อมูลถูกจัดเรียงเรียบร้อย จะรอส่งข้อมูลนั้น หรือรับข้อมูลมาเพื่อ Merge ต่อไป ตามหน้าที่ที่ได้รับมา

3.4.3.4 Sorting ที่ใช้

- Array.sort() ของ Java เป็นฟังก์ชั่น Dual-pivot quicksort ที่ Java นำมา Implement เป็นฟังก์ชั่นของตัวเอง
- Arrays.parallelSort() ของ Java เป็น Multithreaded sort ของ Java ที่ Implement มาจาก Multithreaded merge-sort โดยข้อมูลแต่ละส่วนที่ถูกแบ่งให้แต่ละThreadจะถูกจัดเรียงโดยฟังก์ชั่น Arrays.sort
- ส่วนของ Merge จาก Merge sort

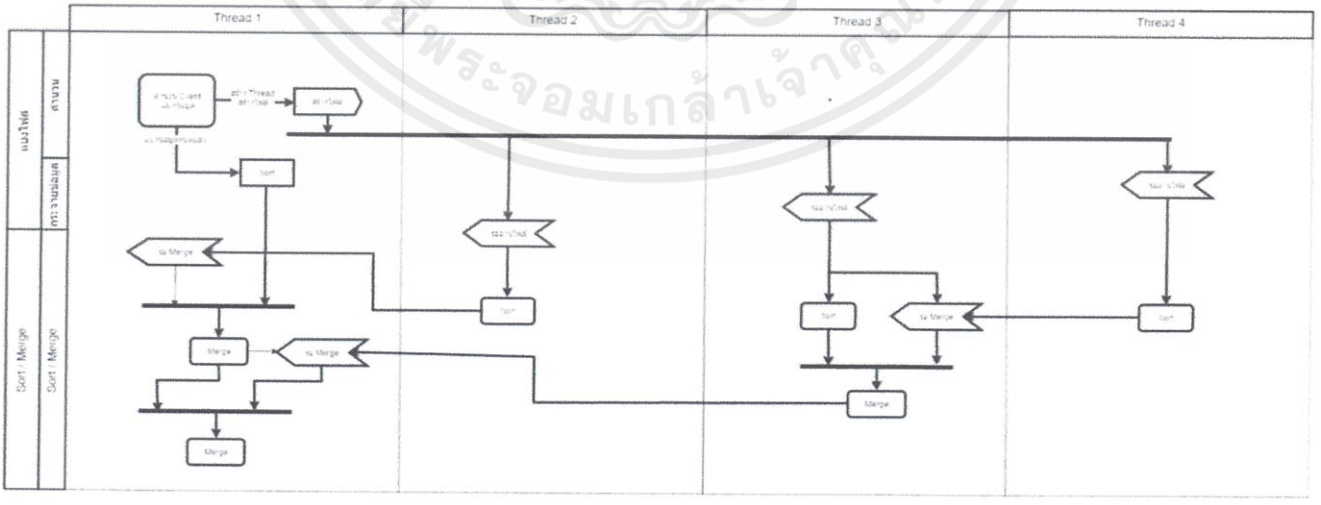
3.4.4 Activity diagram

การทำงานโดยรวม Device 4 เครื่องดังนี้



รูปที่ 3.8 Activity Diagram ของโปรแกรม

การทำงานส่วนของการ Sort



รูปที่ 3.9 การทำงานของโปรแกรมในส่วนของการ Multithread Sort

3.4.5 Architecture

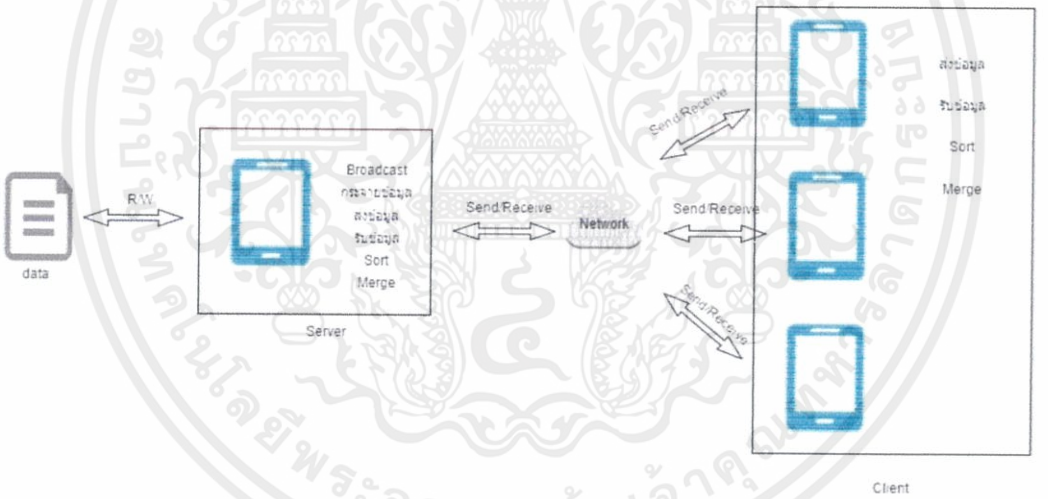
ระบบจะแบ่งออกเป็น 3 ส่วนคือ

- Server
- Client
- Network

Sever จะทำหน้าที่อ่านข้อมูลจากไฟล์ นำไปคำนวณเพื่อแบ่งข้อมูลในไฟล์ว่าต้องแจกข้อมูลให้แต่ละ Client เท่าไหร่แล้วจึงจะส่งข้อมูลแต่ละส่วนและหน้าที่ไปให้ Client หลังจากนั้นก็จะทำการ Sort ข้อมูลส่วนของตัวเอง และรอรับข้อมูลจาก Client เพื่อนำมา Merge เมื่อได้ข้อมูลที่สำเร็จแล้วจึงจะนำไปเขียนลงในไฟล์

Client จะทำหน้าที่รอรับข้อมูลจาก Server เพื่อนำไป Sort แล้วส่งข้อมูลที่ Sort แล้วไปให้ Server หรือ Client ตัวต่อไปเพื่อทำการ Merge

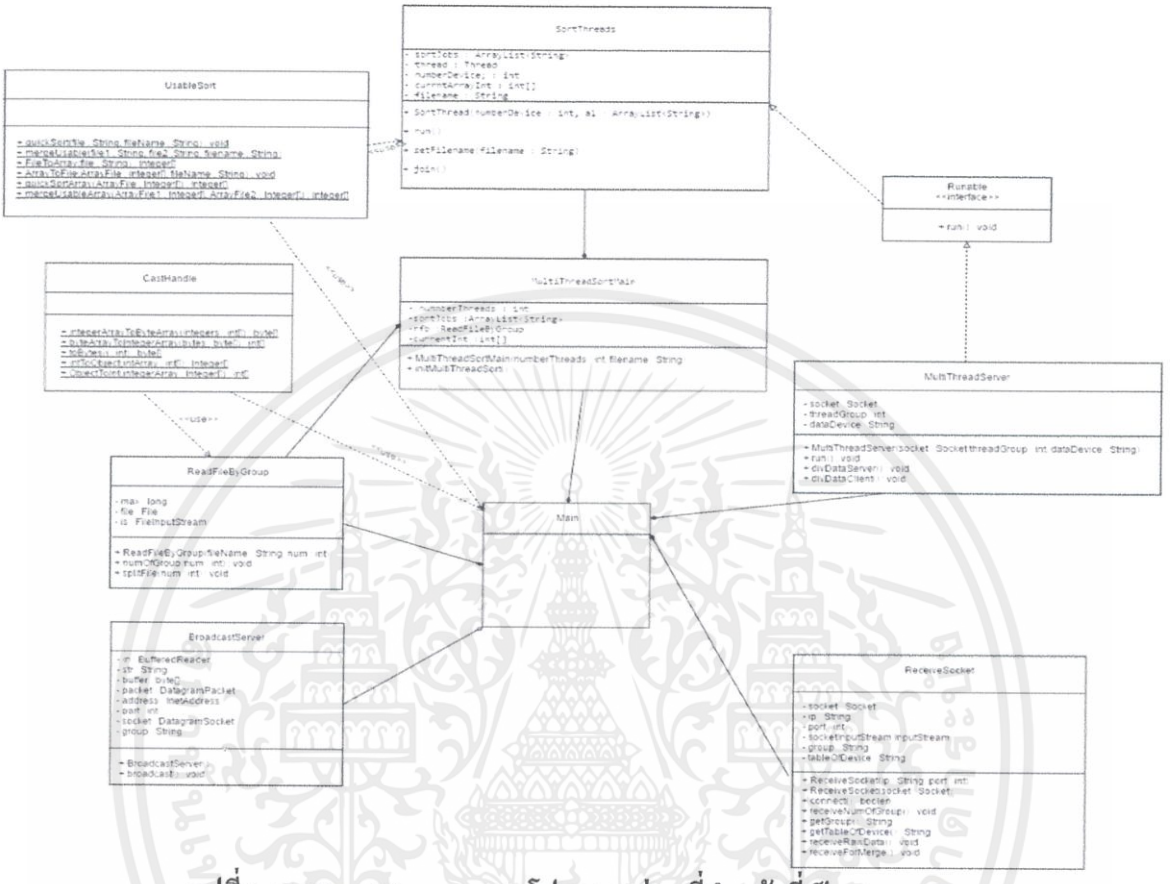
Network ในการส่งหรือรับข้อมูลนั้นจะกระทำผ่าน Network โดยใช้ Socket io



รูปที่ 3.10 Architecture

3.4.6 Class diagram

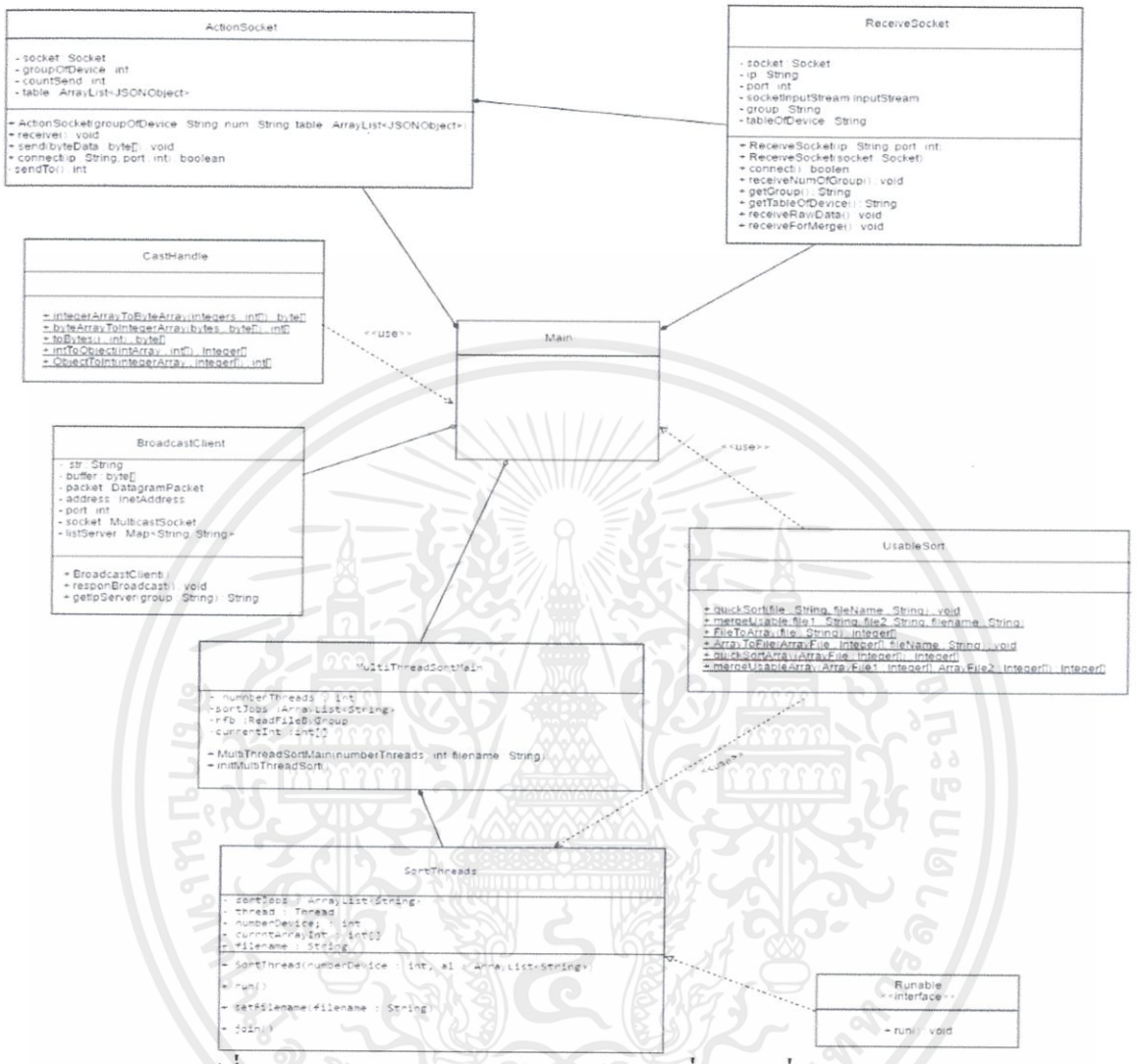
Class Diagramของ Serverในรูปที่ 3.5 ประกอบด้วย



รูปที่ 3.11 Class Diagram ของโปรแกรมส่วนที่ทำหน้าที่เป็น Server

- BroadcastServer ทำหน้าที่เกี่ยวกับ Broadcast กระจายบอก Client ว่า Server นี้อยู่ที่ IP และ Port หมายเลขอะไร
- MultiThreadServer ทำหน้าที่แยก Thread เพื่อส่งข้อมูลหาแต่ละ Client
- ReadFileByGroup ทำหน้าที่อ่านข้อมูลจาก File แล้วแบ่งไฟล์เป็นกลุ่มๆ
- ReceiveSocket ทำหน้าที่ในการรับข้อมูลผ่าน Socket
- CastHandle ทำหน้าที่ในการแปลงชนิดของข้อมูลแบบต่างๆ
- UsableSort ทำหน้าที่เกี่ยวกับการจัดเรียงข้อมูล
- MultiThreadSortMain ทำหน้าที่จัดการการ Sort แบบ MultiThread
- SortThread ทำหน้าที่เกี่ยวกับการ Sort

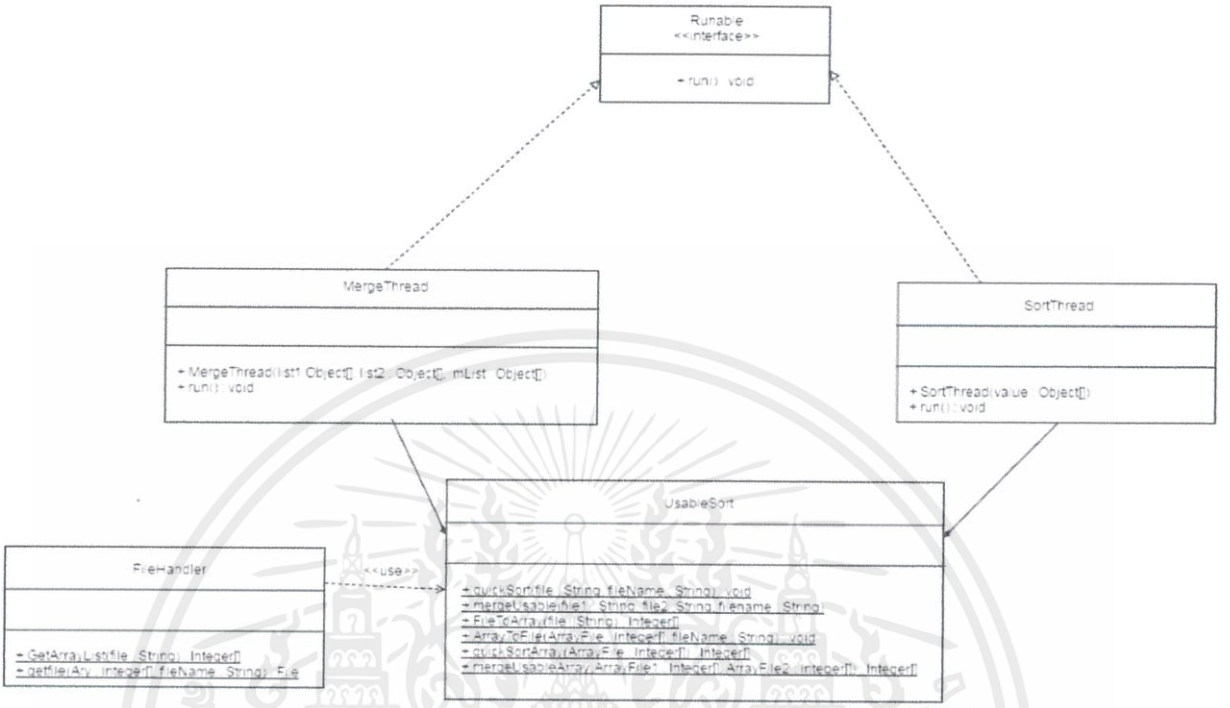
Class Diagram ของ client ประกอบด้วย



รูปที่ 3.12 Class Diagram ของโปรแกรมส่วนที่ทำหน้าที่เป็น Client

- BroadcastClient ทำหน้าที่ในการรับ Broadcast ข้อมูลจาก Server
- ReceiveSocket ทำหน้าที่ในการรับข้อมูลผ่าน Socket
- ActionSocket ทำหน้าที่ในการเลือกว่าจะให้ Client ทำการ Read Socket หรือ Write Scoket
- CastHandle ทำหน้าที่ในการแปลงชนิดของข้อมูลแบบต่างๆ
- UsableSort ทำหน้าที่เกี่ยวกับการจัดเรียงข้อมูล
- MultiThreadSortMain ทำหน้าที่จัดการการ Sort แบบ MultiThread
- SortThread ทำหน้าที่เกี่ยวกับการ Sort

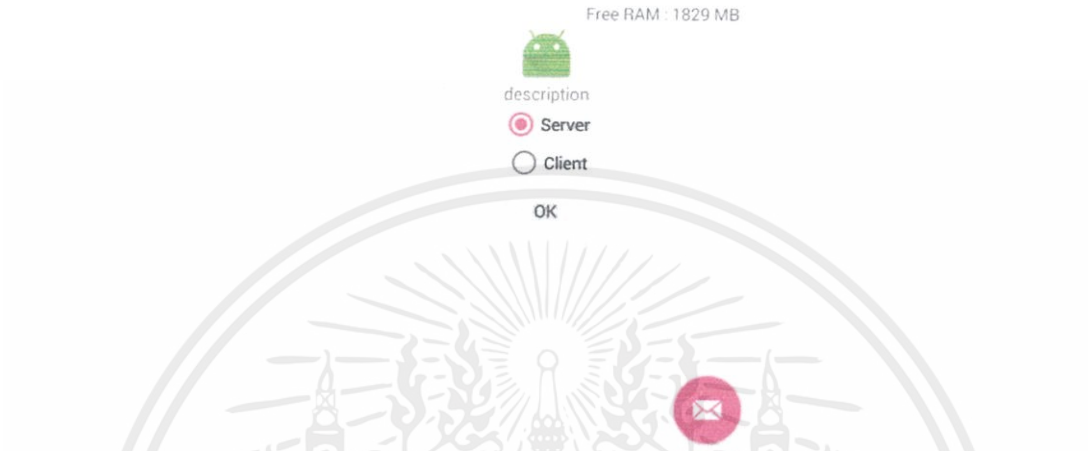
Class Diagram ของ UsableSort ประกอบด้วย



รูปที่ 3.13 Class Diagram ของ UsableSort

- MergeThread ทำหน้าที่ในการ Merge ข้อมูลเป็น Thread
- SortThread ทำหน้าที่ในการ Sort ข้อมูลเป็นโดยแบ่งเป็น Thread
- FileHandle ทำหน้าที่จัดการเกี่ยวกับ File

3.5 หน้าต่าง Activity การทำงานของโปรแกรมบนสมาร์ทโฟน



รูปที่ 3.14 หน้าจอหลัก

รูปที่ 3.14 เป็นรูปหน้าจอการทำงานหลักของโปรแกรม โดยจะมีให้เลือกว่าจะเป็น Server หรือ Client และจะแสดง พื้นที่ที่ยังไม่ได้ใช้งานของ Ram

รูปที่ 3.15 หน้าจอหลักของฝั่ง Server

รูปที่ 3.16 หน้าจอหลักของฝั่ง Client

รูปที่ 3.15 เป็นหน้าจอหลักของ Server โดยในหน้าจอนี้เราสามารถที่จะกำหนดไฟล์ที่เราจะใช้ เลือกจำนวน Thread ที่ต้องการและกรณีที่มี Client เชื่อมต่อเข้ามาเราสามารถเลือกว่าจะแบ่งไฟล์ขนาดเท่าใดได้ด้วย

รูปที่ 3.16 เป็นหน้าจอหลักของ Client โดยจะมีให้ใส่ IP ของ Server และจำนวน Thread ที่จะใช้ และสามารถทำงานได้หลังจากที่ตัว Server ทำการ Broadcast ข้อมูลไปหา Client

รูปที่ 3.17 หน้าจอการทำงานของ Server

รูปที่ 3.18 หน้าจอการทำงานของ Client

รูปที่ 3.17 หน้าจอการทำงานของ Server โดยจะแสดงงานที่ Server ได้ทำและ เวลาการทำงานทั้งหมดในหน่วยมิลลิวินาที(ms)

รูปที่ 3.18 หน้าจอการทำงานของ Client โดยจะแสดงงานที่ Client ได้ทำ

บทที่ 4

การทดลองและผลการทดลอง

4.1 การทดลองบน Java Virtual Machine

การทดลองบน Java Virtual Machine จะมี 2 ส่วน

4.1.1. การทดลองเพื่อเปรียบเทียบเวลาในการเปรียบเทียบการทำงานระหว่างฟังก์ชั่น Arrays.sort() กับ Arrays.parallelSort() เพื่อจัดเรียงข้อมูลภายในแต่ละเครื่อง

4.1.1.1 คุณสมบัติของเครื่องคอมพิวเตอร์ที่ใช้

เครื่องที่ 1 CPU Intel® Core™ i3-2100 3.10 GHz 2 core 4 Thread หน่วยความจำ 8 GB

เครื่องที่ 2 CPU AMD A6-3650 APU 2.60 GHz 4 core 4 Thread หน่วยความจำ 12 GB

เครื่องที่ 3 CPU AMD E-350 APU 1.60 GHz 2 Core 2 Thread หน่วยความจำ 6 GB

เครื่องที่ 4 CPU AMD E-350 APU 1.60 GHz 2 Core 2 Thread หน่วยความจำ 6 GB

ทั้ง 4 เครื่องเชื่อมต่ออยู่บน Ethernet Switch ความเร็ว 10/100 Mbps

4.1.1.2 ตัวแปรการทดลอง

ตัวแปรต้น : จำนวน Thread ที่ใช้ในแต่ละเครื่อง(1,2,4,8), ฟังก์ชั่นในการจัดเรียง (Arrays.sort(), Arrays.parallelSort())

ตัวแปรตาม : เวลาที่ใช้ในแต่ละเงื่อนไข

ตัวแปรควบคุม : ประสิทธิภาพของแต่ละเครื่องคอมพิวเตอร์ตั้งโต๊ะ, จำนวนข้อมูล 100M

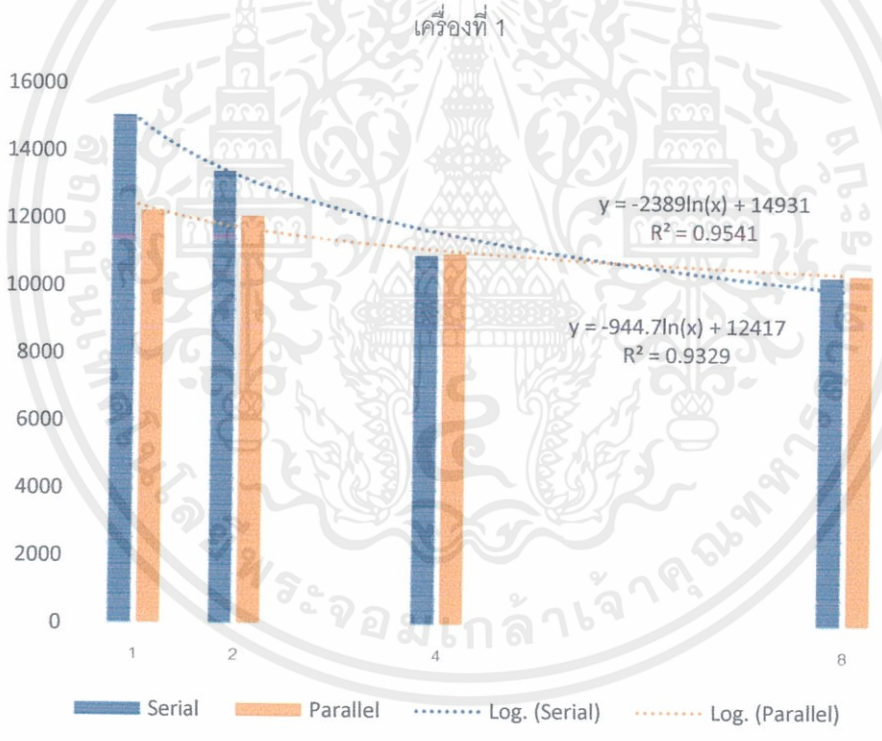
4.1.1.3 วิธีการทดลอง

1. เริ่มต้นรันโปรแกรมในแต่ละเครื่อง
2. เริ่มจับเวลา
3. รอจนโปรแกรมทำงานเสร็จ
4. ตรวจสอบความถูกต้องของข้อมูล
5. ตรวจสอบเวลาที่ใช้ในการรันโปรแกรมในแต่ละเครื่อง
6. บันทึกผลการทดลอง
7. ทำซ้ำข้อ 1 - 6 โดยเพิ่มจำนวน Thread

4.1.1.4 ผลการทดลอง

ตารางที่ 4.1 การทดลองจัดเรียงข้อมูล 100M ในเครื่องที่ 1

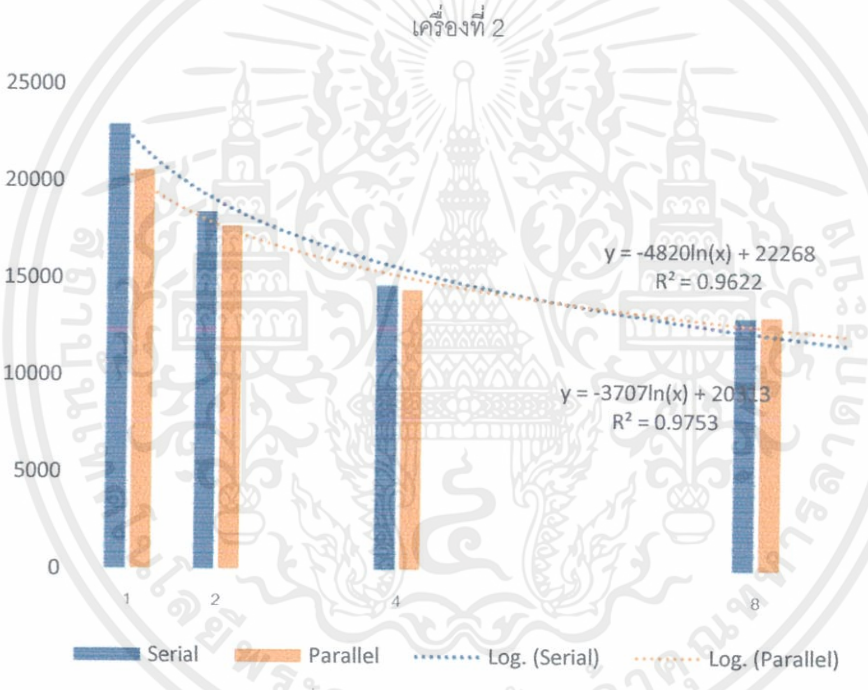
	เวลา(ms)	
จำนวน Thread	Serial	Parallel
1	15,068.00	12,231.50
2	13,408.00	12,090.25
4	10,942.50	11,004.75
8	10,369.50	10,410.50



รูปที่ 4.1 กราฟการทดลองจัดเรียงข้อมูล 100M ในเครื่องที่ 1

ตารางที่ 4.2 การทดลองจัดเรียงข้อมูล 100M ในเครื่องที่ 2

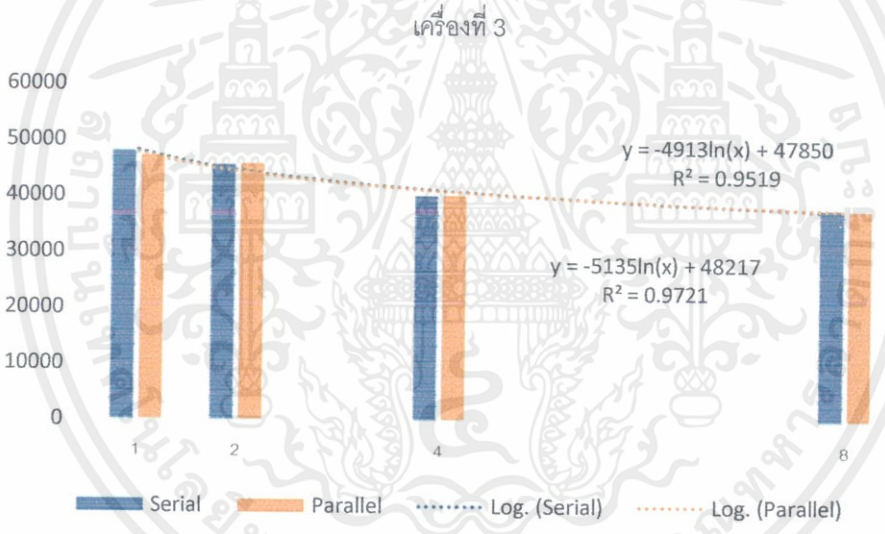
	เวลา(ms)	
จำนวน Thread	Serial	Parallel
1	22,915.75	20,573.00
2	18,426.50	17,716.75
4	14,644.00	14,449.50
8	13,040.25	13,098.00



รูปที่ 4.2 กราฟการทดลองจัดเรียงข้อมูล 100M ในเครื่องที่ 2

ตารางที่ 4.3 การทดลองจัดเรียงข้อมูล 100M ในเครื่องที่ 3

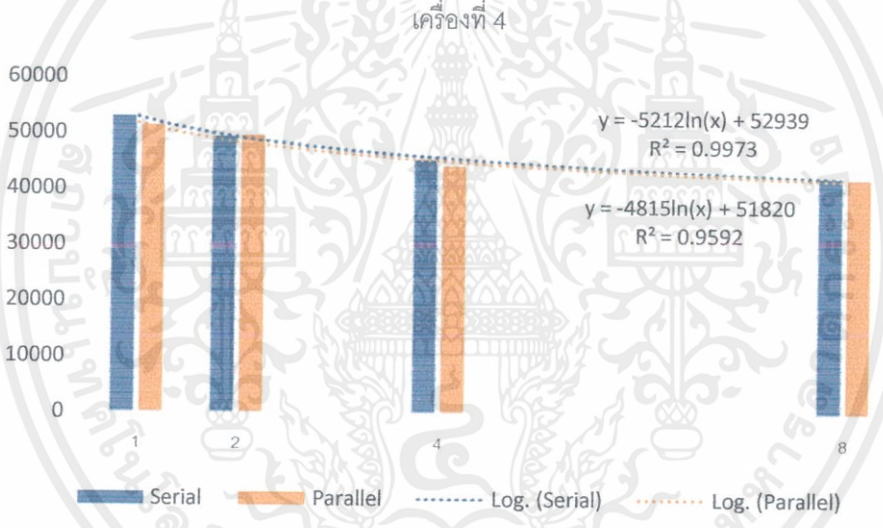
	เวลา(ms)	
จำนวน Thread	Serial	Parallel
1	47,916.50	47,199.75
2	45,562.00	45,815.50
4	40,192.50	40,251.00
8	37,841.50	37,703.75



รูปที่ 4.3 กราฟการทดลองจัดเรียงข้อมูล 100M ในเครื่องที่ 3

ตารางที่ 4.4 การทดลองจัดเรียงข้อมูล 100M ในเครื่องที่ 4

	เวลา(ms)	
จำนวน Thread	Serial	Parallel
1	52,983.00	51,377.75
2	49,436.00	49,607.25
4	45,362.75	44,224.00
8	42,298.00	42,047.25



รูปที่ 4.4 กราฟแสดงเวลาการทดลองจัดเรียงข้อมูล 100M ในเครื่องที่ 4

จากกราฟจะเห็นได้ว่าในแต่ละเครื่อง สำหรับ 1 Thread การใช้ฟังก์ชั่น Arrays.parallelSort() จะใช้เวลาน้อยกว่าฟังก์ชั่น Arrays.sort() แต่เมื่อเพิ่มจำนวน Thread ที่ใช้อัตราการลดลงของเวลาที่ใช้จะแตกต่างกันตามสมการที่แสดงในแต่ละกราฟ และเมื่อใช้ 8 Threads จะพบว่าทั้ง 2 ฟังก์ชั่นใช้เวลาในการทำงานใกล้เคียงกันมาก ทำให้เราเลือกที่จะใช้ 8 Threads และฟังก์ชั่น Arrays.sort() ในการทดลองถัดไป

4.1.2 การทดลองเปรียบเทียบเวลาสำหรับรูปแบบการแบ่งสัดส่วนให้แต่ละเครื่องที่แตกต่างกัน

โดยการทดลองนี้จะทำการรันโปรแกรมบน Java Virtual Machine 2-4 เครื่อง โดยจะทดลองหัวข้อหลักๆ 2 เรื่องนั่นคือ

- ทดลองโดยให้จำนวนข้อมูลที่ Server ได้รับเท่ากันทุกครั้งที่จำนวนเครื่องเปลี่ยนไป แต่จำนวนข้อมูลที่ Client ได้รับในแต่ละเครื่องจะมีจำนวนเท่ากัน
- ทดลองโดยให้จำนวนข้อมูลที่ Client ได้รับเท่ากันทุกครั้งที่จำนวนเครื่องเปลี่ยนไป

โดยการทดลองนี้ แต่ละเครื่องจะใช้ฟังก์ชั่น Array.sort() และใช้ 8 Threads ในการจัดเรียงข้อมูลจำนวน 100M

4.1.2.1 คุณสมบัติของเครื่องคอมพิวเตอร์ที่ใช้

เครื่องที่เป็น Server : CPU Intel® Core™ i3-2100 3.10 GHz 2 core 4 Thread หน่วยความจำ 8 GB

เครื่องที่ 2 CPU AMD A6-3650 APU 2.60 GHz 4 core 4 Thread หน่วยความจำ 12 GB

เครื่องที่ 3 CPU AMD E-350 APU 1.60 GHz 2 Core 2 Thread หน่วยความจำ 6 GB

เครื่องที่ 4 CPU AMD E-350 APU 1.60 GHz 2 Core 2 Thread หน่วยความจำ 6 GB

ทั้ง 4 เครื่องเชื่อมต่ออยู่บน Ethernet Switch ความเร็ว 10/100 Mbps

4.1.2.2 ตัวแปรการทดลอง

ตัวแปรต้น : สัดส่วนของข้อมูลที่ใช้

ตัวแปรตาม : เวลาที่ใช้ในแต่ละเงื่อนไข

ตัวแปรควบคุม : ประสิทธิภาพของแต่ละเครื่องคอมพิวเตอร์ตั้งโต๊ะ
จำนวนข้อมูล 100M

4.1.2.3 วิธีการทดลอง

1. เริ่มต้นรันโปรแกรม
2. เริ่มจับเวลาหลังจากเชื่อมต่อสำเร็จตามจำนวนเครื่องที่กำหนดไว้
3. รอจนโปรแกรมทำงานเสร็จ
4. ตรวจสอบความถูกต้องของข้อมูล
5. ตรวจสอบเวลาที่ใช้ในการรันโปรแกรม
6. บันทึกผลการทดลอง
7. ทำซ้ำข้อ 1 - 6 โดยการเปลี่ยนสัดส่วนและจำนวนเครื่องตามที่กำหนดไว้

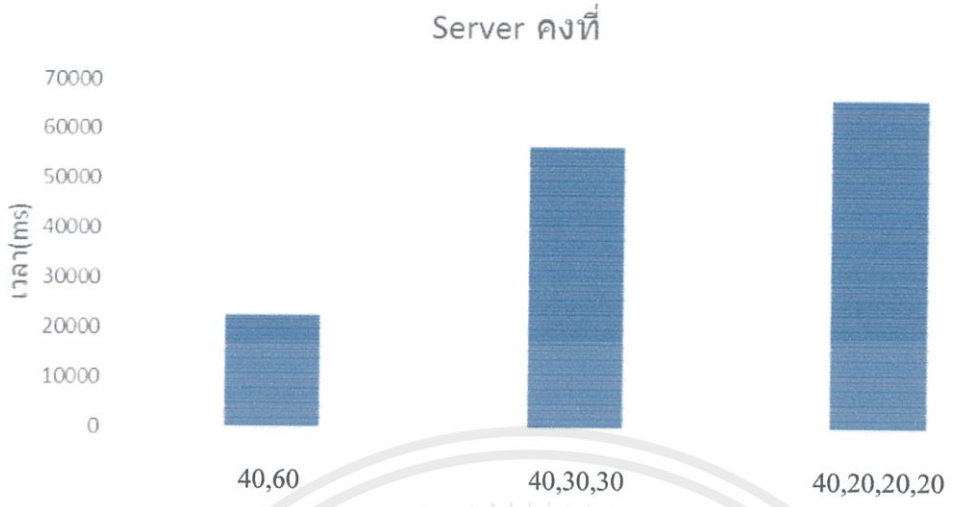
4.1.2.4 ผลการทดลอง

ตารางที่ 4.5 การทดลองเปลี่ยนสัดส่วนของข้อมูลและจำนวนเครื่องที่ใช้ในการส่งให้แต่ละเครื่องโดยคงสัดส่วนข้อมูลของ Serverไว้ ที่ใช้ 8 Thread และ Serial sort

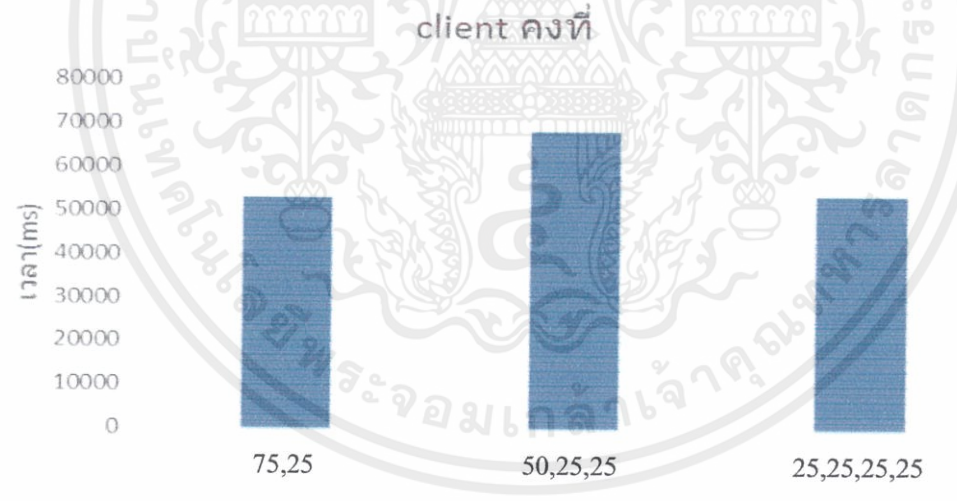
สัดส่วน	เวลา(ms)
40,60	53,009.4
40,30,30	68,177.4
40,20,20,20	53,439.8

ตารางที่ 4.6 การทดลองเปลี่ยนสัดส่วนของข้อมูลและจำนวนเครื่องที่ใช้ในการส่งให้แต่ละเครื่องโดยคงสัดส่วนข้อมูลของ Client ไว้ ที่ใช้ 8 Thread และ Serial sort

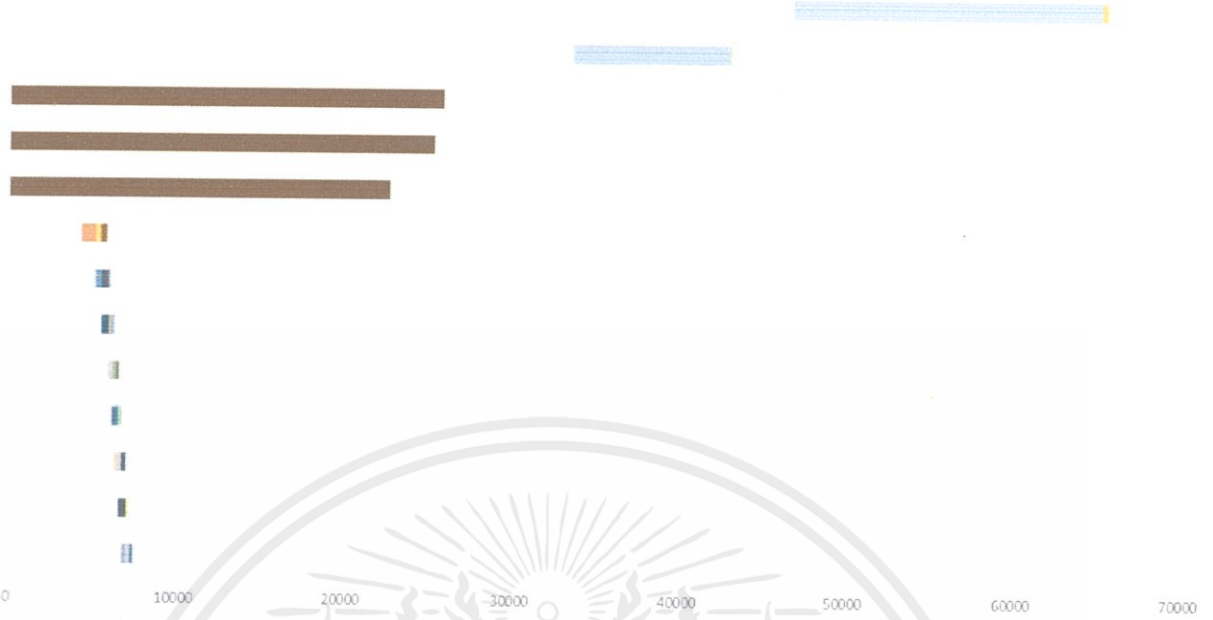
สัดส่วน	เวลา(ms)
75,25	22,512.0
50,25,25	56,702.6
25,25,25,25	66,132.2



รูปที่ 4.5 กราฟการทดลองเปลี่ยนสัดส่วนของข้อมูลและจำนวนเครื่องที่ใช้ในการส่งให้แต่ละเครื่องโดยคงสัดส่วนข้อมูลของ Server ไว้ ที่ใช้ 8 Thread และ Serial sort



รูปที่ 4.6 กราฟการทดลองเปลี่ยนสัดส่วนของข้อมูลและจำนวนเครื่องที่ใช้ในการส่งให้แต่ละเครื่องโดยคงสัดส่วนข้อมูลของ Client ไว้ ที่ใช้ 8 Thread และ Serial sort



รูปที่ 4.7 Log การทำงานของสัดส่วนที่แบ่งให้แต่ละเครื่องเป็น 25 เปอร์เซ็นต์เท่ากันหมด

จากการทดลองนี้ จะเห็นได้ว่าสำหรับการทดลองที่คงสัดส่วนของข้อมูลบน Server ไว้ และเพิ่มจำนวนเครื่องกับลดข้อมูลที่ส่งให้ Client แต่ละเครื่อง ในการใช้ 2 เครื่องกับ 4 เครื่องในการทำงานนั้นจะใช้เวลาใกล้เคียงกัน แต่การทำงานที่ใช้ 3 เครื่องจะใช้เวลาในการทำงานมากกว่า สาเหตุมาจาก เมื่อเทียบการทำงานระหว่าง การใช้ 4 เครื่อง กับ 3 เครื่องนั้น การใช้ 3 เครื่องจะใช้เวลาในการรอข้อมูลมากกว่า เพราะการทำงานที่ใช้ 4 เครื่องนั้น Server จะรอข้อมูล 20 เปอร์เซ็นต์เครื่องที่ 2 ในขณะที่เครื่องที่ 3 รอข้อมูลจำนวน 20 เปอร์เซ็นต์ในการทำงานไปพร้อมกัน ทำให้เสียเวลาน้อยลงในการทำงานทั้งหมด ตรงกันข้ามกันการทำงานที่ใช้ 3 เครื่อง เวลาที่ Server ใช้ในการรอข้อมูล จะมากกว่า เพราะหลังจากที่ Server จัดเรียงข้อมูลส่วนของตัวเองเสร็จจะไม่มีงานที่ต้องทำเพิ่มจนกว่าจะได้รับข้อมูลจะเครื่องที่ 2 ทำให้ใช้เวลาโดยรวมมากกว่าการทำงาน 4 เครื่อง

สำหรับการทดลองที่คงสัดส่วนของข้อมูลให้ Client แต่ละตัวไว้จะเห็นได้ว่า ในการเพิ่มจำนวนขึ้นจะใช้เวลามากขึ้น แต่แนวโน้มการเพิ่มของเวลาลดลง

4.2 การทดลองบนสมาร์ทโฟน

4.2.1 การทดลองบนสมาร์ทโฟน 1 เครื่อง

ในการทดลองบนสมาร์ทโฟน Android จะทดลองเพื่อดูประสิทธิภาพในการทำงาน

4.2.1.1 คุณสมบัติของสมาร์ทโฟนที่ใช้

เครื่องประสิทธิภาพสูง

Samsung Galaxy A8 : CPU Octa-core (4x1.8 GHz Cortex-A53 & 4x1.3 GHz Cortex-A53) หน่วยความจำ 2 GB

เครื่องประสิทธิภาพต่ำ

Sony Xperia SL : Dual-core 1.7 GHz Scorpion หน่วยความจำ 1 GB

4.2.1.2 ตัวแปรการทดลอง

ตัวแปรต้น : จำนวน Thread ที่ใช้(1, 2, 4, 8)

จำนวนข้อมูลที่นำมาจัดเรียง(1M, 2M, 4M, 5M, 6M ตัว)

ตัวแปรตาม : เวลาที่ใช้ในแต่ละเงื่อนไข

ตัวแปรควบคุม : ประสิทธิภาพของแต่ละสมาร์ทโฟน

4.1.2.3 วิธีการทดลอง

1. เริ่มต้นรันโปรแกรม
2. เริ่มจับเวลา
3. รอจนโปรแกรมทำงานเสร็จ
4. ตรวจสอบความถูกต้องของข้อมูล
5. ตรวจสอบเวลาที่ใช้ในการรันโปรแกรมในแต่ละเครื่อง
6. บันทึกผลการทดลอง
7. เพิ่มจำนวน Thread และทำซ้ำข้อ 1 – 6
8. เพิ่มขนาดของข้อมูลและทำซ้ำข้อ 1- 7

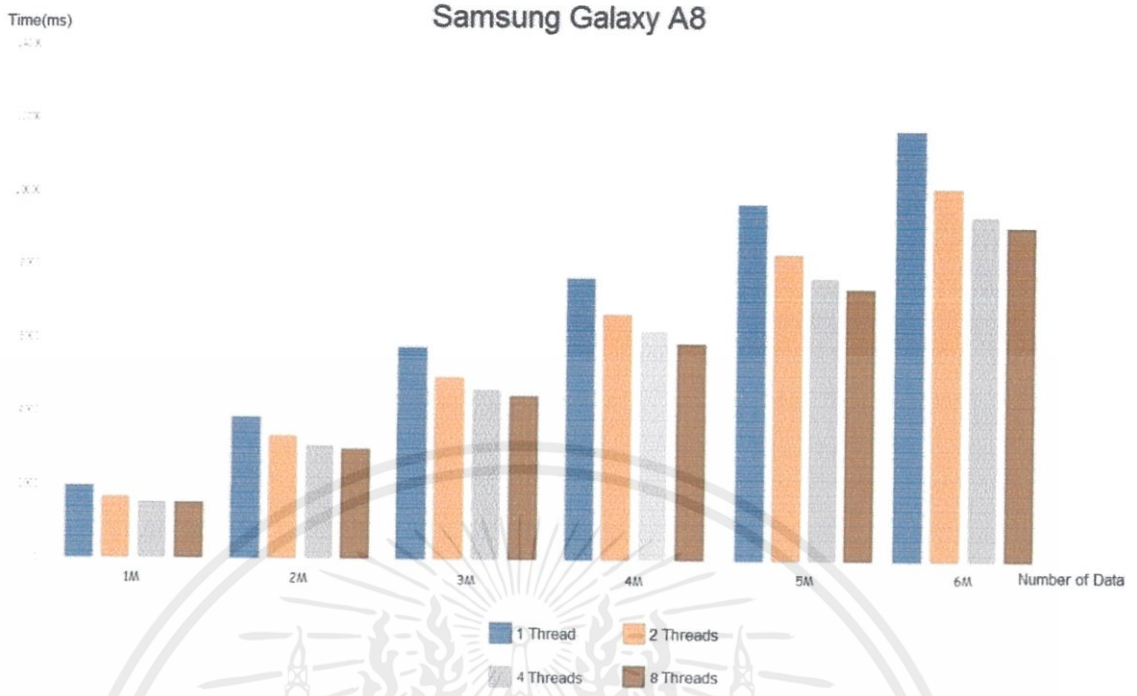
4.2.1.4 ผลการทดลอง

ตารางที่ 4.7 ตารางผลการทดลองบนสมาร์ทโฟน Android Samsung Galaxy A8

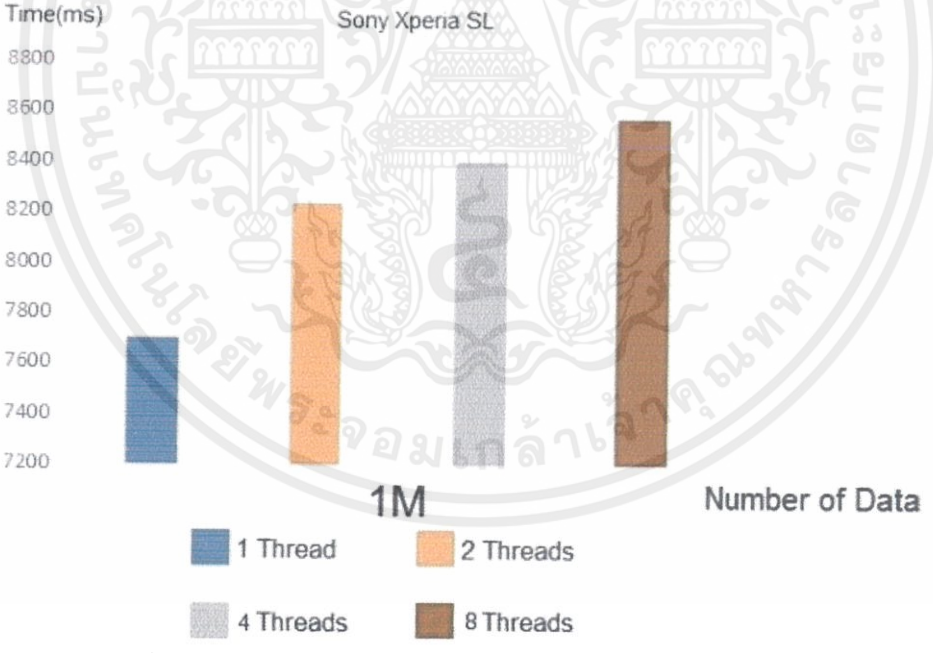
จำนวนข้อมูล (ตัว) / จำนวน Thread	1M	2M	3M	4M	5M	6M
1	1,970.2	3,886.0	5,805.0	7,713.0	9,744.0	11,734.8
2	1,698.4	3,359.0	4,984.2	6,717.8	8,376.0	10,174.4
4	1,572.8	3,108.2	4,663.6	6,280.4	7,723.4	9,425.6
8	1,533.2	3,024.8	4,488.2	5,926.0	7,423.4	9,115.2

ตารางที่ 4.8 ตารางผลการทดลองบนสมาร์ทโฟน Android Sony Xperia SL

จำนวนข้อมูล (ตัว) / จำนวน Thread	1M	2M	3M	4M	5M	6M
1	7,699.6	-	-	-	-	-
2	8,230.2	-	-	-	-	-
4	8,398.0	-	-	-	-	-
8	8,569.2	-	-	-	-	-



รูปที่ 4.8 ผลการทดลองบนสมาร์ทโฟน Android Samsung Galaxy A8



รูปที่ 4.9 ผลการทดลองบนสมาร์ทโฟน Android Sony Xperia SL

จากรูปที่ 4.8 จะพบว่า Samsung Galaxy A8 เมื่อเพิ่มจำนวนข้อมูลจะทำให้เวลาที่ใช้ในการทำงานเพิ่มขึ้นแต่ว่าในแต่ละขนาดข้อมูลหากเพิ่มจำนวน Thread จะทำให้เวลาที่ใช้นั้นลดลง ส่วนใน Sony Xperia SL ที่เป็นเครื่องที่มีประสิทธิภาพต่ำนั้นสามารถทำงานกับข้อมูลได้มากสุดเพียง 1M เท่านั้น และยิ่งเพิ่ม Thread ก็ใช้เวลามากตาม

4.2.2 การทดลองบนสมาร์ทโฟน 2 เครื่อง

ในการทดลองบนสมาร์ทโฟน Android จะทดลองเพื่อเพิ่มประสิทธิภาพให้เครื่องที่มีประสิทธิภาพต่ำโดยจะให้เครื่องที่มีประสิทธิภาพต่ำเป็น Server และแบ่งสัดส่วนให้ตนเองไม่เกิน 1M

4.2.2.1 คุณสมบัติของสมาร์ทโฟนที่ใช้

เครื่องประสิทธิภาพสูง

Samsung Galaxy A8 : CPU Octa-core (4x1.8 GHz Cortex-A53 & 4x1.3 GHz Cortex-A53) หน่วยความจำ 2 GB

เครื่องประสิทธิภาพต่ำ

Sony Xperia SL : Dual-core 1.7 GHz Scorpion หน่วยความจำ 1 GB

4.2.2.2 ตัวแปรการทดลอง

ตัวแปรต้น : จำนวนเครื่องที่เพิ่มเข้ามา

ตัวแปรตาม : เวลาที่ใช้ในแต่ละเงื่อนไข ,
ขนาดของข้อมูลที่ทำงานได้มากขึ้น

ตัวแปรควบคุม : ประสิทธิภาพของแต่ละสมาร์ทโฟน

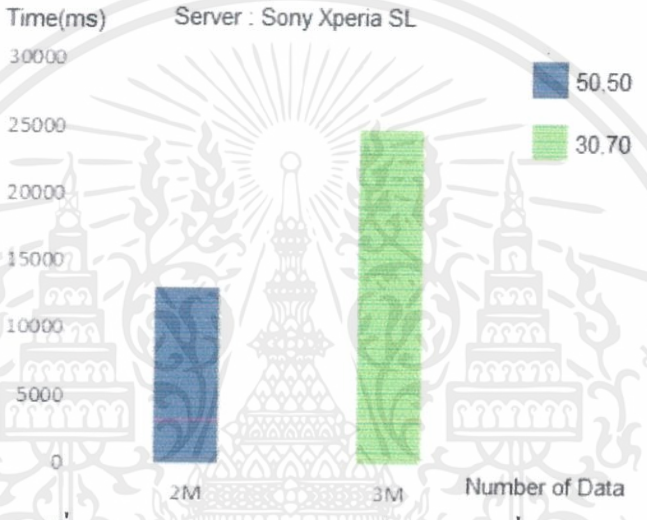
4.2.2.3 วิธีการทดลอง

1. เริ่มต้นรันโปรแกรม
2. เริ่มจับเวลาหลังจากเชื่อมต่อสำเร็จตามจำนวนเครื่องที่กำหนดไว้
3. รอจนโปรแกรมทำงานเสร็จ
4. ตรวจสอบความถูกต้องของข้อมูล
5. ตรวจสอบเวลาที่ใช้ในการรันโปรแกรม
6. บันทึกผลการทดลอง
7. เพิ่มขนาดของข้อมูลและทำซ้ำข้อ 1- 6

4.2.2.4 ผลการทดลอง

ตารางที่ 4.9 ตารางผลการทดลองบนสมาร์ทโฟน 2 เครื่อง

ขนาดของข้อมูล(ตัว) และ สัดส่วน(เปอร์เซ็น)	เวลาที่ใช้
2M (สัดส่วน 50,50)	13,080.8
3M (สัดส่วน 30,70)	24,806.4



รูปที่ 4.10 ผลการทดลองบนสมาร์ทโฟน 2 เครื่อง

จากรูปที่ 4.10 จะพบว่าเครื่องประสิทธิภาพต่ำหรือก็คือ Sony Xperia SL สามารถรับข้อมูลได้มากขึ้นจาก 1M เป็น 3M แต่เวลาที่ใช้ก็เพิ่มขึ้นมากตามไปด้วยเพราะมีการส่งข้อมูลผ่าน Network เพื่อให้อีกเครื่องช่วยในการประมวลผล

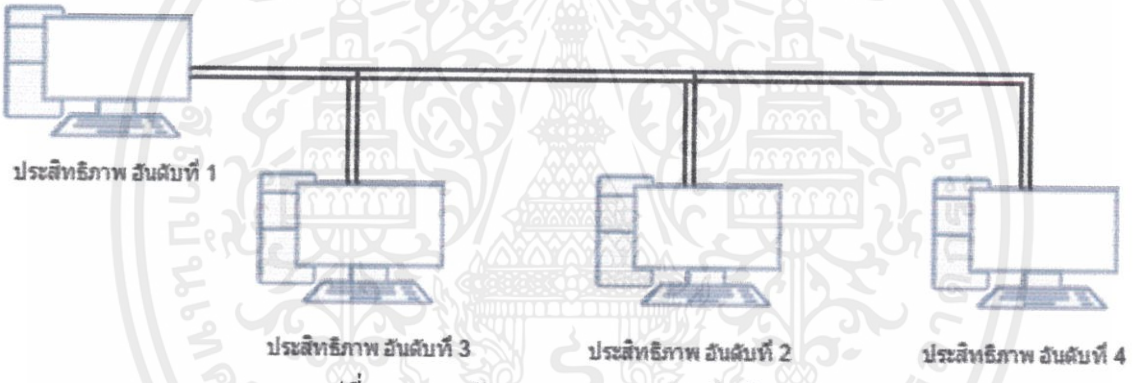
4.3 วิเคราะห์ผลการทดลอง

จากผลการทดลองจะสามารถวิเคราะห์ได้ดังนี้

4.3.1 การทดลองบน Java virtual machine

4.3.1.1 การทดลองแบ่งสัดส่วนของข้อมูล 2 รูปแบบ จากกราฟของ 1 Thread และ 2 Threads จะพบว่า สัดส่วนการแบ่งข้อมูลทั้ง 2 แบบให้ผลที่ใกล้เคียงกันมาก เพราะถึงเราจะแบ่งข้อมูลส่วนใหญ่ไปให้เครื่องที่มีประสิทธิมากมากกว่า แต่ตามหน้าที่ของแต่ละเครื่องนั้นจากการจัดเรียงประสิทธิภาพจะเป็นดังรูปที่ 4.11

จากรูป 4.11 เครื่องที่มีประสิทธิภาพอันดับที่ 2 มีภาระหน้าที่ในการรวมข้อมูลเพื่อส่งไปให้เครื่องที่เป็น Master ได้รับข้อมูล 20 เปอร์เซ็นต์ซึ่งลดภาระงานลงในระดับหนึ่ง แต่เครื่องที่มีประสิทธิภาพอันดับที่ 3 ได้รับข้อมูลเพิ่ม เป็น 30 เปอร์เซ็นต์ซึ่งส่วนนี้จะทำให้ใช้เวลามากขึ้น เลยทำให้ผลที่ได้ใกล้เคียงกัน



รูปที่ 4.11 การจัด Schedule ตามประสิทธิภาพ

4.3.1.2 การทดลองเพิ่ม Thread ที่ใช้

เมื่อนำกราฟทั้ง 2 กราฟมาเปรียบเทียบกันจะเห็นได้ว่าใช้เวลาในการดำเนินการใกล้เคียงกันมากสำหรับ 1 และ 2 Thread เพราะวิธีที่ใช้สำหรับจัดเรียงข้อมูลนั้นเป็น ParallelSort ของ Java ซึ่งใช้ Multithreading ในการ Sort ทำให้ใช้เวลามากสำหรับข้อมูลที่ลดลงจากการแบ่งข้อมูล ซึ่งต่อให้การแบ่ง Thread จะทำให้ลดจำนวนข้อมูลที่ต้องจัดเรียง แต่จะเสียเวลาเพิ่มเติมในการ Merge ของข้อมูลด้วย ทำให้ส่งผลตามที่ได้กล่าวมา

4.3.2 การทดลองบนสมาร์โฟน Android

จากตารางจะเห็นได้ว่าในการรันบนสมาร์ทโฟน Android นั้นยังใช้เวลามาก อาจเป็นเพราะประสิทธิภาพของสมาร์ทโฟนไม่สูงมากเมื่อเทียบกับคอมพิวเตอร์ตั้งโต๊ะที่ใช้รัน Java Virtual Machine แต่การเพิ่มจำนวน Thread ก็ส่งผลให้สมาร์ทโฟนรันได้เร็วขึ้น ซึ่งก็ขึ้นกับประสิทธิภาพของสมาร์ทโฟนที่ใช้ด้วย และการเพิ่มจำนวน Client ที่เข้ามาช่วยในการทำงานนั้นทำให้สมาร์ทโฟนสามารถรองรับข้อมูลได้มากขึ้น

บทที่ 5

สรุปผลการดำเนินงาน

5.1 สรุปการดำเนินงานทั้งหมด

จากการทดลองในบทที่ 4 เราจะสามารถสรุปผลได้ดังนี้

5.1.1 การทดลองบน Java virtual machine

5.1.1.1 การทดลองเพื่อเปรียบเทียบเวลาในการใช้ฟังก์ชั่น Arrays.sort() กับ Arrays.parallelSort() เพื่อจัดเรียงข้อมูลภายในเครื่อง

จะพบว่าการเพิ่มจำนวน Threads ในแต่ละเครื่องจะทำให้เวลาในการใช้ฟังก์ชั่น Arrays.sort() มีประสิทธิภาพมากกว่า

5.1.1.2 การทดลองเพื่อเปรียบเทียบเวลาสำหรับรูปแบบการแบ่งสัดส่วนให้แต่ละเครื่องที่ต่างกัน

จะเห็นได้ว่า การทดลองที่ใช้จำนวนเครื่องที่เป็นเลขคี่ มีแนวโน้มใช้เวลามากกว่าจำนวนเครื่องที่เป็นเลขคู่

5.1.2 สรุปผลการทดลองบนสมาร์โฟน Android

การทดลองบนสมาร์โฟน Android จะพบว่าสามารถรองรับข้อมูลได้น้อยและใช้เวลามากเมื่อเทียบกับคอมพิวเตอร์ตั้งโต๊ะ แต่ผลการทดลองในเครื่องเดียวมีแนวโน้มที่ดีขึ้นเมื่อเพิ่มจำนวน Thread และสามารถรองรับข้อมูลได้มากขึ้นเมื่อเพิ่มจำนวน Client ที่เข้ามาช่วยในการทำงาน

5.2 แนวทางการพัฒนาต่อ

ในการพัฒนาโปรแกรมของเรานั้น เราได้มองไปยังการวิธีการจัดเรียงข้อมูล การจัด Schedule มากกว่าส่วนอื่น ซึ่งยังมีส่วนที่เรายังไม่ได้พัฒนา เช่น Network ในการส่งข้อมูล ซึ่งถ้าสามารถพัฒนาวิธีการในการส่งข้อมูลระหว่างเครื่องได้ ก็จะทำให้โปรแกรมพัฒนาไปอีกระดับ

นอกจากนั้น โปรแกรมของเรายังทำได้เพียงจัดเรียงข้อมูลที่เป็นตัวเลข 32-bits เท่านั้น แต่ในอนาคตสามารถจัดเรียงข้อมูลได้หลากหลายขึ้น ผ่านวิธีการอ่านไฟล์ที่เหมาะกับข้อมูลที่เราจะนำมาใช้เพิ่มเติม

บรรณานุกรม

Jakob Jenkov. java concurrency and multithread (ออนไลน์) แหล่งที่มา :

http://tutorials.jenkov.com/java-concurrency/index.html

John Bell. multithread (ออนไลน์) แหล่งที่มา :

https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/4_Threads.html

wikipedia. java (ออนไลน์) แหล่งที่มา :

https://en.wikipedia.org/wiki/Java_(programming_language)

wikipedia. deadlock (ออนไลน์) แหล่งที่มา :

https://en.wikipedia.org/wiki/Deadlock

wikipedia. OOP (ออนไลน์) แหล่งที่มา :

https://en.wikipedia.org/wiki/Object-oriented_programming

wikipedia. OOP (ออนไลน์) แหล่งที่มา :

https://en.wikipedia.org/wiki/Object-oriented_programming

การจัดเรียงและการค้นหาข้อมูล (ออนไลน์) แหล่งที่มา :

https://goo.gl/xPhtPp

wikipedia. Virtual Box (ออนไลน์) แหล่งที่มา :

https://en.wikipedia.org/wiki/VirtualBox

Ben Jakuben. Genymotion (ออนไลน์) แหล่งที่มา :

https://teamtreehouse.com/library/android-tools/getting-started-with-genymotion/what-is-genymotion

wikipedia. Android (ออนไลน์) แหล่งที่มา :

https://en.wikipedia.org/wiki/Android_(operating_system)

Android studio (ออนไลน์) แหล่งที่มา :

https://developer.android.com/studio/index.html