เรื่อง Talos

จัดทำโดย

นาย รัฐบุรี แท่นนิล 6406021421260

นาย จิลายุทธ บัณฑิต 6406021421171

เสนอ

อาจารย์ นพก้าว ทองใบ

รายงานนี้เป็นส่วนหนึ่งของวิชา Open Source

ภาคเรียนที่ 2 ปีการศึกษา 2564

มหาวิทยาลัยพระจอมเกล้าพระนครเหนือ

คำนำ

รายงานนี้เป็นส่วนหนึ่งของรายวิชา Open Source จัดทำขึ้นเพื่อให้ข้อมูลและความรู้เกี่ยวกับ Talos โดยรายงานได้นำข้อมูลจากต้นฉบับมาแปลเป็นเป็นภาษาไทยเพื่อให้ง่ายต่อการอ่าน ศึกษา และทำความเข้าใจเกี่ยวกับ Talos

จัดทำโดย

นาย รัฐบุรี แท่นนิล

นาย จิลายุทธ บัณฑิต

สารบัญ

ทาลอส (Talos)

ที่ Mozilla ระบบอัตโนมัติระบบแรกๆ ของเราคือเฟรมเวิร์กการทดสอบประสิทธิภาพที่เราขนานนามว่า Talos Talos ได้รับการดูแลอย่างซื่อสัตย์โดยไม่มีการดัดแปลงมากมายตั้งแต่เริ่มก่อตั้งในปี 2550 แม้ว่าสมมติฐานดั้งเดิมและการตัดสินใจด้านการออกแบบจำนวนมากที่อยู่เบื้องหลัง Talos จะสูญหายไปเนื่องจากความเป็นเจ้าของเครื่องมือเปลี่ยนมือ

ในฤดูร้อนปี 2011 ในที่สุด เราก็เริ่มมองถึงความสงสัยในเสียงและการแปรผันของตัวเลข Talos และเราเริ่มสงสัยว่าเราจะทำการปรับเปลี่ยนเล็กน้อยในระบบเพื่อเริ่มปรับปรุงได้อย่างไร เราไม่รู้ว่าเรากำลังจะเปิดกล่องแพนดอร่า

ในบทนี้ เราจะให้รายละเอียดเกี่ยวกับสิ่งที่เราพบเมื่อเราลอกเลเยอร์กลับหลังเลเยอร์ของซอฟต์แวร์นี้ ปัญหาที่เราค้นพบ และขั้นตอนใดบ้างที่เราดำเนินการเพื่อแก้ไขโดยหวังว่าคุณจะได้เรียนรู้จากความผิดพลาดและความสำเร็จของเรา

ภาพรวม (overview)

มาแกะส่วนต่าง ๆ ของ Talos กันเถอะ หัวใจของ Talos คือสายรัดทดสอบที่เรียบง่ายซึ่งสร้างโปรไฟล์ Firefox ใหม่ เริ่มต้นโปรไฟล์ ปรับเทียบเบราว์เซอร์ เรียกใช้การทดสอบที่ระบุ และสุดท้ายรายงานสรุปผลการทดสอบ การทดสอบอยู่ภายในที่เก็บ Talos และเป็นหนึ่งในสองประเภท: หน้าเดียวที่รายงานตัวเลขเดียว (เช่น เวลาเริ่มต้นผ่านตัวจัดการ onload ของหน้าเว็บ) หรือชุดของหน้าที่ถูกวนรอบเพื่อวัดเวลาในการโหลดหน้า ภายในส่วนขยายของ Firefox ถูกใช้เพื่อวนรอบเพจและรวบรวมข้อมูล เช่น หน่วยความจำและเวลาในการโหลดหน้า เพื่อบังคับการรวบรวมขยะ และเพื่อทดสอบโหมดเบราว์เซอร์ต่างๆ เป้าหมายเดิมคือการสร้างสายรัดแบบทั่วไปให้มากที่สุดเท่าที่จะเป็นไปได้เพื่อให้สายรัดนั้นทำการทดสอบทุกรูปแบบและวัดชุดของคุณลักษณะด้านประสิทธิภาพบางอย่างตามที่กำหนดโดยตัวการทดสอบเอง

ในการรายงานข้อมูล สายรัดของ Talos สามารถส่ง JSON ไปยัง Graph Server: เว็บแอปพลิเคชันกราฟภายในที่ยอมรับข้อมูลของ Talos ตราบใดที่ข้อมูลนั้นตรงตามรูปแบบที่กำหนดไว้ล่วงหน้าเฉพาะสำหรับการทดสอบ ค่า แพลตฟอร์ม และการกำหนดค่าแต่ละรายการ Graph Server ยังทำหน้าที่เป็นอินเทอร์เฟซสำหรับตรวจสอบแนวโน้มและการถดถอยของประสิทธิภาพ อินสแตนซ์ในเครื่องของเว็บเซิร์ฟเวอร์ Apache มาตรฐานจะให้บริการหน้าต่างๆ ระหว่างการทดสอบ

องค์ประกอบสุดท้ายของ Talos คือเครื่องมือการรายงานการถดถอย ทุกครั้งที่เช็คอินที่เก็บข้อมูล Firefox จะมีการเรียกใช้การทดสอบ Talos หลายครั้ง การทดสอบเหล่านี้จะอัปโหลดข้อมูลไปยัง Graph Server และสคริปต์อื่นจะใช้ข้อมูลจาก Graph Server และตรวจสอบว่ามีการถดถอยหรือไม่ หากพบการถดถอย (เช่น การ

วิเคราะห์ของสคริปต์ระบุว่าโค้ดที่เช็คอินทำให้ประสิทธิภาพในการทดสอบแย่ลงอย่างมาก) สคริปต์จะส่งอีเมลข้อความไปยังรายชื่อผู้รับจดหมายและบุคคลที่ตรวจสอบโค้ดที่ละเมิด

แม้ว่าสถาปัตยกรรมนี้ซึ่งสรุปไว้ในรูปที่ 8.1 ดูเหมือนจะค่อนข้างตรงไปตรงมา แต่ Talos แต่ละชิ้นก็เปลี่ยนแปลงไปตลอดหลายปีที่ผ่านมา เนื่องจาก Mozilla ได้เพิ่มแพลตฟอร์ม ผลิตภัณฑ์ และการทดสอบใหม่ๆ ด้วยการกำกับดูแลระบบทั้งหมดเพียงเล็กน้อยในฐานะโซลูชันแบบครบวงจร Talos ต้องการทำงานที่จริงจัง

- เสียงรบกวน สคริปต์ที่เฝ้าดูข้อมูลที่เข้ามาทำให้เกิดสัญญาณรบกวนในการทดสอบมากเท่ากับการถดถอยที่เกิดขึ้นจริงและไม่สามารถเชื่อถือได้
- ในการพิจารณาการถดถอย สคริปต์ได้เปรียบเทียบการเช็คอินแต่ละครั้งกับ Firefox กับค่าสำหรับการเช็คอินสามครั้งก่อนหน้าและสามครั้งหลังจากนั้น ซึ่งหมายความว่าผลลัพธ์ของ Talos สำหรับการเช็คอินของคุณอาจไม่สามารถใช้ได้เป็นเวลาหลายชั่วโมง
- Graph Server มีข้อกำหนดที่เข้มงวดว่าข้อมูลขาเข้าทั้งหมดจะเชื่อมโยงกับแพลตฟอร์ม สาขา ประเภทการทดสอบ และการกำหนดค่าที่กำหนดไว้ก่อนหน้านี้ ซึ่งหมายความว่าการเพิ่มการทดสอบใหม่ทำได้ยากเนื่องจากต้องใช้คำสั่ง SQL กับฐานข้อมูลสำหรับการทดสอบใหม่แต่ละครั้ง
- Talos นั้นใช้งานได้ยากโดย Talos เนื่องจากต้องใช้ข้อกำหนดทั่วไปที่จริงจังเกินไป มีขั้นตอน "กำหนดค่า" เพื่อสร้างสคริปต์การกำหนดค่าที่จะใช้เพื่อเรียกใช้การทดสอบในขั้นตอนต่อไป

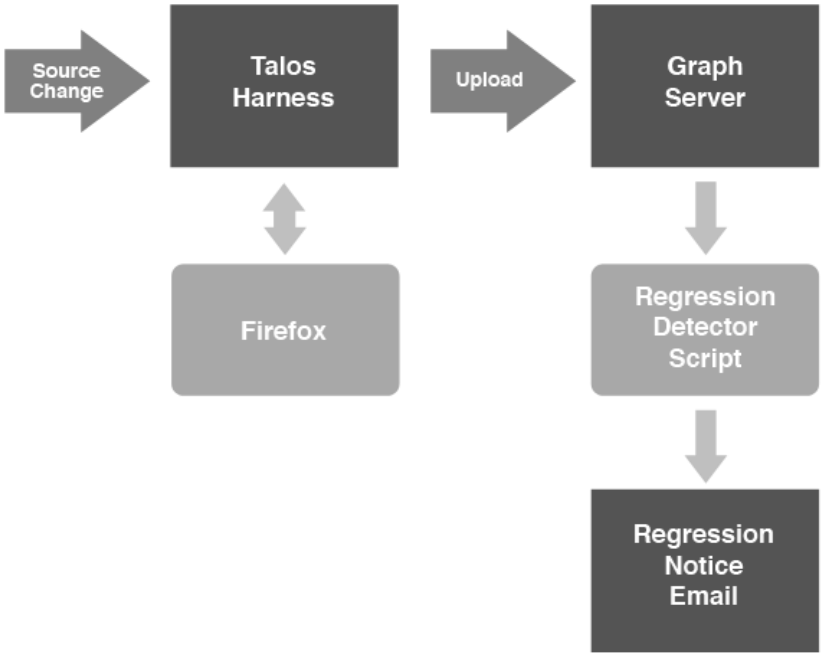

Figure 8.1 - Talos architecture

ขณะแฮ็คสายรัด Talos ในช่วงฤดูร้อนปี 2011 เพื่อเพิ่มการรองรับแพลตฟอร์มและการทดสอบใหม่ๆ เราพบผลลัพธ์จากวิทยานิพนธ์ของอาจารย์ Jan Larres ซึ่งเขาได้ตรวจสอบเสียงรบกวนจำนวนมากที่ปรากฏในการทดสอบ Talos เขาวิเคราะห์ปัจจัยต่างๆ รวมถึงฮาร์ดแวร์ ระบบปฏิบัติการ ระบบไฟล์ ไดรเวอร์ และ Firefox ที่อาจส่งผลต่อผลการทดสอบ Talos จากการทำงานนั้น Stephen Lewchuk ได้อุทิศการฝึกงานของเขาเพื่อพยายามลดเสียงรบกวนที่เราเห็นในการทดสอบเหล่านั้นทางสถิติ

จากงานและความสนใจของพวกเขา เราเริ่มจัดทำแผนเพื่อขจัดหรือลดเสียงรบกวนในการทดสอบ Talos เรารวบรวมแฮ็กเกอร์เพื่อควบคุมตัวเอง นักพัฒนาเว็บเพื่ออัปเดต Graph Server และนักสถิติเพื่อกำหนดวิธีที่เหมาะสมที่สุดในการทดสอบแต่ละครั้งเพื่อให้ได้ผลลัพธ์ที่คาดการณ์ได้โดยมีสัญญาณรบกวนน้อยที่สุด

ทำความเข้าใจสิ่งที่คุณกำลังวัด (Understanding What You Are Measuring)

เมื่อทำการทดสอบประสิทธิภาพ จำเป็นต้องมีการทดสอบที่เป็นประโยชน์ซึ่งให้คุณค่าแก่ผู้พัฒนาผลิตภัณฑ์และช่วยให้ลูกค้าเห็นว่าผลิตภัณฑ์นี้ทำงานอย่างไรภายใต้เงื่อนไขบางประการ สิ่งสำคัญคือต้องมีสภาพแวดล้อมที่ทำซ้ำได้ เพื่อให้คุณสามารถสร้างผลลัพธ์ได้ตามต้องการ แต่สิ่งที่สำคัญที่สุดคือทำความเข้าใจว่าคุณมีการทดสอบอะไรบ้างและสิ่งที่คุณวัดจากการทดสอบเหล่านั้นคืออะไร

ไม่กี่สัปดาห์ในโครงการของเรา เราทุกคนได้เรียนรู้เพิ่มเติมเกี่ยวกับระบบทั้งหมด และเริ่มทดลองกับพารามิเตอร์ต่างๆ เพื่อรันการทดสอบที่แตกต่างกัน คำถามหนึ่งที่เกิดซ้ำคือ “ตัวเลขหมายความว่าอย่างไร” นี้ไม่ได้ตอบง่าย การทดสอบหลายครั้งดำเนินมาหลายปีแล้ว โดยแทบไม่มีเอกสารประกอบเลย

ที่แย่ไปกว่านั้น ยังไม่สามารถสร้างผลลัพธ์แบบเดียวกันในเครื่องซึ่งได้รับรายงานจากการทดสอบอัตโนมัติ เห็นได้ชัดว่าสายรัดตัวเองทำการคำนวณ (มันจะลดค่าสูงสุดต่อหน้าแล้วรายงานค่าเฉลี่ยสำหรับรอบที่เหลือ) และเซิร์ฟเวอร์กราฟก็ทำเช่นกัน (ลดค่าหน้าสูงสุดแล้วเฉลี่ยหน้าด้วยกัน) . ผลลัพธ์ที่ได้คือไม่มีข้อมูลในอดีตที่สามารถให้คุณค่าได้มาก และไม่มีใครเข้าใจการทดสอบที่เรากำลังดำเนินการอยู่

เรามีความรู้บางอย่างเกี่ยวกับการทดสอบหนึ่งโดยเฉพาะ เรารู้ว่าการทดสอบนี้นำเว็บไซต์ 100 อันดับแรกที่มีการสแนปชอตในเวลาและโหลดทีละหน้า โดยทำซ้ำ 10 ครั้ง Talos โหลดหน้าเว็บ รอเหตุการณ์ mozAfterPaint (เหตุการณ์มาตรฐานที่เริ่มทำงานเมื่อ Firefox วาดภาพผ้าใบสำหรับหน้าเว็บ) จากนั้นบันทึกเวลาตั้งแต่โหลดหน้าจนถึงรับกิจกรรมนี้ เมื่อดูจุดข้อมูล 1,000 จุดที่เกิดจากการทดสอบครั้งเดียว ไม่มีรูปแบบที่ชัดเจน ลองนึกภาพการต้ม 10,000 คะแนนเหล่านั้นให้เป็นตัวเลขเดียวและติดตามตัวเลขนั้นเมื่อเวลาผ่านไป จะเกิดอะไรขึ้นถ้าเราทำให้ CSS แยกวิเคราะห์เร็วขึ้น แต่การโหลดรูปภาพช้าลง เราจะตรวจพบได้อย่างไร? เป็นไปได้ไหมที่หน้า 17 จะช้าลงหากหน้าอื่น ๆ ทั้งหมด 99 หน้ายังคงเหมือนเดิม? ในการแสดงวิธีคำนวณค่าใน Talos เวอร์ชันดั้งเดิม ให้พิจารณาตัวเลขต่อไปนี้

สำหรับค่าการโหลดหน้าต่อไปนี้

หน้า 1: 570, 572, 600, 503, 560

หน้า 2: 780, 650, 620, 700, 750

หน้า 3: 1220, 980, 1000, 1100, 1200

อย่างแรก สายรัด Talos เองจะลดค่าแรกและคำนวณค่ามัธยฐาน:

หน้า 1: 565.5

หน้า 2: 675

หน้า 3: 1050

ค่าเหล่านี้จะถูกส่งไปยังเซิร์ฟเวอร์กราฟ Graph Server จะลดค่าสูงสุดและคำนวณค่าเฉลี่ยโดยใช้ค่าเหล่านี้ต่อหน้า และจะรายงานว่าค่าหนึ่งค่า

$$\frac{565.5 + 675}{2} = 620.25$$

ค่าสุดท้ายนี้จะแสดงเป็นกราฟเมื่อเวลาผ่านไป และอย่างที่คุณเห็น ค่านี้จะสร้างค่าโดยประมาณซึ่งไม่ดีสำหรับอะไรมากไปกว่าการประเมินประสิทธิภาพแบบคร่าวๆ นอกจากนี้ หากตรวจพบการถดถอยโดยใช้ค่าเช่นนี้ จะเป็นการยากมากที่จะย้อนกลับและดูว่าหน้าใดทำให้เกิดการถดถอย เพื่อให้นักพัฒนาสามารถนำไปยังปัญหาเฉพาะเพื่อแก้ไขได้

เรามุ่งมั่นที่จะพิสูจน์ว่าเราสามารถลดเสียงรบกวนในข้อมูลจากการทดสอบ 100 หน้านี้ได้ เนื่องจากการทดสอบวัดเวลาในการโหลดหน้าเว็บ อันดับแรกเราต้องแยกการทดสอบออกจากอิทธิพลอื่นๆ ในระบบ เช่น การแคช เราเปลี่ยนการทดสอบให้โหลดหน้าเดิมซ้ำแล้วซ้ำอีก แทนที่จะวนไปมาระหว่างหน้าต่างๆ เพื่อให้วัดเวลาในการโหลดสำหรับหน้าที่ส่วนใหญ่แคชไว้ แม้ว่าวิธีการนี้ไม่ได้บ่งชี้ว่าผู้ใช้ปลายทางท่องเว็บอย่างไร แต่ก็ลดสัญญาณรบกวนบางส่วนในข้อมูลที่บันทึกไว้ ขออภัย การดูจุดข้อมูลเพียง 10 จุดสำหรับหน้าเว็บหนึ่งๆ ไม่ใช่ขนาดตัวอย่างที่มีประโยชน์

โดยการเปลี่ยนขนาดตัวอย่างและการวัดค่าเบี่ยงเบนมาตรฐานของค่าการโหลดหน้าเว็บจากการทดสอบหลายๆ ครั้ง เราได้พิจารณาแล้วว่าสัญญาณรบกวนลดลงหากเราโหลดหน้าเว็บอย่างน้อย 20 ครั้ง หลังจากการทดลองหลายครั้ง วิธีการนี้พบจุดที่น่าสนใจที่มีโหลด 25 รายการและละเว้นการโหลด 5 รายการแรก กล่าวอีกนัยหนึ่ง โดยการตรวจสอบค่าเบี่ยงเบนมาตรฐานของค่าของการโหลดหน้าเว็บหลายหน้า เราพบว่า 95% ของผลลัพธ์ที่รบกวนเราเกิดขึ้นในการโหลดห้าครั้งแรก แม้ว่าเราจะไม่ได้ใช้จุดข้อมูล 5 จุดแรก แต่เราเก็บข้อมูลเหล่านี้ไว้เพื่อให้เราสามารถเปลี่ยนแปลงการคำนวณทางสถิติของเราในอนาคตได้หากต้องการ

การทดลองทั้งหมดนี้ทำให้เรามีข้อกำหนดใหม่บางประการสำหรับการรวบรวมข้อมูลที่ Talos กำลังดำเนินการอยู่

- ข้อมูลทั้งหมดที่รวบรวมต้องเก็บไว้ในฐานข้อมูล ไม่ใช่แค่ค่าเฉลี่ยของค่าเฉลี่ย
- การทดสอบต้องรวบรวมข้อมูลที่เป็นประโยชน์อย่างน้อย 20 จุดต่อการทดสอบ (ในกรณีนี้ต่อหน้า)
- เพื่อหลีกเลี่ยงการปิดบังการถดถอยในหน้าหนึ่งโดยการปรับปรุงในอีกหน้าหนึ่ง จะต้องคำนวณแต่ละหน้าแยกกัน ไม่มีค่าเฉลี่ยในหน้าต่างๆ อีกต่อไป
- การทดสอบแต่ละครั้งที่รันต้องมีนักพัฒนาที่เป็นเจ้าของการทดสอบและเอกสารเกี่ยวกับสิ่งที่กำลังรวบรวมและเหตุผล
- เมื่อสิ้นสุดการทดสอบ เราต้องสามารถตรวจพบการถดถอยของหน้าใดๆ ในขณะรายงานผลได้

การใช้ข้อกำหนดใหม่เหล่านี้กับระบบ Talos ทั้งหมดเป็นสิ่งที่ควรทำ แต่สำหรับระบบนิเวศที่เติบโตขึ้นรอบๆ Talos จะเป็นภาระหน้าที่สำคัญในการเปลี่ยนไปใช้รูปแบบใหม่นี้ เรามีการตัดสินใจว่าเราจะปรับโครงสร้างใหม่หรือเขียนระบบใหม่

เขียนใหม่ เทียบกับ ปรับปรุงโครงสร้าง (Rewrite vs. Refactor)

จากการวิจัยของเราเกี่ยวกับสิ่งที่ต้องเปลี่ยนแปลงใน Talos เรารู้ว่าเราจะทำการเปลี่ยนแปลงครั้งใหญ่ อย่างไรก็ตาม การเปลี่ยนแปลงทางประวัติศาสตร์ทั้งหมดของ Talos ที่ Mozilla มักประสบกับความกลัวว่าจะ "ทำลายตัวเลข" อยู่เสมอ Talos หลายชิ้นถูกสร้างขึ้นในช่วงหลายปีที่ผ่านมาโดยผู้มีส่วนร่วมที่มีเจตนาดีซึ่งการเพิ่มเติมนั้นสมเหตุสมผลในตอนนั้น แต่ไม่มีเอกสารหรือการดูแลทิศทางของห่วงโซ่เครื่องมือ มันจึงกลายเป็นการปะติดปะต่อของรหัสที่ยากต่อการทดสอบ แก้ไขหรือทำความเข้าใจ

เนื่องจากเรากลัวสสารมืดที่ไม่มีเอกสารในฐานโค้ด รวมกับปัญหาที่เราจะต้องตรวจสอบการวัดใหม่ของเรากับการวัดแบบเก่า เราจึงเริ่มความพยายามในการปรับโครงสร้างใหม่เพื่อแก้ไข Talos และ Graph Server ในสถานที่ อย่างไรก็ตาม เห็นได้ชัดว่าหากไม่มีการสร้างโครงสร้างใหม่ขนาดใหญ่ของสคีมาฐานข้อมูล ระบบ Graph Server จะไม่สามารถนำเข้าข้อมูลดิบทั้งชุดจากการทดสอบประสิทธิภาพได้ นอกจากนี้ เราไม่มีวิธีที่สะอาดหมดจดในการใช้วิธีทางสถิติที่วิจัยใหม่ของเรากับแบ็กเอนด์ของเซิร์ฟเวอร์กราฟ ดังนั้นเราจึงตัดสินใจเขียน Graph Server ใหม่ตั้งแต่ต้น โดยสร้างโปรเจ็กต์ชื่อ Datazilla นี่ไม่ใช่การตัดสินใจง่ายๆ เนื่องจากโครงการโอเพนซอร์สอื่นๆ ได้แยกฐานโค้ดของเซิร์ฟเวอร์กราฟสำหรับการทำงานอัตโนมัติของตนเอง ที่ด้านบังเหียนของ Talos ของสมการ เรายังสร้างต้นแบบตั้งแต่เริ่มต้น เรายังมีต้นแบบที่ใช้งานได้ซึ่งทำการทดสอบอย่างง่ายและมีโค้ดที่เบากว่าประมาณ 2,000 บรรทัด

ขณะที่เราเขียน Graph Server ใหม่ตั้งแต่ต้น เราก็กังวลว่าจะก้าวไปข้างหน้าด้วยต้นแบบนักวิ่งทดสอบ Talos รุ่นใหม่ของเรา ความกลัวของเราคือเราอาจสูญเสียความสามารถในการใช้ตัวเลข "แบบเก่า" เพื่อที่เราจะสามารถเปรียบเทียบแนวทางใหม่กับแบบเก่าได้ ดังนั้นเราจึงละทิ้งต้นแบบของเราและแก้ไขสายรัด Talos ทีละ

น้อยเพื่อแปลงเป็นเครื่องกำเนิดข้อมูลในขณะที่เหลือชิ้นส่วนที่มีอยู่ซึ่งดำเนินการโดยเฉลี่ยเพื่ออัปโหลดไปยังระบบ Graph Server แบบเก่า นี่เป็นการตัดสินใจที่ไม่ดีอย่างยิ่ง เราควรจะสร้างสายรัดแยกต่างหากแล้วเปรียบเทียบสายรัดแบบใหม่กับสายรัดแบบเก่า

พยายามสนับสนุนการไหลของข้อมูลเดิมและวิธีการใหม่ในการวัดข้อมูลสำหรับแต่ละหน้าพิสูจน์แล้วว่ายาก ในด้านบวก มันบังคับให้เราปรับโครงสร้างโค้ดภายในจำนวนมากให้กับเฟรมเวิร์ก และปรับปรุงบางสิ่งในหลายๆ อย่าง แต่เราต้องทำทั้งหมดนี้ทีละน้อยบนชิ้นส่วนของระบบอัตโนมัติที่ทำงานอยู่ ซึ่งทำให้เราต้องปวดหัวหลายครั้งในแท่นขุดเจาะการผสานรวมอย่างต่อเนื่องของเรา

จะดีกว่ามากที่จะพัฒนาทั้งกรอบงาน Talos และ Datazilla ระบบการรายงานแบบคู่ขนานกันตั้งแต่เริ่มต้น โดยทิ้งโค้ดเก่าทั้งหมดไว้เบื้องหลัง โดยเฉพาะอย่างยิ่งเมื่อพูดถึงการแสดงละคร มันจะง่ายกว่ามากในการกำหนดระบบใหม่โดยไม่ต้องพยายามเชื่อมโยงในการสร้างข้อมูลการพัฒนาสำหรับระบบ Datazilla ที่กำลังจะมีขึ้นในการรันอัตโนมัติ เราคิดว่าจำเป็นต้องทำเช่นนี้เพื่อให้เราสามารถสร้างข้อมูลการทดสอบด้วยบิลด์จริงและโหลดจริง เพื่อให้แน่ใจว่าการออกแบบของเราจะปรับขนาดอย่างเหมาะสม ในท้ายที่สุด ข้อมูลบิลด์นั้นไม่คุ้มกับความซับซ้อนในการปรับเปลี่ยนระบบการผลิต หากเรารู้ในตอนนั้นว่าเรากำลังเริ่มโครงการยาวหนึ่งปีแทนที่จะเป็นโครงการหกเดือนที่คาดการณ์ไว้ เราจะเขียน Talos และเฟรมเวิร์กผลลัพธ์ใหม่ตั้งแต่ต้น

การสร้างวัฒนธรรมการแสดง (Creating a Performance Culture)

เนื่องจากเป็นโครงการโอเพ่นซอร์ส เราจึงต้องน้อมรับแนวคิดและการวิพากษ์วิจารณ์จากบุคคลและโครงการอื่นๆ ไม่มีผู้อำนวยการฝ่ายพัฒนาบอกว่าสิ่งต่างๆ จะทำงานอย่างไร เพื่อให้ได้ข้อมูลมากที่สุดและตัดสินใจได้ถูกต้อง จำเป็นต้องดึงผู้คนจำนวนมากจากหลายทีม โปรเจ็กต์เริ่มต้นด้วยนักพัฒนาสองคนบนเฟรมเวิร์ก Talos สองคนบน Datazilla/Graph Server และนักสถิติสองคนที่ยืมตัวมาจากทีมเมตริกของเรา เราเปิดโปรเจ็กต์นี้ให้กับอาสาสมัครของเราตั้งแต่เริ่มต้น และดึงหน้าใหม่หลายคนมาที่ Mozilla รวมถึงคนอื่นๆ ที่ใช้ Graph Server และการทดสอบ Talos สำหรับโครงการของตนเอง ขณะที่เราทำงานร่วมกัน ค่อยๆ ทำความเข้าใจว่าการเปลี่ยนลำดับของการทดสอบใดจะทำให้เราได้ผลลัพธ์ที่มีเสียงรบกวนน้อยลง เราจึงติดต่อเพื่อรวมนักพัฒนา Mozilla หลายคนไว้ในโปรเจ็กต์ การพบกันครั้งแรกของเรากับพวกเขานั้นยากเย็นแสนเข็ญ เนื่องจากมีการเปลี่ยนแปลงครั้งใหญ่ที่เราเสนอให้ทำ ความลึกลับของ “Talos” ทำให้สิ่งนี้ขายยากสำหรับนักพัฒนาหลายคนที่ใส่ใจเรื่องประสิทธิภาพเป็นอย่างมาก

ข้อความสำคัญที่ต้องใช้เวลาพอสมควรในการแก้ไขคือเหตุใดการเขียนองค์ประกอบขนาดใหญ่ของระบบใหม่จึงเป็นความคิดที่ดี และเหตุใดเราจึงไม่สามารถ ข้อเสนอแนะที่พบบ่อยที่สุดคือการเปลี่ยนแปลงเล็กน้อยในระบบที่มีอยู่ แต่ทุกคนที่เสนอคำแนะนำนั้นไม่รู้ว่าระบบพื้นฐานทำงานอย่างไร เรานำเสนองานมากมาย เชิญคน

จำนวนมากมาที่การประชุมของเรา จัดการประชุมพิเศษแบบครั้งเดียวในครั้งเดียว บล็อก โพสต์ ทวีต ฯลฯ เราทำทุกวิถีทางที่ทำได้เพื่อให้ได้รับคำบอกเล่า เพราะสิ่งเดียวที่น่ากลัวกว่าการทำงานทั้งหมดนี้เพื่อสร้างระบบที่ดีกว่าคือการทำงานทั้งหมดและไม่มีใครใช้

เป็นเวลาหนึ่งปีแล้วตั้งแต่การทบทวนปัญหาเสียงของ Talos ครั้งแรกของเรา นักพัฒนาต่างตั้งตารอสิ่งที่เราจะเปิดตัว กรอบงาน Talos ได้รับการปรับโครงสร้างใหม่เพื่อให้มีโครงสร้างภายในที่ชัดเจน และสามารถรายงานไปยัง Datazilla และเซิร์ฟเวอร์กราฟเก่าได้พร้อมกัน เราได้ตรวจสอบแล้วว่า Datazilla สามารถจัดการกับขนาดของข้อมูลที่เรากำลังส่ง (ข้อมูล 1 TB ต่อหกเดือน) และตรวจสอบตัวชี้วัดของเราสำหรับผลการคำนวณ สิ่งที่น่าตื่นเต้นที่สุดคือ เราได้ค้นพบวิธีส่งการวิเคราะห์การถดถอย/การปรับปรุงแบบเรียลไทม์บนพื้นฐานต่อการเปลี่ยนแปลงไปยังแผนผัง Mozilla ซึ่งถือเป็นชัยชนะครั้งใหญ่สำหรับนักพัฒนา

ดังนั้น เมื่อมีคนผลักดันการเปลี่ยนแปลงใน Firefox นี่คือสิ่งที่ Talos ทำ

• Talos รวบรวม 25 จุดข้อมูลสำหรับแต่ละหน้า

• ตัวเลขทั้งหมดเหล่านี้ถูกอัปโหลดไปยัง Datazilla

• Datazilla ดำเนินการวิเคราะห์ทางสถิติหลังจากวางจุดข้อมูลห้าจุดแรก (95% ของเสียงรบกวนพบได้ใน 5 จุดข้อมูลแรก)

• จากนั้น การทดสอบ T-Test ของ Welch จะใช้ในการวิเคราะห์ตัวเลขและตรวจสอบว่ามีสิ่งผิดปกติใดๆ ในข้อมูลต่อหน้าหรือไม่ เมื่อเทียบกับแนวโน้มก่อนหน้าจากการกดครั้งก่อน1

• จากนั้นผลลัพธ์ทั้งหมดของการวิเคราะห์ T-Test จะถูกผลักผ่านตัวกรอง False Discovery Rate ซึ่งช่วยให้มั่นใจว่า Datazilla สามารถตรวจจับผลบวกที่ผิดพลาดที่เกิดจากเสียงรบกวน2

• สุดท้าย หากผลลัพธ์อยู่ในเกณฑ์ความคลาดเคลื่อนของเรา Datazilla จะรันผลลัพธ์ผ่านอัลกอริธึมการปรับให้เรียบแบบเอ็กซ์โพเนนเชียลเพื่อสร้างเส้นแนวโน้มใหม่3 หากผลลัพธ์ไม่อยู่ในเกณฑ์ความคลาดเคลื่อนของเรา ผลลัพธ์จะไม่สร้างเส้นแนวโน้มใหม่และหน้าจะถูกทำเครื่องหมายเป็น ความล้มเหลว.

• เรากำหนดตัวชี้วัดว่าผ่าน/ไม่ผ่านโดยรวมตามเปอร์เซ็นต์ของหน้าที่ส่งผ่าน ผ่าน 95% คือ "ผ่าน"

ผลลัพธ์จะกลับมาที่สายรัดของ Talos ในแบบเรียลไทม์ จากนั้น Talos สามารถรายงานไปยังสคริปต์บิลด์ได้ไม่ว่าจะมีการถดถอยของประสิทธิภาพหรือไม่ก็ตาม ทั้งหมดนี้เกิดขึ้นกับ 10-20 Talos รันเสร็จสิ้นทุกนาที (ด้วยเหตุนี้ข้อมูล 1 TB) ในขณะที่อัปเดตการคำนวณและสถิติที่จัดเก็บไว้ในเวลาเดียวกัน

การนำสิ่งนี้จากโซลูชันที่ใช้งานได้มาแทนที่โซลูชันที่มีอยู่ต้องใช้ทั้งสองระบบเคียงข้างกันสำหรับ Firefox เวอร์ชันเต็ม กระบวนการนี้ช่วยให้แน่ใจว่าเราจะดูการถดถอยทั้งหมดที่รายงานโดย Graph Server ดั้งเดิม และ

ตรวจสอบให้แน่ใจว่าเป็นจริงและรายงานโดย Datazilla ด้วย เนื่องจาก Datazilla รายงานแบบต่อหน้าแทนที่จะเป็นระดับชุดทดสอบ จึงจำเป็นต้องมีการปรับให้เข้ากับ UI ใหม่และวิธีที่เรารายงานการถดถอย

มองย้อนกลับไป น่าจะเร็วกว่าถ้าเปลี่ยนสายรัด Talos แบบเก่าที่ด้านหน้า อย่างไรก็ตาม การปรับโครงสร้างใหม่ Mozilla ได้นำผู้มีส่วนร่วมรายใหม่ๆ มาสู่โครงการ Talos การปรับโครงสร้างใหม่ยังบังคับให้เราเข้าใจการทดสอบได้ดีขึ้น ซึ่งแปลเป็นการแก้ไขการทดสอบที่เสียหายจำนวนมากและปิดการทดสอบโดยมีค่าเพียงเล็กน้อยหรือไม่มีเลย ดังนั้น เมื่อพิจารณาว่าจะเขียนใหม่หรือจัดองค์ประกอบใหม่ เวลาทั้งหมดที่ใช้ไปจึงไม่ใช่ตัวชี้วัดเดียวที่ต้องตรวจสอบ

บทสรุป (Conclusion)

ในปีที่แล้ว เราได้เจาะลึกทุกส่วนของการทดสอบประสิทธิภาพอัตโนมัติที่ Mozilla เราได้วิเคราะห์สายรัดทดสอบ เครื่องมือการรายงาน และความสมบูรณ์ทางสถิติของผลลัพธ์ที่สร้างขึ้น ตลอดปีนั้น เราใช้สิ่งที่เราเรียนรู้เพื่อทำให้เฟรมเวิร์กของ Talos ง่ายต่อการบำรุงรักษา เรียกใช้ได้ง่ายขึ้น ตั้งค่าได้ง่ายขึ้น ทดสอบแพตช์ทดลองได้ง่ายขึ้น และมีโอกาสเกิดข้อผิดพลาดน้อยลง เราได้สร้าง Datazilla เป็นระบบที่ขยายได้สำหรับการจัดเก็บและเรียกข้อมูลตัวชี้วัดประสิทธิภาพทั้งหมดของเราจาก Talos และการทำงานอัตโนมัติในอนาคต เราได้เริ่มต้นการวิเคราะห์ทางสถิติประสิทธิภาพการทำงานใหม่ และสร้างการตรวจจับการถดถอย/การปรับปรุงที่ปฏิบัติได้ทางสถิติ เราได้ทำให้ระบบทั้งหมดเหล่านี้ใช้งานง่ายขึ้นและเปิดกว้างมากขึ้น เพื่อให้ผู้มีส่วนร่วมจากทุกที่สามารถดูโค้ดของเราและแม้แต่ทดลองวิธีใหม่ๆ ในการวิเคราะห์ทางสถิติเกี่ยวกับข้อมูลประสิทธิภาพของเรา ความมุ่งมั่นอย่างต่อเนื่องของเราในการตรวจสอบข้อมูลครั้งแล้วครั้งเล่าในแต่ละขั้นตอนของโครงการ และความเต็มใจของเราที่จะทิ้งข้อมูลที่พิสูจน์แล้วว่าไม่สามารถสรุปผลได้หรือไม่ถูกต้องช่วยให้เรายังคงมุ่งเน้นในขณะที่เราขับเคลื่อนโครงการขนาดยักษ์นี้ไปข้างหน้า การนำผู้คนจากทุกทีมที่ Mozilla และอาสาสมัครใหม่จำนวนมากเข้ามาช่วยให้ความพยายามเป็นจริงและยังช่วยสร้างการฟื้นคืนชีพในการตรวจสอบประสิทธิภาพและการวิเคราะห์ข้อมูลในหลายพื้นที่ของความพยายามของ Mozilla ส่งผลให้มีข้อมูลขับเคลื่อนประสิทธิภาพมากขึ้น- วัฒนธรรมที่เน้น