บทที่ 3

การพัฒนาระบบ

ในที่นี้ เราจะกล่าวถึงพื้นฐานของสถาปัตยกรรม Smart Message ที่เราใช้ในการพัฒนาระบบของเรา รวมถึงกล่าวถึงข้อดีของสถาปัตยกรรมดังกล่าว ก่อนที่เราจะอธิบายถึงรายละเอียดการพัฒนาระบบของเราต่อไป

3.1 สถาปัตยกรรม Smart Message

Smart Message (SM) เป็น Mobile Agent สำหรับเครือข่ายไร้สายของระบบฝังตัว แต่ละ Mobile Agent ประกอบไปด้วยส่วนที่เป็นรหัสคำสั่งและส่วนที่เป็นข้อมูล นอกจากนี้ยังมีสถานะการทำงานขนาดย่อมอยู่ด้วย สถาปัตยกรรมนี้แตกต่างจากวิธีการติดต่อสื่อสารทั่วไป หรือ วิธีการติดต่อสื่อสารโดยการส่งข้อความถามตอบกันอยู่มาก โปรแกรมประยุกต์ SM หากต้องการข้อมูลบางอย่างจากโหนดหนึ่ง จำเป็นต้องโอนย้ายตัวเองไปยังโหนดที่มีข้อมูลนั้น แล้วประมวลผลหรือรันอยู่บนโหนดนั้นอีกที การจะทำเช่นนั้นได้ SM จะต้องรันอัลกอริธึ่มในการหาเส้นทาง ขนรหัสคำสั่งของอัลกอริธึ่มนั้นเป็นส่วนหนึ่งของ SM สำหรับใช้รันบนโหนดที่อยู่ขณะนั้น เพื่อใช้คำนวนหาโหนดถัดไปที่ต้องโอนย้ายตัวเองต่อไป เพื่อเข้าสู่โหนดเป้าหมายอีกทีหนึ่ง ส่วนรหัสคำสั่งนี้จะถูกแคชไว้ที่โหนดที่เคยผ่านไป เพื่อลด Cost ที่เกิดขึ้นจากการโอนย้ายรหัสคำสั่งที่เหมือนกันในอนาคต เมื่อเวลาผ่านไป ค่า Cost นี้จะถูกเฉลี่ยออกไปเนื่องจาก Locality ในเชิงเวลาและเชิงพื้นที่ของโปรแกรมประยุกต์ SM

สถาปัตยกรรม SM ประกอบไปด้วยเครื่องจำลอง SM (SMVM) และ Tag Space โดย SMVM มีพื้นฐานมาจาก KVM ของค่าย Sun โดยมีการพัฒนาต่อให้สนับสนุน Tag Space และสามารถโอนย้ายโปรแกรมไปรันต่อบนเครื่องอื่นได้ SMVM จึงเหมาะสมสำหรับอุปกรณ์เคลื่อนที่ที่มีทรัพยากรจำกัด และมีหน่วยความจำแค่ 160KB ก็ยังทำงานได้ ส่วน Tag Space เป็นเนื้อที่ในหน่วยความจำที่มีชื่อกำกับ สามารถทำให้การเชื่อมต่อกับ I/O และหน่วยความจำของ SMVM อยู่ในรูปแบบเดียวกันได้ การเข้าถึง I/O หรือหน่วยความจำจะต้องกระทำการผ่าน Tag Object เท่านั้น การเข้าถึง I/O หรือหน่วยความจำโดยตรงจะไม่สามารถกระทำได้ใน SMVM

ใน SM ไม่มี Java thread หรือ pre-emption ทำให้โมเดลในการทำงานของ SM นั้นค่อนข้างง่าย มีเพียง SM แค่ SM เดียวเท่านั้นที่จะ active ใน SMVM ในขณะที่ SM อื่นจะรออยู่ในคิวจนกว่า SM ที่ active จะทำหนึ่งในสิ่งต่อไปนี้คือ ทำงานเสร็จ โอนย้ายตัวเองไปทำงานต่อที่โหนดอื่น หรือ หยุดรอเหตุการณ์อะไรบางอย่าง

ข้อดีของการใช้ SM เป็น Platform ในการพัฒนาของเรามีอยู่มาก ข้อดีที่มีนัยสำคัญมากอันหนึ่งคือความสามารถในการโปรแกรมใหม่ได้หลังการติดตั้งใช้งาน ฟังก์ชันสำหรับ Aggregation หรือ Predicate สามารถติดตั้งได้ในขณะทำงานตามปกติ ข้อดีอีกข้อหนึ่งคือ SM Tag ซึ่งทำให้การเชื่อมต่อหรือเรียกใช้หน่วยความจำและ I/O มีรูปแบบเดียวกัน ความสามารถข้อนี้ทำให้การพัฒนาของเราง่ายขึ้นมาก โดยเฉพาะอย่างยิ่ง ส่วนการเข้าถึงทรัพยากรในลักษณะคล้ายการเข้าถึงตัวแปรของงานเรา

3.2 การพัฒนา DRN บน SM

โปรแกรมเชิงมหภาคถูกพัฒนาเป็น Smart Message ตัวหนึ่งที่จะถูกติดตั้งบนโหนดของผู้ใช้ โดยทั่วไปแล้ว โปรแกรม SM สามารถโอนย้ายตัวเองไปทำงานที่โหนดใดก็ได้ แต่ในการพัฒนาของเรา SM หลักของโปรแกรมเชิงมหภาคของเราจะไม่ทำการโอนย้ายตัวเองแต่อย่างใด ส่วนการดึงข้อมูลจากโหนดอื่นจะทำโดยให้ SM หลักสร้าง SM ลูกออกมา แล้วให้ SM ลูกโอนย้ายตัวเองไปยังโหนดเหล่านั้นเพื่อไปนำข้อมูลกลับมาอีกทีหนึ่ง

เมื่อตัวแปรทรัพยากรถูกประกาศด้วย Predicate และอายุขัยของการผูก ปกติ SM หลักจะไม่ทำการผูกตัวแปรทันที แต่การผูกจะทำเมื่อจำเป็นเท่านั้น เช่น เมื่อตัวแปรถูกอ้างอิงถึง การที่เซตของโหนดที่ตรงตามเงื่อนไขการผูกมักเปลี่ยนแปลงบ่อย การผูกตัวแปรเผื่อไว้ล่วงหน้าก่อนการใช้งานจริงมักไม่มีประโยชน์ เพราะเมื่อถึงเวลาต้องใช้จริง ตัวแปรก็ต้องถูกผูกใหม่อยู่ดี โอเวอร์เฮดของการผูกไว้ก่อนจึงสูญเปล่า แนวคิดนี้คล้ายกับวิธีการหาเส้นทางเมื่อต้องการใช้เท่านั้นในเครือข่ายไร้สายเฉพาะกิจ

การผูกตัวแปรทรัพยากรนั้น เราจะต้องทำการค้นหาว่าโหนดใดในระบบขณะนั้นมีคุณสมบัติตรงตามเงื่อนไขบ้าง SM หลักจะทำการสร้าง SM ค้นหา ซึ่งภายในประกอบไปด้วยเงื่อนไขสำหรับการค้นหาโหนดและค้นหาเส้นทางไปยังโหนดเหล่านั้น ถ้าไม่มีเงื่อนไขเชิงภูมิศาสตร์ปนอยู่ SM ค้นหาจะทำการ Flood

เครือข่ายโดยการทำสำเนาตัวเองแล้วโอนย้ายสำเนาเหล่านั้นไปยังโหนดเพื่อนบ้านทุกตัวจนกว่าโหนดทุกตัวในเครือข่ายจะมีสำเนาของ SM ค้นหานี้

ถ้ามีเงื่อนไขเชิงภูมิศาสตร์ปรากฏอยู่และบ่งบอกพื้นที่เป้าหมายไว้ด้วย SM ค้นหาจะโอนย้ายตัวเองไปยังโหนดเพื่อนบ้านที่ใกล้จุดเป้าหมายมากที่สุด ทำเช่นนี้ไปจนกระทั่งถึงโหนดหนึ่งที่อยู่ในพื้นที่เป้าหมาย หลังจากถึงแล้ว จึง Flood ตัวเองไปยังโหนดทุกตัวในพื้นที่เป้าหมายเพื่อตรวจสอบว่าโหนดใดตรงตามเงื่อนไขบ้างอีกทีหนึ่ง

ในทุกโหนดที่ SM ค้นหาได้เคยโอนย้ายมาถึง SM ค้นหาจะสร้าง Tag เพื่อทำเครื่องหมายไว้บนโหนดเหล่านั้น Tag นี้มีประโยชน์ในการใช้แยกแยะว่าโหนดดังกล่าว SM ค้นหาได้เคยผ่านมาแล้วหรือไม่ SM ค้นหาจะจบการทำงานของตนเองหากมันมาถึงโหนดที่ตนเองเคยทำเครื่องหมายไว้แล้ว นอกจาก Tag เครื่องหมายแล้ว SM ค้นหายังสร้าง Tag เพื่อใช้เป็นเส้นทางเดินทางกลับไปหาโหนดของผู้ใช้ตลอดเส้นทางที่มันผ่านไป โดยการจดจำว่าโหนดล่าสุดก่อนหน้าที่จะมาอยู่บนโหนดปัจจุบันนี้คือโหนดใด ดังนั้น Tag เส้นทางสู่ผู้ใช้นี้บนโหนดทั้งหลายจะประกอบกันเป็นต้นไม้รวมข้อมูลสำหรับรวบรวมข้อมูลจากโหนดทุกโหนดที่ตรงตามเงื่อนไขได้

เมื่อค้นพบโหนดที่ตรงตามเงื่อนไข SM ค้นหาจะโอนย้ายตัวเองกลับไปหา SM หลักโดยอาศัยเส้นทางจากต้นไม้รวมข้อมูล ระหว่างการเดินทางกลับ SM ค้นหาจะสร้าง route-to-id tag และ route-to-resource tag บนโหนดปัจจุบันเพื่อใช้จดจำโหนดล่าสุดก่อนหน้าโหนดปัจจุบันของการเดินทางกลับนี้ route-to-id tag ทั้งหลายเมื่อต่อรวมกันจะก่อให้เกิดเส้นทางไปสู่โหนดที่ตรงตามเงื่อนไขได้ เส้นทางนี้มีประโยชน์ต่อการเข้าถึงข้อมูลแบบอนุกรม ในขณะที่ route-to-resource tag จะก่อให้เกิดต้นไม้ multicast สำหรับการส่งคำขอการเข้าถึงไปยังโหนดที่ตรงตามเงื่อนไขแบบขนาน

ทันทีที่เดินทางกลับมาถึงโหนดผู้ใช้ SM ค้นหาจะรายงาน SM หลักเกี่ยวกับโหนดที่ตรงตามเงื่อนไข SM หลักจะเพิ่มโหนดที่อยู่ในรายงานดังกล่าวลงไปในเซตที่ผูกอยู่กับตัวแปรทรัพยากร นอกจากนี้ SM หลักจะทำการตั้งเวลาผูกใหม่ของตัวแปรทรัพยากรให้มีค่าเป็นอายุขัยการผูกต่อไป

ในการพัฒนานี้ การเข้าถึงแบบอนุกรมโดยใช้ตัววนซ้ำจะไม่ก่อให้เกิดการค้นหาทรัพยากรใหม่ SM หลักจะสร้าง SM เข้าถึง โดย SM เข้าถึงนี้จะโอนย้ายตัวเองไปยังโหนดที่ผูกไว้โดยใช้ route-to-id tag สำหรับโหนดนั้น ในทางกลับกัน การเข้าถึงตัวแปรทรัพยากรอาจนำไปสู่การค้นหาทรัพยากร ถ้าอายุขัยของ

การผูกนั้นหมดลง หรือตัวแปรไม่ได้ถูกไว้ในขณะนั้น เมื่อตัวแปรถูกผูกเสร็จ SM หลักจะสร้าง SM เข้าถึงที่มีหน้าที่โอนย้ายตัวเองไปยังโหนดที่ผูกไว้ทั้งหลายแบบขนาน โดยใช้ต้นไม้ Multicast ทันทีที่ถึงโหนดที่ผูกไว้ SM เข้าถึงจะทำงานตามที่โปรแกรมไว้คือ การเข้าถึงทรัพยากรบนโหนดเหล่านั้น อาจประมวลผลบางอย่างก่อนนำผลลัพธ์ที่ได้กลับไปแจ้งโหนดผู้ใช้ต่อไป

การเข้าทรัพยากรแบบขนานทำให้เราสามารถรวมข้อมูลระหว่างทางได้เมื่อมีโอกาส อันนำไปสู่การประหยัดพลังงานได้ ดังนั้น บนโหนดข้อต่อของต้นไม้รวมข้อมูล SM เข้าถึงจะหยุดรอ SM เข้าถึงตัวอื่นจนกว่า SM เข้าถึงจากทุกกิ่งของต้นไม้จะเดินทางมาถึงโหนดข้อต่อดังกล่าว หรือจนกว่าจะหมดเวลาของการรอคอย SM ทั้งหลายที่มาถึงที่โหนดข้อต่อจะถูกรวมเข้าเป็น SM ตัวเดียว โดย SM ที่เป็นผลลัพธ์ของการรวมจะโอนย้ายตัวเองกลับไปที่โหนดผู้ใช้ เวลาของการรอคอยในการพัฒนานี้มีค่าคงที่และค่อนข้างพื้นฐานมาก การพัฒนาที่ดีกว่านี้คือการใช้ความลึกของโหนดข้อต่อในการตั้งค่าเวลาการรอคอย โหนดยิ่งอยู่ลึกจะต้องรอคอยนาน การคำนวนความลึกของโหนดข้อต่อสามารถทำได้ในช่วงที่ SM ค้นหาเดินทางกลับจากโหนดที่ตรงตามเงื่อนไขไปยังโหนดผู้ใช้ เนื่องจากโหนดที่ตรงตามเงื่อนไขคือใบไม้ของต้นไม้ ความลึกของโหนดที่ตรงตามเงื่อนไขคือศูนย์ SM ค้นหาแต่ละตัวจะมีตัวนับความลึกนี้ ซึ่งจะเพิ่มขึ้นทีละหนึ่งสำหรับทุก Hop ระหว่างการเดินทางกลับ SM ค้นหาจะสร้าง Tag ความลึกไว้บนทุกโหนดระหว่างการเดินทางกลับ หากโหนดดังกล่าวยังไม่ Tag ความลึก โดยเขียนค่าปัจจุบันของตัวนับความลึกลงไปบน Tag ความลึกดังกล่าวของโหนด แต่หากโหนดมี Tag ความลึกอยู่แล้ว SM จะเปรียบเทียบค่าตัวนับความลึกของตนกับค่า Tag ความลึก ค่าใดมากกว่า จะใช้ค่านั้นเป็นค่าใหม่ของตัวนับและ Tag ความลึก

นอกจากนี้ เรายังสามารถปรับปรุงระบบให้ดีขึ้นได้ด้วยการรวม SM ค้นหาและ SM เข้าถึงให้เป็น SM ค้นหาและเข้าถึงในตัวเดียวกันเลย SM ตัวใหม่นี้จะทำงานคล้ายกับ SM ค้นหา หากแต่ว่าเมื่อมันค้นพบโหนดที่ตรงตามเงื่อนไข มันจะเข้าถึงทรัพยากรบนโหนดทันที แล้วย้ายตัวเองกลับไปหา SM หลักเพื่อรายงานเกี่ยวกับโหนดที่ตรงตามเงื่อนไขรวมทั้งผลของการเข้าถึงด้วยในขั้นตอนเดียว การรวมกันของ SM ดังกล่าวสามารถปรับปรุงเรื่องการประหยัดพลังงานและช่วยลดเวลาในการเข้าถึงของระบบ ส่วนการรวมกันของ SM ดังกล่าวเป็นเรื่องที่สามารถนำไปพัฒนาต่อได้สำหรับผู้ที่สนใจต่อไป