บทที่ 4

การพัฒนาโปรแกรมตามมาตรฐาน MPI

มาตรฐาน MPI เป็นมาตรฐานที่ใช้ในการพัฒนาโปรแกรมที่ใช้ในการทดลองการประมวลผลแบบขนาน ในบทนี้นำเสนอหลักการทำงานโดยสังเขปของมาตรฐาน MPI และฟังก์ชันหลักที่ใช้ในการพัฒนาโปรแกรมในการทดลอง

4.1 มาตรฐาน MPI

Message Passing Interface (MPI) เป็นมาตรฐานการเขียนโปรแกรมในโครงสร้างแบบส่งผ่านข้อความที่ได้รับความนิยมอย่างกว้างขวาง เนื่องมาจากประสิทธิภาพการทำงานที่ดีบนเครื่องคอมพิวเตอร์หลายๆ แบบ โดย MPI ไม่ใช่ภาษาใหม่แต่จะเป็นชุดของโปรแกรมย่อยที่สามารถเรียกใช้จากโปรแกรมที่เขียนด้วยภาษา C หรือ Fortran 77

ในการออกแบบมาตรฐาน MPI จะดึงลักษณะเด่นจากมาตรฐานการเขียนโปรแกรมแบบส่งผ่านข้อความที่มีอยู่แล้ว และมาตรฐาน MPI ได้กำหนดคุณสมบัติพิเศษ เช่น ชนิดข้อมูลที่ผู้ใช้กำหนด, ช่องทางสื่อสารที่มีความทนทาน, การเก็บรวบรวมข้อมูลแบบประสิทธิภาพสูง และการกำหนดขอบเขตการสื่อสาร ซึ่งยังไม่มีมาตรฐานก่อนหน้านี้ที่มีคุณสมบัติพิเศษครบถ้วนเช่นนี้ โดยการพัฒนามาตรฐาน MPI เป็นไปในลักษณะเปิดอาศัยบุคคลจากภาคอุตสาหกรรม, ภาคการศึกษา และนักวิจัยจากหน่วยวิจัยของรัฐบาล

4.1.1 ลักษณะการทำงาน

ในบทความ [12] ได้อธิบายลักษณะการทำงานโดยสังเขปของโปรแกรมที่เขียนตามมาตรฐาน MPI ไว้ดังนี้

4.1.1.1 การสื่อสารระหว่างจุดต่อจุด

ลักษณะของการส่งข้อความแบบที่ง่ายที่สุดคือ การส่งระหว่างจุดต่อจุด โดยข้อความจะถูกส่งจากโปรเซสที่เป็นผู้ส่งไปยังโปรเซสที่เป็นผู้รับ มาตรฐาน MPI จะมีฟังก์ชันที่ใช้สำหรับการรับส่งข้อความระหว่างโปรเซส ซึ่งจะมีการกำหนดชนิดของข้อมูลที่อยู่ในข้อความ ชนิดของข้อมูลมีความสำคัญเนื่องจากมาตรฐาน MPI ได้รับการออกแบบให้ใช้กับระบบเครือข่ายที่มีคอมพิวเตอร์ต่างชนิดกัน (network of heterogeneous workstation) ดังนั้นในการส่งข้อมูลไปยังเครื่องที่มีสถาปัตยกรรมแตกต่างกัน จะสามารถเปลี่ยนข้อมูลให้ตรงกับสถาปัตยกรรมได้ถูกต้อง นอกจากนี้การรับส่งข้อความต้องมีการกำหนดป้าย (tag) ทำให้การรับข้อความสามารถเลือกข้อความให้ตรงกับที่ต้องการรับได้ โดยอาจจะต้องให้ป้ายของข้อความที่จะรับต้องตรงกันถึงจะรับข้อความ หรืออาจจะให้ป้ายไม่จำเป็นต้องตรงกัน

ลักษณะของการรับและการส่งข้อความสามารถแบ่งได้เป็น การรับส่งในลักษณะที่หยุดรอ (blocking) และการรับส่งในลักษณะที่ไม่หยุดรอ (non-blocking) การรับส่งในลักษณะที่หยุดรอจะทำการรอจนกว่าการรับส่งนั้นจะเสร็จเรียบร้อยถึงจะมีการดำเนินการตามโปรแกรมต่อไป ในขณะที่การรับส่งในลักษณะที่ไม่หยุดรอจะมีการดำเนินการตามโปรแกรมต่อไปทันที โดยจะมีฟังก์ชันเพื่อทดสอบว่าการรับส่งนั้นเสร็จสิ้นหรือยัง

การส่งข้อความทั้งแบบที่หยุดรอ และแบบที่ไม่หยุดรอจะมีการทำงานให้เลือกหลายแบบดังนี้ แบบแรกคือ แบบมาตรฐาน (standard) โดยเมื่อการส่งข้อความเสร็จสิ้นไม่ได้หมายความว่าการรับข้อความต้องเริ่มขึ้นแล้ว และไม่สามารถทราบได้ว่าข้อความได้ถูกเก็บในที่พักข้อมูล (buffer) โดย MPI หรือไม่ แบบที่สองคือ แบบเก็บในที่พักข้อมูล (buffered) โดยจะรับประกันว่ามีการจองหน่วยความจำเพื่อเป็นที่พักข้อมูล และการส่งจะเสร็จสิ้นทันทีหลังจากเรียกใช้ฟังก์ชันเพื่อส่งข้อความ ข้อดีก็คือสามารถคาดเดาได้ว่าผู้ส่งผู้รับไม่ต้องประสานเวลากัน และถ้าระบบเครือข่ายมีปัญหา สิ่งที่จะเกิดขึ้นก็มีการกำหนดไว้แล้วคือแสดงความผิดพลาดออกมา แบบที่สามคือ แบบประสานเวลา (synchronous) โดยการส่งข้อความจะเสร็จสิ้นเมื่อโปรเซสที่เป็นผู้รับได้รับข้อความเรียบร้อยแล้ว แบบที่สี่คือ แบบพร้อม (ready) ซึ่งจะคล้ายกับแบบเก็บในที่พักข้อมูลเพราะการส่งจะเสร็จสิ้นทันที แต่จะต่างจากการส่งแบบอื่นตรงที่ต้องมีผู้รับรอรับข้อความอยู่ก่อนแล้ว มิฉะนั้นสิ่งที่จะเกิดขึ้นไม่ได้ถูกกำหนดไว้ในมาตรฐาน MPI การส่งแบบนี้ต้องการที่จะตัดการตอบรับจากผู้รับหรือจากใช้ที่พักข้อมูลเหมือนการส่งแบบอื่นๆ ซึ่งจะช่วยเพิ่มประสิทธิภาพในการทำงาน

4.1.1.2 ชนิดข้อมูลที่ผู้ใช้กำหนด

การเรียกใช้ฟังก์ชันการสื่อสารของ MPI ต้องระบุชนิดของข้อมูล ซึ่งอาจจะเป็นชนิดพื้นฐาน เช่น ตัวเลขจำนวนเต็ม เป็นต้น ข้อมูลที่ใส่ลงไปในข้อความเพื่อทำการส่งจะเป็นชุด (array) ของข้อมูลชนิดเดียวกัน แต่มาตรฐาน MPI ก็ให้ผู้ใช้สามารถกำหนดชนิดข้อมูลใหม่ ชนิดข้อมูลที่กำหนดขึ้นจะเป็นการกำหนดจากข้อมูลชนิดพื้นฐานหลายชนิดที่รวมเป็นกลุ่ม และระบุการวางตำแหน่งของชนิดข้อมูลแต่ละชนิดในกลุ่มนั้น ทำให้การส่งข้อมูลแต่ละครั้งสามารถส่งข้อมูลที่มีความซับซ้อน และประกอบด้วยข้อมูลหลายชนิดได้

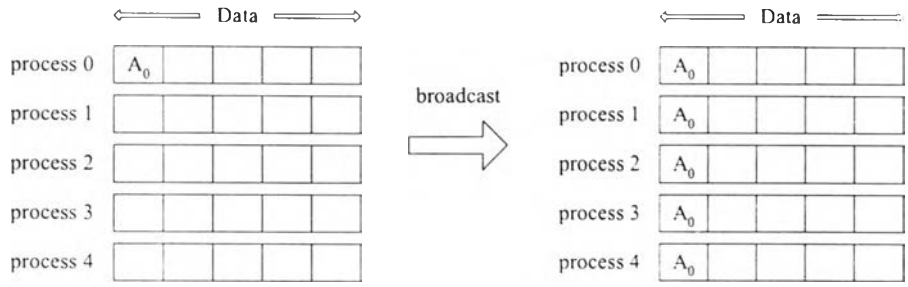
4.1.1.3 การสื่อสารเพื่อเก็บข้อมูล (Collective Communication)

การสื่อสารแบบนี้เป็นการสื่อสารกับกลุ่มของโปรเซส ซึ่งต่างจากแบบระหว่างจุดต่อจุดที่เป็นการสื่อสารระหว่างสองโปรเซส โดยการทำงานของการสื่อสารเพื่อเก็บข้อมูลกับการสื่อสารระหว่างจุดจะแยกออกจากกัน เช่น เมื่อใช้การสื่อสารเพื่อเก็บข้อมูลแล้ว ฟังก์ชันการรับข้อมูลของการสื่อสารระหว่างจุดจะไม่ได้รับข้อมูลของการสื่อสารเพื่อเก็บข้อมูล นอกจากนี้มาตรฐานยังไม่ได้กำหนดว่าการสื่อสารแบบนี้ทำงานเป็นแบบประสานเวลาหรือไม่[1] ผู้ใช้จึงควรระวังเพราะจะทำให้โปรแกรมไม่สามารถนำไปใช้กับเครื่องคอมพิวเตอร์แบบอื่นได้

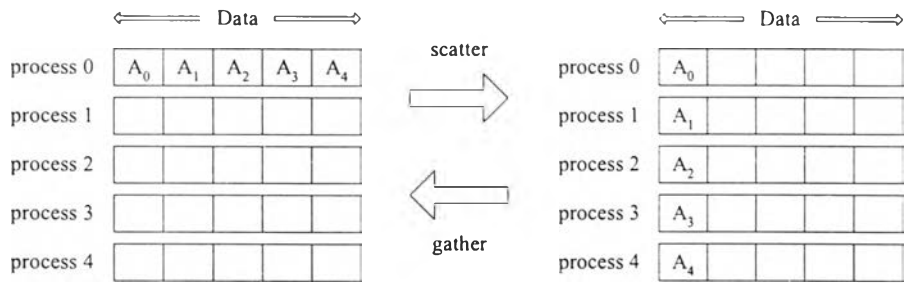
ฟังก์ชันการสื่อสารเพื่อเก็บข้อมูลที่สำคัญมีดังต่อไปนี้

- **การกีดขวาง (barrier)** ทำหน้าที่ให้ทุกโปรเซสทำการหยุดรอจนกว่าทุกโปรเซสจะมีการเรียกใช้ฟังก์ชันนี้ ประโยชน์ก็เพื่อใช้ทำการประสานเวลาของแต่ละโปรเซส โดยฟังก์ชันนี้ไม่มีการรับส่งข้อมูล
- **การส่งกระจาย (broadcast)** ทำหน้าที่ส่งข้อมูลจากโปรเซสหนึ่งไปยังโปรเซสที่เหลือทุกโปรเซส ดังรูป 4.1
- **การรวบรวม (gather)** ทำหน้าที่เก็บรวบรวมข้อมูลจากทุกโปรเซสมาเก็บไว้ที่โปรเซสที่เป็นผู้รับ ดังรูป 4.2

[1] ยกเว้นการกีดขวาง (barrier)

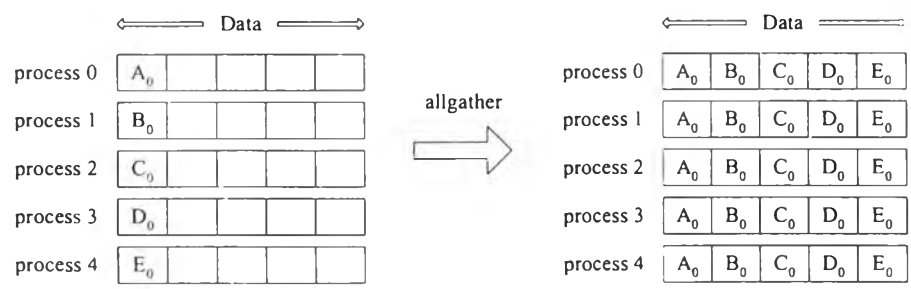


รูปที่ 4.1 การทำงานของฟังก์ชันการส่งกระจาย (broadcast)

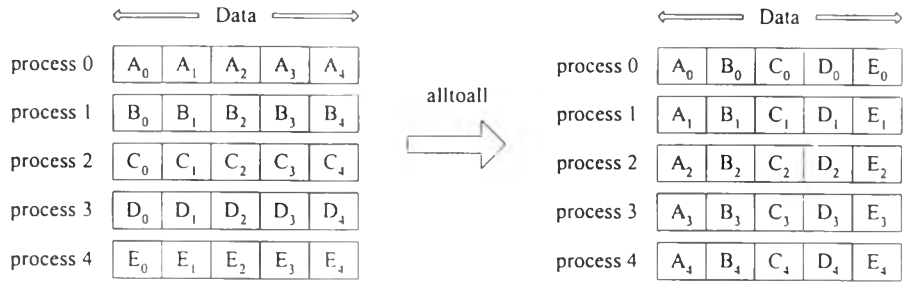


รูปที่ 4.2 การทำงานของฟังก์ชันการแยกส่ง (scatter) และการรวบรวม (gather)

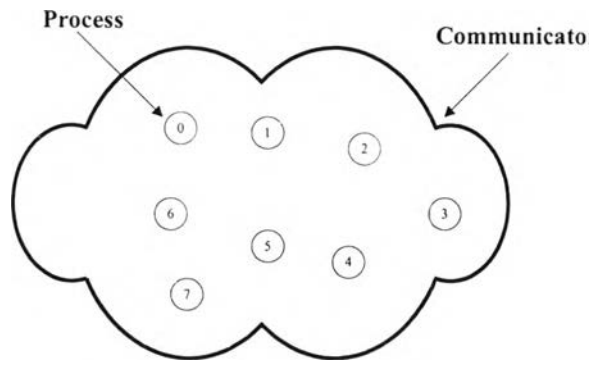
- **การแยกส่ง (scatter)** ทำหน้าที่กระจายข้อมูลจากโปรเซสหนึ่งไปยังโปรเซสที่เหลือทุกโปรเซส แตกต่างจากการส่งกระจาย (broadcast) ที่ข้อมูลที่ส่งไปให้แต่ละโปรเซสจะไม่เหมือนกัน ดังรูป 4.2
- **การรวบรวมทั้งหมด (allgather)** จะคล้ายกับการใช้ฟังก์ชันการรวบรวม (gather) และตามด้วยการส่งกระจาย (broadcast) ข้อมูลที่ได้จากการรวบรวม ดังรูป 4.3
- **ทั้งหมดสู่ทั้งหมด (alltoall)** จะคล้ายกับการทำการรวบรวม (gather) เป็นชุด ดังรูป 4.4
- **การดำเนินการลดลงทั้งหมด (global reduction operation)** ใช้สำหรับการหาผลลัพธ์จากคำนวณกับข้อมูลที่กระจายอยู่ในกลุ่มโปรเซส เช่น ทุกโปรเซสมีข้อมูลเป็นเลขจำนวนเต็ม แล้วต้องการหาผลรวมของตัวเลขทั้งหมด หรือหาค่ามากที่สุด โดยผู้ใช้สามารถกำหนดฟังก์ชันการคำนวณเองได้



รูปที่ 4.3 การทำงานของฟังก์ชันการรวบรวมทั้งหมด (allgather)



รูปที่ 4.4 การทำงานของฟังก์ชันทั้งหมดสู่ทั้งหมด (alltoall)



รูปที่ 4.5 ลักษณะของกลุ่มสื่อสาร

4.1.1.4 กลุ่มสื่อสาร (communicator)

ฟังก์ชันการสื่อสารของ MPI ต้องระบุกลุ่มสื่อสาร และโปรเซสสามารถสื่อสารได้กับโปรเซสที่อยู่ในกลุ่มสื่อสารเดียวกันเท่านั้น ทุกกลุ่มสื่อสารจะประกอบด้วยกลุ่มของโปรเซส โดยโปรเซสจะถูกจัดลำดับ และมีหมายเลขกำกับเริ่มต้นตั้งแต่ 0 ดังรูป 4.5 โปรเซสสามารถอยู่ในหลายกลุ่มสื่อสาร นั่นก็คือโปรเซสจะอยู่ในหลายกลุ่ม โดยแต่ละกลุ่มก็จะได้หมายเลขกำกับไม่เหมือนกัน หมายเลขของโปรเซลนี้จะใช้ระบุลงไปเมื่อทำการเรียกใช้ฟังกชันสื่อสาร

โปรแกรมที่เขียนตามมาตรฐาน MPI จะเริ่มต้นการทำงานด้วย `MPI_COMM_WORLD` ซึ่งเป็นกลุ่มสื่อสารเริ่มต้นที่ประกอบไปด้วยโปรเซสทุกโปรเซส

4.1.2 ฟังก์ชันที่ใช้ในการทดลอง

ในส่วนนี้จะกล่าวถึงฟังก์ชันของ MPI เท่าที่จำเป็นสำหรับการทดลองในวิทยานิพนธ์ฉบับนี้ ฟังก์ชันที่แสดงเป็นฟังก์ชันที่ใช้กับภาษา C สำหรับรายละเอียดเพิ่มเติมสามารถหาได้จากหนังสือเกี่ยวกับมาตรฐาน MPI เช่น [28]

4.1.2.1 การเริ่มต้นโปรแกรม MPI

ฟังก์ชันแรกที่เรียกใช้สำหรับโปรแกรมตามมาตรฐาน MPI คือ MPI_Init ฟังก์ชันนี้จะทำการสร้างกลุ่มสื่อสารเริ่มต้นคือ MPI_COMM_WORLD ซึ่งต้องทำการเรียกใช้ก่อนจะสามารถเรียกใช้ฟังก์ชัน MPI อื่น และการเรียก MPI_Init มากกว่า 1 ครั้งจะทำให้เกิดข้อผิดพลาด

```
int MPI_Init(int *argc, char ***argv)
```

4.1.2.2 การจบการทำงานโปรแกรม MPI

เมื่อจบการจบการทำงานของโปรแกรมจะเรียกใช้ฟังก์ชัน MPI_Finalize หลังจากที่การสื่อสารทั้งหมดเสร็จสิ้นแล้ว ฟังก์ชันนี้จะลบโครงสร้างข้อมูลของ MPI ทั้งหมด แต่จะไม่ยกเลิกการสื่อสารที่ยังไม่เสร็จสิ้น และหลังจากที่เรียกใช้ฟังก์ชันนี้แล้ว จะไม่สามารถเรียกใช้ฟังก์ชันของ MPI ได้อีก แม้แต่ MPI_Init

```
int MPI_Finalize(void)
```

4.1.2.3 การหาข้อมูลเกี่ยวกับกลุ่มสื่อสาร

แต่ละโปรเซสสามารถหาหมายเลขของโปรเซสในกลุ่มสื่อสารด้วยฟังก์ชัน MPI_Comm_rank และสามารถหาจำนวนโปรเซสในกลุ่มสื่อสารด้วยฟังก์ชัน MPI_Comm_size โดย comm จะเป็นการระบุกลุ่มสื่อสาร, rank และ size จะเป็นค่าที่ส่งคืนจากฟังก์ชัน

```
int MPI_Comm_rank(MPI_Comm comm, int *rank)
int MPI_Comm_size(MPI_Comm comm, int *size)
```

4.1.2.4 การรับและการส่งข้อความแบบประสานเวลา

การส่งข้อความใช้ฟังก์ชัน MPI_Ssend ซึ่งทำงานในแบบประสานเวลา โดยฟังก์ชันจะไม่เสร็จสิ้นจนกว่าจะมีการเริ่มต้นรับข้อมูล และสำหรับการรับข้อความใช้ฟังก์ชัน MPI_Recv

ข้อความที่จะใช้ส่งโดย MPI_Ssend และข้อความที่รับโดย MPI_Recv จะเก็บในหน่วยความจำซึ่งอ้างถึงโดย message, count จะระบุความยาวของชุดข้อมูลที่ส่งหรือรับ, datatype เป็นการบอกชนิดของข้อมูล, dest และ source จะเป็นการระบุหมายเลขของโปรเซสที่เป็นปลายทางและต้นทางตามลำดับ, tag เป็นการกำหนดป้ายดังที่ได้อธิบายในส่วน 4.1.1.1, comm เป็นการระบุกลุ่มสื่อสาร และ status เป็นการเก็บสถานะของการรับข้อมูล

```
int MPI_Ssend(void *message, int count, MPI_Datatype datatype, int dest, int tag,
MPI_Comm comm)
int MPI_Recv(void *message, int count, MPI_Datatype datatype, int source, int
tag, MPI_Comm comm, MPI_Status *status)
```

4.1.2.5 การรับและการส่งข้อความแบบไม่ประสานเวลา

การส่งข้อความใช้ฟังก์ชัน MPI_Isend และการรับข้อความใช้ฟังก์ชัน MPI_Irecv ซึ่งทำงานในแบบไม่ประสานเวลา โดยหลังจากการเรียกฟังก์ชันจะกลับคืนจากฟังก์ชันทันที ในการทดสอบว่าการรับและการส่งข้อความเรียบร้อยหรือยังใช้ฟังก์ชัน MPI_Test

พารามิเตอร์ของการรับและการส่งที่เพิ่มจากแบบประสานเวลาคือ request ซึ่งจะใช้ในการตรวจสอบว่าการรับส่งเสร็จสิ้นแล้วหรือไม่ด้วย MPI_Test โดย MPI_Test ต้องการพารามิเตอร์อีก 2 อันคือ flag ซึ่งจะเก็บค่าไม่เท่ากับศูนย์ในกรณีที่การรับส่งเสร็จสิ้น และ status เก็บสถานะของการทำงาน

```
int MPI_Isend(void *message, int count, MPI_Datatype datatype, int dest, int tag,
MPI_Comm comm, MPI_Request *request)
```

```
int MPI_Irecv(void *message, int count, MPI_Datatype datatype, int source, int
tag, MPI_Comm comm, MPI_Request *request)
```

```
int MPI_Test(MPI_Request *request, int *flag, MPI_Status *status)
```

4.1.2.6 การส่งกระจาย (broadcast)

ฟังก์ชัน MPI_Bcast จะใช้ส่งกระจายข้อมูลแบบส่งกระจายจากโปรเซสราก (root process) และทุกโปรเซสจะได้รับข้อความ 1 ชุดจะโปรเซสราก โดยทุกโปรเซสต้องกำหนดโปรเซสรากให้เหมือนกัน

message, count, datatype และ comm จะเหมือนกับฟังก์ชันการรับส่งข้อความ แต่พารามิเตอร์ที่เพิ่มเข้ามาคือ root ซึ่งจะเป็นการระบุหมายเลขโปรเซสราก

```
int MPI_Bcast(void *message, int count, MPI_Datatype datatype, int root, MPI_Comm
comm)
```

4.1.2.7 การประสานเวลา

การประสานเวลาใช้ฟังก์ชัน MPI_Barrier โดยโปรเซสที่เรียกใช้ฟังก์ชันนี้แล้วจะหยุดรอจะกว่าทุกโปรเซสใน comm ทำการเรียกใช้ฟังก์ชันนี้

```
int MPI_Barrier(MPI_Comm comm)
```