SYMMETRIC MULTIPROCESSORS(SMP)

ขณะที่ความต้องการทางด้านประสิทธิภาพเพิ่มมากขึ้นและราคาของ microprocessor จะหยุดอยู่กับที่ผู้ขายจึงนำเข้าระบบ SMP มาแนะนำให้รู้จัก SMP เป็นสถาปัตยกรรมคอมพิวเตอร์ HARDWARE และด้วยระบบการทำงานที่สะท้อนถึงสถาปัตยกรรมนั้นๆ SMP จึงสามารถนิยามได้ว่าเป็นระบบคอมพิวเตอร์ชนิดเดียวที่ทำงานตามลักษณะนี้

1. มีความสามารถที่เปรียบได้กับ processor 2 ตัวหรือมากกว่านั้น
2. มีการแบ่งหน่วยความจำหลักออกเป็นส่วนๆและต่อ I/O ได้ง่ายการเชื่อมต่อใช้ระบบ bus หรือ การ connection ปกติ ดังนั้นเวลาในการเข้าถึงข้อมูลก็จะเท่ากับ processor เพียงตัวเดียว
3. processor ทุกตัวจะแบ่งการเข้าถึงอุปกรณ์ I/O โดยจะส่งผ่านช่องโดยแต้ละช่องจะแตกต่าง กำหนดโดยความยาวผ่านไปยังอุปกรณ์
4. คอมพิวเตอร์ทุกเครื่องสามารถทำงานได้ด้วยตัวของมันเอง
5. ระบบทุกระบบทำงานร่วมกันและมีระบบหนึ่งที่ ควบคุมการทำงานระหว่าง processors แต่ละตัว

หัวข้อ 1ถึง 4 จะทำ self-explanatory หัวข้อ 5 อธิบายตรงกันข้ามซึ่งเกี่ยวกับความไม่รอบคอบของระบบ multiprocessing เช่น ตามที่ cluster ในภายหลังและ physical unit ของ interaction คือ สามรถใช้ถ้อยคำหรือแฟ้มข้อมูลที่สมบูรณ์ใน SMPซึง ข้อมูลเฉพาะกลุ่มสามารถประกอบเป็นระดับของ interaction และ สามารถทำให้อยู่ในระดับสูง จากการทำเป็น 2เท่า ของหน่วยประมวลผล

ระบบปฏิบัติการ ของ SMPตารางหน่วยประมวลผล หรือ threads across all ของ processor ซึ่ง SMPมีตัวเลขประโยชน์ที่สามารถเป็นไปได้เหนือกว่าสถาปัตยกรรม uniprocessor ซึ่งรวมตามมาที่หลัง

- performance: ถ้างานที่ทำโดยคอมพิวเตอร์สามารถบันทึกดังนั้นสามารถบันทึกแบ่งงานที่ทำใน parellel ซึ่ง ระบบเกี่ยวกับ processor หลายๆตัวจะทำให้มีประสิทธิภาพมากกว่าซึ่งหนึ่งในนั้นจะมี processor ของชนิดเดียวกัน(รูปหน้า 16.3)
- Availability: ใน symmetric multiprocessor หนึ่ง ๆ เพราะว่าทุกๆ processor ตัวใดๆไม่สามารถหยุดทำงานได้เพราะฉะนั้นระบบที่ทำงานเชื่อมต่อกันจะมีประสิทธิภาพลดลง
- Incremental growth : ผู้ใช้สามารถจะเพิ่มประสิทธิภาพระบบได้โดยการเพิ่ม processor
- Scaling: ผู้ที่สามารถเสนอผลิตภัณฑ์จึงแตกต่างทั้งด้านราคาและประสิทธิภาพ, ลักษณะพิเศษพื้นฐานของ processor ในระบบ

มันเป็นส่วนสำคัญที่เป็นไปได้ว่าจะดีกว่า รับประกันได้ ระบบทำงานจะกำหนดการทำงานด้วยเครื่องมือและฟังก์ชั่นที่ที่เป็นประโยชน์คล้ายๆในระบบ SMPลักษณะพิเศษของ SMP คือการมีชีวิตอยู่ของ processor ระบบจะสามารถช่วยเหลือกันใน processsor และสามารถทำงานพร้อมกันได้

หัวข้อ 1ถึง 4 จะทำ self-explanatory หัวข้อ 5 อธิบายตรงกันข้ามซึ่งเกี่ยวกับความไม่รอบคอบของระบบ multiprocessing เช่น ตามที่ cluster ในภายหลังและ physical unit ของ interaction คือ สามารถใช้ถ้อยคำหรือแฟ้มข้อมูลที่สมบูรณ์ใน SMPซึง ข้อมูลเฉพาะกลุ่มสามารถประกอบเป็นระดับของ interaction และ สามารถทำให้อยู่ในระดับสูง จากการทำเป็น 2เท่า ของหน่วยประมวลผล

ระบบปฏิบัติการ ของ SMPตารางหน่วยประมวลผล หรือ threads across all ของ processor ซึ่ง SMPมีตัวเลขประโยชน์ที่สามารถเป็นไปได้เหนือกว่าสถาปัตยกรรม uniprocessor ซึ่งรวมตามมาที่หลัง

- performance: ถ้างานที่ทำโดยคอมพิวเตอร์สามารถบันทึกดังนั้นสามารถบันทึกแบ่งงานที่ทำใน parellel ซึ่ง ระบบเกี่ยวกับ processor หลายๆตัวจะทำให้มีประสิทธิภาพมากกว่าซึ่งหนึ่งในนั้นจะมี processor ของชนิดเดียวกัน(รูปหน้า 16.3)

- Availability: ใน symmetric multiprocessor หนึ่ง ๆเพราะว่าทุกๆ processor ตัวใดๆไม่สามารถหยุดทำงานได้เพราะฉะนั้นระบบที่ทำงานเชื่อมต่อกันจะมีประสิทธิภาพลดลง

- Incremental growth : ผู้ใช้สามารถจะเพิ่มประสิทธิภาพระบบได้โดยการเพิ่ม processor

- Scaling: ผู้ที่สามารถเสนอผลิตภัณฑ์จึงแตกต่างทั้งด้านราคาและประสิทธิภาพ, ลักษณะพิเศษพื้นฐานชอง processor ในระบบ

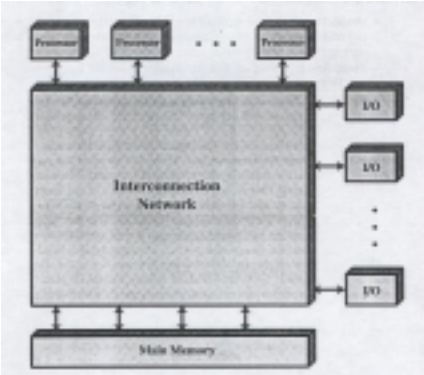
มันเป็นส่วนสำคัญที่เป็นไปได้ว่าจะดีกว่า รับประกันได้ ระบบทำงานจะกำหนดการทำงานด้วยเครื่งมือและฟังก์ชั่นที่ที่เป็นประโยชน์คล้ายๆในระบบ SMPลักษณะพิเศษของ SMP คือการมีชีวิตอยู่ของ processor ระบบจะสามารถช่วยเหลือกันใน processsor และสามารถทำงานพร้อมกันได้

การจัดระเบียบ

รูปที่16.4 บรรยายเกี่ยวกับคำศัพท์ การจัดระเบียบของระบบ multiprocessor มี 2 processor หรือ มากกว่านั้น แต่ละ processor จะบรรจุตัวของมันเองรวมทั้งควบคุมหน่วย,ALU,หารลงทะเบียนและ แคช แต่ละ processor จะเข้าไปแบ่งหน่วยความจำกันและ I/O จะทำแผนการผ่านบางสิ่งที่มาจากกลไกของ interconnection processor สามารถติดต่อสื่อสารกับเครื่องอื่นๆได้โดยผ่านหน่วยความจำ (ข้อความและสถานภาพของคำแนะนำที่มาจากพื้นที่ของข้อมูลปกติ) มันอาจจะเป็นไปได้สำหรับ processor ที่จะเปลี่ยนสัญญาณได้อย่างถูกต้อง หน่วยความจำมักจะถูกจัดระเบียบในเวลาเดียวกันหลายครั้ง เพื่อที่จะเข้าไปแบ่งแยกส่วนของหน่วยความจำทั่วไปของตัวมันเอง และช่อง I/O และทรัพยากรในการแบ่งด้วยเช่นกัน

การจัดระเบียบของระบบ multiprocessor สามารถแบ่งได้ดังนี้

1. การแบ่งเวลา หรือ bus ปกติ
2. หน่วยความจำ multiport
3. หน่วยศูนย์กลางการควบคุม



รูป 16.4

การแบ่ง time ของ bus

การแบ่ง time ของ bus คือ กลไกที่ง่ายที่สุดสำหรับการสร้างระบบ multiprocessor (รูป 16.5) การสร้าง และ interfaces เป็นการง่ายเหมือนกับการทำสำหรับระบบ single processor ชื่อจะใช้ bus interconnection ซึ่ง bus ประกอบด้วย ตัวควบคุม ที่อยู่ และบรรทัดข้อมูล การทำให้ DMA สะดวกต่อการเคลื่อนย้ายจาก I/O processors,ลักษณะเด่นที่ตามมาจะมีดังนี้

- **Addressing** : มันต้องเป็นไปได้ที่จะเห็นความแตกต่างของหน่วย บน bus เพื่อที่จะค้นหาแหล่งที่มาและข้อมูลของจุดหมายปลายทาง
- **Arbitration** : บาง I/O หน่วย สามารถทำหน้าที่เป็นหัวหน้าได้อย่างชั่วคราว กลไกประกอบด้วยความสมบูรณ์ที่ตัดสินชี้ขาดในการข้อร้องสำหรับ การควบคุม bus , การใช้ชนิดของโครงการ
- **Time sharing** : เมื่อหน่วยใดหน่วยหนึ่งกำลังควบคุม bus อยู่หน่วยอื่นๆ ก็จะต้อง lock ถ้ามันจำเป็น ต้องระงับการปฏิบัติการจนกระทั่ง การเข้าถึงจะประสบความสำเร็จ

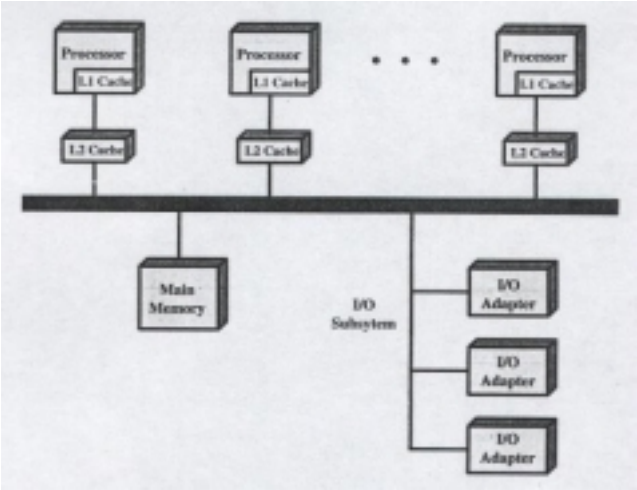
ลักษณะเด่นของ uniprocessor เหล่านี้ คือ การใช้การได้อย่างตรงไปตรงมาในโครงร่าง SMP ในกรณีหลัง,ตอนนี้มี multiprocessor เช่นเดียวกับ multiple I/O processor ความพยายามทั้งหมด จะได้รับการเข้าถึง1 หรือ มากกว่าหน่วยความจำโดยผ่าน bus

การจัดระเบียบ bus มีข้อได้เปรียบมากมาย เมื่อเปรียบเทียบกับทางเข้าอื่นๆ

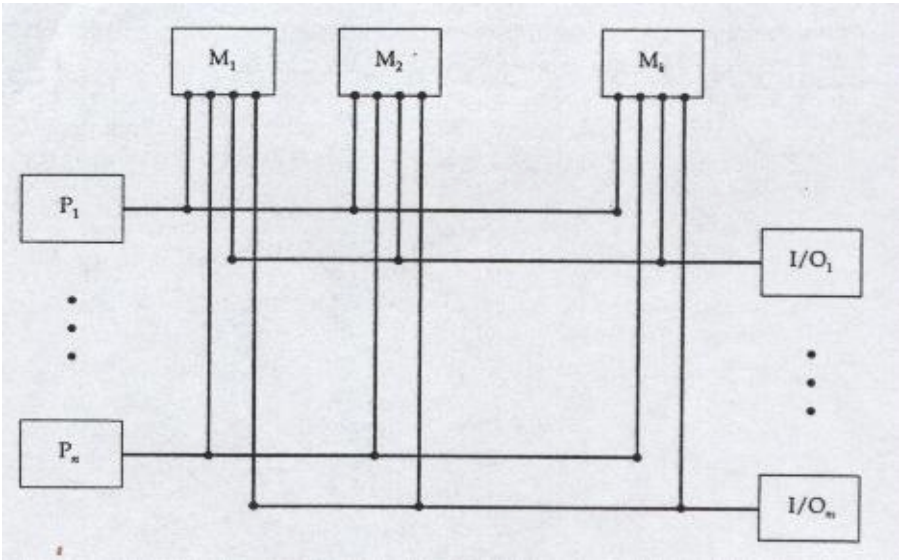
- **Simplicity** : นี่คือทางเข้าที่งายที่สุดในการจัดระเบียบ multiprocessor interface ทางกายภาพ และที่อยู่,การตัดสินชี้ขาด และการหาเหตุผลการแบ่งเวลาของแต่ละ processor ยังคงเหมือนกับระบบ single-processor
- **Flexibility** : มันเป็นเรื่องง่ายที่จะขยายระบบโดยการผูกมัด processor กับ bus
- **Reliability** : bus คือ passive ที่จำเป็นขนาดกลาง และความล้มเหลวของบางแผนหารที่ผูกมัด ไม่ควรเป็นเหตุความผิดพลาดของระบบทั้งหมด

ข้อเสียเปรียบที่สำคัญ ในการวัดระเบียบ bus คือ การปฏิบัติการกล่าวถึงหน่วยความจำทั้งหมดของ bus ผ่าน bus ธรรมดาตลอดดังนั้นรวดเร็วของระบบถูกจำกัดโดยเวลาของ bus เพื่อที่การปฏิบัติการที่ดีขึ้นมันคือความต้องการที่จะจัดหาแต่ละ processor ด้วยการ แคช หน่วยความจำนี้ควรจะลดจำนวนของการเข้าถึง bus อย่างดี โดยปกติ workstation และ PC SMPs มี 2 ระบบของ แคช ด้วยL1 cache เกี่ยวกับภายใน (เหมือนกับภายใน chip ใน processor) และ L2 cache ไม่ภายในก็ภายนอก

การใช้ แคช แนะนำการพิจารณาที่ออกแบบใหม่ เพราะแต่ละท้องถิ่นของ แคช จะบรรจุจินตนาการของส่วนของหน่วยความจำถ้าคำถูกเปลี่ยนในแต่ละ แคช มันสามารถทำให้คำหนึ่งคำเป็นโมฆะไปได้ใน แคช อื่นๆ ในการปกป้องนี้ processor อื่นๆต้องถูกเตือนซึ่งการ update เกิดขึ้น ปัญหานี้เป็นที่รู้จักกันในนามปัญหา *cache coherence* และโดยทั่วไปนี้ค่อนข้างจะมีที่อยู่ใน hardware มากกว่าโดยระบบการดำเนินการ พวกเรากล่าวออกมาในข้อ 10.3



รูป 16.5



รูป 16.6

ช่องหลายช่องในหน่วยความจำ

ช่องต่างๆหน่วยความจำสามารถเข้าถึงได้โดยตรงการเข้าถึงศูนย์กลางชิ้นส่วนหน่วยความจำได้อย่างอิสระโดยแต่ละเครื่องประมวลผลและชิ้นส่วน I/O (ข้อ 16.6). ลอจิกที่เกี่ยวข้องกับหน่วยความจำที่ต้องการที่ขัดกัน วิธีที่ใช้บ่อยๆกับข้อตกลงที่ขัดกันต้องกำหนดมีเครื่องหมายลำดับก่อนหลังอย่างถาวรเพื่อแต่ละช่องหน่วยความจำ.โดยชนิดส่วนติดต่อฟิสิกส์และไฟฟ้าแต่ละช่องจะเหมือนกันไม่ว่าใช้สำหรับทำงานอะไรในชิ้นส่วนหน่วยความจำช่องเดี่ยวไปจนถึงระดับที่เล็กๆหรือไม่มีการแก้ไขถูกต้องการสำหรับเครื่องประมวลผลหรือชิ้นส่วน I/O เพื่อช่องต่างๆให้เหมาะสมกับหน่วยความจำ.

หน่วยความจำที่มีหลายช่องจะเข้าถึงได้ยากมากกว่าแบบบัสความต้องการผลรวมที่ถูกต้องทางลอจิกเพื่อที่จะได้รับเพิ่มระบบหน่วยความจำ อย่างไรก็ตามการป้องกันดีกว่าเพื่อประสิทธิภาพเพราะว่าแต่ละเครื่องประมวลผลมีการทำงานให้แต่ละชิ้นส่วนหน่วยความจำ. ข้อได้เปรียบอื่นๆของการมีหลายช่องไว้เพื่อแก้ไขส่วนของหน่วยความจำเหมือน "private" เพื่อหน่วยความจำหรือเครื่องประมวลผลมากขึ้นหรือ ชิ้นส่วน I/O.ความสามารถนี้ยอมสำหรับการเพิ่มความปลอดภัยปะทะการเข้าถึงที่ไม่ให้สิทธิและสำหรับ ที่ใช้บันทึกของทำงานประจำ ในพื้นที่ของหน่วยความจำไม่สามารถแก้ไขได้ง่ายโดยเครื่องประมวลผลอื่นๆ

ส่วนจุดอื่นๆรูปแบบการเขียนใช้สำหรับคอนโทรลแคชเพราะว่าไม่มีความหมายที่สะดวกอื่นๆเพื่อเตือนเครื่องประมวลผลอื่นๆของหน่วยความจำใหม่ๆ

หน่วยศูนย์กลางการควบคุม

ศูนย์กลางการควบคุมจนกระทั่งข้อมูลที่แยกออกมาเป็นทางข้างหลังและ 4 ระหว่างชิ้นส่วนอิสระ คือ เครื่องประมวลผล, หน่วยความจำ, I/O หน่วยควบคุมที่สามารถเก็บข้อมูลชั่วคราวตามความต้องการและกระทำชี้ขาดและเวลาในฟังก์ชัน มันยังสามารถผ่านสถานะและควบคุมข้อความระหว่างเครื่องประมวลผลและกระทำการเตือนการปรับปรุงแคชเพราะว่าลอจิกทั้งหมดสำหรับการจัดระเบียบ ค่าที่ตั้งไว้ในหน่วยประมวลผลหลายทางถูกเพ่งเลง

ในศูนย์กลางการจนกระทั่งส่วนติดกับ I/O, หน่วยความจำ, และเครื่องประมวลผลยังคงไม่ได้ถูกรบกวน ซึ่งได้เตรียมลักษณะที่เปลี่ยนแปลงและความเรียบง่ายของส่วนเชื่อมต่อกันของระบบของการเข้าหาbus ส่วนการแก้ไขข้อเสียหายของการเข้าหาคือหน่วยควบคุมที่สลับซับซ้อนทั้งหมดซึ่งมันเป็นสิ่งที่ทำให้ประสิทธิภาพช้าลง

The central control unit structure ที่เคยเป็นของระบบหลัก multiple-processor เช่น large-scale members of the IBM S/370 family และมันยังดีเลิศจนทุกวันนี้

Multiprocessor Operating System Design Considerations

SMP operating system จัดการประมวลผลและแหล่งที่มาของคอมพิวเตอร์อื่นๆซึ่งผู้ใช้ได้แลเห็นองค์ประกอบเดียวoperating system การควบคุมแหล่งที่มาของระบบ ในข้อเท็จจริงเช่น รูปร่างภายนอกจะปรากฏ single-processor multiprogramming system ในส่วนทั้ง2อย่างคือSMPและuniprocessor cases multiple jobs หรือprocessor จะคล่องตัวที่ระยะเวลาหนึ่งและมันเป็นความรับผิดชอบของ operating systemที่จะกำหนดแบบแผนการปฏิบัติงานและจักสรรแหล่งที่มา userจะสร้างการประยุกต์นั้นจะใช้multiple-processorหรือmultiple threadsภายในการประมวลผลโดยปราศจากการพิจารณาจากsingle-processor หรือmultiple-processor ซึ่งจะหาได้ดังนั้นmultiprocessor operating จะต้องจัดหาให้โดยภาระหน้าที่ทั้งหมดของmultiprogramming systemที่เพิ่มขึ้นเป็นจุดเด่นพิเศษที่จัดหาให้multiple-processor ระหว่างการออกแบบการแก้ไขประเด็นสำคัญมีดังต่อไปนี้

-**Simultaneous concurrent process:** ปกติระบบปฏิบัติการต้องการเข้าไปในการประมวลผลต่างๆที่ปฏิบัติในเวลาเดียวกันเหมือนIS การปฏิบัติงานของ multiple-processor จะคล้ายหรือแตกต่างกับส่วนของระบบปฏิบัติการ tableและการควบคุมโครงสร้างจะต้องจัดการอย่างเหมาะสมถูกต้องหลีกเลี่ยวสภาพที่ไม่เหมาะสมต่อไป

-**Scheduling:** ตารางสารบัญจะถูกแสดงโดยการประมวลผลหลายๆการประมวลผลดังนั้นควรหลีกเลี่ยงการขัดแย้งและตารางสารบัญจะต้องกำหนดตัวแปรในการประมวลผล

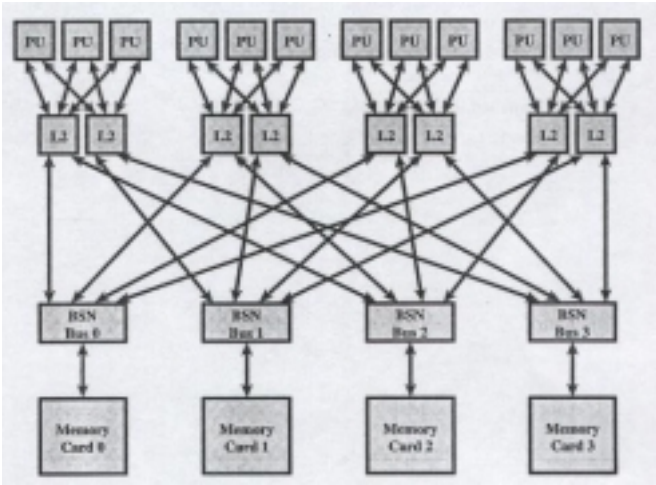
-**Sychronization:** multiple active processes อาจเป็นไปได้ที่จะถูกแบ่งจากที่ที่ว่างหรือถูกแบ่งแหล่งที่มา I/O และก็จะต้องทำที่จะจัดหาให้เกิดในเวลาเดียวกันที่มีประสิทธิภาพเป็นความสะดวกซึ่งบังคับใช้แยกออกไป

-**Memory management:** การจัดการ memory บน multiprocessor จะต้องจัดการกับจุดที่สำคัญทั้งหมดบนเครื่องระบบปฏิบัตการหน่วยประมวลผล ต้องการความสามารถซึ่งหาได้ที่มีลักษณะความสัมพันธ์กันของฮาร์ดแวร์ เช่นเดียวกับช่องหลายช่องในหน่วยความจำที่จะสำเร็จในประสิทธิภาพที่ดีที่สุด The paging mechanisms บนความแตกต่างในการประมวลผลทจะเท่ากันที่บังคับใช้อย่างสม่ำเสมอเมื่อการประมวลผลต่างๆได้แบ่งหน้าหรือส่วนและตัดสินใจในส่วนที่ทำหน้าที่แทน

-**Reliability and fault tolerance:** ระบบปฏิบัติการจะเตรียมลดข้อขัดข้องให้ดีขึ้น ตารางสารบัญและส่วนอื่นๆของระบบปฏิบัติการจะต้องยอมรับในส่วนน้อยของการประมวลผลและจัดการปรับปรุงโครงสร้างใหม่ให้สอดคล้องกัน

A Mainframe SMP

PC และ workstation SMPs จะใช้แผนการการติดต่อระหว่าง bus มันเป็นการให้ข่าวสารข้อมูลที่จะพิจารณาการเข้าถึงซึ่งมีให้เลือกซึ่งใช้สำหรับการดำเนินการเร็วๆนี้ของthe IBM S/390 mainframe family [MAK97]



รูป 16.7

รูป 16.7 วาดให้เห็นองค์กรครอบคลุมของS/390 SMP. องค์กรของระบบวัดระยะจากตัวที่ไม่ใช่หน่วยประมวลผลกับหนึ่งการ์ดหน่วยความจำหลักถึงระบบที่สูงที่สุดกับ 10 เครื่องประมวลผลและ 4 การ์ดหน่วยความจำค่าที่ตั้งไว้รวมถึงค่าที่ตั้งไว้หรือ 2 เครื่องประมวลผลเพิ่มเติมซึ่งให้เป็น เครื่องประมวลผล I/O ส่วนประกอบคีย์ของค่าที่ตั้งไว้มีดังต่อไปนี้:

-หน่วยเครื่องประมวลผล (PU) : นี่คือ CISC microprocessor , ซึ่งมีการใช้คำสั่งมากที่สุดคือ hardwired และนอกจากนี้ยังกระทำโดย firmware. แต่ละ PU รวมถึง64-KBL1 แคชซึ่งเป็น unifid (ข้อมูลและคำสั่ง)ขนาดของแคช L1 ต้องมีขนาดพอดีกับชิป PU และเพื่อบรรลุการเข้าถึงรอบชิป

-แคช L2: แต่ละแคช L2 บรรจุ 384 KB. แคช L2 ถูกจัดเรียงในกลุ่ม x2, กับทีละ 3PUs ที่สนับสนุนกลุ่มและการตระเตรียมเข้าถึงสู่ช่องว่างหน่วยความจำหลักทั้งหมด.

-การปรับเครือข่ายที่เปลี่ยนบัสให้เหมาะสม (BSN): BSNs อยู่ระหว่างการติดต่อแคช L2 และหน่วยความจำหลัก. แต่ละBSN ยังรวมถึงระดับ 3(L3) ซึ่งขนาดของแคชคือ 2 MB.

-การ์ดหน่วยความจำ:แต่ละการ์ดจัดขึ้น 8 GB ของหน่วยความจำสำหรับทั้งหมดของความจุ32 GB .

มีความสามารถที่น่าสนใจจำนวนมากค่าที่ตั้งไว้ใน S/390 SMP

-การต่อแบบ switch

-การแบ่งแบบ L2 cache

-L3 cache

การต่อแบบ switch

การแบ่ง bus เดี่ยวเป็นส่วนหนึ่งของการเตรียมการของ SMPs สำหรับ PCs และ workstations โดยการเตรียมนี้ bus เดี่ยวจะกลายเป็นปัญหาของการออกแบบ แต่ S/390 จะจัดการด้วย 2 วิธี วิธีแรก ความจำหลักจะแยกเป็น 4 การ์ด แต่ละการ์ดจะมีตัวควบคุมการเก็บข้อมูลของมันเองซึ่งสามารถที่จะส่งผ่านข้อมูลไปยังหน่วยความจำด้วยความเร็วสูง ในการ load ไปยังหน่วยความจำจะถูกตัดโดย factor หนึ่งใน 4 ตัว เพราะแต่ละเส้นทางจะไม่เกี่ยวข้องกันเลย วิธีที่สองคือการต่อจากตัวประมวลผลไปยังการ์ดหน่วยความจำเดี่ยวที่แบ่งไว้ จะไม่อยู่ในรูปแบบของการแบ่ง bus แต่จะเหมือนกับ point-to-point links มากกว่า แต่ละ links จะต่อกลุ่มตัวประมวลผลที่มี 3 ตัว ด้วย L2 cache กับ BSN ใน BSN ทางกลับกันแล้วการกระทำกับแบบ switch นั้นสามารถที่จะผ่านข้อมูลไปโดยใช้การต่อ 5 links โดยมี L2 links 4 ตัว BSN ที่ต่อกับ physical links 4 ตัว กับ logical data bus หนึ่งตัว ด้วยเหตุนี้จึงเกิดข้อดีคือการย้อนกลับไปเป็นแบบ L2 links 3 ตัว ซึ่งใช้ในการต่อ แคช

การแบ่งแบบ L2 แคช

ในการใช้ แคช แบบ two-level สำหรับ SMP ซึ่งตัวประมวลผลแต่ละตัวจะมี แคช แบบ dedicate L1 และ L2 เมื่อเร็วๆนี้ได้มีความคิดที่น่าสนใจเกี่ยวกับการแบ่ง L2 แคช ที่ก้าวหน้าขึ้น โดยในแบบแรกๆของ S/390 SMP จะเท่ากับรุ่นที่ 3 ที่ IBM ได้ทำประโยชน์ของ แคช แบบ dedicate L2 ในแบบล่าสุดการแบ่งแบบ L2 นี้จะถูกใช้มาก โดย 2 ข้อที่เปลี่ยนไปคือ:

1. การย้ายจากแบบรุ่น 3 ไปเป็นรุ่น 4 รุ่นที่ล่าสุด IBM จะเร็วเป็น 2 เท่าของ microprocessor ถ้าไม่เปลี่ยนจะเกิดปัญหากับ bus ได้ในขณะเดียวกันก็ต้องการจะนำ รุ่น 3 มาใช้ใหม่เท่าที่ทำได้

2. การวิเคราะห์ S/390 workloads แสดงให้เห็นขนาดที่ใหญ่ของการแบ่งคำสั่งและข้อมูลที่จะใช้

ข้อพิจารณาเหล่านี้ นำมาใช้กับ S/390 รุ่น 4 ทีมออกแบบได้วิเคราะห์ประโยชน์ของการใช้ L2 แคช 1 ตัวหรือหลายตัว ซึ่งจะแบ่งโดยตัวประมวลผลหลายตัว ในครั้งแรกๆ อาจจะดูเหมือนว่าไม่ดีนัก การผ่านข้อมูลจะช้ามากเพราะตัวประมวลผลจะคอยทำการผ่านไปยัง แคช L2 แบบเดี่ยว อย่างไรก็ตาม จำนวนที่น่าจะพอความจริงแล้วจะแบ่งไปโดยตัวประมวลผลหลายตัว ดังนั้น การแบ่ง แคช จะเพิ่ม throughput ให้มากว่าตัวที่ถ่วงมันไว้ ข้อมูลที่ถูกแบ่งและพบใน แคช ที่แบ่งจะถูกบรรจุอย่างรวดเร็ว ถ้าพบได้แถว bus

Table 16.1 การออกแบบ S/390 G4 ในส่วนของเวลาที่ใช้เริ่มกับ แคช โดยใช้หน่วยประมวลผลทั้งหมด, ในปัจจุบันมีการปรับปรุงประสิทธิภาพโดยใช้ระบบ VIA ซึ่งมีประสิทธิภาพมากกว่า,VIA

ใช้ระบบบัสที่สมบูรณ์ แต่การวิเคราะห์ประสิทธิภาพควรใช้ แคช ร่วมกับ BSNs buses จึงจะเกิดประโยชน์สูงสุดของ แคช และลดปัญหาในการส่งข้อมูล . ค่าของ แคชเป็นสิ่งยืนยันถึงการวัดประสิทธิภาพและแสดงถึงการปรับปรุง แคช hit rates ดูจากตารางการใช้ G3 organnization [MAK 97] .

L3 Cache

ส่วนประกอบอื่นๆที่น่าสนใจของ S/390 SMP คือการใช้ แคช ระดับ3 (L3 Cache) . L2 Cache เป็นตำแหน่งใน BSNs ดังนั้น L3 Cache อยู่ระหว่าง L2 Cache และ memory card อีกหนึ่งตัว. การส่งข้อมูลจะเร็วกว่าการส่งจาก main memory ใน Cache Line จะทำงานร่วมกับ processor อื่นๆ.

ตาราง 16.1 แสดงผลลัพธ์ของ แคช ทั้ง 3 ระดับในระบบ SMP ชนิด S/390 . Access Penalty (PU cycles) เป็น latency ระหว่างข้อมูลไป แคช ลดหลั่นตามลำดับและครั้งแรกจะส่งค่ากลับ 16 byte data. L1 Cache ทำให้เกิด Hit rates 89 % , ส่วนที่เหลืออีก 11 % ของเม็มโมรี่จะแยกเป็น L2 เท่ากับ 5 % , L3 เท่ากับ 3 % และการเข้าถึงเม็มโมรี่ใช้เพียง 3 % เท่านั้น .

16.4 clusters

รูปแบบหนึ่งที่เป็นที่นิยมในการออกแบบระบบคอมพิวเตอร์ คือ clustering

clustering คือ การเชื่อมต่อระบบคอมพิวเตอร์หลายๆ ระบบให้ทำงานร่วมกันอย่างมีประสิทธิภาพสุงสุด เราสามารถกำหนดกลุ่มของ cluster ที่เชื่อมต่อกันให้คอมพิวเตอร์ทั้งหมดทำงานร่วมกันได้แบบเครื่องจักร โดยคอมพิวเตอร์ทุกเครื่องจัดเป็นระบบใหญ่ คือสามารถทำงานได้ด้วยตัวของมันเอง จากรูปแบบของ Cluster ในเรื่องเกี่ยวกับการเขียน เครื่องแต่ละเครื่องใน Cluster จะเปรียบเสมือนกับแป้นพิมพ์ (ปุ่มต่างๆ)

[BREW 97] รายชื่อ 4 หัวข้อที่สามารถบอกถึงผลสำเร็จของ Clustering มันสามารถทำงานบรรลุวัตถุประสงค์หรือออกแบบสิ่งที่ต้องการได้

-Absdute scalability : เป็นไปได้ที่จะสร้าง Cluster ใหญ่ๆ ที่ทำงานได้เหมือนกับเครื่องจักร แต่ละ Cluster สามารถมีเครื่องจักรประกอบได้ถึง 1 โหล ซึ่งก็คือ มีเครื่องประมวลผลได้หลายๆเครื่อง

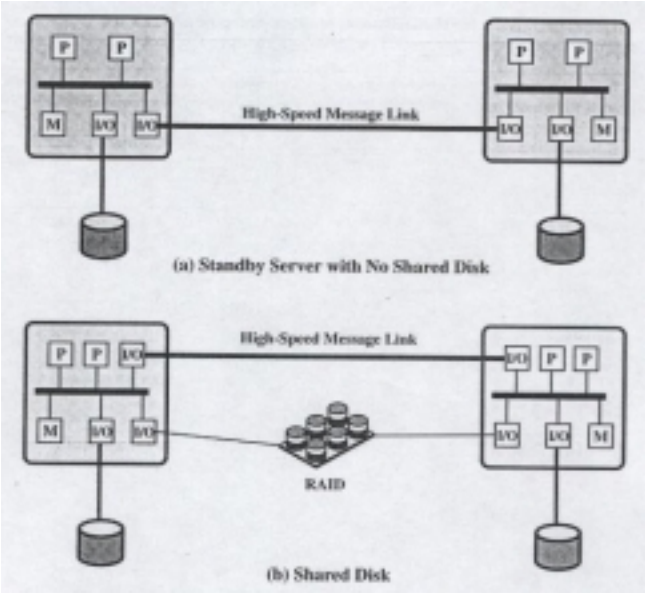
-Incremental sca;ability : cluster สามารถเพิ่มระบบใหม่ๆเข้าไปในระบบเล็กๆ โดย user สามารถสร้างระบบขนาดพอประมาณและสามารถขยายได้ตามต้องการโดยมีระบบเล็กทำหน้าที่แทนระบบใหญ่ๆ

-High availability : เพระว่าแต่ละเครื่องใน Cluster คือ สามารถทำงานได้ด้วยตัวของมันเอง จึงสามารถทำงานต่างไได้หลายงาน โดย Software จะทำงานได้โดยอัตโนมัติ

-Superior price/perfromance : ในการสร้างงานระบบ Cluster สามารถทำงานได้เท่ากับหรือมากกว่า ดีกว่าคอมพิวเตอร์ที่ทำงานเดียวๆเพียงเครื่องเดียวและมีค่าใช้จ่ายน้อย

Cluster Conficguration

ในการอธิบายถึง Cluster สามารถอธิบายได้หลายอย่าง บางทีก็จะแบ่งออกเป็นประเภทหรือไม่คอมพิวเตอร์ใน Cluster ก็จะแบ่งกันเข้าถึงข้อมูลใน disk ดังรูป 16.9 แสดง 2 เครื่องใน Cluster ในการเชื่อมต่อ 1 ครั้ง สามารถติดต่อแลกเปลี่ยนข้อความที่มีความสำคัญพอๆกันได้ใน Cluster ในการเชื่อมต่อกันสามารถทำได้ง่ายโดยใช้ระบบ LAN แต่ในบางกรณีที่มีเครื่องคอมพิวเตอร์จำนวนมากๆ Cluster จะใช้การเชื่อมต่อแบบ LAN หรือ WAN เพื่อติดต่อระหว่างกันดังในรูป เครื่องคอมพิวเตอร์แต่ละเครื่องจะสามารถทำงานได้มากขึ้นโดยไมาจำเป็นต้องใช้เครื่องคอมพิวเตอร์มากๆ แต่ได้ประสิทธิภาพและผลประโยชน์เพิ่มขึ้น



รูป 16.9

รูป 16.9 เป็นทางเลือกที่จะ shared – disk ของ cluster ในกรณีนี้โดยทั่วๆไปก็ยังสามารถส่งข้อมูลระหว่างกันได้ ในการเพิ่มระบบย่อยใน cluster จะทำหน้าที่ควบคุมคอมพิวเตอร์หลายๆเครื่องใน cluster ดังในรูปเราเรียกระบบย่อยๆนี้ว่าระบบ RAID. โดยระบบ RAID ทำงานคล้ายกับ shared – disk แต่มีประโยชน์มากกว่า คือสามารถทำงานหลายอย่างที่คอมพิวเตอร์เครื่องเดียวทำไม่ได้ให้สำเร็จ

ในการเลือก cluster ที่สามารถทำงานได้ดี ดูจากการเลือกทำงาน Hewlett Packard [HP96] กำหนดการแยกประเภทโดยดูจากการทำงานของ function , ซึ่งเราต้องพิจารณา.

Gluster sersus

ตาราง 16.2 วิธีการรวมกลุ่ม : ผลประโยชน์และขีดจำกัด

วิธีการรวมกลุ่ม	คำอธิบาย	ผลประโยชน์	ขีดจำกัด
Passive standby	Sever ที่รองลงมาจะรับช่วงในกรณีของ sever ที่สำคัญเกิดความผิดพลาด	ง่ายต่อการดำเนินการ	มูลค่าสูงเพราะsever รองลงมาหาไม่ได้สำหรับงานกระบวนการ
Active secondary	Sever ที่รองลงมามักจะใช่สำหรับงานกระบวนการเช่นกัน	มูลค่าลดลงเพราะ sever ที่รองลงมาสามารถใช่สำหรับกระบวนการได้	มีความซับซ้อนเพิ่มขึ้น
Separate servers (sever ที่แบ่งเป็น-ส่วนๆ)	Sever ที่แบ่งเป็นส่วนๆจะมี disk ของตัวมันเองข้อมูลคือการบันทึก อย่างต่อเนื่อง จากหลักสำคัญไปสู่ sever ที่รองลงมา	มีการหามาได้สูง	เครือข่ายและ severที่เหมาะสมเหนือพอที่จะบันทึกการปฏิบัติการ

Servers connected to disks (sever ติดต่อกับ – disk)	Sever มีสายไฟฟ้า เหมือนกับ disk แต่ละ sever ของมันจะมี disk ของมัน ถ้า sever ตัวหนึ่งเสีย disk ของมันจะถูกรับช่วงโดย sever ตัวอื่น	เครือข่ายลดลง	บ่อยครั้งที่ต้องการ disk ที่สะท้อนได้ หรือ เทคโนโลยี RAID ที่จะชดเชยสำหรับการเสี่ยงของ disk ที่ผิดพลาด
Servers share disks (sever ที่แบ่ง disk กัน)	Sever ที่พลาดการแบ่งเข้าไปใน disk ในเวลาเดียวกัน	เครือข่ายต่ำและมี sever ที่เหนือกว่าการเสี่ยงของการ downtime เป็นเหตุโดย disk ที่ผิดพลาด	ต้องการ lock การจัดการ software บ่อยๆ ที่เกิดกับการสะท้อนของ disk หรือ เทคโนโลยี RAID

โดยปกติแล้ว วิธีเก่าๆที่เรารู้จักกันในนาม **passive standy** คือวิธีง่ายๆที่จะทำให้ computer เครื่องหนึ่งจัดการ load ได้ทั้งหมด ขณะที่ computer เครื่องอื่นๆยังคงไม่ว่องไวในการเตรียมพร้อมการรับช่วงเหตุการณ์ของการไม่ทำงานในหลักสำคัญได้ การประสานงานของเครื่องจักร ความว่องไวหรือหลักสำคัญ บางครั้งคราวจะส่ง heartbeat ข้อความเพื่อที่จะเตรียมเครื่องจักรให้พร้อม ควรจะหยุดข้อความเหล่านี้ไว้ก่อนจะมาถึง การเตรียมจะสันนิษฐานว่า sever หลักสำคัญจะผิดพลาด และใส่ตัวมันเองลงในการปฏิบัติการ วิธีการแก้ไขมาได้เพิ่มขึ้น แต่ไม่ได้ทำให้การปฏิบัติการดีขึ้นเลย นอกจากนี้ถ้าคำแนะนำเพียงอย่างเดียวที่แลกเปลี่ยนระหว่างระบบ 2 ระบบได้คือข้อความจาก heartbeat และถ้าสองระบบนั้นไม่แบ่ง disk กัน การเตรียมพร้อมจะทำให้การปฏิบัติหน้าที่กลับมาและจะไม่เข้าไปที่ database เพื่อจัดการหลักสำคัญ

โดยทั่วไปแล้ว passive standby จะไม่ถูกส่งไปอย่างเป็นกลุ่มๆ ระยะเวลาเป็นกลุ่มๆจะถูกเก็บไว้สำหรับ computer interconnect มากมายซึ่ง คือการทำงานอย่างว่องไวทั้งหมดในขณะที่ การรักษาภาพพจน์ของระบบเดียวออกไปสู่โลกภายนอก คำว่า **active secondary** มักจะใช้ในการส่งไปยังโครงร่าง 3 วิธีของการรวมกลุ่มสามารถพิสูจน์ได้คือ การแบ่งแยก sever การไม่แบ่งบัน และแบ่งหน่วยความจำ

วิธีการลงมือรวมกลุ่มของแต่ละเครื่อง computer คือ การแบ่งแยก **separate sever** ด้วย disk ของมันเอง และไม่มี disk แบ่งกันระหว่างระบบ (รูป16.9a) การจัดการจะให้การปฏิบัติเช่นเดียวกับการหามาได้สูงในกรณีของการจัดการบางชนิด หรือการจัด software ลงในตาราง มักเป็นที่ต้องการกำหนดรายได้ที่ลูกค้าขอในการ sever เพื่อว่าการ load คือการสมดุลและการใช้ประโยชน์ได้สูงคือ การประสบผลสำเร็จ มันคือควรมีความสามารถในการ failover ซึ่งหมายความว่า ถ้า computer มีการผิดพลาดขณะที่การปฏิบัติตามการใช้ computer เครื่องอื่นๆที่รวมกันเป็นกลุ่มสามารถรับ และถ้าให้การใช้เสร็จสมบูรณ์ ตั้งแต่สิ่งนี้เกิดขึ้น ข้อมูลต้องอัดสำเนาเป็นประจำ ท่ามกลางระบบเพื่อที่แต่ละระบบเพื่อที่แต่ละระบบ จะเข้าไปที่ข้อความปัจจุบันของระบบอื่นๆการ overhead ของข้อมูลนี้จะเปลี่ยน การหามาได้ที่แน่นอนที่มูลค่า ของการลงโทษในการปฏิบัติการ การลดการสื่อสารโดยการ overhead ตอนนี้มักจะ รวมกลุ่มที่ประกอบด้วย sever ที่ติดต่อกับ desk ธรรมดา (รูป16.9b) ความหลากหลายในการทำงาน คือ การเรียกว่า **shared nothing** ในการลงมือปฏิบัติ desk ธรรมดา คือ การแบ่งแยกประมาณ และแต่ละปริมาณจะถูกตัวมันเองแบ่งโดย computer เดียวถ้า computer นั้นผิดพลาด การรวมกลุ่มจะ

ต้อง reconfigured เพื่อที่ computer หลายอย่างที่แบ่ง desk เหมือนกัน ในเวลาเดียวกันเรียกว่าการทำงานแบบ **share disk** เพื่อที่ computer แต่ละเครื่อง จะเข้าไปในปริมาณของ desk ทั้งหมดการทำงานนี้ เป็นที่ต้องการในการใช้ lock การจ่ายอุปกรณ์บางชนิด เพื่อที่จะให้แน่นใจว่า ข้อมูลนั้นสามารถเข้าไป โดย computer เครื่องหนึ่ง

<u>การปฏิบัติการระบบที่ออกแบบออกมา</u>

เต็มไปด้วยการใช้ที่เป็นประโยชน์ของ การรวมกลุ่มของของโครงสร้าง hardware ต้องการการเพิ่ม ระบบเดียวในการดำเนินระบบ

<u>การจัดการไม่ทำงาน</u>

ความล้มเหลวถูกจัดการอย่างไร โดยการรวมกลุ่มขึ้นอยู่กับ การรวมกลุ่มวิธีที่เคยทำ โดยปกติการทำงาน 2 อย่าง สามารถถูกนำมาจัดการกับการไม่ทำงาน การรวมกลุ่มที่หามาได้สูง และการรวมกลุ่มที่อดทนอย่างผิดพลาด การรวมกลุ่มที่หามาได้สูง ให้ความเป็นไปได้สูงกว่า ทรัพยากรทั้งหมดจะอยู่ในประโยชน์ ถ้าการไม่ทำงานเกิดขึ้น เช่นระบบเสื่อมหรือจำนวน disk สูญหายหรือแม้แต่คำถามในกระบวนการสูญหาย คำถามาที่สูญหายบางคำถาม ถ้ามันพยายามอีกครั้ง จะเป็นประโยชน์ โดย computer ตัวอื่นๆ ในการรวมกลุ่ม อย่างไรก็ตามระบบการทำงานแบบการรวมกลุ่ม จะไม่รับรองเกี่ยวกับความมั่นคง ของการปฏิบัติตามการจัดการบางส่วนเหล่านี้ต้องการการจัดการที่เป็นระบบที่มีประโยชน์

การรวมกลุ่มที่อดทนอย่างผิดพลาด จะแน่ใจได้ว่าทรัพยากรทั้งหมดหามาได้เสมอ นี้คือการได้รับความสำเร็จโดยการใช้ disk แบ่งกันอย่างเหลือเฟือ และกลไกสำหรับการทำความผิดในการจักการจะกลับมาและการกระทำการจัดการอย่างสมบูรณ์

หน้าที่ของการเปลี่ยนประโยชน์และทรัพยากรข้อมูล เกิดจากความผิดพลาดของระบบที่ระบบตัวเลือกในการรวมกลุ่มจริงเป็น **failover** ความเกี่ยวโยงของหน้าที่ คือการนำมาใช้อีกของประโยชน์ และทรัพยากรข้อมูลเป็นระบบต้นกำเนิด มันจะผสมส่งเป็น **failblack** มันสามารถทำได้อย่างอัตโนมัติ แต่นี้ต้องการเพียงอย่างเดียว ถ้าปัญหามันผสมอย่างแท้จริง และไม่เหมือนกับการเกิดขึ้นอีกถ้า failback ไม่อัตโนมัติ มันสามารถเป็นเหตุในภายหลัง ในการพลาดทรัพยากรที่จะกลับมาและออกมาระหว่าง computer ผลในการปฏิบัติงาน และปัญหาที่ฟื้นขึ้น

<u>การ load ที่สมดุล</u>

การรวมกลุ่มต้องการความสามารถที่เป็นผลสำหรับการสมดุลของ load ท่ามกลาง computer ที่หามาได้ มันรวมทั้งความต้องการ ซึ่งการรวมกลุ่มเกี่ยวกับส่วนนี้ เมื่อ computer เครื่องใหม่ถูกเพิ่มขึ้นรวมเป็นกลุ่มอุปกรณ์การ load สมดุลควรจะอัตโนมัติรวมทั้ง computer นี้ในประโยชน์ที่ลัดจากเรื่องเวลากลไกผลิตในส่วนกลาง ต้องการการระลึกได้ว่าประโยชน์สามารถปรากฏบนส่วนหนึ่งของการรวมกลุ่ม และอาจจะโยกย้ายจากส่วนหนึ่งของการรวมกลุ่ม และอาจจะโยกย้ายจากส่วนหนึ่งไปยังอีกที่หนึ่ง

ทั้ง cluster และ (smp) symetric multiprocessor กำหนดรูปลักษณ์ให้สามารถรองรับการประยุกต์สิ่งต่างๆได้ตามต้องการ.ทั้งการแก้ปัญหา และประโยชน์ทางด้านการติดต่อค้าขาย

ในการเข้าถึงข้อมูลและการจัดการ smp ทำงานได้ดีกว่า cluster.เดิม smp คือการประมวลผลอย่างเดียวที่ใกล้เคียงกับการประยุกต์การเขียน smp สามารถที่จะวางแผนการทำงาน คุณประโยชน์อื่นๆข้อ smp ก็พอจะเทียบได้กับ cluster และสิ่งสำคัญสุดท้ายคือผลงานที่ได้จาก smp คงที่แน่นอน.

นอกเหนือจากผลประโยชน์ที่ได้จากการเข้าถึงข้อมูลแล้วผลลัพธ์ใน cluster ยังมีประสิทธิภาพสูงพอที่ตลาดต้องการ. Cluster ดีกว่า smps ในด้านของการเพิ่ม / สมบูรณ์ของ scalability. Cluster ดีกว่าในเรื่องของประโยชน์ที่ได้รับเพราะว่าระบบที่ประกอบขึ้นมาสามารถอ่านและทำงานได้มากเกินจำเป็น
642-646

ในส่วนของผลิตภัณฑ์มี2ทางที่จะเข้าถึงได้เพื่อที่จะมาให้ระบบ multiple-processor เพื่อสนับสนุนการประยุกต์ใช้คือ SMPs และ cluster สำหรับบางปีการเข้าถึงในทางอื่นๆและผลิตภัณฑ์ของ NUMA ซึ่งเกี่ยวกับการค้าได้ค้นคว้าเมื่อไม่นานมานี้ ก่อนการดำเนินงานต่อไปเราจะกำหนดขอบข่ายในบางส่วนของ NUMA ในการวิจัยเฉพาะสาขานั้นๆ

-**Uniform memory access (UMA) :** processorทั้งหมดเข้าถึงในส่วนทั้งหมดของmain memory การใช้loads and stores memory ยอมรับเวลาของprocessor ก็เช่นเดียวกับบริเวณทั้งหมดของmemory The access times จะถูกเลือกจากหลากหลายprocessorsเช่นเดียวกันและองค์ประกอบของSMได้อธิบายแล้วในเนื้อหาอื่นคือUMA

-**Non uniform memory access (NUMA) :** processorsทั้งหมดได้เข้าถึงทุกส่วนของmain memory โดยการloads and stores The memory access time ของprocessor จะแตกต่างไปขึ้นอยู่กับบริเวณของ main memory ที่ถูกเข้าถึงคำกล่าวที่เป็นจริงของprocessorทั้งหมด อย่างไรก็ตามสำหรับการประมวลที่ต่างๆกัน memory regions อาจจะช้าหรือเร็ว

-**Cache-coherence NUMA (CC-NUMA) :** ระบบNUMA cache coherenceจะถูกบำรุรักษาระหว่างcache ของการประมวลผลต่างๆ

ระบบNUMA จะปราศจากcache coherence ซึ่งจะมีปริมาณมากหรือน้อยที่จะcluster (รวมตัวกัน)ผลิตภัณฑ์เกี่ยวกับการค้าได้ถูกยอมรับในการพิจารณาเมื่อไม่นานมานี้คือ ระบบCC-NUMAซึ่งพิเศษทั้งหมดจากทั้งคู่คือ SMPs and cluster ตัวอย่างระบบอยู่ในความจริงถูกกล่าวถึงในเฉพาะสาขานั้นๆซึ่งเกี่ยวกับการค้าเช่นระบบ CC-NUMA

Motivation

Motivation กับระบบ SMP เกี่ยวกับการปฏิบัติ limit เพื่อของจำนวน/ตัวเลขโปรเซสเซอร์นั้นสามารถถูกใช้แล้ว cache effective scheme ลด traffic รถประจำทางระหว่างบางหนึ่งโปรเซสเซอร์และหน่วยความจำสำคัญ. เช่นของจำนวน/ตัวเลขโปรเซสเซอร์เพิ่มขึ้น, รถประจำทางนี้ traffic อีกด้วยเพิ่มขึ้นอีกด้วย คือรถประจำทางใช้เพื่อแลกเปลี่ยนcache-coherenceสัญลักษณ์ต่อไป adding เพื่อ burden ที่จุดบางส่วน becomes รถประจำทางการกระทำ bottleneck. การกระทำ degradation seems เพื่อ limit ของจำนวน/ตัวเลขโปรเซสเซอร์ใน SMP เพื่อโครงสร้างที่ใดที่หนึ่งระหว่าง 16 และ 64 โปรเซสเซอร์ สำหรับตัวอย่าง silicon graphics' กำลัง challenge SMP คือจำกัดเพื่อ 64 r 10000 โปรเซสเซอร์ในเดี่ยว system; โพ้นจำนวน/ตัวเลขนี้การกระทำ degrades substantially.

โปรเซสเซอร์ LIMITใน SMP หนึ่งคือของ movtivations driving behind ของการพัฒนา clusterระบบ อย่างไรก็ตามกับcluster แต่ละ node มีมัน own ส่วนตัวสำคัญ memory; โปรแกรมประยุกต์ไม่พบกว้างใหญ่ global หน่วยความจำ. ใน effect, coherency คือ maintained ในค่อนข้างโปรแกรมกว่าฮาร์ดแวร์. หน่วยความจำนี้ granularity affects และการกระทำเพื่อ achieve maximum การกระทำ โปรแกรมจะ

ต้องเป็น tailored เพื่อนี้ environment. หนึ่ง approach เพื่อ achieving large-scale multiprocessing ในขณะที่ retaining ของ flavor SMP คือ NUMA สำหรับตัวอย่าง graphics silicon origin NUMA ระบบถูกออกแบบเพื่อรองรับถึง 1024 MIPS 10000 โปรเซสเซอร์. [WHIT97] และ NUMA-Q sequent ระบบถูกออกแบบเพื่อรองรับถึง 252 pentium ii โปรเซสเซอร์ [love96].

objective กับ NUMA คือเพื่อคงไว้ transparent system-wide หน่วยความจำในขณะที่ permitting multiple multiprocessor nodes, แต่ละกับมัน own รถประจำทางหรืออื่น internal interconnect ระบบ.

Organization

รูปร่าง 16.10 depicts typical CC-NUMA organization. มี multiple independent nodes แต่ละของที่ซึ่งคือ, ใน effect, organization SMP. THUS แต่ละ node บรรจุ multiple โปรเซสเซอร์, แต่ละกับมัน own l1 และ l2 caches, plus หน่วยความจำสำคัญ. คือ node อาคารพื้นฐาน block ของ CC-NUMA overall organization สำหรับตัวอย่าง, แต่ละ silicon graphics origin node รวมอยู่ MIPS000 processors แต่ละ sequent NUMA-Q node รวมอยู่สี่ pentium ii โปรเซสเซอร์. nodes คือ interconnected โดยหมายถึงของบางส่วน communications facilityที่ซึ่งจะสามารถเป็น switching กลไก , แหวน, หรือบางส่วนอื่น networking facility.

แต่ละ node ใน CC-NUMA รวมอยู่ระบบบางส่วนหน่วยความจำสำคัญ. จากของจุดภาพของโปรเซสเซอร์ อย่างไรก็ตามมีเพียงเดี่ยว addressable หน่วยความจำกับแต่ละ location having unique system-wideที่อยู่เมื่อโปรเซสเซอร์ initiates ทางเข้าหน่วยความจำ, ถ้าหน่วยความจำ requested location ไม่ใช่ใน processor's นั้น cache, จากนั้น cache l2 initiates fetch การปฏิบัติการถ้าบรรทัดต้องการคือในท้องถิ่นส่วนแบ่งสำคัญหน่วยความจำของคือบรรทัด fetched across รถประจำทางท้องถิ่น ถ้าบรรทัดต้องการคือใน remote ส่วนแบ่งของหน่วยความจำสำคัญ, จากนั้น request อัตโนมัติถูกส่งข้างนอกเพื่อ fetch บรรทัดนั้น across เครือข่าย interconnection, deliver มันเพื่อรถประจำทางท้องถิ่น, และจากนั้น deliver มันเพื่อ cache requesting บนรถประจำทางนั้น. ตลอด/ทุกๆของกิจกรรมนี้คืออัตโนมัติและ transparent เพื่อและโปรเซสเซอร์มัน cache.นนี้โครงสร้าง cache

coherence คือ central เกี่ยวข้องalthough implementations differ เช่นเพื่อรายละเอียด, ในเรื่องgeneralเราสามารถบอกว่าแต่ละ node จะต้องคงไว้พวกบางส่วนของไดเร็กทอรี่นั้น gives มันของindication ของlocationต่างๆ ส่วนหนึ่งของmemoryและข่าวสารสถานภาพของcacheและเราสามารถได้ตัวอย่างจาก [PFIS98]การเรียงลำดับตำแหน่งที่มาก่อนจะอธิบายdataคืออ่านจากmemoryระยะไกลโดยการใช้กลไกการทำงานของhardwareซึ่งทำการดำเนินการที่ชัดเจนในการประมวลผล กลไกการทำงานที่นิยมบางcache coherence แบบร่างที่ต้องการระบบต่างๆที่แตกต่างกันและถูกต้องเป็นที่ยอมรับ อันดับแรกเช่นเดียวกับส่วนของตำแหน่งที่มาก่อน directory ของnode1จะเก็บrecordซึ่งบางremote cache ซึ่งจะcopy line ที่บรรจุไว้ เมื่อต้องการที่ร่วมมือร่าง(แบบร่าง)ในการดูแลในเรื่องการปรับปรุงสำหรับตัวอย่าง ถ้าการปรับปรุงถูกต้องในcacheและเราสามารถออกข่าวnodeอื่นๆ

directoryของnodeบางชิ้นนั้นซึ่งยอมรับเช่นเดียวกับการออกข่าวสามารถได้ข้อสรุปถ้าlocal cacheซึ่งมีlineและถ้าเป็นสาเหตุที่ทำให้มันถูกขจัด ถ้าlocation memoryที่แท้จริงที่เป็นnodeในฐานะผู้รับการแจ้งข่าวในขณะนั้น directoryของnodeต้องการคงอยู่ของสิ่งที่บอกการเข้าซึ่งlineของmemoryไม่เหมาะที่จะใช้ต่อไป

และยังคงอยู่จนกระทั่งwrite backเกิดขึ้น ถ้าการประมวลผลอื่นเรียกร้องlineที่ไม่เหมาะจะใช้ต่อไปขณะนั้นlocal directoryจะบังคับให้write back update memoryก่อน

NUMA Pros และ Cons

ข้อดีหลัก ๆ ของ CC-NUMA คือ มันสามารถส่งผลให้ประสิทธิภาพที่ระดับที่สูงกว่าเมื่อเทียบกับ SMP ในระดับเดียวกัน ความต้องการภายนอกส่วนใหญ่ของการเปลี่ยนแปลงซอฟต์แวร์ประกอบด้วยโหนด NUMA หลายๆโหนดและการเดินทางของบัสบนโหนดพิเศษจำเพาะอันใดอันหนึ่งเป็นข้อจำกัดไปยังความต้องการสิ่งนั้นบัสสามารถที่จะจัดการได้ อย่างไรก็ตามถ้าการเข้าถึงหน่วยความจำไปยังโหนดที่ห่างไกลเป็นจำนวนมากประสิทธิภาพก็จะเริ่มน้อยลงเหตุผลที่เชื่อว่าประสิทธิภาพน้อยลงสามารถหลีกเลี่ยงได้ ขั้นแรกโดยใช้แคช L1และL2 ออกแบบให้การเข้าถึงหน่วยความจำที่ห่างออกไป ถ้ามีซอฟต์แวร์ที่ดีมากขึ้นตามเวลาไปด้วยเวลานั้นหน่วยความจำที่ห่างไกลก็จะไม่เกินขีดความสามารถ ขั้นที่สองถ้าซอฟต์แวร์มีตำแหน่งที่ว่างที่ดีและถ้าหน่วยความจำเสมือนที่ใช้เวลานั้นข้อมูลที่ต้องการใช้จะขึ้นอยู่กับการจำกัดของการใช้หน้าเสมอๆ สิ่งนั้นสามรถที่จะเริ่มบรรจุไปยังหน่วยความจำเฉพาะที่ได้แล้วส่งต่อไปดำเนินการประยุกต์ ท้ายที่สุดแผนการหน่วยความจำเสมือนสามารถที่จะเพิ่มได้โดยการรวมในระบบปฏิบัติการหน้าหนึ่งกลไกการย้ายที่นั้นคือเคลื่อนที่จากหน่วยความจำเสมือนไปยังโหนดนั้นอย่างเสมอๆ

ข้อเสียของการเข้าถึง CC-NUMA ดูเหมือนว่าจะดี โดยพิจารณารายละเอียด 2 ประการใน[PFIS 98] ข้อแรก CC-NUMA ไม่เข้าใจง่ายเหมือน SMP การเปลี่ยนแปลงซอฟต์แวร์คือ ความต้องการเคลื่อนที่ไปยังระบบปฏิบัติการหนึ่งและการนำไปใช้จาก SMP หนึ่งไปยังระบบ CC-NUMA รวมไปถึงการจัดสรรหน้าการบันทึกจนเสร็จ การกำหนดกระบวนการและการบรรจุให้สมดุลโดยระบบปฏิบัติการ ข้อสองเกี่ยวข้องกับลักษณะการใช้งานโดยการใช้งานของระบบ CC-NUMA ค่อนข้างที่จะมีผลลัพธ์ที่ซับซ้อนและขึ้นอยู่กับความถูกต้องของการปฏิบัติของระบบ CC-NUMA

การคำนวณเวกเตอร์

ถึงแม้ว่าประสิทธิภาพของเมนเฟรมคอมพิวเตอร์เอนกประสงค์จะปรับปรุงต่อไปอย่างไม่หยุดการนำไปใช้จะช้ากว่าของเมนเฟรมในรุ่นเดียวกันเป็นความต้องการคอมพิวเตอร์ที่แก้ปัญหาทางคณิตศาสตร์ปัญหาของกระบวนการที่แท้จริงอย่างเช่น เกิดขึ้นในวิชาการรวมไปถึงแอโรไดนามิกและวิชาที่ว่าด้วยแผ่นดินไหว อุตุนิยมวิทยา ออโตมิก นิวเคลียร์และพลาสมาฟิสิกส์

ปัญหาอีกอย่างหนึ่งคือ การบรรยายลักษณะภาพโดยต้องการความแม่นยำสูงและโปรแกรมที่เกิดขึ้นใหม่กระทำกับการปฏิบัติการจุดลอยตัวเลขคณิตบนอาเรย์ขนาดใหญ่ของจำนวน โดยปัญหาส่วนมากจะเกิดขึ้นแบ่งเป็นประเภทที่รู้จักเหมือน CONTINUOUS-FIELD SIMULATION ในสิ่งจำเป็นทางสภาวการณ์ทางกายภาพความสามารถที่จะอธิบายโดยพื้นผิวและบริเวณใน 3 มิติ พื้นผิวนั้นประมาณโดยจุด นิยามความแตกต่างของชุดสมการการปฏิบัติการทางกายภาพของพื้นผิวทุกจุด สมการเหมือนการแสดงอาเรย์หนึ่งของค่าและสัมประสิทธิ์และผลเฉลยรวมอยู่ด้วย การดำเนินการย้อนกลับทางเลขคณิตบนอาเรย์ของข้อมูล

การจัดการกับชนิดของปัญหา ซูเปอร์คอมพิวเตอร์มีการพัฒนาสัญลักษณ์ความสามารถพิเศษของเครื่องกลหนึ่งร้อยของหนึ่งล้านของการปฏิบัติการจุดลอยตัวต่อวินาทีและราคาใน 10 ถึง 15 ล้านดอลลาร์ในความแตกต่างของเมนเฟรม ซึ่งการออกแบบมัลติโปรแกรมมิ่งและอินพุท-เอาท์พุท ซูเปอร์คอมพิวเตอร์คือสูงสุดสำหรับการคำนวณชนิดตามตัวเลขที่เที่ยงตรง

ซูเปอร์คอมพิวเตอร์มีขีดจำกัดการใช้เพราะว่าราคาและความจำกัดทางการตลาด โดยเปรียบเทียบกับความเล็กน้อยของเครื่องกลเป็นการดำเนินการจากศูนย์กลางการค้นคว้าโดยมากและตัวแทนของบางรัฐบาลประกอบด้วยหน้าที่ของวิทยาศาสตร์และวิศวกรรมเหมือนประกอบด้วยพื้นที่อื่นๆของคอมพิวเตอร์เทคโนโลยี ค่าคงที่นั้นขึ้นอยู่กับการเพิ่มประสิทธิภาพของซูเปอร์คอมพิวเตอร์ ปัจจุบันการประยุกต์ในแอโรไดนามิกและนิวเคลียร์ฟิสิกส์มากเหมือน 10^{13} การปฏิบัติการเลขคณิตเป็นที่น่าสนใจมากกว่าเวลาของคอมพิวเตอร์ 2 วันในซูเปอร์คอมพิวเตอร์รุ่นเดียวกัน ความต้องการสำหรับปัญหาเดี่ยว ดังนั้นเทคโนโลยีและประสิทธิภาพของซูเปอร์คอมพิวเตอร์ค่อยๆเปลี่ยนไป

ชนิดอื่นๆของระบบนั้นมีการออกแบบที่อยู่ที่ต้องการสำหรับการคำนวณเวกเตอร์โดยเหมือนหน่วยประมวลผลของอาเรย์ ถึงแม้ว่าซูเปอร์คือสูงสุดสำหรับการคำนวณเวกเตอร์มันเป็นคอมพิวเตอร์เอนกประสงค์ ความสามารถของการจัดการหน่วยประมวลผลสเกลาร์และงานการประมวลผลข้อมูลทั่วไปทั่วไปหน่วยประมวลผลอาเรย์ไม่รวมถึงการประมวลผลของสเกลาร์ พวกนั้นเป็นโครงแบบเหมือนอุปกรณ์ภายนอกโดยผู้ใช้เมนเฟรมและมินิคอมพิวเตอร์

<u>การคำนวณในการเข้าถึงเวคเตอร์</u>

หลักในการออกแบบ supercomputer หรือ อาร์เรย์ ให้ยอมรับนั้นหน้าที่หลักคือการคำนวณด้านเลขคณิต บนอาร์เรย์หรือเวคเตอร์ ของเลขทศนิยม จุดประสงค์ทั่วไปของคอมพิวเตอร์,ต้องการสั่งให้ทำซ้ำผ่านทางอาร์เรย์ ตัวอย่างเช่น พิจารณาที่เวคเตอร์ 2 ตัว(อาร์เรย์ 1 มิติ) ของตัวเลข, A,Bและ C เราต้องการที่จะเพิ่มทั้ง a และb เก็บผลลัพธ์ไว้ที่ c ในตัวอย่าง16.11 นี้ต้องการเพิ่มต่างหาก(this requirs six addition), จะเพิ่มความเร็วในการคำนวณอย่างไร ? มีหลายด้านที่จะให้การคำนวณเวคเตอร์ประสบผลสำเร็จ เราจะอธิบายโดยมีภาพประกอบ พิจารณาการคูณเวคเตอร์ C=A*B, เมื่อ A,Bและ C เป็นเมตริกจัตุรัส สูตรของ C คือ

$$c_{i,j} = \sum a_{i,k} * b_{k,j}$$

เมื่อ A,B,C ประกอบด้วย $a_{i,j}$, $b_{i,k}$, $c_{i,j}$ ตามลำดับในตัวอย่าง16.12 แสดงโปรแกรมภาษา FORTRAN ของการคำนวณสเกลลาแบบทั่วไป

สิ่งหนึ่งที่เป็นการปรับปรุงประสิทธิภาพที่เกี่ยวข้องกับ **<u>vectros processing</u>** ตัวอย่างที่ 16.12 เป็นภาษา FORTRAN กับโครงสร้างใหม่ที่อนุญาตให้คำนวณเวคเตอร์ที่ระบุได้ สัญกรณ์ (J=1,N) บอกถึงดัชนีทั้งหมด J ในวงเล็บกระทำใน single operation

โปรแกรมใน 16.12b ชี้ให้เห็นถึงส่วนประกอบทั้งหมดของ i แถวเป็นตัวคำนวณใน parallel บางส่วนในแถวเป็นผลรวมและผลรวมนั้น (ข้าม k) เป็นขั้นๆ การคูณในเวคเตอร์ N2 เท่านั้นที่สั่งให้อัลกอริทึมเปรียบเทียบกับ N3 สเกลลาอัลกอริทึม

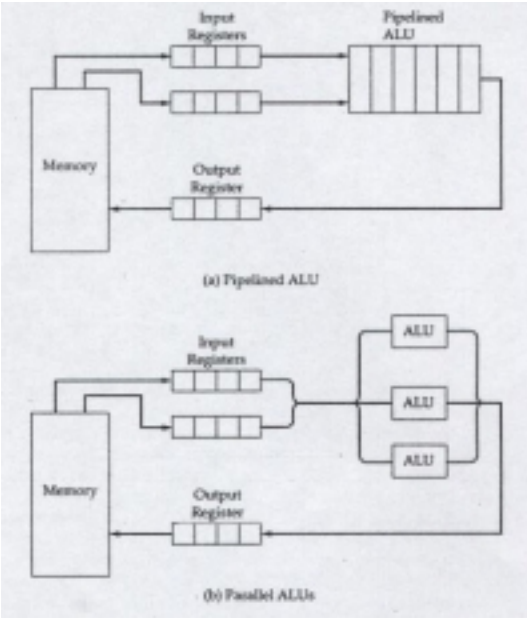
ส่วนการเข้าถึงแบบอื่นๆ , <u>parallel processing</u> อธิบายใน 16.12c <u>parallel processing</u> มี N เป็นตัวแปลอิสระ เรามักจะเลือกใช้เพยงวิธีใดวิธีหนึ่ง แบ่งส่วนในการคำนวณไปที่หน่วยประมวลผลหลายตัว ใน 2 วิธีแรก วิธีที่1 FORK เป็นกรณีอิสระเริ่มทำงานที่ n ในระหว่างนั้นจะปฏิบัติตามคำสั่งที่อยู่ใน FORK ทุกครั้งที่การทำงานที่ FORK จะทำให้เกิดการคำนวณครั้งใหม่ , JOIN เป็นคำสั่งสำคัญที่แปลผกผันกับ FORK .

กรณีคำสั่ง JOIN N เป็นการดำเนินการอิสระเป็นการรวมกันของ 1 คำสั่งส่งไปที่ JOIN ระบบปฏิบัติการ(OS) จะรวมเข้าด้วยกัน, และกระทำต่อจนกระทั่งN ครั้ง 3 หัวข้อเด่นๆ :

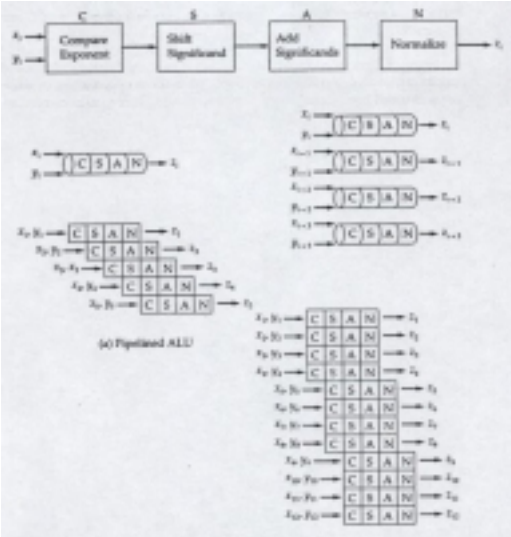
-PipelineALU

-Parallel ALU

-Parallel processors



รูป 16.13



รูป 16.14

รูป 16.13 อธิบายถึงสองวิธีแรก คำอธิบายในเรื่อง pipelining อยู่ในบทที่ 11 เสนอเกี่ยวกับการคำนวณการทำงานของ ALU ตัวเลขทศนิยมค่อนข้างจะซับซ้อน มีคำอธิบายพร้อมภาพประก อยู่ในตัวอย่างที่

16.14 a ในตัวอย่างนี้จะแบ่งตัวเลขทศนิยมเป็น 4 ตำแหน่ง : เปรียบเทียบ , shift , add , normalize ตัวเลขในตำแหน่งแรกของเวคเตอร์เป็นการแสดงเหตุการณ์ที่ต่อเนื่องกัน 4 ส่วนของตัวเลขจะเกิดขึ้นที่ pipeline

หน่วยประมวลผล จะไปดึงข้อมูลและทำงานซ้ำๆกันลักษณะเป็นวงกลม pipeline จะรักษาไว้จนเต็มและเก็บเวลาที่ได้รับ คล้ายกับว่า pipeline เป็นการเก็บเวลาไว้เท่านั้น pipelined ALU เป็นกลุ่มของตัวแปรข้อมูลในพื้นที่ที่ต่อเนื่องกัน การทำงานของตัวเลขทศนิยมจะแยกไว้ต่างหากไม่ได้ไปเพิ่ม speedup โดย pipeline speedup เป็นผลสำเร็จจากเวคเตอร์ของตัวดำเนินการแสดงไปที่ ALU

Pipeline สามารถทำงานเพิ่มมากขึ้นถ้าส่วนประกอบของเวคเตอร์ซึ่งสามารถหามาได้ในรีจิสเตอร์มากกว่าในmain memory ดังแสดงไว้ในตัอย่าง 16.13a ส่วนประกอบบางส่วนของเวคเตอร์ดำเนินการเป็นตัวโหลด บล็อคในเวคเตอรรีจิสเตอร์ , เวคเตอร์ดำเนินการเปรียบเสมือนธนาคารขนาดใหญ่ของรีจิสเตอร์ , การทำงานของเวคเตอร์ที่เข้าไปในเมมโมรีจะเริ่มต้นและจบที่นี่

กลไกการทำงานอธิบายไว้ในภาพตัวอย่าง 16.14 กล่าวถึงการทำงานภายในของ pipelining เช่นการทำงานอันเดียว (C=A+B) pipelining ยอมให้มีการคูณเวคเตอร์สำหรับการทำงาน แบบ parallel กลไกนี้สามารถทำงานเพิ่มมากขึ้นกับ *pipelining across opertion* . สิ่งหลัง, มีลำดับการทำงานของเวคเตอร์เลขคณิต, และโครงสร้าง pipelining เป็นการเพิ่มความเร็วในการทำงาน

การเข้าถึงอีกวิธีหนึ่ง, กล่าวถึง *chaining* เป็นการค้นพบบนCray supercomputers, หลักเกณฑ์พื้นฐานของ *chaining* กล่าวคือ การเริ่มทำงานของเวคเตอร์จะเร็วเท่ากับส่วนแรกของ เวคเตอร์ดำเนินการที่สามารถหามาได้และหน่วยของฟังก์ชั่น (เช่น บวก ลบ คูณ หาร) โดยพื้นฐาน , chaining ใช้แก้ปัญหาจากหนึ่งหน่วยฟังก์ชั่นในทันทีภายในฟังก์ชั่นอื่นหรือที่เกี่ยวข้อง ถ้ารีจิสเตอร์ของเวคเตอร์เป็นตัวใช้, ผลลัพธ์จะไม่มีการเก็บในเม็มโมรีและสามารถใช้ก่อนที่การทำงานเวคเตอร์จะทำงานเสร็จสบูรณ์

ตัวอย่าง, เพื่อคำนวณ C=(s*A)+B , เมื่อ A,Bและ C เป็นเวคเตอร์และ s เป็นสเกลลา, Cray จะกระทำ 3 คำสั่งครั้งแรก ส่วนสำคัญในการดึงข้อมูลออกสำหรับบรรจุเข้าไปในตัวคูณใน pipelined ทันที , ผลลัพธ์ที่ได้จะส่งไปที่ตัวเพิ่มของ pipeline , และผลรวมเป็นพื้นที่ในรีจิสเตอร์ของเวคเตอร์เท่ากับตัวบวกที่สมบูรณ์

ตัวอย่างที่ดีขององค์กร ALU pipelined สำหรับเป็นvectorการประมวลผลคือความสะดวกเป็น vector พัฒนาสำหรับ IBM 370 สถาปัตยกรรมและเพิ่มบน highend 3090 ชุด[PADE88,พับ87].
ความสะดวกนี้คือข้อกำหนดเพิ่มเติมที่เพิ่มบนเพื่อระบบพื้นฐานแต่ถูกรวมอย่างสูงกับมัน resembles เป็นvector ความสะดวกค้นพบบน supercomputers, เช่น ครอบครัว Cray.

ความสะดวก IBM ทำให้การใช้ของregisterเป็นvectorจำนวนมาก. แต่ละregisterคือธนาคารของc scalar register เพื่อคำนวณเป็นvectorรวม C =A+B, เป็นvectorและ B ถูกโหลดเข้าไปใน 2 registerเป็นvector register fromthese ข้อมูลถูกผ่าน througth ALU เร็วเท่ากับเป็นไปได้, และผลลัพธ์ถูกผ่าน
ตั้งอยู่ใน registerเป็นvector 1 ใน 3. กระบวนการคำนวณซ้อนกัน, และการโหลดของข้อมูลสิ่งที่นำเข้าเข้าไปใน the registerในกล่อง, ผลลัพธ์ใน สำคัญการเพิ่มความเร็วข้ามการคำนวณ ALU ปรกติ.
องค์กร

สถาปัตยกรรมเป็นvector IBM, และ ALUs เป็นvector pipelined ที่คล้าย, ตระเตรียม

การชำระหนี้ที่เพิ่มข้ามวนซ้ำของ scalar คำสั่งเลขคณิตใน 3 วิธี.

- ที่ซ่อมและโครงสร้าง predetermined ของข้อมูลเป็นvectorอนุญาติ housekeeoing คำสั่งข้างในthe วนซ้ำเพื่อถูกเคลื่อนย้ายโดยเร็วกว่าภายใน (อุปกรณ์หรือ microcoded) การคำนวณเครื่อง.

-การเข้าถึงข้อมูลและเลขคณิต opertionns บนธาตุเป็นvector successiveมากมาย สามารถดำเนินต่อไปอย่างร่วมกันโดยการซ้อนกันการคำนวณเช่นนั้นใน pipelined design หรือโดยการกระทำการคำนวณต่างๆใน parallel.

-การใช้registerเป็นvectorสำหรับส่งเพื่อหลีกเลี่ยงข้อมูลที่ใช้บันทึกเพิ่มเติม

รูป 16.16 การแสดงองค์กรทั่วไปของความสะดวกเป็นvector

ความสะดวกเป็นvectorถูกเห็นเพื่อเป็นที่แยกออกมาทางกายภาพที่เพิ่มบนเครื่องประมวลผล, สถาปัตยกรรมของมัน คือชนิดของระบบ/370 สถาปัตยกรรมและคือที่สามารถเข้ากันได้กับมัน.

ความสะดวกเป็นvectorถูกรวมเข้าไปในระบบ/370 สถาปัตยกรรมในวิธีดังต่อไปนี้:

-การมีอยู่ระบบ/370 instructionns ถูกใช้สำหรับ การคำนวณ scalarทั้งหมด.

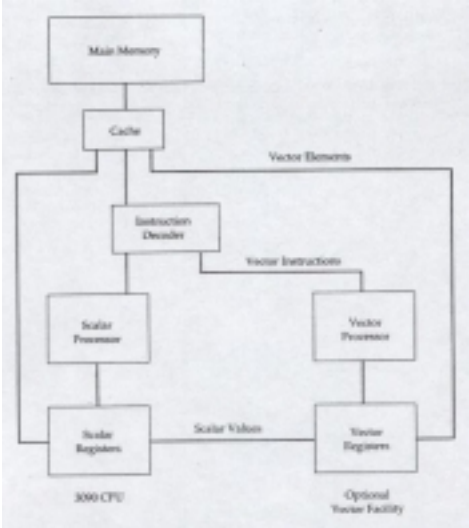
-การคำนวณเลขคณิตบนธาตุเป็นvectorบุคคลผลิตอย่างแน่นอนเหมือนกัน ผลลัพธ์เป็นทำที่คล้ายคลึงระบบ/370 คำสั่ง scalar. ยกตัวอย่างเช่น ,คำสั่งการตัดสินการออกแบบสนใจนิยามของผลลัพธ์ในจุดซึ่งลอยอยู่ หารการคำนวณ. ถ้าผลลัพธ์แน่นอน, เป็นมันสำหรับ scalar จุดซึ่งลอยอยู่แผนก, หรือควรการโดยประมาณถูกยอมซึ่งจะอนุญาติสูงกว่าเร่งความเร็วเครื่องมือแต่สามารถบางเวลาแนะนำข้อผิดพลาดในข้อผิดพลาดหรือเพิ่มเติมตำแหน่งบิทคำสั่งต่ำ? descision ถูกทำเพื่อ uphold ความเข้ากันได้ที่สมบูรณ์กับระบบ/370 สถาปัตยกรรมที่expense ของการชำระหนี้น้อย degradation.

-คำสั่งเป็นvectorคือ interruptible, และการทำให้สำเร็จของเขาทั้งหลายสามารถถูกต่อจากจุดของการชะงักหลังจัดสรรการกระทำได้ถูกใช้, ในกริยาอาการที่สามารถเข้ากันได้กับระบบ/370 แผนการชะงักโปรแกรม.

-ข้อยกเว้นเลขคณิตเหมือนกับ ,หรือชนิดของ,ข้อยกเว้นสำหรับthe scalar คำสั่งเลขคณิต ระบบ/370, และทำงานประจำที่ซ่อมขึ้นที่คล้ายสามารถถูกใช้. Toaccommodate สิ่งนี้, ดัชนีการชะงักเป็นvectorถูกจ้างดัชนีนั้นแสดงที่ตั้งใน register เป็นvectorว่าถูกมีผลต่อโดยข้อยกเว้น(e.q., overflow) ด้วยเหตุนี้, เมื่อการทำให้สำเร็จของคำสั่งเป็นvector ต่อ, คนเหมาะสม สถานที่ในregisterเป็นvectorถูกเข้าถึง

-ข้อมูลเป็นvectorอาศัยอยู่ในเสมือนจริงที่ใช้บันทึก, กับความผิดพลาดหน้าการถูกจัดการในกริยาอาการมาตรฐาน.

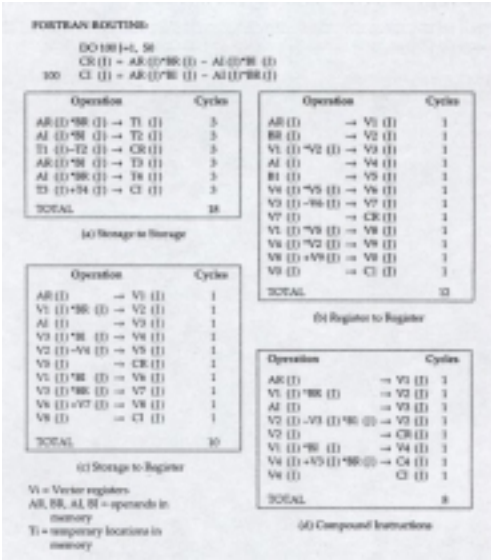
ระดับนี้ของการรวมตระเตรียมคุณประโยชน์จำนวนมาก. การมีอยู่ระบบปฏิบัติการสามารถสนับสนุนความสะดวกเป็นvectorกับชนิดน้อย. การมีอยู่โปรแกรมแอปปลิเคชั่น,ผู้ตรวจสอบภาษา, และซอฟแวร์อื่นๆสามารถคือวิ่งไม่เปลี่ยนแปลง. ซอฟแวร์ซึ่งสามารถ ใช้ข้อได้เปรียบของความสะดวกเป็นvectorสามารถถูกแก้ไขตามที่ dasired.



รูป 16.16

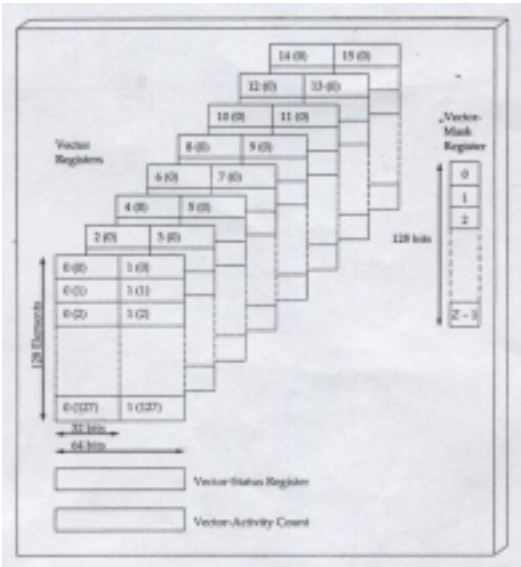
Register

ความรู้คีย์ในการออกแบบของความสะดวกเป็นพาหะคือไม่ว่า operands ถูกค้นหาที่ตั้งในลงทะเบียนหรือหน่วยความจำ. องค์กร IBM ถูกอ้างอิงเพื่อเป็นลงทะเบียน-เพื่อ-registerเพราะว่าเป็นvector operands, ทั้งสิ่งที่นำเข้าและสิ่งที่นำออก, สามารถถูกในregisterเป็นvector สิ่งนี้เข้าใกล้ยังถูกใช้บน Cray supercomputer.หัวข้อเข้าใกล้, ใช้บนเครื่องข้อมูลคอนโทรล, ต้องรับ operands โดยตรงจากหน่วยความจำ ไม่ข้อได้เปรียบหลักของการใช้ของregisterเป็นvectorคือว่านักเขียนโปรแกรมหรือผู้ตรวจสอบต้องพาพวกเขาเข้าไปในregister. ยกตัวอย่างเช่น, สมมติว่าความยาวของregisterเป็นvectorคือK และความยาวของเป็นvectorเพื่อถูกประมวลผลคือ N>K. ในสิ่งนี้ทางเลือก, เป็นvectorวนซ้ำต้องถูกกระทำ. ซึ่งการคำนวณถูกกระทำบน K ณ 1 เวลา และวนซ้ำคือเวลา N/K ที่วนซ้ำ. ข้อได้เปรียบหลักของregisterเป็นvecterเข้าใกล้คือว่าการคำนวณคือว่า decoupled จากหน่วยความจำหลักช้ากว่าและแทนใช้สถานที่ primarily กับregister speedup ซึ่งสามารถถูกบรรลุการใช้registerถูกแสดงในรูป 16.17 [PADE88]. ทำงานประจำ FORTRAN คูณเป็นพาหะA โดยB เป็นพาหะเพื่อผลิต C เป็นvector, ที่ซึ่งแต่ละเป็นvectorมีส่วน (AR,BR,CR) จริงๆ และ imaginaryส่วน (AI,BI,CI). The 3090 สามารถกระทำหนึ่งการเข้าถึงที่ใช้บันทึกหลักต่อเครื่องประมวลผล, หรือนาฬิกา, รอบ(อันใดอันหนึ่งอ่านหรือเขียน), registerซึ่งมีสามารถ sustain 2 การเข้าถึงสำหรับการอ่านและหนึ่งสำหรับการเขียนต่อรอบ, และผลิตหนึ่งผลลัพธ์ต่อรอบในหน่วยเลขคณิตของมัน.ให้เราทึกทักเอาการใช้ของคำสั่งซึ่งสามารถเจาะจง 2 ข้อมูลoperands และผลลัพธ์. แบ่งของตัวเลขแสดงตัวเลขนั้นกับหน่วยความจำที่เพื่อในการทำซ้ำกระบวนการคำนวณแต่ละครั้งใช้ทั้งหมด 18 cycle แต่ตอนนี้ได้ลดลงเหลือ 12 cycle เท่านั้น อย่างไรก็ตามปริมาณ vector จะถูกบรรจุใน vector register ตามลำดับ เพื่อจะนำไปคำนวณและเก็บไว้ใน หน่วยความจำในภายหลังสำหรับ vector ใหญ่ๆ ค่าที่ถูกปรับหรือเสียเวลาไปนั้นเมื่อสังเกตแล้วจะน้อยมาก



รูป 16.17

รูปที่16.17 จะแสดงถึงการเก็บข้อมูลทั้งสองแบบของหน่วยความจำ และเครื่องหมายใน register ของหนึ่งคำสั่ง ซึ่งสามารถที่จะลดเวลาลงได้ 10 cycle ต่อการทำซ้ำหนึ่งครั้ง ซึ่งต่อมาชนิดของคำสั่งได้ถูกรวบรวมไว้ในสถาปัตยกรรมของ vector



รูป 16.18

รูป 16.18 จะแสดงให้เห็นเป็นตัวอย่างของ register ซึ่งอยู่ใน IBM 3090 เป็น vector อย่างง่ายมี vector register จำนวน 16 – 32 บิต และสามารถจับคู่เป็น vector register แบบ 864 บิต แต่ละ register ปกติแล้วจะมีรูปแบบค่าตัวเลขเป็นแบบจำนวนเต็ม ,จุดทศนิยม ดังนั้น vector register อาจจะใช้ แบบ 32 บิต และ 64 บิต สำหรับค่าแบบจำนวนเต็ม และ 32 บิตและ 64 บิตสำหรับค่าแบบจุดทศนิยม

ในทางสถาปัตยกรรมได้ระบุไว้ว่าแต่ละ register ปกติแล้วจะมี 8 – 512 หน่วย ตัวเลือกของความยาวที่จริงแล้วเกี่ยวข้องกับการออกแบบ trade-off เวลาที่ใช้ในการทำ vector operation ประกอบด้วยส่วนของ pipeline startup และ register ตัวที่ใช้บวกกับ 1 cycle ต่อ 1 หน่วย vector ด้วยเหตุนี้ประโยชน์ของเลขจำนวนที่มากๆ ของ register จะต้องถูกทำให้สมดุล โดยบวกกับเวลาที่ต้องการเก็บข้อมูลและการนำข้อมูลกลับไปยังที่เดิมของ vector register ใน process switch และบวกกับค่าเวลาที่เหมาะสมและเขตของช่องว่าง ทั้งหมดนี้ได้ถูกนำไปใช้ประโยชน์ในแบบ 128 หน่วยต่อ 1 registerใน 3090 ที่นิยมในปัจจุบัน

ในระบบการบวกของสาม register จะถูกใช้ใน vector อย่างง่าย register ใน vector-mask จะมี mask bit ซึ่งจะถูกใช้เลือกโดยหน่วยใน vector register เพื่อใช้ในการประมวลผลสำหรับคำสั่งเฉพาะ vector-status register จะมี field ควบคุมอยู่ เช่น ตัว vector count ซึ่งเป็นตัวกำหนดว่าจะใช้กี่หน่วยในการทำงานคำสั่งของ vector instructions

ส่วนประกอบของคำสั่ง

ก่อนนี้ได้พิจารณาว่าการจัดการคำสั่งสามารถที่จะทำการ overlapped โดยวิธีแบบลูกโซ่ เพื่อเพิ่มประสิทธิภาพ โดยที่ผู้ออกแบบ IBM vector อย่างง่ายจะไม่เลือกที่จะใช้วิธีแบบนี้ด้วยเหตุผล 2-3 ข้อ ระบบ/370 ของสถาปัตยกรรมจะต้องมีการขยายเพื่อจัดการกับ interruption ที่มีความซับซ้อน และการเปลี่ยน

corresponding จะถูกใช้ใน software ความรู้พื้นฐานอีกอย่างหนึ่งคือ ค่าของการรวมของตัวเพิ่มหน่วยควบคุมและ register ที่เป็นทางเข้าไปใน vector อย่างง่ายสำหรับ ระบบ chaining ทั่วไป

การกระทำทั้งสามจะถูกเตรียมโดยเข้าไปรวมในขั้นตอนหนึ่ง (ขั้น opcode) ส่วนใหญ่แล้วการเรียงลำดับในกระบวนการคำนวณของ vector , การคูณจะตามมาด้วย การลบ , การเพิ่มหรือการบวกรวม ใน storage-to-register คำสั่ง MULTIPLY-AND-ADD จะใช้ดังตัวอย่างนี้คือ การนำ vector ตัวหนึ่งมาจากตัวบันทึกตัวหนึ่งมาคูณด้วย vector จาก register ตัวหนึ่ง และบวกผลลัพธ์ที่ได้กับ vector ตัวที่สามใน register วิธีที่ใช้จากคำสั่ง MULTIPLY-AND-ADD และ MULTIPLY-AND-SUBTRACT ดังตัวอย่างในรูป 16.17 เวลารวมที่ใช้ในการทำนี้จะถูกลดลงจาก 10 เหลือ 8 cycle

อีกรูปแบบที่ไม่เหมือนของ chaining ส่วนประกอบคำสั่งที่ไม่ใช้ประโยชน์จากการเพิ่ม register สำหรับหน่วยความจำชั่วคราวของค่ากลางของผลลัพธ์ และยังต้องการ register เป็นตัวผ่านอย่างน้อยอีกหนึ่งตัว ดังตัวอย่างในรูปแบบของ chain

$$A \rightarrow VR1$$

$$VR1 + VR2 \rightarrow VR1$$

ในกรณีนี้การเก็บทั้งสองค่าใน register VR1 จะนำมาใช้ในสถาปัตยกรรม IBM มี storage-to-register หนึ่งตัว คำสั่ง ADD นี้จะมีการบวกเท่านั้นที่จะแทนที่ใน VR1 ส่วนประกอบของคำสั่งจะหลีกเลี่ยงที่ไปยัง machine-state ของตัวเลขหนึ่งตัวของคำสั่งซึ่งง่ายต่อการบันทึกและการเก็บข้อมูลคือที่โดย OSและการจัดการของ interrupts

Table 16.3 IBM 3090 Vector Facility: Arithmetic and Logical Instructions

Operation	Data Types: Floating-Point: Long	Short	Binary or Logical	Operand Locations			
Add	FL	FS	BI	V+V→V	V+S→V	Q+V→V	Q+S→V
Subtract	FL	FS	BI	V−V→V	V−S→V	Q−V→V	Q−S→V
Multiply	FL	FS	BI	V×V→V	V×V→V	Q×V→V	Q×S→V
Divide	FL	FS	—	V/V→V	V/S→V	Q/V→V	Q/S→V
Compare	FL	FS	BI	V:V→V	V:S→V	Q:V→V	Q:S→V
Multiply and add	FL	FS	—		V×V+S→V	V+Q×V→V	V+Q×S→V
Multiply and subtract	FL	FS	—		V−V×S→V	V−Q×V→V	V−Q×S→V
Multiply and accumulate	FL	FS	—	P+:V→V	P+:S:V		
Complement	FL	FS	BI	-V→V			
Positive absolute	FL	FS	BI	|V|→V			
Negative absolute	FL	FS	BI	-|V|→V			
Maximum	FL	FS	—			Q:V→Q	
Maximum absolute	FL	FS	—			Q:V→Q	
Minimum	FL	FS	—			Q:V→Q	
Shift left logical	—	—	LO	·V→V			
Shift right logical	—	—	LO	·V→V			
And	—	—	LO	V&V→V	V&S→V	Q&V→V	Q&S→V
OR	—	—	LO	V|V→V	V|S→V	Q|V→V	Q|S→V
Exclusive-OR	—	—	LO	V@V→V	V@S→V	Q@V→V	Q@S→V

Explanation: **Data Types**
FL Long floating point
FS Short floating point
BI Binary integer
LO Logical

Operand Locations
V Vector register
S Storage
Q Scalar (general or floating point register)
P Partial sums in vector register
· Operand operation

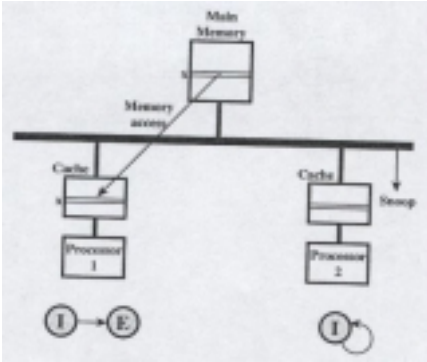
ตาราง 16.13

The instruction set

ตาราง 16.3 จะสรุปเกี่ยวกับตัวเลขและ logical operation ซึ่งจะเป็นตัวอธิบายให้กับสถาปัตยกรรมของ vector โดยมีคำสั่ง memory-to-register load และ register-to-memory store โดยข้อความนั้นคำสั่งหลายคำสั่งใช้รูปแบบ three-operand ด้วยเหตุนี้คำสั่งเหล่านั้น จึงมีตัวเลขของตัวแปรตัวหนึ่งขึ้นอยู่กับตำแหน่งของ operand , source operand 1 ตัว หรืออาจจะเป็น vector register 1 ตัว , storage , หรือ scalar register 1 ตัว แต่เป้าหมายมักจะเป็น vector register นอกจากสำหรับการเปรียบเทียบ ผลลัพธ์ของมันจะไปยัง vector-mask register ด้วยตัวแปรเหล่านี้เลขโดยรวมของ opcodes คือ171 ซึ่งเป็นจำนวนที่ดีมาก อย่างไรก็ตามก็ยังไม่แพงจนทำให้เป็นแค่ฝัน ครั้งหนึ่งเครื่อง machine ได้เตรียมหน่วยตัวเลขและเส้นทางของข้อมูลที่จะให้ค่า operands จากตัวเก็บข้อมูล scalar register และ vector register ไปยัง vector pipeline ค่า hardwareส่วนใหญ่จะทำให้เกิดความเสียหาย แต่สถาปัตยกรรมทำได้ด้วยความแตกต่างของค่าเพียงเล็กน้อย เตรียมตั้งค่าตัวแปรไว้มากๆ เพื่อใช้ประโยชน์ของ register เหล่านั้นและ pipeline

คำสั่งส่วนมากในตาราง16.3 จะอธิบายอยู่แล้วในตัวในคำสั่งการบวกทั้งสองคำสั่งจะอธิบายได้ดีกว่า operation การเพิ่มจะทำการบวกกันของหน่วยของ vector เดี่ยวหรือ หน่วยของผลลัพธ์ของ 2 vector คำสั่งเหล่านี้จะเสนอรูปแบบของปัญหาที่น่าสนใจเราจะต้องชอบใช้เพราะมันจะเร็วมาก การทำงานที่เต็มประสิทธิภาพของ ALU pipeline ความยากอยู่ที่การบวกของ 2 เลขที่ใส่ใน pipeline หาไม่ง่ายนัก กระทั่งผ่านไป 2-3 cycle จึงจะหาได้ เหตุนี้หน่วยที่ 3 ของ vector ไม่อาจเพิ่มโดยการบวกของ 2 หน่วยแรกกระทั่ง 2 หน่วยนั้นผ่านไปยัง pipeline ได้สมบูรณ์ เมื่อผ่านปัญหานี้ได้หน่วยใน vector จะถูกบวกดังเช่นการจะทำ partial sum 4 ตัว ในส่วนที่เฉพาะเจาะจงหน่วย 0,4,8,12,…,124 จะถูกบวกเพิ่มในคำสั่งนั้นเพื่อให้เกิด partial sum 0 ; หน่วย 1,5,9,13, …,125 อยู่ใน partial sum 1; หน่วย 2,6,10,14,…,126 อยู่ใน partial sum 2 ; หน่วย 3,7,11,15,…,127 partial sum 3; partial sum แต่ละตัวสามารถผ่านไปยัง pipelineโดยเร็วที่สุดเพราะว่าความช้าใน pipeline จะทำให้เสียเวลาไป 4 cycle vector

register ส่วนหนึ่งจะถูกใช้เก็บ partial sums เมื่อหน่วยทั้งหมดของ vector แรกๆจะทำการประมวลผล partial sum ทั้ง 4 จะถูกบวกกันให้ได้ผลลัพธ์สุดท้าย ประสิทธิภาพของเฟสที่2นี้ไม่น่าพอใจ เพราะมีเพียง 4 หน่วยvectorเท่านั้นที่เกี่ยวข้อง



รูป 16.19

16.7 RECOMMENDED READING

[CATA94]ของหลักการ multiprocessors และตรวจสอบ SPARC-based SMPs ในรายละเอียดอีกด้วย SMPs คือcovered ในรายละเอียดบางส่วนและใน [STON93] [WHAN93]. [PFIS98] คือ essential การอ่านสำหรับบางคนที่สนใจในหนังสือ clusters และฮาร์ดแวร์โปรแกรมการออกแบบ issues และ contrasts clusters กับ SMPs และ NUMAs อีกด้วย หนังสือบรรจุรายละเอียดทางเทคนิคของ SMP และ NUMA การออกแบบ issues

ดีเยี่ยม survey ของ relating issues เพื่อ cache coherence คือใน multiprocessors [LIJI93] [TOMA93] บรรจุ reprints ของกระดาษลูกกุญแจบนเรื่อง

ดี discussions ของ vector computation สามารถพบใน [STON93]และ[HWAN93].

16.8 PROBLEMS

16.1 ปล่อยให้ α เป็นของ percentage โปรแกรมโค้ดนั้นสามารถเป็น executed ทันเวลาเดียวกันโดย n โปรเซสเซอร์ในคอมพิวเตอร์ระบบ. ตั้งสมมติฐานนั้นโค้ด remaining จะต้องเป็น executed sequentially โดยเดี่ยว โปรเซสเซอร์. แต่ละโปรเซสเซอร์มีอัตรา execution ของ x MIPS

a. สำหรับการแสดงความเห็น MIPS effective อัตราเมื่อการใช้สำหรับระบบ exclusive execution ของโปรแกรมนี้ในของเรื่อง n , α และx

b. ถ้า n = 16 และ x = 4 MIPS การตัดสินใจของ value αนั้นจะยอมแพ้การกระทำระบบของ 40 MIPS.

16.2 multiprocessor กับ eight โปรเซสเซอร์มี 20 ได้ต่อติด tape ไดรว์. มีจำนวน/ตัวเลขกว้างใหญ่ของ jobs submitted เพื่อระบบนั้นแต่ละ require maximum ของ4 tape ไดรว์เพื่อสมบูรณ์ execution. ตั้งสมมติ

ฐานนั้นแต่ละ job เริ่มกำลังรันกับเพียงสาม tape ไดรว์สำหรับระยะเวลายาวก่อน requiring tape fourth ไดรว์สำหรับระยะเวลาสั้น toward ของจบการปฏิบัติการมันอีกด้วยตั้งสมมติฐานจัดให้ endless ของเช่นนั้น jobs.

a. ในOS scheduler จะไม่เริ่ม job unless มี4 tape ไดรว์เป็นไปได้. เมื่อ job คือ started, ไดรว์4 คือ assigned immediately และไม่ใช่การนำออกเผยแพร่จนกระทั่ง หมด job อะไรคือ maximum และจำนวน/ตัวเลขอย่างน้อยที่สุดของ tape ไดรว์นั้นจะซ้าย idle เช่น result ของนี้ policy?

b. แนะนำ policy alternative เพื่อปรับปรุง tape ไดรว์ utilization และที่เวลาเหมือน ๆ กันหลีกเลี่ยงระบบ deadlock. อะไรคือจำนวน/ตัวเลข maximum ของ job นั้นสามารถเป็นในที่ progress once?อะไรคือบนจำนวน/ตัวเลขสิ่งผูกมัดของ idling tape drives?

16.3 คุณสามารถ foresee ปัญหาบางกับ cache write-once approach บน -based รถประจำทาง multiprocessors? ถ้าดังนั้น, แนะนำ solution.

16.4 ไตร่ตรอง situation ในสองที่ซึ่งโปรเซสเซอร์ใน SMP โครงสร้าง, ตรงเวลา, require ทางเข้าเพื่อบรรทัดเหมือน ๆ กันของข้อมูลจากหน่วยความจำสำคัญ. ทั้งสองโปรเซสเซอร์มี cache และใช้โปรโตคอล MESI initially, ทั้งสอง caches มีทำสำเนา invalid ของบรรทัด. รูปร่าง 16.19 depicts consequence ของอ่านของบรรทัด xโดยโปรเซสเซอร์ P1. ถ้านี้คือของเริ่ม sequence ของ accesses, draw รูปร่าง subsequent สำหรับ sequence ดังต่อไปนี้

1. P2 reads x. .
2. P1 write to x (for clarity, label the line in P1's cache x').
3. P1 write to x (label the line in P1's cache x").
4. P2 reads x .

16.5 รูปร่าง 16.20 shows สอง diagrams สภาพเป็นไปได้ cache coherence โปรโตคอล deduce และอธิบายแต่ละโปรโตคอล, และ compare แต่เพื่อ MESI.

16.6 ตรวจสอบกับ SMPทั้งสอง L1 และ L2 caches การใช้โปรโตคอล MESI. เช่นได้อธิบายใน 16.3 ส่วน, หนึ่งของสภาพสี่คือเกี่ยวเนื่องกับแต่ละบรรทัดใน L2 cache. คือตลอด/ทุกๆสภาพสี่อีกด้วยสำหรับต้องการแต่ละบรรทัดใน L1 ถ้า cache? ดังนั้น, ทำไม? ถ้าไม่, อธิบายที่ซึ่งสภาพสามารถเป็น eliminated.

16.7 หัวข้อที่ 16.1 แสดงให้เห็นถึงประสิทธิภาพของ three-level cache arrangement สำหรับ S/390 IBM จุดประสงค์ของปัญหานี้คือเพื่อตัดสินใจหรือไม่ของ inclusion เสมอ third ของ cache seems worthwhile. ตัดสินใจ penalty ทางเข้า (average จำนวน/ตัวเลขของ pu cycles) สำหรับระบบกับเพียง cache L1, และ normalize นั้น value เพื่อ 1.0 จากนั้นตัดสินใจทางเข้า normalized penalty เมื่อทั้งสองและ L1 L2 cache ถูกใช้และ penalty ทางเข้าเมื่อตลอด/ทุกๆสาม caches ถูกใช้แล้ว. สมุดโน้ตของ amount improvement ในหีบแต่ละและสภาพทัศนะของคุณบน value L3 ของ cache.

16.8 code segment ต้องการเพื่อเป็น executed 64 เวลาสำหรับ evaluation ของ vector arithmetic การแสดงความเห็น ดังต่อไปนี้ D(I) = A(I) + B(I) * C(I) for 0 <= I <=63.

Load R1,B(I) /R1---Memory (α+I)/

Load R2,C(I)	/R2---Memory (β+I)/
Multiply R1,R2	/R1---(R1)*(R2)/
Load R3,A(I)	/R3---Memory (γ+I)/
Add R3,R1	/R3---(R3)+(R1)/
Load D1,R3	/Memoryθ---(R3)/

ที่ไหน R1,R2,and R3 คือโปรเซสเซอร์ลงทะเบียน และ $\alpha,\beta,\gamma,\theta$เป็นหน่วยความจำสำคัญของการเริ่ม addresses ของ B(I),C(I),A(I),and D(I), respectively. ตั้งสมมติฐานสี่ clock cycles สำหรับแต่ละโหลดหรือเก็บ/สะสม. สอง cycles สำหรับเพิ่ม, และ eight cycles สำหรับบนทั้ง multiplier uniprocessor หรือเดี่ยวโปรเซสเซอร์ใน SIMD เครื่องจักร.

a. จำนวน/ตัวเลขทั้งหมดของโปรเซสเซอร์ cycles ต้องการเพื่อ execute โค้ดนี้ segment repeatedly 64 เวลาบน SISD uniprocessor คอมพิวเตอร์ sequentially, ignoring ตลอด/ทุกๆ เวลาอื่นหน่วงเวลา.

b. ไตร่ตรองของผู้ใช้ uniprocessor SIMD กับ 64 การประมวลผมธาตุเพื่อ execute การปฏิบัติการ vector ใน synchronized หก vector การแนะนำบน 64-component vector ข้อมูลและทั้งสอง driven โดย clock same-speed execution ทั้งหมดเวลาบน SIMD เครื่องจักร, ignoring การแนะนำการออกอากาศและอื่นหน่วงเวลา.

c. อะไรถูกรับ speedup ของคอมพิวเตอร์ SIMDบน computer SISD ?

16.9 ผลิต vestorized รุ่นของโปรแกรมดังต่อไปนี้

```
      DO 20 I = 1,N
      B(I, 1)  = 0
      DO 10 J = 1,M
      A(I) = A(I) + B(I, J) * C(I, J)
10    CONTINUE
      D(I) = E(I) + A(I)
20    CONTINUE
```

16.10 uniprocessor คอมพิวเตอร์สามารถปฏิบัติการใน scalar หรือทั้ง vector โหมด กับ computation performed nine ความเร็วในโหมด vector benchmark โปรแกรม took เวลา t บนคอมพิวเตอร์นี้. ของเวลานี้ 25% ในโหมด vector และเวลา remaining ในโหมด scalar.

a. speedup effective ด้านล่าง mentioned ข้างบนเงื่อนไขเช่น compared เพื่อไม่การใช้ vector โหมด. อีกด้วย calculate α ของ percentage โค้ดนั้นมีเป็น vectorized (compiled ดังนั้นเช่นเพื่อใช้โหมด mode) vector ใน preceding โปรแกรม.

b. เรานั้นสองอัตราส่วนความเร็วระหว่าง vector และ scalar โหมดโดยฮาร์ดแวร์ improvements. calculate speedup effective นั้นคือ achieved.

c. speedup เหมือน ๆ กัน obtained ในคือ (b) obtained จาก compiler improvements ค่อนข้างกว่าฮาร์ดแวร์ improvement. อะไรจะ vectorization ใหม่อัตราส่วน α นั้นจะได้รับรองโดยสำหรับ compiler benchmark เหมือน ๆ กัน program?