แผนการบริหารการสอนประจำบทที่ 10

เนื้อหาประจำบท

บทที่ 10 โครงสร้างของหน่วยเก็บข้อมูลสำรอง

1. การจัดสรรตารางและโครงสร้างของหน่วยเก็บข้อมูลเพื่อการใช้งาน
2. การบริหารจัดการเพื่อการดำเนินการของดิสก์
3. การจัดการพื้นที่ที่ใช้ในการสับเปลี่ยน

จุดประสงค์เชิงพฤติกรรม

เมื่อศึกษาบทที่ 10 แล้วนักศึกษาสามารถ

1. สามารถอธิบายรายละเอียดโครงสร้างและการจัดสรรหน่วยเก็บข้อมูล
2. สามารถอธิบายการจัดการในการจัดระเบียบ การแบ่งพาร์ติชั่นของหน่วยเก็บข้อมูล
3. สามารถอธิบายการจัดการพื้นที่ในการสับเปลี่ยน และความน่าเชื่อถือของหน่วยเก็บข้อมูล

กิจกรรมการเรียนการสอนประจำบท

1. ผู้สอนอธิบายหลักการทำงานของระบบปฏิบัติการ พร้อมยกตัวอย่างประกอบการบรรยาย
2. ให้ผู้เรียนศึกษาเอกสารประกอบการเรียนการสอน ศึกษาทำความเข้าใจและซักถาม
3. ให้ผู้เรียนทำแบบฝึกหัดและงานที่ได้รับมอบหมาย
4. ทดสอบย่อยหลังจบบทเรียน

สื่อการเรียนการสอน

1. สื่ออิเล็กทรอนิกส์ประกอบการสอนวิชาระบบปฏิบัติการ
2. เอกสารประกอบการสอนวิชาระบบปฏิบัติการ
3. หนังสืออ่านประกอบค้นคว้าเพิ่มเติม

การวัดผลและประเมินผล

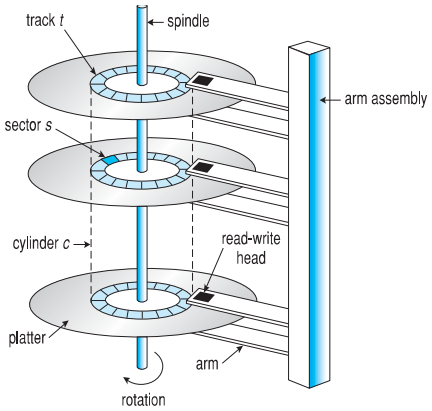
1. สังเกตจากการซักถามในระหว่างการเรียน
2. สังเกตจากความสนใจและความตั้งใจ
3. ประเมินจากการอภิปรายกลุ่มย่อย และการทำแบบฝึกหัด
4. ประเมินจากการสอบระหว่างภาคและปลายภาค

บทที่ 10
โครงสร้างของหน่วยเก็บข้อมูลสำรอง

ในบทนี้จะอธิบายถึงโครงสร้างของหน่วยเก็บข้อมูลในระดับต่ำที่สุดในระบบแฟ้มข้อมูล คือ โครงสร้างของหน่วยเก็บข้อมูลสำรอง

10.1 โครงสร้างของดิสก์

ดิสก์ คือ ก้อนของหน่วยเก็บข้อมูลสำรองสำหรับระบบคอมพิวเตอร์ยุคใหม่ ส่วนประกอบภายในฮาร์ดดิสก์จะประกอบด้วยแผ่นวงกลมที่มีขนาด 2-5.25 นิ้วเรียงซ้อนกัน ซึ่งเราเรียกแผ่นวงกลมนี้ว่า Disk ซึ่งจะถูกบรรจุอยู่ภายในกล่องที่ปิดสนิทกันฝุ่นและอากาศ เพื่อไม่ให้เข้าไปทำความเสียหายแก่ดิสก์ด้านใน การอ่านข้อมูลนั้นภายในฮาร์ดดิสก์จะมีหัวอ่าน 2 หัวต่อแผ่นดิสก์ 1 จาน ซึ่งจะช่วยกันอ่านและเขียนข้อมูล ซึ่งพื้นผิวของจานเราจะเรียกว่า Platter โดยจะถูกแบ่งส่วนภายในจานซึ่งเรียกว่า Track ส่วนวิธีการนับจะนับจากวงนอกสุดของจาน จะเป็น Track 0 ถ้าจำนวนของ Track ยิ่งมากความจุของฮาร์ดดิสก์จะมีมากขึ้นตามด้วย และ Track แบ่งออกเป็นส่วน ๆ ตามแนวเส้นผ่านศูนย์กลางที่เรียกว่าเซ็กเตอร์ (Sector) และไซลินเดอร์ (Cylinders) จะใช้เรียกตำแหน่งของ Track ของ Platter



ภาพที่ 10.1 แสดงกลไกของดิสก์
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.468)

เครื่องขับดิสก์ (Disk drive) ยุคใหม่ ถูกพูดถึงเหมือนเป็นอาร์เรย์ 1 มิติขนาดใหญ่ของบล็อกทางตรรกะ (Logical block) ซึ่งบล็อกทางตรรกะคือหน่วยที่เล็กที่สุดของการโอนย้ายข้อมูล ขนาดของบล็อกทางตรรกะมักมีขนาด 512 ไบต์ ถึงแม้ว่าดิสก์บางตัวจะสามารถทำ “การจัดระเบียบระดับต่ำ” (Low-level formatted) เพื่อเลือกขนาดของบล็อกที่ต่างออกไป เช่น 1024 ไบต์ อาร์เรย์ 1 มิติของบล็อกทางตรรกะถูกจับคู่ (Map) ไปบนเซกเตอร์ของดิสก์แบบเรียงลำดับ เซกเตอร์ 0 คือ เซกเตอร์แรกของแทร็กแรกบนไซลินเดอร์นอกสุด การจับคู่ดำเนินการผ่านแทร็กแล้วผ่านแทร็กไปยังไซลินเดอร์ แล้วผ่านไซลินเดอร์จากนอกสุดไปสู่ชั้นในสุด

โดยใช้การจับคู่แบบนี้ ทำให้การแปลงหมายเลขบล็อกทางตรรกะมาเป็นตำแหน่งของดิสก์จริง ๆ ซึ่งประกอบด้วย หมายเลขไซลินเดอร์ หมายเลขแทร็กภายในไซลินเดอร์นั้น และหมายเลขเซกเตอร์ภายในแทร็กนั้น สามารถเป็นไปได้ในทางปฏิบัติ เป็นการยากที่จะแปลงเลขดังกล่าวด้วยเหตุผล 2 ประการ

1) ดิสก์ส่วนใหญ่มีเซกเตอร์เสีย แต่การจับคู่จะทำได้โดยใช้เซกเตอร์อื่นในดิสก์แทน

2) จำนวนของเซกเตอร์ต่อแทร็กไม่คงที่ แทร็กมาจากจุดศูนย์กลางของดิสก์ ยิ่งแทร็กยาวมากจำนวนเซกเตอร์ก็จะมากตาม

ดังนั้นดิสก์ในยุคใหม่จะถูกจัดระเบียบเป็นโซนของไซลินเดอร์ จำนวนของเซกเตอร์ต่อ แทร็ก คือ ค่าคงที่ภายในโซน แต่เมื่อเราเคลื่อนจากโซนภายในไปสู่โซนภายนอก จำนวนของเซกเตอร์ต่อแทร็กจะเพิ่มขึ้น ตัวอย่างเช่น แทร็กในโซนนอกสุดมีเซกเตอร์ 40% มากกว่าแทร็กในโซนในสุด จำนวนของเซกเตอร์ต่อแทร็กมีจำนวนเพิ่มขึ้นเมื่อเทคโนโลยีของดิสก์ได้รับการปรับปรุง และเป็นเรื่องปกติที่จะมีเซกเตอร์ต่อแทร็กมากกว่า 100 เซกเตอร์ในโซนภายนอกของดิสก์ ในทำนองเดียวกันจำนวนของไซลินเดอร์ต่อดิสก์ก็จะเพิ่มขึ้น

10.2 การจัดตารางของดิสก์

หนึ่งในความรับผิดชอบของระบบปฏิบัติการ คือ ต้องใช้ฮาร์ดแวร์อย่างมีประสิทธิภาพ สำหรับเครื่องขับดิสก์นั่นหมายถึง มีเวลาในการเข้าถึงอย่างรวดเร็ว (Fast access time) และ Disk bandwidth ในเรื่องเวลาในการเข้าถึงมี 2 องค์ประกอบหลัก คือ

1) เวลาในการค้นหา (Seek time) คือ เวลาที่แขนของดิสก์เคลื่อนหัวอ่านไปสู่ไซลินเดอร์ที่มีเซกเตอร์ที่ต้องการ

2) เวลาในการหมุนหัวอ่าน (Rotational latency) คือ เวลาในการรอคอยที่เพิ่มขึ้นสำหรับดิสก์ในการหมุนเซกเตอร์ที่ต้องการมาสู่หัวอ่าน

Disk bandwidth คือ จำนวนไบต์ทั้งหมดที่ถูกโอนย้ายมาจากเวลาทั้งหมดตั้งแต่การร้องขอบริการครั้งแรกจนถึงการโอนย้ายครั้งสุดท้ายเสร็จสิ้น เราสามารถปรับปรุงทั้งเวลาในการเข้าถึง และ Bandwidth โดยการจัดตารางบริการของการร้องขอรับ-ส่งข้อมูลของดิสก์ในลำดับที่ดี เมื่อมีโพรเซสที่ต้องการรับข้อมูลจากดิสก์หรือส่งข้อมูลไปสู่ดิสก์ จึงเกิดการเรียกระบบไปสู่ระบบปฏิบัติการ การร้องขอเจาะจงสารสนเทศหลายชิ้น ดังนี้คือ

1) การปฏิบัติการนี้ คือ การนำเข้าหรือการส่งออก

2) ตำแหน่งของดิสก์ที่ใช้ในการโอนย้ายข้อมูลคืออะไร

3) ตำแหน่งของหน่วยความจำสำหรับการโอนย้ายข้อมูลคืออะไร

4) จำนวนของไบต์ที่ถูกโอนย้ายข้อมูลเป็นเท่าไร

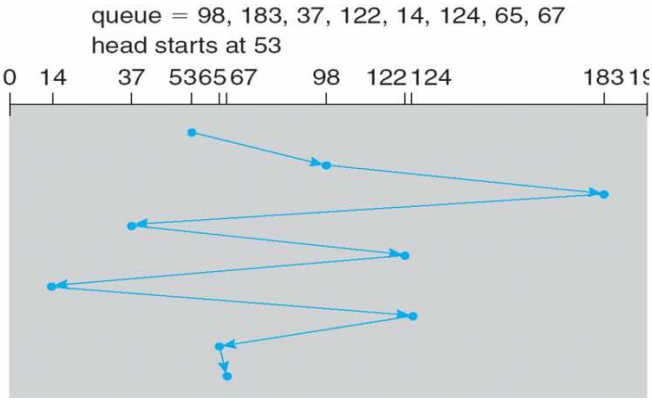
5) ถ้าเครื่องขับดิสก์และตัวควบคุมว่าง การร้องขอจะได้รับบริการทันที ถ้ามันไม่ว่างการร้องขอใหม่จำเป็นต้องนำไปไว้ในแถวคอย

สำหรับระบบ Multiprogramming ที่มีหลายโพรเซสเข้าแถวคอยของดิสก์ (Disk queue) อาจมีการร้องขออยู่หลายตัว ดังนั้นเมื่อการร้องขอหนึ่งได้รับบริการแล้ว ระบบปฏิบัติการจึงมีโอกาสที่จะเลือกการร้องขอมาให้บริการถัดไป

10.2.1 การจัดตารางแบบมาก่อน-ได้ก่อน (FCFS Scheduling)

รูปแบบที่ง่ายที่สุดของการจัดตารางของดิสก์ คือ มาก่อน-ได้ก่อน (First-come, First-served : FCFS) พิจารณาตัวอย่าง ถ้าแถวคอยของดิสก์ มีการร้องขอการรับส่งข้อมูลของบล็อกบนไซลินเดอร์ดังนี้

98 , 183 , 37 , 122 , 14 , 124 , 65 , 67 ตามลำดับ ถ้าหัวอ่านเริ่มต้นที่ไซลินเดอร์ที่ 53 มันจะเริ่มเคลื่อนในครั้งแรกจาก 53 ไป 98 แล้วจึงไป 183 , 37 , 122 , 14 , 124 , 65 และสุดท้ายที่ 67 สำหรับการเคลื่อนย้ายหัวอ่านทั้งหมด 640 ไซลินเดอร์ การจัดตารางนี้คือแผนภาพดังภาพที่ 10.2



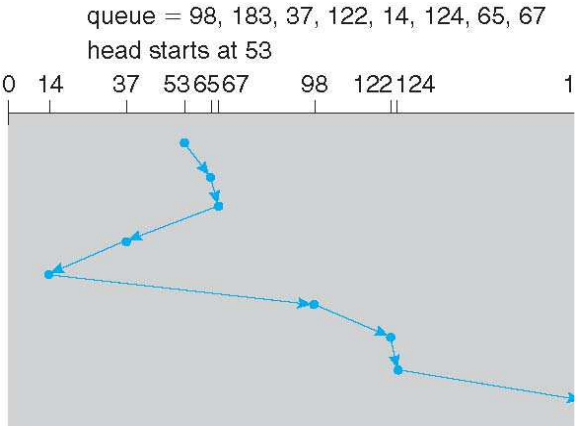
ภาพที่ 10.2 กราฟแสดงการจัดตารางของดิสก์แบบมาก่อน-ได้ก่อน (FCFS)
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.474)

ปัญหาของการจัดตารางวิธีนี้ดูได้จากเส้นกราฟที่มีการเหวี่ยงกว้างจาก 122 ไป 14 แล้วกลับมา 124 ถ้าการร้องขอสำหรับไซลินเดอร์ 37 และ 14 ได้รับบริการต่อกัน ก่อนหรือหลังการร้องขอที่ 122 และ 124 การเคลื่อนย้ายหัวอ่านทั้งหมดน่าจะลดลงอย่างมาก ดังนั้นสมรรถนะ (Performance) ควรจะได้รับการปรับปรุง

10.2.2 การจัดตารางแบบเวลาในการค้นหาสั้นที่สุดได้ก่อน (SSTF Scheduling)

การที่จะบริการการร้องขอทั้งหมดที่อยู่ใกล้ตำแหน่งของหัวอ่านในขณะนั้นก่อนจะย้ายหัวอ่านไกลออกไปเพื่อบริการการร้องขออื่น สมมติฐานนี้คือพื้นฐานสำหรับวิธีเวลาในการค้นหาสั้นที่สุดได้ก่อน (Shortest-seek Time-first : SSTF) วิธี SSTF จะเลือกการร้องขอที่มีเวลาในการค้นหาน้อยที่สุดจากตำแหน่งปัจจุบันของหัวอ่าน เมื่อเวลาในการค้นหาเพิ่มขึ้นด้วยจำนวนของไซลินเดอร์ที่ถูกอ่าน โดยหัวอ่าน SSTF จะเลือกการร้องขอที่ใกล้ที่สุดกับตำแหน่งปัจจุบันของหัวอ่าน

จากตัวอย่างที่ผ่านมา จะเห็นได้ว่าการร้องขอที่ใกล้กับตำแหน่งเริ่มต้นของหัวอ่านที่สุด (53) คือ ที่ไซลินเดอร์ 65 เมื่อเราเคลื่อนมาที่ไซลินเดอร์ 65 การร้องขอที่ใกล้ที่สุดถัดไปคือที่ไซลินเดอร์ 67 จากนั้นการร้องขอที่ไซลินเดอร์ 37 จะใกล้กว่า 98 ดังนั้น 37 จะได้รับบริการถัดไป ต่อมาทำการบริการการร้องขอที่ไซลินเดอร์ 14 แล้วเป็น 98 , 122 , 124 และสุดท้ายที่ 183 (ดังภาพที่ 10.3) ผลลัพธ์ของวิธีการจัดตารางแบบนี้การเคลื่อนย้ายหัวอ่านทั้งหมดมีเพียง 236 ไซลินเดอร์เท่านั้น น้อยกว่า 1 ใน 3 ของระยะทางของวิธี FCFS วิธีนี้ทำให้สมรรถนะได้รับการปรับปรุงอย่างมาก



ภาพที่ 10.3 กราฟแสดงการจัดตารางของดิสก์แบบเวลาในการค้นหาสั้นที่สุดได้ก่อน (SSTF)
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.475)

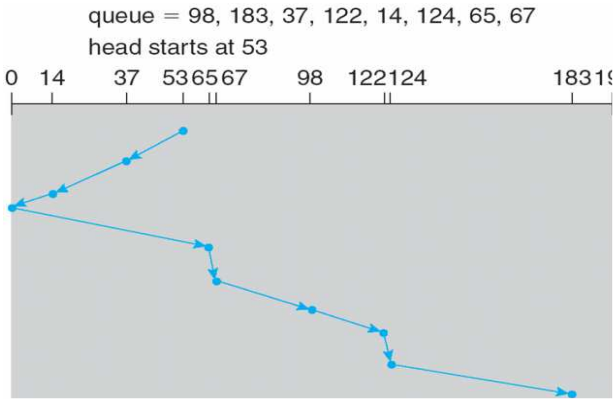
การจัดตารางแบบ SSTF จำเป็นที่สุดสำหรับรูปแบบของการจัดตารางแบบ SJF และเหมือนกับ SJF มันอาจจะเกิดปัญหาการอดตาย (Starvation) ยังจำได้หรือไม่ว่าการร้องขออาจมาถึงที่เวลาต่าง ๆ กัน สมมติว่าเรามีการร้องขอ 2 ตัวในแถวคอย สำหรับไซลินเดอร์ 14 และ 186 และขณะที่กำลังบริการการร้องขอจาก 14 การร้องขอใหม่ใกล้ 14 มาถึง การร้องขอใหม่นี้จะได้รับบริการเป็นรายถัดไป ทำให้การร้องขอที่ 186 ต้องรอก่อน ในขณะที่การร้องขอนี้กำลังได้รับบริการ การร้องขออีกตัวที่ใกล้ 14 มาถึงอีก ในทางทฤษฎีถ้าสายการร้องขออย่างต่อเนื่องที่ใกล้อีกตัวหนึ่งมาถึงจะเป็นเหตุให้การร้องขอสำหรับไซลินเดอร์ 186 จะต้องรออย่างไม่สิ้นสุด สิ่งนี้จะเป็นการเพิ่มแถวคอยการร้องขอให้ยาวขึ้น

ถึงแม้ว่าวิธี SSTF คือ การปรับปรุงอย่างมากจาก FCFS แต่มันยังไม่ดีที่สุด จากตัวอย่าง เราสามารถทำได้ดีกว่าโดยการเคลื่อนหัวอ่านจาก 53 ไป 37 ถึงแม้ว่า 37 จะไม่ใกล้ที่สุด จากนั้นไป 14 ก่อนจะกลับไป 65 , 67 , 98 , 122 , 124 และ 183 กลยุทธ์นี้เป็นการลดการเคลื่อนย้ายหัวอ่านทั้งหมดเหลือ 208 ไซลินเดอร์

10.2.3 การจัดตารางแบบกวาด (SCAN Scheduling)

ในวิธีแบบกวาด (SCAN) แขนของดิสก์เริ่มต้นที่จุดสิ้นสุดจุดหนึ่งของดิสก์ (A) และเคลื่อนไปยังจุดสิ้นสุดอื่น (B) การบริการการร้องขอจะทำได้เมื่อมันมาถึงในแต่ละไซลินเดอร์ จนกระทั่งมันไปถึงจุดสิ้นสุดอื่นของดิสก์ (B) ที่จุดสิ้นสุดอื่น (B) ทิศทางของการเคลื่อนของหัวอ่านจะถูกย้อนกลับและให้บริการต่อไป หัวอ่านจะกวาดไปข้างหน้าและข้างหลังผ่านดิสก์ไปเรื่อย ๆ

พิจารณาตัวอย่างเดิมก่อนจะใช้วิธีจัดตารางแบบกวาด กับการร้องขอบนไซลินเดอร์ 98, 183, 37, 122, 14, 124, 65 และ 67 เราจำเป็นต้องรู้ทิศทางของการเคลื่อนหัวอ่านก่อน โดยถ้าตอนนี้ตำแหน่งปัจจุบันของหัวอ่านคือ 53 ถ้าแขนของดิสก์กำลังเคลื่อนไปทาง 0 หัวอ่านจะบริการ 37 และจากนั้นเป็น 14 ที่ไซลินเดอร์ 0 แขนจะย้อนกลับและจะเคลื่อนผ่านจุดสิ้นสุดอื่นของดิสก์ (ที่ไม่ใช่ 0) การบริการการร้องขอจะทำที่ 65 , 67 , 98 , 122 , 124 และ 183 (ดังภาพที่ 10.4)



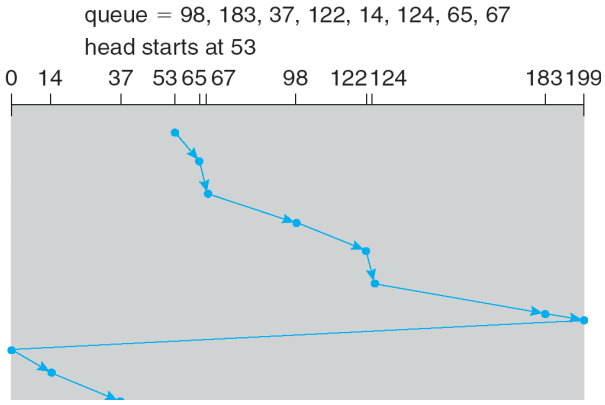
ภาพที่ 10.4 กราฟแสดงการจัดตารางของดิสก์แบบกวาด (SCAN)
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.476)

ถ้าการร้องขอที่จะเข้ามาอยู่ในแถวคอยหน้าของหัวอ่าน มันก็จะได้รับบริการเกือบจะทันที ส่วนการร้องขอที่มาถึงหลังหัวอ่าน ก็จะต้องรอจนกระทั่งแขนเคลื่อนไปถึงจุดสิ้นสุดของดิสก์แล้วย้อนกลับมาอีกครั้ง วิธีแบบกวาด (SCAN algorithm) นี้บางครั้งก็เรียกว่า วิธีแบบบันไดเลื่อน (Elevator algorithm) เพราะแขนของดิสก์มีพฤติกรรมคล้ายบันไดเลื่อนในตึก การบริการเริ่มต้นกับการร้องขอทั้งหมดในขาขึ้นและจากนั้นจะบริการการร้องขอย้อนกลับมาอีกทาง

สมมติว่าการกระจายของการร้องขอสำหรับไซลินเดอร์เป็นแบบเดียวกัน พิจารณาความหนาแน่นของการร้องขอเมื่อหัวอ่านมาถึงจุดสิ้นสุดจุดหนึ่งแล้วย้อนทิศทางกลับ ณ จุดนี้ มีการร้องขอเพียงเล็กน้อยที่สัมพันธ์กันที่อยู่หน้าหัวอ่าน ดังนั้นไซลินเดอร์เหล่านี้ก็จะได้รับบริการ แต่ความหนาแน่นที่มากที่สุดของการร้องขออยู่ที่อีกจุดสิ้นสุดหนึ่งของดิสก์ การร้องขอเหล่านี้ก็ต้องรอนานมาก ดังนั้นทำไมจึงไม่คิดที่จะไปทางด้านนั้นก่อน นั่นคือความคิดของวิธีถัดไป

10.2.4 การจัดตารางแบบกวาดเป็นวง (C-SCAN Scheduling)

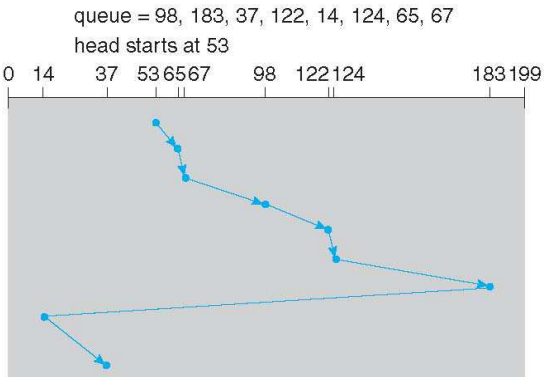
การกวาดเป็นวง (Circular SCAN เรียกย่อว่า C-SCAN) คือ การเปลี่ยนแปลงของการกวาดซึ่งถูกออกแบบมาเพื่อจัดการกับเวลารอคอยที่มากกว่า 1 รูปแบบ เหมือนกับแบบกวาด แบบกวาดเป็นวงจะเคลื่อนหัวอ่านจากจุดสิ้นสุดจุดหนึ่ง (A) ของดิสก์ไปสู่อีกจุดหนึ่ง (B) มันจะกลับไปสู่จุดเริ่มต้นของดิสก์ในทันที โดยไม่บริการการร้องขอใด ๆ ในขากลับนี้ (ดังภาพที่ 10.5) การจัดตารางแบบกวาดเป็นวงนี้เป็นสิ่งจำเป็นสำหรับการจัดการกับไซลินเดอร์แบบวงกลม ซึ่งล้อมรอบจากไซลินเดอร์สุดท้ายไปสู่ไซลินเดอร์แรก



ภาพที่ 10.5 กราฟแสดงการจัดตารางของดิสก์แบบกวาดเป็นวง (C-SCAN)
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.476)

10.2.5 การจัดตารางแบบ LOOK (LOOK Scheduling)

จะเห็นว่า ทั้งแบบกวาดและกวาดเป็นวงจะเคลื่อนแขนดิสก์ไปทั่วทั้งดิสก์ ในทางปฏิบัติไม่มีวิธีไหนที่เอาไปใช้จริงได้ ในความเป็นจริงแขนดิสก์จะไปไกลเพียงแค่การร้องขอสุดท้ายเท่านั้นในแต่ละทิศทาง จากนั้นมันก็จะย้อนกลับไปอีกทางทันทีโดยไม่ต้องไปจุดสิ้นสุดของดิสก์ การทำเช่นนี้เรียกว่า LOOK และ C-LOOK เพราะมันมองการร้องขอก่อนจะเคลื่อนไปยังทิศทางที่ได้ (ดังภาพที่ 10.6)



ภาพที่ 10.6 กราฟแสดงการจัดตารางของดิสก์แบบ C-LOOK
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.477)

10.2.6 การเลือกวิธีการจัดตารางของดิสก์ (Selection of a Disk-Scheduling Algorithm)

วิธี SSTF ดูจะธรรมดาและความเป็นไปได้ตามธรรมชาติ วิธี SCAN และ C-SCAN ใช้ได้ดีสำหรับระบบที่มีภาระหนักบนดิสก์ เพราะมันจะเกิดปัญหาการแช่แข็งได้ยาก สำหรับรายการของการร้องขอโดยเฉพาะ มันเป็นไปได้ที่จะกำหนดลำดับที่ดีที่สุด แต่ในแง่ของการคำนวณจำเป็นต้องหาการจัดตารางที่ดีที่สุด ซึ่งอาจจะไม่ประหยัดไปกว่าแบบ SSTF หรือ SCAN

สำหรับดิสก์ยุคใหม่เวลาในการหมุนหัวอ่าน (Rotational latency) มีขนาดใกล้เคียงกับเวลาในการค้นหาเฉลี่ย (Average seek time) แต่เป็นเรื่องยากสำหรับระบบปฏิบัติการที่จะจัดตารางเพื่อปรับปรุงเวลาในการหมุนหัวอ่าน เพราะดิสก์ยุคใหม่ไม่เปิดเผยตำแหน่งทางกายภาพของบล็อกทางตรรกะ ผู้ผลิตดิสก์ได้ช่วยแก้ปัญหานี้โดยใช้วิธีการจัดตารางของดิสก์ในฮาร์ดแวร์ควบคุม (Controller hardware) ที่สร้างไว้ในเครื่องขับดิสก์ (Disk drive) ถ้าระบบปฏิบัติการส่งการร้องขอเป็นกลุ่มมาที่ตัวควบคุม ตัวควบคุมสามารถจัดแถวคอยพวกมันและจากนั้นก็ทำการจัดตาราง เพื่อปรับปรุงทั้งเวลาในการค้นหาและเวลาในการหมุนหัวอ่าน ถ้ามีแต่สมรรถนะในการรับส่งข้อมูลเท่านั้นที่เราสนใจ ระบบปฏิบัติการจะยินดีมากที่จะโอนความรับผิดชอบของการจัดตารางของดิสก์ไปให้ฮาร์ดแวร์ของดิสก์ทำแทน

ในทางปฏิบัติ ระบบปฏิบัติการมีข้อจำกัดอื่น ๆ สำหรับการบริการการร้องขอ เช่น การจัดสรรหน้าตามคำขอ (Demand paging) อาจมีศักดิ์สูงกว่าการรับส่งข้อมูลของโปรแกรมประยุกต์ (Application I/O) และการเขียนก็เป็นเรื่องที่เร่งด่วนกว่าการอ่าน ถ้า Cache กำลังทำงานออกนอกเพจที่ว่าง ดังนั้นอาจจะต้องการการรับประกันลำดับของกลุ่มของดิสก์ที่จะเขียน เพื่อทำให้ระบบแฟ้มข้อมูลแข็งแรงเพื่อเผชิญกับการชนของระบบ (System crash) ลองมาพิจารณาดูว่าอะไรจะเกิดขึ้นถ้าระบบปฏิบัติการจัดสรรหน้าของดิสก์ให้แฟ้มข้อมูล แล้วโปรแกรมประยุกต์เขียนข้อมูลลงหน้านั้นก่อนที่ระบบปฏิบัติการจะมีโอกาสปรับปรุงหรือแจ้งให้ดิสก์ทราบว่าหน้านั้นว่าง เพื่อทำให้ความต้องการเหมาะสม ระบบปฏิบัติการอาจจะเลือกที่จะทำการจัดตารางของดิสก์เอง แล้วป้อนการร้องขอให้ตัวควบคุมดิสก์ทีละตัว

10.3 การจัดการดิสก์

บทบาทที่สำคัญหน้าที่หนึ่งของระบบปฏิบัติการคือ หน้าที่สำหรับการจัดการเวลาในการใช้ดิสก์ (Disk scheduling) การฟอร์แมตดิสก์ การจัดการเนื้อที่บนดิสก์ที่เสียหาย การจัดการพื้นที่ว่างบนดิสก์ และการดูแลรักษาดิสก์

10.3.1 การจัดระเบียบดิสก์ (Disk Formatting)

ก่อนที่ดิสก์จะสามารถบรรจุข้อมูลได้ มันต้องถูกแบ่งเป็นเซกเตอร์ซึ่งตัวควบคุมดิสก์สามารถอ่านและเขียนได้ กระบวนการนี้เรียกว่า “การจัดระเบียบระดับต่ำ” (Low-level formatting) หรือ การจัดระเบียบเชิงกายภาพ (Physical formatting) การจัดระเบียบระดับต่ำจะเติมโครงสร้างข้อมูลพิเศษลงไปในแต่ละเซกเตอร์ของดิสก์ โครงสร้างข้อมูลสำหรับเซกเตอร์ประกอบด้วย หัวอ่าน (Header) พื้นที่ของข้อมูล ที่มักมีขนาด 512 ไบต์ และส่วนท้ายของหัวอ่าน (Trailer) มีการบรรจุสารสนเทศที่ถูกใช้โดยตัวควบคุมดิสก์ เช่น หมายเลขเซกเตอร์ และ รหัสถูก-ผิด (Error-correcting code : ECC) เมื่อตัวควบคุมเขียนเซกเตอร์ของข้อมูลในระหว่างการรับส่งปกติ ECC จะถูกทำให้ข้อมูลถูกต้องและเป็นข้อมูลล่าสุด ด้วยค่าที่คำนวณจากไบต์ทั้งหมดในพื้นที่ข้อมูล เมื่อเซกเตอร์ถูกอ่าน ECC จะถูกคำนวณใหม่และถูกเปรียบเทียบกับค่าที่เก็บเอาไว้ ถ้าหมายเลขที่ถูกเก็บไว้ และที่ได้จากการคำนวณต่างกัน การไม่เข้าคู่นี้ชี้ให้เห็นว่า พื้นที่ข้อมูลของเซกเตอร์กลายเป็นข้อผิดพลาด และเซกเตอร์ของดิสก์นั้นอาจจะไม่ดี (เกิด Bad sector) ECC คือ รหัสถูก-ผิด เพราะมันบรรจุสารสนเทศที่เพียงพอ ซึ่งถ้ามีข้อมูล 1 หรือ 2 บิตถูกทำให้ผิดพลาด ตัวควบคุมจะสามารถแยกได้ว่าบิตไหนถูกเปลี่ยนแปลง และสามารถคำนวณได้ว่า ค่าที่ถูกต้อง

ควรจะเป็นค่าใด กระบวนการของ ECC จะถูกทำอย่างอัตโนมัติโดยตัวควบคุมไม่ว่าเซกเตอร์จะถูกอ่านหรือถูกเขียน

ฮาร์ดดิสก์ส่วนใหญ่มักถูกจัดระเบียบระดับต่ำ จะถูกดำเนินการมาจากโรงงานซึ่งเป็นส่วนหนึ่งของขั้นตอนในการผลิต การจัดระเบียบนี้ทำให้การผลิตสามารถทดสอบดิสก์และเริ่มต้นจับคู่จากหมายเลขบล็อกทางตรรกะเพื่อไม่ให้เกิดข้อบกพร่องของเซกเตอร์บนดิสก์ สำหรับฮาร์ดดิสก์ทั้งหลาย เมื่อตัวควบคุมดิสก์ถูกชี้แนะให้ทำการจัดระเบียบระดับต่ำ ทำให้สามารถบอกได้ว่ามีจำนวนไบต์ของข้อมูลที่ว่างระหว่าง Header และ Trailer ของเซกเตอร์ทั้งหมดเป็นจำนวนเท่าไร มันเป็นไปได้ที่จะเลือกขนาดต่าง ๆ เช่น 256, 512 และ 1024 ไบต์ การจัดระเบียบดิสก์ด้วยเซกเตอร์ขนาดใหญ่ หมายความว่า มีเซกเตอร์เพียงเล็กน้อยในแต่ละแทร็ก และยังหมายถึงว่ามีหัวอ่านและ trailer เพียงเล็กน้อยที่จะถูกเขียนบนแต่ละแทร็ก ดังนั้นจึงเป็นการเพิ่มพื้นที่ว่างให้แก่ข้อมูลของผู้ใช้ ระบบปฏิบัติการบางระบบสามารถจัดการกับเซกเตอร์ขนาด 512 ไบต์ เท่านั้น

เพื่อใช้ดิสก์เก็บแฟ้มข้อมูล ระบบปฏิบัติการยังคงต้องการบันทึกโครงสร้างข้อมูลบนดิสก์ โดยทำงานเป็น 2 ขั้นตอน คือ

1) ทำการแบ่งส่วน (Partition) ดิสก์เป็นหนึ่งหรือหลายกลุ่มของไซลินเดอร์ ระบบปฏิบัติการสามารถจัดการแต่ละส่วนเป็นเหมือนดิสก์ที่แยกจากกัน ตัวอย่างเช่น พาร์ติชั่นหนึ่งอาจจะเก็บโปรแกรมของระบบปฏิบัติการ ในขณะที่อีกพาร์ติชั่นเก็บแฟ้มข้อมูลของผู้ใช้หลังจากทำการแบ่งส่วน

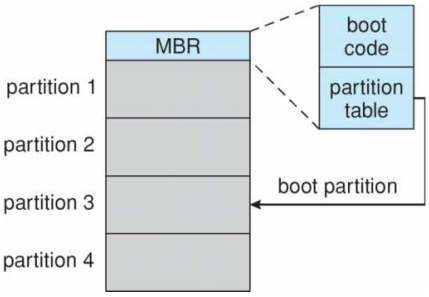
2) การจัดระเบียบเชิงตรรกะ (Logical formatting) หรือ “การทำระบบแฟ้มข้อมูล” (Making a file system) ในขั้นตอนนี้ ระบบปฏิบัติการจะเก็บโครงสร้างข้อมูลของระบบแฟ้มข้อมูลเริ่มแรกไว้บนดิสก์ โครงสร้างข้อมูลเหล่านี้อาจรวมถึงแผนที่ของที่ว่างและการจัดสรรพื้นที่ (FAT) และไดเรกทอรี่ว่างเริ่มต้น

10.3.2 บูตบล็อก (Boot Block)

เมื่อคอมพิวเตอร์เริ่มทำงาน (เช่นเปิดเครื่อง หรือบูตเครื่องใหม่) มันจำเป็นต้องมีโปรแกรมเริ่มต้นเพื่อการทำงาน โปรแกรมนี้คือ Bootstrap เป็นโปรแกรมการเริ่มต้นทั้งหมดของระบบ เริ่มจากซีพียู ตัวควบคุมอุปกรณ์ และหน่วยความจำหลัก แล้วจึงเริ่มนำระบบปฏิบัติการเข้าสู่ระบบ โดยมันจะหาแก่น (kernel) ของระบบปฏิบัติการในดิสก์ แล้วนำแก่นนั้นเข้าสู่หน่วยความจำและกระโดดไปสู่ตำแหน่งเริ่มต้นเพื่อเริ่มการทำงานของระบบปฏิบัติการ

คอมพิวเตอร์ส่วนใหญ่โปรแกรม Bootstrap จะถูกเก็บไว้ใน ROM (Read-only memory) เพราะ ROM ไม่จำเป็นต้องใช้ตั้งแต่แรก และมันเป็นตำแหน่งที่แน่นอนคงที่ซึ่งหน่วยประมวลผลสามารถเริ่มทำงานเพื่อเปิดเครื่องหรือ Reset เครื่อง และเพราะ ROM ใช้อ่านได้อย่างเดียว มันจึงไม่สามารถจะติดไวรัสได้ แต่ปัญหาก็คือ การเปลี่ยนแปลงโค๊ดของ Bootstrap จำเป็นต้องเปลี่ยนชิป ROM ด้วย ด้วยเหตุนี้ ระบบส่วนใหญ่จะเก็บโปรแกรม Bootstrap loader เล็ก ๆ ไว้ใน Boot ROM เมื่อต้องการใช้งานก็ค่อยนำโปรแกรม Bootstrap แบบเต็มเข้ามาจากดิสก์ ดังนั้นโปรแกรม Bootstrap แบบเต็มจะสามารถเปลี่ยนแปลงได้ง่าย เราเพียงแต่นำเวอร์ชั่นใหม่เขียนลงบนดิสก์ โปรแกรม Bootstrap แบบเต็มถูกเก็บไว้ในส่วนที่เรียกว่า Boot blocks ที่ตำแหน่งตายตัวบนดิสก์ ดิสก์ที่มีส่วนที่ใช้บูต คือ Boot disk หรือ System disk

ตัวควบคุมดิสก์จะอ่านคำสั่งใน Boot ROM จาก Boot block เข้าสู่หน่วยความจำ (ยังไม่มีตัวควบคุมอุปกรณ์ใด ๆ ถูกนำเข้าหน่วยความจำในตอนนี้) จากนั้นก็เริ่มทำงานตามคำสั่งดังกล่าว โปรแกรม Bootstrap แบบเต็มสามารถที่จะนำระบบปฏิบัติการจากตำแหน่งที่ไม่ตายตัวมาไว้บนดิสก์ได้ แล้วจึงเริ่มให้ระบบปฏิบัติการทำงาน ดังนั้นโปรแกรม Bootstrap แบบเต็มจึงมีขนาดเล็กได้ ตัวอย่างเช่น MS-DOS ใช้บล็อกขนาด 512 ไบต์ สำหรับการบูตเครื่อง (ดังภาพที่ 10.7)



ภาพที่ 10.7 โครงร่างของดิสก์ในระบบ MS-DOS
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.481)

10.3.3 บล็อกเสีย (Bad Block)

ในระบบดิสก์อย่างง่าย เช่น ดิสก์ที่มีตัวควบคุมแบบ IDE เราสามารถจัดการกับบล็อกเสียได้ (โดยผู้ใช้สั่งให้ทำ) เช่น คำสั่ง format ของ MS-DOS เป็นการจัดระเบียบทางตรรกะและจะทำการหาบล็อกเสียบนดิสก์ ถ้าหาเจอจะเขียนค่าพิเศษไว้ในช่องของ FAT ที่ตรงกันเพื่อบอกให้โปรแกรมย่อยต่าง ๆ ไม่ให้ใช้บล็อกนั้น ถ้าบล็อกเกิดเสียในระหว่างการทำงานปกติ โปรแกรมพิเศษ (เช่น Chkdsk) ต้องถูกสั่งให้ทำงานเพื่อค้นหาบล็อกเสียและล็อคบล็อกเหล่านั้นไว้ไม่ให้ใช้ ข้อมูลที่อยู่ในบล็อกเสียก็จะเสียไปโดยปริยาย

สำหรับดิสก์แบบ SCSI สามารถกู้บล็อกเสียคืนได้ โดยตัวควบคุมจะเก็บรายการของบล็อกเสียบนดิสก์ รายการนี้ถูกเก็บมาตั้งแต่ตอนจัดระเบียบดิสก์ระดับต่ำจากโรงงาน และถูกทำให้ทันสมัยตลอดเวลาของการใช้ดิสก์ การจัดระเบียบระดับต่ำจะจัดเซกเตอร์อะไหล่ที่ระบบปฏิบัติการมองไม่เห็นไว้ แล้วตัวควบคุมจะทำการแทนที่เซกเตอร์ทางตรรกะที่เสียแต่ละเซกเตอร์ด้วยเซกเตอร์อะไหล่ วิธีการนี้เรียกว่า การเก็บเซกเตอร์อะไหล่ (Sector sparing) หรือการเปลี่ยนที่ (Forwarding) ตัวอย่างของการเปลี่ยนแปลงเซกเตอร์เสียอาจเป็นดังนี้

1) ระบบปฏิบัติการพยายามอ่านบล็อกทางตรรกะที่ 87

2) ตัวควบคุมจะทำการคำนวณรหัสถูก-ผิด (ECC) และพบว่าเซกเตอร์นั้นเสีย มันจะรายงานการค้นพบนี้ไปสู่ระบบปฏิบัติการ

3) คราวหน้าเมื่อบูตระบบใหม่ คำสั่งพิเศษจะทำงานเพื่อบอกตัวควบคุม SCSI เพื่อแทนเซกเตอร์ที่เสียด้วยเซกเตอร์อะไหล่

4) หลังจากนั้น เมื่อระบบร้องขอบล็อกทางตรรกะที่ 87 การร้องขอนั้นจะถูกแปลไปยังตำแหน่งของเซกเตอร์ที่ถูกแทนที่โดยตัวควบคุม

จะเห็นได้ว่าการเปลี่ยนทิศทางโดยตัวควบคุมอาจจะทำให้การจัดตารางดิสก์ของระบบปฏิบัติการผิดพลาด ด้วยเหตุนี้ดิสก์ส่วนใหญ่ถูกจัดระเบียบ เพื่อเตรียมเซกเตอร์อะไหล่ไว้เล็กน้อยใน

แต่ละไซลินเดอร์ เมื่อบล็อกเสียถูกพบ ตัวควบคุมจะใช้เซกเตอร์อะไหล่จากไซลินเดอร์เดียวกันถ้าทำได้ อีกทางเลือกหนึ่งของการเก็บเซกเตอร์อะไหล่ ตัวควบคุมสามารถสั่งให้แทนที่บล็อกเสียโดยเซกเตอร์เลื่อน (Sector slipping) ตัวอย่างเช่น สมมติว่าบล็อกทางตรรกะตำแหน่งที่ 17 เสีย และเซกเตอร์อะไหล่แรกที่ใช้ได้อยู่ที่ 202 ดังนั้นเซกเตอร์เลื่อนจะจับตำแหน่งใหม่จาก 17 ไปถึง 202 โดยเคลื่อนตำแหน่งทั้งหมดลงหนึ่งจุด คือ เซกเตอร์ 202 จะถูกใช้เป็นอะไหล่ ดังนั้นเซกเตอร์ 201 จะถูกคัดลอกไป 202 และ 200 จะถูกย้ายไป 201 ไปเรื่อย ๆ จนกระทั่งเซกเตอร์ 18 ถูกคัดลอกลงเซกเตอร์ 19 การเลื่อนเซกเตอร์ด้วยวิธีนี้ ทำให้พื้นที่ของเซกเตอร์ 18 ว่าง ดังนั้นเซกเตอร์ 17 สามารถถูกจับคู่ไปตรงตำแหน่งนั้นแทนได้

การแทนที่ของบล็อกเสียโดยทั่วไปยังไม่เป็นกระบวนการอัตโนมัติเสียทีเดียว เพราะข้อมูลในบล็อกเสียมักจะเสียไป ดังนั้นแฟ้มข้อมูลที่ใช้งานบล็อกนั้นต้องถูกซ่อมแซม (เช่น การกู้คืนจาก Backup เทป) โดยต้องการให้คนเป็นผู้แก้ไข

10.4 การจัดการพื้นที่ที่ใช้ในการสับเปลี่ยน

หน่วยความจำเสมือนใช้พื้นที่ในดิสก์เสมือนส่วนขยายของหน่วยความจำหลัก เพราะว่าการเข้าถึงดิสก์ทำได้ช้ากว่าการเข้าถึงหน่วยความจำ ดังนั้นการใช้พื้นที่ที่ใช้ในการสับเปลี่ยน (Swap space) จะมีผลกระทบมากต่อสมรรถนะของระบบ เป้าหมายหลัก สำหรับการออกแบบและการนำพื้นที่ที่ใช้ในการสับเปลี่ยนไปใช้ คือ เตรียมอัตรางานที่ได้ต่อหนึ่งหน่วยเวลา (Throughput) ให้ดีที่สุดสำหรับระบบหน่วยความจำเสมือน

10.4.1 การใช้พื้นที่ที่ใช้ในการสับเปลี่ยน (Swap-Space Use)

พื้นที่ที่ใช้ในการสับเปลี่ยน ถูกใช้ได้หลายวิธีโดยหลายระบบปฏิบัติการที่ต่างกัน ขึ้นอยู่กับการใช้อัลกอริทึ่มในการจัดการหน่วยความจำ ตัวอย่างเช่น ระบบที่ใช้การสับเปลี่ยน (Swapping) อาจใช้พื้นที่ที่ใช้ในการสับเปลี่ยน (Swap space) เพื่อเก็บภาพของกระบวนการทั้งหมดไว้ รวมทั้งตอน (Segment) ของรหัสโปรแกรม (Code) และข้อมูลด้วยระบบการแบ่งเป็นหน้า (Paging system) อาจจะง่ายในการเก็บหน้าซึ่งถูกผลักออกจากหน่วยความจำหลัก จำนวนของพื้นที่ที่ใช้ในการสับเปลี่ยน ที่ต้องการของระบบขึ้นอยู่กับจำนวนของหน่วยความจำทางกายภาพ

10.4.2 ตำแหน่งของพื้นที่ที่ใช้ในการสับเปลี่ยน (Swap-Space Location)

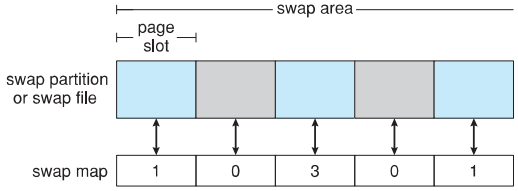
มีพื้นที่อยู่สองพื้นที่ที่เหมาะใช้ในการสับเปลี่ยน คือ พื้นที่ที่ใช้ในการสับเปลี่ยนสามารถถูกตัดออกจากระบบแฟ้มข้อมูลปกติ หรืออยู่ในส่วนของดิสก์ที่แยกออกมา (Separate disk partition) ถ้าการสับเปลี่ยนพื้นที่ทำได้ง่ายกับแฟ้มข้อมูลขนาดใหญ่ภายในระบบแฟ้มข้อมูล โปรแกรมย่อยระบบแฟ้มข้อมูลปกติ (Normal file-system routines) ควรจะถูกใช้เพื่อสร้าง เพื่อระบุชื่อ (ตั้งชื่อ) และเพื่อจัดสรรพื้นที่ให้กับแฟ้มนั้น ๆ วิธีนี้นำไปใช้จริงได้ง่ายแต่ไม่มีประสิทธิภาพ การดำเนินการสำรวจโครงสร้างของไดเรกทอรี่ และโครงสร้างของข้อมูลการจัดสรรดิสก์ต้องใช้เวลา การสูญเสียพื้นที่ย่อยภายนอกทำให้เพิ่มเวลาในการสับเปลี่ยนขึ้นมาก เพราะต้องค้นหาหลายครั้งในช่วงของการอ่านหรือเขียนภาพของโพรเซส สามารถปรับปรุงสมรรถนะได้โดยการเก็บตำแหน่งของบล็อกในหน่วยความจำทางกายภาพและโดยการใช้เครื่องมือ

พิเศษเพื่อจัดสรรบล็อกที่ติด ๆ กัน ทางกายภาพให้กับการสับเปลี่ยนแฟ้มข้อมูล (Swap file) แต่ต้นทุนในการท่องไปในโครงสร้างข้อมูลของระบบแฟ้มข้อมูลยังคงมีอยู่

อีกวิธีหนึ่ง คือ ใช้ส่วนของดิสก์ที่แยกออกมา เพื่อสร้างพื้นที่ ๆ ใช้ในการสับเปลี่ยน จึงไม่มีระบบแฟ้มข้อมูลหรือโครงสร้างของไดเรกทอรี่บนพื้นที่นี้ ตัวจัดการหน่วยเก็บพื้นที่ ๆ จะใช้ในการสับเปลี่ยนจะถูกใช้ในการจัดสรรหรือยึดบล็อกคืนมาสู่ระบบ ตัวจัดการนี้ใช้อัลกอริทึ่มที่ดีที่สุดสำหรับความเร็ว เพื่อให้ได้ประสิทธิภาพการสูญเสียพื้นที่ย่อยภายในอาจจะเพิ่มขึ้น แต่ก็พอรับได้เพราะข้อมูลในพื้นที่ ๆ ใช้ในการสับเปลี่ยนมักจะอยู่ในช่วงสั้นกว่าแฟ้มข้อมูลในระบบแฟ้มข้อมูล และพื้นที่ ๆ ใช้ในการสับเปลี่ยนมักจะถูกเข้าถึงอยู่บ่อยครั้ง ข้อเสียของวิธีนี้คือสร้างพื้นที่ ๆ ใช้ในการสับเปลี่ยนอย่างคงที่เอาไว้มากในระหว่าง การแบ่งส่วนของดิสก์ การเพิ่มพื้นที่ ๆ ใช้ในการสับเปลี่ยนให้มากขึ้นทำได้โดยการแบ่งส่วนของดิสก์ใหม่อีกครั้ง (ซึ่งเกี่ยวกับการย้ายหรือการทำลาย และการกู้คืนส่วนของระบบแฟ้มข้อมูลอื่นจากการ Backup) หรือโดยการเพิ่มพื้นที่ที่ใช้ในการสับเปลี่ยนจากพื้นที่อื่น ๆ

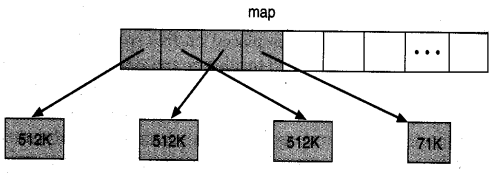
10.4.3 การจัดการพื้นที่ที่ใช้ในการสับเปลี่ยน (Swap-Space Management)

พื้นที่ที่ใช้ในการสับเปลี่ยนถูกจัดสรรให้แก่โพรเซส เมื่อโพรเซสเริ่มต้น พื้นที่จะถูกจัดสรรให้เพื่อเก็บโปรแกรมอย่างเพียงพอ โดยเก็บหน้าของข้อความ (Text page) หรือตอนของข้อความ (Text segment) และตอนของข้อมูล (Data segment) ของโพรเซส ก่อนการจัดสรรพื้นที่ต้องการทั้งหมด ด้วยวิธีนี้เป็นการป้องกันโพรเซสจากการทำงานนอกพื้นที่ ๆ ใช้ในการสับเปลี่ยนในขณะที่กำลังทำงาน (Execute) เมื่อโพรเซสเริ่มต้นทำงาน ข้อความจะถูกแบ่งเป็นหน้าจากระบบแฟ้มข้อมูล หน้าเหล่านี้จะถูกเขียนเมื่อจำเป็นและถูกอ่านย้อนหลังจากตรงนั้น ดังนั้นแฟ้มข้อมูลจะถูกพิจารณาเพียงครั้งเดียว



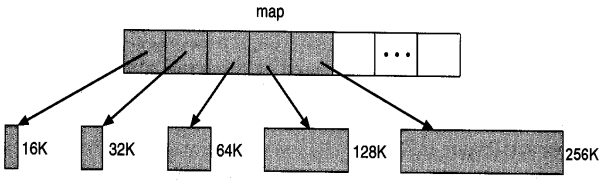
ภาพที่ 10.8 แสดงแผนที่ของการสับเปลี่ยนตอนของข้อความ
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.484)

สำหรับในแต่ละหน้าของข้อความหน้าจากตอนของข้อมูลจะถูกอ่านจากระบบแฟ้มข้อมูล หรือถูกสร้างขึ้น (ถ้ายังไม่มี) และถูกเขียนที่พื้นที่ ๆ ใช้ในการสับเปลี่ยน และย้อนกลับไปยังหน้าที่ต้องการ วิธีที่ดีที่สุดวิธีหนึ่ง (เช่น เมื่อผู้ใช้ 2 คน ทำงาน Editor ตัวเดียวกัน) คือ โพรเซสในหน้าของข้อความที่เหมือนกันควรร่วมกันใช้หน้าเหล่านี้ ทั้งในหน่วยความจำทางตรรกะและในพื้นที่ ๆ ใช้ในการสับเปลี่ยนแก่น (Kernel) จะใช้แผนที่ในการสับเปลี่ยน (Swap map) เพื่อตามรอยพื้นที่ที่ใช้ในการสับเปลี่ยนตอนของข้อความจะมีขนาดคงที่ ดังนั้นพื้นที่ที่ใช้ในการสับเปลี่ยนจะถูกจัดสรรให้ก้อนละ 512K ยกเว้นก้อนสุดท้าย จะเก็บส่วนที่เหลือ (อาจไม่ถึง 512K) ดังแสดงในภาพที่ 10.9



ภาพที่ 10.9 แสดงแผนที่ของการสับเปลี่ยนตอนของข้อความ

แผนที่การสับเปลี่ยนตอนของข้อมูลค่อนข้างซับซ้อน เพราะตอนของข้อมูลสามารถเพิ่มขึ้นได้ตลอดเวลา แผนที่ดังกล่าวมีขนาดคงที่แต่เก็บตำแหน่งที่ใช้ในการสับเปลี่ยนที่เป็นบล็อกที่มีขนาดไม่คงที่ ให้ดัชนี i คือ บล็อกที่แผนที่ที่ใช้ในการสับเปลี่ยนชี้อยู่มีขนาด 2^i x 16 K โดยมากที่สุดได้ 2 เมกกะไบต์ โครงสร้างของข้อมูลนี้แสดงได้ดังภาพที่ 10.10



ภาพที่ 10.10 แสดงแผนที่ที่ใช้ในการสับเปลี่ยนตอนของข้อมูล

เมื่อโพรเซสพยายามเพิ่มตอนของข้อมูลก่อนบล็อกสุดท้ายในพื้นที่ที่ใช้ในการสับเปลี่ยน ระบบปฏิบัติการจะจัดบล็อกให้อีกบล็อกหนึ่ง ที่มีขนาดเป็น 2 เท่าของบล็อกก่อนหน้านี้ ผลลัพธ์ของวิธีการนี้ โพรเซสเล็กก็จะใช้บล็อกเล็กตามไปด้วย นั่นเท่ากับเป็นการลดการเสียพื้นที่ บล็อกของโพรเซสขนาดใหญ่ก็จะหาพบได้รวดเร็ว ดังนั้นแผนที่ที่ใช้ในการสับเปลี่ยนจึงมีขนาดเล็กตามไปด้วย ขนาดของบล็อกที่เล็กที่สุดและใหญ่ที่สุดไม่คงที่ และสามารถเปลี่ยนแปลงได้โดยการบูตเครื่องใหม่

10.5 ความน่าเชื่อถือของดิสก์

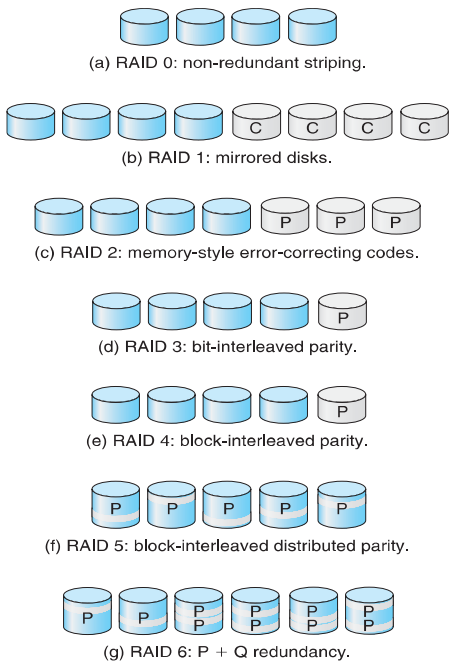
ในระบบคอมพิวเตอร์ดิสก์ยังคงมีอัตราความผิดพลาดสูง และความผิดพลาดเหล่านั้นทำให้ข้อมูลสูญหายและเสียเวลาการกู้คืน อาจต้องใช้เวลาหลายชั่วโมง ในการปรับปรุงเทคนิคในการใช้ดิสก์หลาย ๆ ทางเกี่ยวข้องกับการใช้ดิสก์หลายตัวทำงานประสานกันเพื่อเป็นการปรับปรุงในด้านความเร็ว การแกะดิสก์ (Disk striping) เราใช้กลุ่มของดิสก์เป็นเหมือนหน่วยเก็บข้อมูล 1 หน่วย บล็อกของข้อมูลแต่ละบล็อกถูกแบ่งเป็นบล็อกย่อยหลาย ๆ บล็อก ซึ่งบล็อกย่อยจะถูกเก็บในแต่ละดิสก์ เวลาที่ต้องการเพื่อใช้ในการย้ายบล็อกไปสู่หน่วยความจำจะลดลงอย่างเหลือเชื่อ เพราะดิสก์ทุกตัวโอนย้ายบล็อกย่อยของพวกมันแบบขนาน ถ้าดิสก์มีการหมุนพร้อมกัน (Synchronized) สมรรถนะจะดีขึ้น เพราะดิสก์ทั้งหมดจะพร้อมที่จะโอนย้ายบล็อกย่อยของมันในเวลาเดียวกัน

การจัดการนี้มักเรียกว่า อาร์เรย์ที่เหลือเฟือของดิสก์อิสระ (Redundant array of independent disk : RAID) วิธีนี้เป็นการเพิ่มความน่าเชื่อถือของระบบหน่วยเก็บข้อมูลโดยการเก็บข้อมูลที่เหลือเฟือ การจัดการ RAID ที่ง่ายที่สุด เรียกว่า การสำรองข้อมูล (Mirroring หรือ shadowing) โดยเก็บสำเนาของแต่ละดิสก์ วิธีนี้คุ้มค่าเพราะดิสก์หลายตัวถูกใช้เก็บข้อมูลเดียวกัน 2 ครั้ง แต่ในทางกลับกันวิธีนี้จะเร็วเป็น 2 เท่าเมื่อเป็นการอ่าน เพราะครึ่งหนึ่งของการร้องขอที่จะอ่านจะถูกส่งไปยังแต่ละดิสก์

การจัดการ RAID เรียกว่า ความตรวจสอบบล็อกขาออก (Block interleaved parity) โดยใช้พื้นที่ของดิสก์เพียงเล็กน้อยในการเก็บบล็อกตรวจสอบ (Parity block) ตัวอย่างเช่น สมมติว่ามีดิสก์ 9 ตัวในอาร์เรย์ โดยบล็อกของข้อมูลทุก ๆ 8 ตัว ที่ถูกเก็บในอาร์เรย์จะมีอยู่ 1 บล็อกที่เป็นบล็อกตรวจสอบที่ถูกเก็บไปด้วย ตำแหน่งของบิตแต่ละบิตในบล็อกตรวจสอบนี้ควรจะเก็บค่าสำหรับตำแหน่งของบิตที่ตรงกันในแต่ละบล็อกของข้อมูลทั้ง 8 เหมือนกับการคำนวณบิตตรวจสอบที่ 9 ในหน่วยความจำหลักสำหรับแต่ละ 8 บิต-ไบต์ ถ้าบล็อกของดิสก์หนึ่งเสีย บิตของข้อมูลทั้งหมดต้องถูกลบ แต่ยังสามารถคำนวณใหม่จากบล็อกของข้อมูลบล็อกอื่นบวกกับบล็อกตรวจสอบ ดังนั้นดิสก์เสียตัวเดียวจึงไม่ทำให้ข้อมูลเสียหาย

10.5.1 ระดับของ RAID

Data striping คือการแบ่งข้อมูลออกเป็นส่วน ๆ แล้วนำแต่ละส่วนไปเก็บในฮาร์ดดิสก์แต่ละตัว การทำ Striping นี้จะช่วยให้การอ่าน หรือเขียนข้อมูลใน Disk array มีประสิทธิภาพมากขึ้น เพราะแต่ละไฟล์จะถูกแบ่งเป็นส่วน ๆ กระจายไปเก็บในส่วนที่ต่างกันของฮาร์ดดิสก์หลายตัว โดยฮาร์ดดิสก์เหล่านั้นทำงานไปด้วยกันแบบขนาน (Parallel) จึงทำให้การเข้าถึงข้อมูลนั้นเร็วกว่าฮาร์ดดิสก์แบบตัวเดียวอย่างแน่นอน เราสามารถอธิบาย Raid ประเภทต่าง ๆ ดังนี้ (ภาพที่ 10.11 แสดงระดับของ RAID)



ภาพที่ 10.11 แสดง RAID ระดับต่าง ๆ
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.487)

1) RAID 0 คือการเอาฮาร์ดดิสก์มากกว่า 1 ตัวมาต่อร่วมกันในลักษณะ Non-redundant ซึ่ง RAID 0 นี้มีจุดประสงค์เพื่อที่จะเพิ่มความเร็วในการอ่าน/เขียนข้อมูลฮาร์ดดิสก์โดยตรง ไม่มีการเก็บข้อมูลสำรอง ดังนั้นถ้าฮาร์ดดิสก์ตัวใดตัวหนึ่งเกิดเสียหาย ก็จะส่งผลให้ข้อมูลทั้งหมดไม่สามารถใช้งานได้ทันทีดังแสดงภาพที่ 10.11(a) จะเห็นว่าข้อมูลจะถูกแบ่งไปเก็บที่ฮาร์ดดิสก์ทั้ง 3 ตัว จุดเด่นของ RAID 0 คือความเร็วในการเข้าถึงข้อมูล แต่ข้อเสียก็คือหากฮาร์ดดิสก์ตัวใดตัวหนึ่งเสียหาย จะส่งผลกับข้อมูลทั้งระบบทันที

2) RAID 1 มีอีกชื่อหนึ่งว่า Disk mirroring จะประกอบไปด้วยฮาร์ดดิสก์ 2 ตัวที่เก็บข้อมูลเหมือนกันทุกประการ เสมือนการสำรองข้อมูล หากฮาร์ดดิสก์ตัวใดตัวหนึ่งเกิดเสียหาย ระบบยังสามารถดึงข้อมูลจากฮาร์ดดิสก์อีกตัวหนึ่งมาใช้งานได้ตามปกติดังแสดงภาพที่ 10.11(b) จุดเด่นของ RAID 1 คือความปลอดภัยของข้อมูล ไม่เน้นเรื่องประสิทธิภาพและความเร็วเหมือนอย่าง RAID 0 แม้ว่าประสิทธิภาพในการอ่านข้อมูลของ RAID 1 จะสูงขึ้นก็ตาม

3) RAID 2 ข้อมูลทั้งหมดจะถูกตัดแบ่งเพื่อจัดเก็บลงฮาร์ดดิสก์แต่ละตัวใน Disk array โดยจะมีฮาร์ดดิสก์ตัวหนึ่งเก็บข้อมูลที่ใช้ตรวจสอบและแก้ไขข้อผิดพลาด (Error checking and correcting : ECC) ซึ่งเป็นการลดเปอร์เซ็นต์ที่ข้อมูลจะเสียหายหรือสูญเสียไป เมื่อมีการส่งข้อมูลไปบันทึกใน Disk array จะเห็นได้ว่ามีฮาร์ดดิสก์ที่เอาไว้เก็บค่า ECC โดยเฉพาะ ถ้าเกิดการปรากฏว่าฮาร์ดดิสก์ตัวใดตัวหนึ่งเสียหาย ระบบก็จะสามารถสร้างข้อมูลทั้งหมดในฮาร์ดดิสก์ตัวนั้นขึ้นมาได้ใหม่ โดยอาศัยข้อมูลจากฮาร์ดดิสก์ ตัวอื่น ๆ และจากค่า ECC ที่เก็บเอาไว้ ซึ่งการทำ ECC นี้ส่งผลให้ฮาร์ดดิสก์ ทั้งระบบต้องทำงานค่อนข้างมากทีเดียว และ RAID 2 นั้นจะเห็นได้ว่าต้องใช้ฮาร์ดดิสก์จำนวนมากในการเก็บค่า ECC ซึ่งทำให้ค่อนข้างสิ้นเปลือง ดังแสดงภาพที่ 10.11(c)

4) RAID 3 มีลักษณะที่คล้ายกับ RAID 2 แต่แทนที่จะตัดแบ่งข้อมูลในระดับบิตเหมือน RAID 2 ก็จะตัดเก็บข้อมูลในระดับ Byte แทนและการตรวจสอบและแก้ไขข้อผิดพลาดของข้อมูล จะใช้ Parity แทนที่จะเป็น ECC ทำให้ RAID 3 มีความสามารถในการอ่านและเขียนข้อมูลได้อย่างรวดเร็ว เพราะมีการต่อฮาร์ดดิสก์แต่ละตัวแบบ stripe และใช้ฮาร์ดดิสก์ที่เก็บ Parity เพียงแค่ตัวเดียวเท่านั้นดังแสดงภาพที่ 10.11(d) แต่ถ้านำ RAID 3 ไปใช้ในงานที่มีการส่งผ่านข้อมูลในจำนวนที่น้อย ๆ ซึ่ง RAID 3 ต้องกระจายข้อมูลไปทั่วทั้งฮาร์ดดิสก์จะทำให้เกิดปัญหาที่เรียกว่า คอขวด ขึ้นกับฮาร์ดดิสก์ที่เก็บ Parity ไม่ว่าข้อมูลจะมีขนาดใหญ่ขนาดไหน RAID 3 ต้องเสียเวลาไปสร้างส่วน Parity ทั้งสิ้น ยิ่งข้อมูลมีขนาดเล็ก ๆ แต่ Parity ต้องสร้างขึ้นตลอด ทำให้ข้อมูลถูกจัดเก็บเสร็จก่อนการสร้าง Parity ทั้งระบบต้องมารอให้สร้าง Parity เสร็จก่อน จึงจะทำงานต่อไปได้นั่นเอง RAID 3 เหมาะสำหรับใช้ในงานที่มีการส่งข้อมูลจำนวนมาก ๆ เช่น งานตัดต่อ Video เป็นต้น

5) RAID 4 มีลักษณะโดยรวมเหมือนกับ RAID 3 ทุกประการ ยกเว้นเรื่องการตัดแบ่งข้อมูลที่ทำในระดับ Block แทนที่จะเป็น Bit หรือ Byte ดังแสดงภาพที่ 10.11(e) ซึ่งทำให้การอ่านข้อมูลแบบ Random ทำได้รวดเร็วกว่า อย่างไรก็ตามแต่ยังคงมีปัญหาคอขวดที่กล่าวใน RAID 3

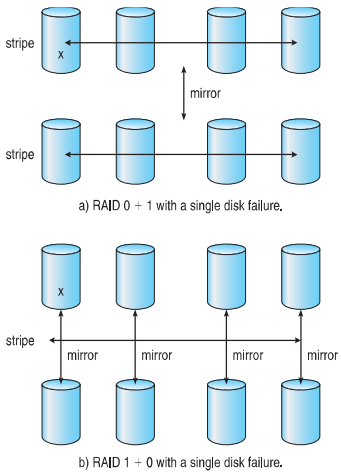
6) RAID 5 มีการตัดแบ่งข้อมูลในระดับ Block เช่นเดียวกับ RAID 4 แต่จะไม่ทำการแยกฮาร์ดดิสก์ตัวใดตัวหนึ่งเพื่อเก็บ Parity ในการเก็บ Parity ของ RAID 5 นั้น จะทำการกระจาย Parity ไปยังฮาร์ดดิสก์ทุกตัวดังแสดงในภาพที่ 10.11(f) โดยปะปนไปกับข้อมูลปกติ จึงช่วยลดปัญหาคอขวด ซึ่งเป็นปัญหาที่สำคัญใน RAID 3 และ RAID 4 คุณสมบัติอีกอันหนึ่งที่น่าสนใจอย่างมากของ RAID 5 คือ

เทคโนโลยี Hot Swap คือ สามารถทำการเปลี่ยนฮาร์ดดิสก์ในกรณีที่เกิดปัญหาได้ ในขณะที่ระบบยังทำงานอยู่ เหมาะสำหรับงาน Server ต่าง ๆ ที่ต้องทำงานต่อเนื่อง

10.5.2 การนำ RAID มาใช้ร่วมกัน

การใช้ RAID ในระดับเดียวนั้นอาจจะไม่สามารถติดตั้งระบบปฏิบัติการต่าง ๆ ได้ครบตามความต้องการ ดังนั้น เพื่อให้ได้เครื่องมือในการทำงานครบถ้วนนั้น การนำข้อดีของ RAID ในแต่ละระดับมารวมกันเพื่อใช้งาน ซึ่งนั่นหมายถึง ถ้าเราสามารถรวมข้อดีของ RAID ในแต่ละลำดับขั้นไว้ด้วยกัน อาจทำให้การทำงานมีประสิทธิภาพมากขึ้นก็เป็นได้ และเป็นเหตุผลของการกำเนิดระดับตัวใหม่ตัวนี้ของ RAID การใช้งานของ RAID ในระดับใหม่นั้นเกิดจากการนำเอาข้อดีของ RAID ในแต่ละระดับมารวมกันซึ่งประสิทธิภาพของการทำงานย่อมเพิ่มขึ้น

นอกจากนี้ลองมาดูการรวมตัวของ RAID ในระดับต่าง ๆ RAID 0 เป็นระดับที่มีประสิทธิภาพการทำงานที่ดีที่สุด และเป็นตัวเดียวที่สามารถนำมาร่วมตัวกับระดับอื่นได้มากที่สุด เพราะไม่ใช่ว่าทุกระดับของ RAID จะสามารถนำมารวมกันได้ โดยทั่วไประดับที่นิยมนำมารวมกันของ RAID ก็จะมี RAID 0+1 และ RAID 1+0 ซึ่งความแตกต่างระหว่าง 0+1 และ 1+0 จะเห็นได้จากชื่อที่เรียกใช้งาน การเลือกใช้ตัวใดนั้นคงต้องเลือกเอาจาก RAID ตัวที่มีข้อผิดพลาดเกิดขึ้นน้อยที่สุดเมื่อนำไปใช้งานจริงการใช้งานของ RAID ทั้งสองตัวนี้ ต้องเตรียมฮาร์ดไดรฟ์เอาไว้ 4 ตัว (แสดงในภาพที่ 10.12 RAID 0 + 1 และ 1 + 0)



ภาพที่ 10.12 แสดง RAID 0 + 1 และ 1 + 0
ที่มา: Abraham, S. Peter, B. G., & Greg, G. Operating system concepts 9th ed. (2013, p.490)

จากนี้จะเริ่มมาดูการทำงานของ RAID 0+1 กันก่อน การรวมกันระหว่าง RAID 0 เพื่อให้ได้ประสิทธิภาพการทำงาน และ RAID 1 เพื่อให้เกิดความผิดพลาดในการปฏิบัติการให้น้อยที่สุดนั้น สำหรับระดับที่มีการรวมนี้ไปแล้ว เมื่อการแยกส่วนจัดเก็บข้อมูลถูกนำมาไว้ในส่วนข้อมูลสำรอง ซึ่งหมายถึงต้องมีฮาร์ดไดรฟ์เพื่อปฏิบัติการนี้ถึง 8 ตัว จึงจะสามารถแยกฮาร์ดไดรฟ์แต่ละตัวนั้นเป็น 2 หมวด ในทุก ๆ 4 ไดรฟ์ และนำ RAID 0 มาประยุกต์ใช้งานร่วมกันไป ซึ่งจะทำให้มีการแยกเก็บข้อมูลไว้ด้วยกัน 2 หมวด เมื่อประยุกต์ RAID 1 เป็นการแยกเก็บข้อมูลแบบ 2 หมวดแล้ว และมีการสำรองข้อมูลไว้ 1 หมวดไว้ในส่วนที่

เหลือ ถ้าฮาร์ดไดรฟ์ตัวที่ถูกแยกส่วนไว้ เกิดความเสียหายหรือสูญหาย และหมวดอื่นที่แยกเก็บข้อมูลไว้ไม่ได้มารองรับข้อมูลในส่วนนี้ การเก็บข้อมูลสำรองเอาไว้จะทำให้ความเสียหายนั้นน้อยลง RAID 1+0 เป็นการประยุกต์ใช้โดยการนำ RAID 1 มาใช้ก่อน จากนั้นจึงนำ RAID 0 มาไว้ในไดรฟ์ ในการที่จะประยุกต์ใช้ RAID 1 นั้นผู้ใช้งานต้องแบ่งไดรฟ์เอาไว้ด้วยกัน 8 ไดรฟ์ทั้งหมด 4 ชุด ซึ่งแต่ละชุดจะมีทั้งหมด 2 ไดรฟ์ โดยในแต่ละชุดนั้นก็คือการสำรองข้อมูลและมีการประยุกต์ข้อมูลเพื่อนำมาใช้งาน จากนั้น จึงประยุกต์ RAID 0 มารวมกันในส่วนนี้จะมีการจำแนกข้อมูลเอาไว้ 4 ชุดด้วยกัน ซึ่งโดยหลัก ๆ แล้วจะมีการจำแนกข้อมูลตามจำนวนของชุดข้อมูลสำรอง

การรวมกันเช่นนี้ จะช่วยลดความผิดพลาดที่อาจเกิดขึ้นได้ดีกว่าการรวมตัวของ RAID 0+1 ซึ่งการทำงานเช่นนี้สำหรับหนึ่งไดรฟ์ภายในชุดสำรองข้อมูลจะมีการปฏิบัติงานได้ดี และส่วนที่ถูกจัดหมวดนั้นยังคงทำหน้าที่ได้ดีเช่นเดิม ซึ่งในทางปฏิบัติแล้วคุณสามารถเก็บข้อมูลได้ถึงครึ่งหนึ่งของไดรฟ์ หากว่าไดรฟ์นั้นเกิดความผิดพลาดในการใช้งาน วิธีการเช่นนี้จะทำให้คุณไม่ต้องเสียข้อมูลไปทั้งหมดเพราะซึ่งวิธีการนี้จะไม่สามารถนำไปใช้ได้กับระดับ RAID 0+1 เพราะมีการแบ่งไดรฟ์ออกเป็นสองส่วนเท่านั้น ความนิยมในตัว RAID 0+1 และ 1+0 นั้นค่อนข้างที่จะนิยมใช้พอกันซึ่งขึ้นอยู่กับความต้องการใช้งานของผู้ใช้มากกว่า เมื่อราคาฮาร์ดไดรฟ์ถูกลงและมีจำนวนเพิ่มขึ้น อาจจะเหลือเการใช้ฮาร์ดไดรฟ์เพียง 4 ตัว

ในการปฏิบัติงานจึงไม่ใช่เรื่องน่าแปลกแต่อย่างใดในปัจจุบัน อย่างไรก็ตามผู้ใช้งานยังคงต้องเกิดความเสียหายในการจัดเก็บถึง 50 เปอร์เซ็นต์ของพื้นที่ใช้งานอยู่ดีเมื่อคุณต้องทำงาน โดยมี การสำรองข้อมูลบริษัทผู้ผลิตระบบปฏิบัติการ และเซิร์ฟเวอร์นั้นมักเสียพื้นที่ส่วนหนึ่งบนเซิร์ฟเวอร์เพื่อทำให้การจัดเก็บข้อมูลเกิดประสิทธิภาพมากที่สุด และเกิดความเสียหายน้อยที่สุด ซึ่งการรวมตัวของ RAID ในระดับต่าง ๆ จะหมายรวมไปถึงการรวมตัวของ RAID ในระดับอื่นด้วย เช่น RAID 0+3,3+0.0+5.5+0,1+5 และ 5+1 ซึ่งระดับต่าง ๆ เหล่านี้เมื่อถูกนำมาใช้งานก็ต้องเป็นการใช้งานร่วมกับฮาร์ดแวร์ที่มีราคาแพงด้วย และใช่ว่าจะมีการนำระดับดังกล่าวมาใช้งานได้ทั้งหมด

10.5.3 ประโยชน์ของ RAID

กล่าวได้ว่าประสิทธิภาพของ RAID จะขึ้นอยู่กับแอพพลิเคชันที่ใช้กับระดับของ RAID ด้วย แต่โดยทั่วไปแล้ว RAID จะถูกใช้เพื่อป้องกันความผิดพลาดในการเก็บข้อมูล และเพิ่มระดับการจัดเก็บด้วยที่สำคัญป้องกันการจัดเก็บข้อมูลในกรณีที่ฮาร์ดไดรฟ์เกิดทำงานผิดพลาด จะเห็นว่าคุณสมบัตินี้สำคัญต่อการบริหารข้อมูลที่สำคัญขององค์กร หรือแม้กระทั่งส่วนบุคคลด้วย นอกจากนั้นจะทำให้ทุกคนไม่ต้องกังวลถึงการสูญเสียพื้นที่หลายกิกะไบต์ให้กับไฟล์ประเภทต่าง ๆ รวมทั้งการกำจัดข้อมูลที่ไม่จำเป็นออกไปของ RAID ทำให้ระบบฐานข้อมูลของคุณนั้นดีขึ้น RAID ระบบเดียวที่ไม่มีคุณสมบัติกำจัดข้อมูลที่ไม่จำเป็นรวมทั้งระบบป้องกันข้อมูลผิดพลาดก็คือ RAID 0

นอกจากนี้ RAID ยังสามารถขยายหน่วยความจำ โดยการผนวกหลาย ๆ ไดรฟ์เข้าด้วยกัน แต่ประสิทธิผลนั้น ขึ้นอยู่กับระดับของ RAID ที่ใช้ด้วย ซึ่งโดยปกติแล้วระดับที่ใช้สำหรับการสำรองการเก็บข้อมูลนั้น ต้องการหน่วยความจำสูงขึ้นเป็นสองเท่า และเหตุผลสุดท้ายที่ควรมาใช้ RAID ก็คือการเพิ่มประสิทธิภาพในการเก็บข้อมูลให้ดีขึ้น เพราะในแต่ละระดับของ RAID ที่เลือกใช้ ความแตกต่างของประสิทธิภาพก็จะแตกต่างกันไป โดยเฉพาะการใช้งานกับแอพพลิเคชันที่ต้องการความเร็วสูงนั้น RAID น่าจะเหมาะสมที่สุด

10.6 การใช้งานหน่วยเก็บข้อมูลชนิดคงที่

โดยนิยามแล้ว ข้อมูลที่อยู่ในหน่วยเก็บข้อมูลชนิดคงที่จะไม่มีทางสูญหายไปไหน การนำหน่วยเก็บข้อมูลไปใช้ เราจำเป็นต้องคัดลอกข้อมูลที่ต้องการไปไว้ยังอุปกรณ์ที่ใช้เก็บข้อมูลหลาย ๆ ที่ (อาจเก็บในดิสก์ด้วย ในเทปด้วย) ด้วยโหมดที่เป็นอิสระจากความล้มเหลว เราต้องการวิธีที่จะทำให้ข้อมูลทันสมัย (Update) ที่รับประกันได้ว่าความผิดพลาดในระหว่างการทำข้อมูลให้ทันสมัยจะไม่ทำให้ข้อมูลเสียหาย และเมื่อเราทำการกู้คืนจากที่ผิดพลาด เราต้องสามารถทำให้ข้อมูลทั้งหมดมีค่าที่ถูกต้อง ถ้าเกิดมีการล้มเหลวอีกในระหว่างการกู้คืนจะต้องทำอย่างไร ดังนั้นดิสก์จะทำงานจนได้ผลลัพธ์หนึ่งในสามข้อนี้ คือ

1) สำเร็จอย่างสมบูรณ์แบบ (Successful completion) ข้อมูลถูกเขียนลงดิสก์ได้อย่างถูกต้อง

2) ล้มเหลวบางส่วน (Partial failure) ความล้มเหลวเกิดขึ้นระหว่างการโอนย้ายข้อมูล ดังนั้นเซกเตอร์บางส่วนจะถูกเขียนด้วยข้อมูลใหม่ และเซกเตอร์ที่ถูกเขียนในระหว่างเกิดการล้มเหลวอาจจะผิดพลาด

3) ล้มเหลวทั้งหมด (Total failure) ความล้มเหลวเกิดขึ้น ก่อนที่ดิสก์จะเริ่มเขียน ดังนั้นค่าของข้อมูลบนดิสก์ก็จะเหมือนเดิมก่อนที่จะเกิดการเขียนใด ๆ

ถ้าความล้มเหลวเกิดในระหว่างการเขียนบล็อก เมื่อระบบตรวจพบแล้วจะทำการกู้บล็อกนั้นคืน ระบบต้องเก็บบล็อกทางกายภาพ 2 บล็อก ไว้ให้บล็อกทางตรรกะแต่ละบล็อก ผลลัพธ์ที่ได้มีดังนี้

1) เขียนข้อมูลลงบนบล็อกทางกายภาพบล็อกแรก
2) เมื่อเขียนบล็อกแรกเรียบร้อย ก็เขียนข้อมูลเดียวกันลงยังบล็อกทางกายภาพอีกบล็อก
3) ประกาศว่าการทำงานเสร็จสิ้นหลังจากการเขียนบล็อกที่สองสำเร็จเท่านั้น

ในระหว่างการกู้ระบบจากความล้มเหลว บล็อกทางกายภาพแต่ละคู่จะถูกทดสอบ ถ้าทั้ง 2 บล็อกเหมือนกันและไม่มีการพบข้อผิดพลาดก็ไม่ต้องทำอะไร ถ้ามีบล็อกหนึ่งพบว่ามีข้อผิดพลาดจะทำการแทนที่เนื้อหาในบล็อกนั้นด้วยข้อมูลของอีกบล็อกหนึ่ง ถ้าทั้งสองบล็อกไม่พบข้อผิดพลาดแต่เนื้อหาต่างกัน จะเอาเนื้อหาของบล็อกที่สองเข้าไปเก็บแทนที่ในบล็อกที่หนึ่ง กระบวนการในการกู้คืนนี้ประกันได้ว่าการเขียนข้อมูลลงในหน่วยเก็บข้อมูลต้องเสร็จสมบูรณ์หรือไม่ผลลัพธ์ที่ได้ต้องไม่เปลี่ยนแปลง

10.7 สรุป

ดิสก์ไดรฟ์เป็นอุปกรณ์ที่เป็นประเภทหน่วยเก็บข้อมูลสำรอง (Secondary-storage I/O) สำหรับเครื่องคอมพิวเตอร์ โดยปกติอุปกรณ์หน่วยเก็บข้อมูลสำรอง จะเป็น Magnetic disk หรือไม่ก็ Magnetic tape ดิสก์ไดรฟ์สมัยใหม่จะถูกสร้างขึ้นด้วย One-dimensional array of logical disk block ขนาดใหญ่ ดิสก์สามารถติดเข้ากับระบบคอมพิวเตอร์ได้โดย 1 ใน 2 ทาง เชื่อมต่ออุปกรณ์เข้ากับคอมพิวเตอร์โดยตรง ผ่านทางช่องพอร์ตอินพุตหรือเอาท์พุต ความต้องการข้อมูลสำหรับการอินพุตและเอาท์พุตของข้อมูลในดิกส์ จะถูกกำหนดโดยระบบแฟ้มข้อมูลและระบบความจำเสมือน ความต้องการแต่ละอย่างจะระบุที่อยู่บนดิสก์เพื่อใช้เป็นตัวอ้างอิง ซึ่งอยู่ในรูปแบบของ Logical block number disk-

scheduling algorithm สามารถที่จะพัฒนา Effective bandwidth, ระยะเวลาตอบสนองเฉลี่ย (Average response time) และความแตกต่างของเวลาในการตอบสนอง (Variance in response time) ให้ดีขึ้นได้

ในระบบคอมพิวเตอร์ความน่าเชื่อถือในการเก็บรักษาข้อมูลในดิกส์ ให้มีความถูกต้องแม่นยำ และการกู้คืนระบบมีความสำคัญอย่างยิ่ง มีการปรับปรุงเทคนิคในการใช้ดิสก์หลาย ๆ ทางเกี่ยวข้องกับการใช้ดิสก์หลายตัวทำงานประสานกันเพื่อเป็นการปรับปรุงในด้านความเร็ว มีการใช้กลุ่มของดิสก์เป็นเหมือนหน่วยเก็บข้อมูล 1 หน่วย Data Striping คือการแบ่งข้อมูลออกเป็นส่วน ๆ แล้วนำแต่ละส่วนไปเก็บในดิกส์แต่ละตัว การทำ Striping นี้จะช่วยให้การอ่าน หรือเขียนข้อมูลในดิกส์อาเรย์มีประสิทธิภาพมากขึ้น เพราะแต่ละแฟ้มข้อมูลจะถูกแบ่งเป็นส่วน ๆ กระจายไปเก็บในส่วนที่ต่างกันของดิกส์หลายตัว โดยดิกส์เหล่านั้นทำงานไปด้วยกันแบบขนาน (Parallel) จึงทำให้การเข้าถึงข้อมูลนั้นเร็วกว่าดิกส์แบบตัวเดียว

เทคโนโลยี RAID คือ การนำเอาดิกส์ตั้งแต่ 2 ตัวขึ้นไปมาทำงานร่วมกันเสมือนเป็นดิกส์ ตัวเดียวที่มีประสิทธิภาพสูงขึ้น หรือมีโอกาสที่จะสูญเสียข้อมูลน้อยลงในกรณีที่เกิดความผิดพลาดของฮาร์ดแวร์ กลุ่มของดิกส์ที่เอามาทำงานร่วมกันในเทคโนโลยี RAID จะถูกเรียกว่าดิกส์อาเรย์ โดยระบบปฏิบัติการและซอฟต์แวร์จะเห็นดิกส์ทั้งหมดเป็นตัวเดียว ซึ่งการทำ RAID นี้ จะเป็นการเพิ่มประสิทธิภาพของการเก็บรักษาข้อมูลอย่างมาก มี RAID แบบต่าง ๆ ที่มีความสามารถต่างกันและถูกเอามาใช้ในงานที่แตกต่างกัน แล้วแต่ผู้ใช้งานจะนำไปใช้งาน

แบบฝึกหัดท้ายบทที่ 10

จงตอบคำถามต่อไปนี้

1) จงอธิบายถึงหน้าที่ของระบบปฏิบัติการในการจัดการดิสก์คืออะไร มีรายละเอียดอย่างไรบ้าง

2) จงอธิบาย Device Driver พร้อมวาดภาพประกอบ

3) จงอธิบายความหมายคำว่า Sector, Track และ Cylinder

4) จงอธิบายข้อดีข้อเสียของวิธีการจัดการใช้ดิสก์ทั้งแบบ FCFS, SSTF, SCAN, C-SCAN และ C-LOOK

5) กำหนดให้ดิสก์ของระบบคอมพิวเตอร์ระบบหนึ่งถูกแบ่งออกเป็น 200 ไซลินเดอร์ (0-199) ขณะนี้หัวอ่านของดิสก์อยู่ที่ไซลินเดอร์ที่ 100 ถ้ามีการขอใช้ที่ไซลินเดอร์ต่าง ๆ ตามลำดับดังนี้ 55, 58, 39, 18, 90, 160, 150, 38 และ 184 จงตอบคำถามระบบจะจัดสรรการใช้ดิสก์อย่างไร เมื่อกำหนดให้ใช้การจัดการใช้ดิสก์ด้วยวิธีต่อไปนี้

5.1) การจัดเวลาแบบ First Come First Served: FCFS

5.2) การจัดเวลาแบบ Shortest Seek Time First: SSTF

5.3) การจัดเวลาแบบ SCAN

5.4) การจัดเวลาแบบ C-SCAN (Circular-SCAN)

5.5) การจัดเวลาแบบ C-LOOK

6) กำหนดให้ ณ ปัจจุบันหัวอ่านดิสก์อยู่ที่ตำแหน่ง 80 และในดิสก์มีจำนวนไซลินเดอร์ทั้งหมด 200 ไซลินเดอร์ โดยไซลินเดอร์ที่จะอ่านมีดังนี้ 10 84 72 180 12 160 155 120 50 จงวาดภาพการจัดการเวลาการใช้ดิสก์ ตามโจทย์ต่อไปนี้

6.1) การจัดเวลาแบบ First Come First Served: FCFS

6.2) การจัดเวลาแบบ Shortest Seek Time First: SSTF

6.3) การจัดเวลาแบบ SCAN

6.4) การจัดเวลาแบบ C-SCAN (Circular-SCAN)

6.5) การจัดเวลาแบบ C-LOOK

7) การจัดการพื้นที่ที่ใช้ในการสับเปลี่ยนคืออะไร มีประโยชน์อย่างไร

8) ถ้าองค์กรแห่งหนึ่งมีข้อมูลที่สำคัญในการเก็บเป็นอย่างมาก ถ้าจำเป็นในการเลือกใช้ RAID เพื่อนำมาใช้งานในองค์กรนี้ ควรจะใช้ RAID ระดับใด จงอธิบายเหตุผลในการเลือกใช้มาให้เข้าใจ

9) การนำ RAID ระดับต่าง ๆ มาใช้ร่วมกันมีประโยชน์อย่างไร จงอธิบายโดยให้เหตุผลหลักในการนำ RAID มาใช้ร่วมกัน

เอกสารอ้างอิง

กลุ่มระบบเครือข่ายคอมพิวเตอร์ กรมตรวจบัญชีสหกรณ์. สืบค้นวันที่ 2 กรกฎาคม 2557 จาก http://netgrp.cad.go.th/ewt_news.php?nid=216&filename=index

พิเชษฐ์ ศิริรัตนไพศาลกุล. (2548). **ระบบปฏิบัติการ.** กรุงเทพฯ : ซีเอ็ดยูเคชั่น.

Abraham Silberschatz, Peter Baer Galvin, Greg Gagne. (2013). **Operating System Concepts. 9th ed.** Wiley & Sons, Inc.