การค้นหาเบสเป้าหมายในลำดับของดีเอ็นเอ โดยวิธีการเขียนโปรแกรมแบบหลายโปรเซสพร้อมกันด้วยภาษาเออแลง

Searching for Markers in DNA Sequences with Multiprocessing using Erlang

วีรศักดิ์ ช่องงูเหลือม, นิตยา เกิดประสพ, กิตติศักดิ์ เกิดประสพ
สาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี
111 ถนนมหาวิทยาลัย ตำบลสุรนารี อำเภอเมือง จังหวัดนครราชสีมา 30000
E-mail:singpor@gmail.com, nittaya@sut.ac.th, kerdpras@sut.ac.th

บทคัดย่อ

ในงานวิจัยทางด้านชีวสารสนเทศ ต้องอาศัยอัลกอริทึมในการที่จะทำความเข้าใจในขบวนการต่างๆทางด้านชีววิทยา โดยอาศัยการคำนวณทางด้านสถิติ และ เทคนิคการค้นหาข้อมูลต่างๆเข้าช่วย โดยวิธีเหล่านี้จะมีข้อมูลที่ต้องใช้วิเคราะห์ปริมาณมากและมีขั้นตอนในการประมูลผลซับซ้อนสูงดังนั้นจึงใช้คอมพิวเตอร์เข้ามาช่วยจัดการ ในงานวิจัยเรื่องนี้จะใช้เทคนิคการเขียนโปรแกรมแบบพร้อมกัน โดยใช้ภาษาเออแลง ซึ่งออกแบบมาให้ง่ายต่อการเขียนโปรแกรมแบบหลายๆโปรเซสพร้อมกัน และ มีลักษณะเป็นภาษาเชิงฟังก์ชัน เพื่อจัดการกับข้อมูลทางด้านชีวสารสนเทศ คือ ข้อมูลของลำดับเบสของดีเอ็นเอ โดยที่โปรแกรมจะทำการค้นหาว่าลำดับเบสที่กำหนด ปรากฏอยู่ในตำแหน่งใดบ้างของลำดับดีเอ็นเอทั้งหมด โดยเขียนโปรแกรมขึ้น 2 แบบเปรียบเทียบกัน แบบแรกใช้เพียงโปรเซสเดียวและค้นหาเรียงลำดับตั้งแต่ต้นจนหมดลำดับ อีกแบบใช้วิธีการแบ่งสตริงออกเป็นส่วนๆโดยอาศัยเทคนิคแบ่งให้มีการซ้อนทับกันอย่างละครึ่งทำให้ผลการค้นหาไม่ขาดตรงส่วนรอยต่อที่แบ่ง แล้วจึงส่งไปค้นหาแบบหลายโปรเซสพร้อมกัน และเปรียบเทียบการทำงานของทั้งสองแบบ

คำสำคัญ: ลำดับดีเอ็นเอ, การโปรแกรมแบบหลายโปรเซส, ภาษาเออแลง

Abstract

Research in bioinformatics needs statistical computing and searching algorithms to comprehend biological processes. These algorithms and techniques have to deal with huge amount of data and complex processes. Therefore the use of computer and advanced programming technique implemented with Erlang, which is a functional language that supports multiprocessing computation. We propose algorithms as well as the source code to search for specific markers in the DNA sequences. The uniprocessing and multiprocessing styles are implemented and compared. The comparative results show the reduction in search time when increase the number of processes.

Keywords: DNA sequence, multiprocessing, Erlang

1. คำนำ

ชีวสารสนเทศศาสตร์(Bioinformatics) หรือ ชีววิทยาเชิงคำนวณ(Computational Biology) เป็นสาขาที่ใช้ความรู้จากคณิตศาสตร์ประยุกต์ สถิติศาสตร์, สารสนเทศศาสตร์, และวิทยาการคอมพิวเตอร์เพื่อแก้ปัญหาทางชีววิทยา [1]

ความเข้าใจในเรื่องเกี่ยวกับโครงสร้างโมเลกุลของดีเอ็นเอและบทบาทที่มีผลต่อชีวิตของเรา ช่วยให้รู้ความเชื่อมโยงระหว่างโรคและข้อบกพร่องต่างๆที่เกิดจากพันธุกรรมและเป็นประโยชน์ในหลายๆด้านของการดำรงชีวิตของสิ่งมีชีวิตทั้งหลาย

การศึกษาทางด้านชีวสารสนเทศ จะเกี่ยวข้องกับข้อมูลทางพันธุศาสตร์และอณูชีววิทยา เช่น ข้อมูลรหัสพันธุกรรม, ข้อมูลลำดับรหัสโปรตีน, ปริมาณชีวโมเลกุล (ระดับการแสดงออกของยีนต่างๆ) แต่ละชนิด (mRNA และโปรตีน) และข้อมูลหมายเหตุ (annotation data) โดยผู้ศึกษาวิจัย จะนำข้อมูลเหล่านี้ ไปใช้ในงานอย่างเช่น การจัดเรียงลำดับรหัสโปรตีน การจัดโครงสร้างโปรตีน การทำนายโครงสร้างโปรตีน การค้นหายีน หรือ การสร้างหุ่นจำลองของวิวัฒนาการ เนื่องจากข้อมูลที่ใช้มีจำนวนมาก ทำให้ใช้พื้นที่จัดเก็บมากตามไปด้วย และบ่อยครั้งที่ข้อมูลมีความซับซ้อน ทำให้ต้องใช้การประมวลผลมากขึ้นเช่นกัน การศึกษาด้านชีวสารสนเทศ จึงเป็นการศึกษาวิธีการจัดเก็บสืบค้น และประมวลผลข้อมูลเหล่านี้ ให้เป็นไปได้โดยสะดวก และมีประสิทธิภาพ

โปรแกรม Basic Local Alignment Search Tool [2] หรือเรียกชื่อย่อว่า BLAST เป็นเครื่องมือที่ออกแบบมาสำหรับเปรียบเทียบและค้นหาลำดับรหัสพันธุกรรมบางส่วนในฐานข้อมูลของรหัสพันธุกรรมทั้งหมด เป็นเครื่องมือหนึ่งที่ใช้กันอย่างกว้างขวางซึ่งใช้อัลกอริทึมที่มีความซับซ้อนและต้องการหน่วยความจำและเวลาในการประมวลผลอย่างมาก

ในงานวิจัยชิ้นนี้จะสร้างเครื่องมือขึ้นมาเพื่อค้นหาลำดับของรหัสพันธุกรรมบางส่วนว่าอยู่ในตำแหน่งใดบ้างของข้อมูลรหัสพันธุกรรมทั้งหมด โดยที่จะอาศัยการประมวลผลแบบหลายๆโปรเซส (multiprocessing) พร้อมกัน ด้วยภาษาเออแลง (Erlang) ซึ่งจะให้ประสิทธิภาพดีกว่าการค้นหาเพียงโปรเซสเดียวเป็นลำดับๆไป และด้วยความสามารถของภาษาเออแลง ทำให้เราสามารถออกแบบโปรแกรมให้หลายๆโปรเซสทำงานร่วมกันได้อย่างง่ายดาย มีความสามารถแบบภาษาเชิงฟังก์ชัน ทำให้ไม่ต้องกังวลในเรื่องการใช้ตัวแปรร่วมกันระหว่างหลายๆโปรเซส ไม่ต้องกังวลเรื่องการล็อกค่าตัวแปรต่างๆเหมือนกับภาษาอื่นๆ

2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ลำดับของดีเอ็นเอ (DNA Sequences)

ดีเอ็น (DNA) เป็นชื่อย่อของสารพันธุกรรม มีชื่อแบบเต็มๆว่า กรดดีออกซีไรโบนิวคลีอิก (Deoxyribonucleic Acid) ที่พบในเซลล์ของสิ่งมีชีวิตทุกชนิด[3] ดีเอ็นเอ มีรูปร่างเป็นเกลียวคู่ คล้ายบันไดเวียนขวา ขาหรือราวของบันไดแต่ละข้างก็คือการเรียงตัวของนิวคลีโอไทด์ (Nucleotide) นิวคลีโอไทด์เป็นโมเลกุลที่ประกอบด้วยน้ำตาล (Deoxyribose Sugar), ฟอสเฟต (Phosphate) (ซึ่งประกอบด้วยฟอสฟอรัสและออกซิเจน) และไนโตรจีนัสเบส (Nitrogenous Base) เบสในนิวคลีโอไทด์มีอยู่สี่ชนิดด้วยกัน ได้แก่ อะดีนีน (adenine, A), ไทมีน (thymine, T), ไซโทซีน (cytosine, C) และ กัวนีน (guanine, G) ขาหรือราวของบันไดสองข้างหรือนิวคลีโอไทด์ถูกเชื่อมกันด้วยเบส โดยที่ A จะเชื่อมกับ T ด้วยพันธะไฮโดรเจนแบบพันธะคู่ หรือ double bonds และ C จะเชื่อมกับ G ด้วยพันธะไฮโดรเจนแบบ พันธะสามหรือ triple bonds ข้อมูลทางพันธุกรรมของสิ่งมีชีวิตต่างๆ ก็เกิดขึ้นจากการเรียงลำดับของเบส (A,T,C,G) ในดีเอ็นเอนั่นเอง

2.2 ภาษา Erlang

Erlang คือภาษาคอมพิวเตอร์ซึ่งมีหลายๆความสามารถ [4] โดยส่วนใหญ่จะมีความเกี่ยวข้องกับระบบปฏิบัติการมากกว่าภาษาโปรแกรมอื่นๆเช่นความสามารถเกี่ยวกับ การประมวลผลแบบหลายๆงานพร้อมๆกัน, การจัดตารางเวลา, การจัดการหน่วยความจำ, การประมวลผลแบบกระจาย, ระบบเครือข่าย, ฯลฯ คุณลักษณะสำคัญของภาษาเออแลงมีรายละเอียดดังต่อไปนี้

2.2.1 Concurrency

เออแลงมีโปรเซสที่เบามาก ความต้องการหน่วยความจำของแต่ละโปรเซสมีความเป็นพลวัต แต่ละโปรเซสไม่มีการใช้หน่วยความจำร่วมกัน และ การติดต่อสื่อสารของแต่ละโปรเซสใช้การส่งข้อความแบบไม่เป็นจังหวะเดียวกัน(Asynchronous) เออแลงรองรับการสร้างโปรแกรมที่มีจำนวนโปรเซสมากๆ โดยไม่จำเป็นต้องอาศัยกลไกการสร้างโปรเซสของระบบปฏิบัติการ

2.2.2 Distribution

เออแลงถูกออกแบบมาเพื่อให้ทำงานกับสภาพแวดล้อมแบบกระจายซึ่งการทำงานของเออแลงเวอชวลแมชชีนจะถือว่าเป็นเออแลงโหนดหนึ่งโหนด ระบบการกระจายของเออแลงคือเครือค่ายของโหนดแต่ละโหนด โดยแต่ละโหนดสามารถสร้างโปรเซสแบบขนานให้ทำงานบนโหนดอื่นๆได้

2.2.3 Robustness

เออแลงมีการตรวจจับข้อผิดพลาดพื้นฐานหลายๆแบบ ซึ่งถูกนำมาใช้ในโครงสร้างระบบ fault-tolerent ตัวอย่างเช่น โปรเซสใดๆสามารถตรวจสอบสถานะของโปรเซสอื่นๆได้ แม้ว่าจะทำงานกันอยู่คนละโหนด โปรเซสที่อยู่ในระบบแบบกระจายสามารถกำหนดค่าให้โหนดอื่นๆเริ่มทำงานที่โหนดสำรองได้กรณีที่เกิดข้อผิดพลาดขึ้น

2.2.4 Soft Real-Time

เออแลงรองรับการทำงานแบบ soft real-time ซึ่งต้องการเวลาในการตอบสนองในระดับ มิลลิวินาที

2.2.5 Hot code upgrade

ในหลายๆระบบ ไม่สามารถจะหยุดการทำงานเพื่อทำการปรับปรุงระบบได้ แต่เออแลงอนุญาตให้แก้ไขโค้ดโปรแกรมได้ขนาดที่โปรแกรมกำลังทำงาน

2.2.6 Incremental code loading

ผู้ใช้งานสามารถควบคุมรายละเอียดของโค้ดที่ถูกโหลดได้ ในระบบแบบฝังตัว โค้ดทั้งหมดจะถูกโหลดตอนบูตระบบ ในระบบที่กำลังพัฒนา โค้ดจะถูกโหลดเมื่อจำเป็นต้องใช้งาน

2.2.7 External interfaces

เออแลงโปรเซสสามารถติดต่อกับโปรเซสภายนอกได้เหมือนกับที่ติดต่อกับเออแลงโปรเซสด้วยกันเอง กลไกนี้ใช้เพื่อติดต่อกับระบบปฏิบัติการและโปรแกรมที่สร้างจากภาษาอื่น

2.3 ประวัติภาษา Erlang

เออแลงถูกสร้างขึ้นครั้งแรกเมื่อปี ค.ศ. 1986 ที่ห้องปฏิบัติการคอมพิวเตอร์ของบริษัทอิริกสัน[5] โดยถูกออกแบบมาเพื่อใช้เป็นภาษาที่ดีสำหรับการเขียนโปรแกรมทางด้านระบบโทรศัพท์ ต่อมาเมื่ออิริกสันใช้งานภาษาเออแลงในงานต่างๆประสบความสำเร็จเป็นอย่างมาก โดยก่อนหน้านี้ ลิขสิทธิ์ของเออแลงยังไม่เป็นโอเพนซอร์สจนกระทั่ง 2 ธันวาคม ค.ศ. 1998, ก็ได้มีการทำให้ลิขสิทธิ์ของเออแลงเป็นโอเพนซอร์ส ทำให้มีการร่วมกันพัฒนาเออแลงของนักพัฒนาอย่างกว้างขวางขึ้นจนมีผู้ใช้และโปรแกรมที่สร้างจากเออแลงอยู่มากในบริษัทชั้นนำในปัจจุบัน

2.4 Multiprocessing with Erlang

การเขียนโปรแกรมด้วย Erlang จะเขียนในลักษณะเชิงฟังก์ชัน (Functional Programming) โดยจะแยกกลุ่มฟังก์ชันออกเป็นโมดูล (Module) และโมดูลของ Erlang จะถูกเรียกให้ทำงานผ่านทาง Virtual Machine ของ Erlang

ใน Erlang นั้นจะมีฟังก์ชันที่เอาไว้สำหรับสร้าง process ย่อยๆ ให้ทำงานเป็นลักษณะ multiprocessing ได้ คือฟังก์ชันที่ชื่อว่า spawn(Module, Function ,Arguments) โดยจะรับค่าสามค่าคือ ชื่อโมดูลของฟังก์ชันที่จะเรียก ชื่อฟังก์ชันที่จะเรียก และ อาร์กิวเมนท์ ที่ต้องการส่งให้ฟังก์ชัน หลังจากเรียกใช้ spawn แล้ว Erlang จะสร้างโปรเซสย่อยที่มีหมายเลขเฉพาะเรียกว่า Process identifier หรือ PID ขึ้นมา และจะแยกโปรเซสการทำงานให้ทำงานฟังก์ชันนี้

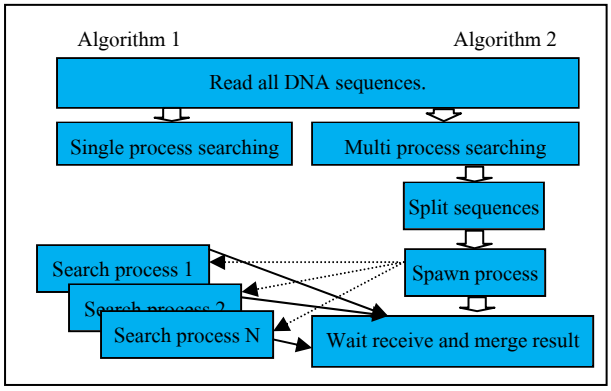
ในแต่ละโปรเซส เราสามารถส่งค่า (message) ไปให้กับโปรเซสอื่นได้โดยอาศัยค่า PID ของโปรเซสนั้นแล้วใช้โอเปอเรเตอร์ ! ตามด้วยข้อมูลที่ต้องการส่ง เช่น PID ! {msg,"hello"}; เพื่อส่งข้อความ "hello" ตามตัวอย่างในรูปที่ 1

```
-module(echo).
-export(main/0).
%% Echo Message
main() -> PID = spawn(?MODULE,process_loop,[]),
          PID ! {msg,"hello"},
          PID ! exit.
process_loop() -> receive
                    {msg,Msg} -> io:format("~w~n",[Msg]),
                                 process_loop();
                    exit -> ok
                  end.
```

รูปที่ 1 ตัวอย่างการสร้างโปรเซสแล้วส่งเมสเซส และการใช้ receive เพื่อรอรับ message

3. ระเบียบวิธีวิจัย

ในงานวิจัยชิ้นนี้จะทำการสร้างโปรแกรมด้วยภาษา Erlang เพื่อค้นหาเบสเป้าหมายในลำดับของดีเอ็นเอ ด้วยเทคนิคการค้นหาในสองรูปแบบคือ ค้นหาแบบเรียงลำดับโดยใช้เพียง process เดียวและค้นหาแบบพร้อมกันด้วยโปรเซสย่อยหลายโปรเซส เพื่อทำการเปรียบเทียบเวลาที่ใช้ในการค้นหา โดยต้นแบบที่สร้างขึ้นจะประกอบด้วยส่วนรับข้อมูล ซึ่งจะแยกออกเป็นอีกหนึ่งโมดูล และส่วนที่ทำหน้าที่ค้นหา โดยที่ส่วนที่ทำหน้าที่ค้นหาของการค้นหาแบบหลายๆ โปรเซสพร้อมกันนั้นจะมีส่วนที่แบ่งข้อมูลออกเป็นส่วนๆเพื่อเตรียมสำหรับส่งไปให้แต่ละโปรเซสค้นหาก่อนด้วย โครงสร้างหลักเป็นดังรูปที่ 2



รูปที่ 2 โครงสร้างหลักของโปรแกรม

ข้อมูลของลำดับ DNA ที่นำมาใช้เป็นการสุ่มขึ้นมาโดยอาศัยเว็บของ http://www.bioinformatics.org/sms2/random_dna.html เป็นเครื่องมือช่วยในการสุ่มข้อมูล ในการทดลองนี้จะใช้ข้อมูลของลำดับ DNA ทั้งหมด 100,000,000 ลำดับเบส

3.1 ค้นหาแบบเรียงลำดับด้วย process เดียว

ในการค้นหาลำดับของเบสในลำดับของดีเอ็นเอทั้งหมดที่ต้องการ โดยวิธีแบบเรียงลำดับ ที่อาศัยการทำงานแค่เพียงโปรเซสเดียวนั้น มีขั้นตอนแสดงได้ดังต่อไปนี้

Algorithm 1 Searching with single process

Input : ไฟล์ที่เก็บ ลำดับของ dna ทั้งหมด (dnaseq.txt) และ ลำดับที่ต้องการจะค้น

Output : เซตของตำแหน่งที่ค้นเจอ

Process :

1. อ่านลำดับดีเอ็นเอที่เก็บอยู่ในไฟล์เก็บในตัวแปร DNA
2. ทำการค้นหาลำดับ S ในตัวแปร DNA
   2.1. ดึงค่า N ตัวแรกของ DNA โดย N คือขนาดความยาวของ S
   2.2. นำค่าที่ได้มาเปรียบเทียบกับ S ถ้าตรงกัน ให้เก็บตำแหน่งที่ค้นเจอของ DNA ไว้ในตัวแปรลิสต์ R แล้วทำการค้นหาอีกครั้งโดยตัด DNA ออกไป N ตัว ถ้าไม่ตรงกัน ค้นหาต่อโดยตัด DNA ออกหนึ่งตัว
   2.3. ทำจนลิสต์ DNA กลายเป็นลิสต์ว่างแล้วส่งผลลัพธ์ที่ลิสต์ R ออกไป

3.2 ค้นหาแบบแยกเป็นหลายๆโปรเซส

ในการค้นหาลำดับของเบสในลำดับของดีเอ็นเอทั้งหมดที่ต้องการ ที่อาศัยการทำงานแบบหลายๆโปรเซส มีขั้นตอนแสดงได้ดังต่อไปนี้

Algorithm 2 Searching with multiprocessing

Input : ไฟล์ที่เก็บ ลำดับของ dna ทั้งหมด (dnaseq.txt) และ ลำดับที่ต้องการจะค้น

Output : เซตของตำแหน่งที่ค้นเจอ

Process :

1. อ่านลำดับดีเอ็นเอที่เก็บอยู่ในไฟล์เก็บในตัวแปร DNA
2. ทำการแบ่ง DNA เป็นลิสต์ย่อยๆแล้วเก็บไว้ในตัวแปร DNAList
3. แบ่งโปรเซสในการค้นหา โดยส่งลำดับที่ต้องการค้นหา S และลำดับย่อยๆใน DNAList ที่จะถูกค้นเป็น input ให้กับแต่ละ process
4. รอรับผลลัพธ์จากการทำงานของแต่ละโปรเซส
   4.1. นำผลลัพธ์ที่ได้เก็บในตัวแปร R
   4.2. รอรับอีกครั้งจนได้ผลครบทุกโปรเซส
5. นำผลลัพธ์ที่ได้แปลงเป็นข้อมูลแบบเซตเพื่อขจัดผลลัพธ์ที่ซ้ำกันของแต่ละโปรเซส
6. แปลงกลับเป็นลิสต์

3.3 วิธีการแบ่งลำดับดีเอ็นเอเพื่อส่งให้แต่ละโปรเซส

ในการแบ่งลำดับดีเอ็นเอออกเป็นส่วนย่อยๆนั้น ถ้าเมื่อเราแบ่งลำดับออกปกติ จะทำให้ถ้ามีผลการค้นที่ถูกต้องอยู่ตรงรอยเชื่อมต่อขาดหายไป ดังนั้นจึงต้องใช้วิธีการแบ่งโดยให้มีส่วนเหลื่อมล้ำกัน ในที่นี้ให้มีส่วนซ้ำกันครึ่งหนึ่ง วิธีการแบ่งแสดงได้ดัง อัลกอริทึมต่อไปนี้

Algorithm 3 Split sequences

Input : ลิสต์ของลำดับทั้งหมด และ ขนาดที่จะแบ่ง

Output : ลิสต์ของลำดับที่แบ่งแล้วตามขนาดที่ต้องการ

1. ดึง N ตัวแรกของลิสต์ DNA ออกมาโดย N คือขนาดที่ต้องการ
2. เก็บลำดับย่อยที่ได้ใน R
3. ตัดทิ้งลำดับ DNA ออกแค่เพียง (N/2)
4. เรียกซ้ำ ขั้นตอนแรกอีกครั้ง จนกว่าจะหมดลำดับใน DNA

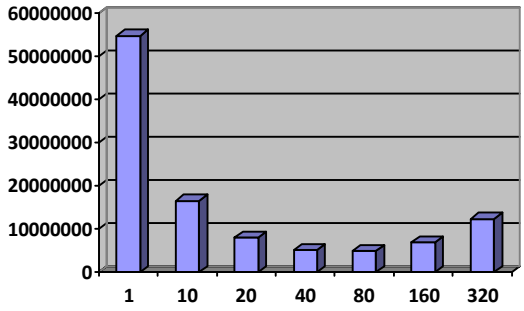
4. การทดสอบประสิทธิภาพ

การทดสอบประสิทธิภาพของการทำงานของทั้ง 2 อัลกอริทึม จะใช้ฟังก์ชัน tc ที่อยู่ในโมดูล timer ของ erlang เป็นตัวจับเวลา โดยผลลัพธ์จะได้เวลาในหน่วย ไมโครวินาที ข้อมูลของลำดับเบสที่นำมาทดสอบจะเป็นลำดับที่สุ่มมาโดยจะนำมาทั้งสิ้นจำนวน 100,000,000 ลำดับเบส ลำดับที่นำมาทดสอบในการค้นหาได้แก่ “att” โดยจะทำการเปรียบเทียบแบบเรียงลำดับที่ใช้แค่โปรเซสเดียว กับ การแบ่งโปรเซสเป็นจำนวน 10, 20, 40, 80, 160, 320 โปรเซส ผลการจับเวลาเป็นไปตามตารางในรูปที่ 3

จำนวนโปรเซส	เวลาที่ใช้ค้นหา (microsecond)	จำนวนที่พบ
1	54,687,522	1,615,860
10	16,558,977	1,615,860
20	8,049,573	1,615,860
40	5,118,960	1,615,860
80	4,989,144	1,615,860
160	6,965,558	1,615,860
320	12,342,060	1,615,860

รูปที่ 3 ตารางผลการใช้เวลาของแต่ละขนาดของโปรเซส

และเมื่อนำผลจากตารางมาสร้างกราฟเปรียบเทียบจะได้กราฟตามรูปที่ 4



รูปที่ 4 กราฟแสดงการเปรียบเทียบการใช้เวลา (แกนตั้ง) ตามจำนวนโปรเซส (แกนนอน)

จากผลการทำงานของโปรแกรมเมื่อเทียบจำนวนที่ค้นหาพบของโปรแกรมแบบลำดับที่มีโปรเซสเดียวกับหลายๆโปรเซสแล้วนั้น ได้จำนวนที่ค้นหาได้เท่าๆกัน และเมื่อเทียบเวลาที่ใช้ในการค้นหาซึ่งเวลาที่จับของการทำงานแบบหลายโปรเซสนั้น ได้รวมขั้นตอนการแบ่งลำดับดีเอ็นเอเอาไว้แล้ว เมื่อเรากระจายโปรเซสไปหลายๆโปรเซสผลการทำงานของโปรแกรมจะทำงานได้เร็วขึ้นกว่าเดิมมาก จากจำนวนที่ทดลองที่ 10,20,40,80 โปรเซส และเมื่อทำการเพิ่มจำนวนของโปรเซสไปจนถึง 160, 320 แล้วนั้น เวลากลับเพิ่มขึ้นมา เนื่องจากว่าในขั้นตอนการทำงานของโปรแกรมก่อนจะแยกโปรเซส เราได้ทำการแบ่งสตริงออกเป็นส่วนๆหลายๆส่วน ถ้ายิ่งโปรเซสมากขึ้น ก็จะเสียเวลาตรงส่วนนี้ไปด้วย ในการเพิ่มจำนวนโปรเซสที่ละ 2 เท่าเช่นในช่วง 10,20,40,80 ตามการคาดคะเนเวลาควรจะลดลงเป็น 1/x โดยที่ x เป็นจำนวนโปรเซส แต่เนื่องจากมีขั้นตอนการแบ่งจำนวนลำดับเพิ่มเข้ามาจึงทำให้มีการเบี่ยงเบนจากที่คาดคะเน

5. สรุป

การนำความสามารถของการทำงานแบบหลายๆโปรเซสพร้อมๆกันมาช่วยในการสร้างโปรแกรมเพื่อใช้ในการแก้ปัญหาทางด้านชีวสารสนเทศ ช่วยทำให้โปรแกรมทำงานได้อย่างมีประสิทธิภาพมากขึ้น โดยอาศัยความสามารถของภาษาเออแลง ทำให้สร้างโปรแกรมในลักษณะนี้ได้อย่างง่ายๆโดยไม่ต้องใช้ library เพิ่มเติมแล้วยังไม่ต้องคอยกังวลเรื่องการเข้าถึงทรัพยากร หรือการใช้ตัวแปร ร่วมกันของหลายๆโปรเซสเพราะการทำงานของ เออแลง เป็น functional programming ที่การสร้างตัวแปรนั้นกำหนดค่าได้แค่ครั้งเดียว การส่งค่าให้กับโปรเซสต่างๆเป็นไปในลักษณะก๊อปปี้ค่า ไม่ใช่การอ้างอิงตัวแปรเดียวกัน เรายังสามารถนำเอาความสามารถด้านการประมวลผลแบบกระจายของภาษาเออแลง เพื่อกระจายโปรเซสการทำงานไปในหลายๆเครื่องได้ ยังมีอีกหลายๆปัญหาทางด้านชีวสารสนเทศที่เหมาะสมกับการประมวลผลแบบหลายโปรเซสพร้อมกัน

จากการนำเอาแนวคิดเรื่องการสร้างโปรแกรมให้ทำงานได้หลายๆโปรเซสพร้อมๆกัน แล้วนำผลลัพธ์ที่ได้มารวมกัน ทำให้เราออกแบบโปรแกรมที่ช่วยในการค้นหาลำดับเบสของดีเอ็นเอ ในข้อมูลดีเอ็นเอทั้งหมด ที่เราต้องการได้เร็วกว่าการค้นหาแบบเรียงลำดับไปโดยใช้เพียงโปรเซสเดียวในการทำงาน จากผลการทดสอบพบว่าเมื่อเพิ่มจำนวนโปรเซสถึง 80 โปรเซสแล้วค่าของเวลาที่ลดลงจากการทำงานแค่โปรเซสเดียวถึง 49,698,378 มิลลิวินาที คิดเป็น 91%

โปรแกรมที่พัฒนานี้ก็ยังมีข้อจำกัดอยู่บ้างเนื่องจากต้องมีขั้นตอนการเตรียมแบ่งข้อมูลก่อนส่งไปให้แต่ละโปรเซสทำงานซึ่งจะทำให้การแบ่งจำนวนโปรเซสมากจนเกินไปก็จะเสียเวลากับส่วนนี้ และขั้นตอนการนำผลลัพธ์การทำงานของแต่ละโปรเซสกลับมารวมกัน ยังใช้เวลานานอยู่บ้าง เพราะว่าในตัวอย่างโปรแกรมนี้อ่านข้อมูลจากไฟล์ ผู้วิจัยต้องการเน้นตรงส่วนการแยกโปรเซสในการทำงานมากกว่า และเป็นการทดลองเฉพาะเรื่องเวลาการทำงาน งานวิจัยนี้ยังไม่มีการทดสอบการใช้งานหน่วยความจำว่าใช้ไปมากน้อยเท่าใด

เอกสารอ้างอิง

[1] "ชีวสารสนเทศศาสตร์", Wikipedia July 2010 http://th.wikipedia.org/wiki/ชีวสารสนเทศศาสตร์

[2] Kuha Mahalingam and Omar Bagasra, "Bioinformatics Tools: Searching for Markers in DNA/RNA Sequences", Proceeding of The 2008 International Conference on Bioinformatics and Computational Biology (BIOCOMP' 08), Las Vegas, Nevada, July 14-17, 2008.

[3] "ดีเอ็นเอ (DNA) คืออะไร (What is DNA ?)", July 2010 http://www.thaibiotech.info/what-is-dna.php

[4] "Open-source Erlang - White Paper" , http://www.erlang.org/white_paper.html

[5] Joe Armstrong, "A history of Erlang", Proceedings of the third ACM SIGPLAN conference on History of Programming languages, San Diego, California,PP. 6-1 - 6-2,June 9-10, 2007.