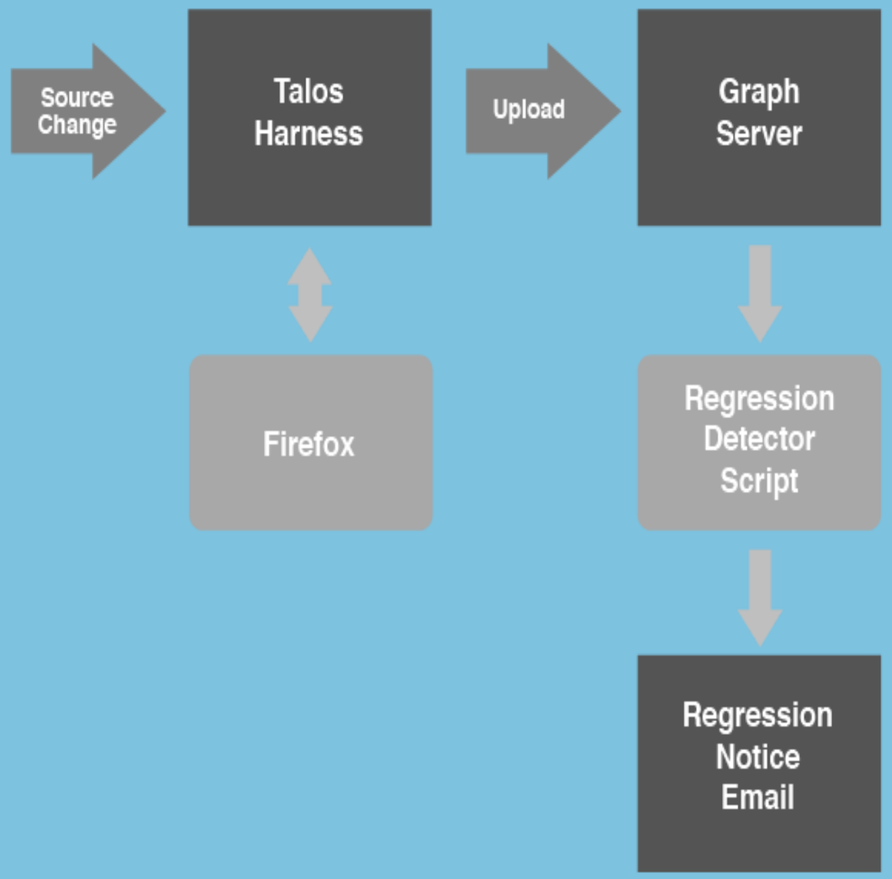
Talos

Clint Talbert and Joel Maher

The Performance of Open Source Applications

Talos



Topics Covered

Overview

Talos คือ Talos เป็นชุดทดสอบที่เรียบง่ายซึ่งสร้างโปรไฟล์ Firefox ใหม่ เริ่มต้นโปรไฟล์ ปรับเทียบเบราว์เซอร์ เรียกใช้การทดสอบที่ระบุ และสุดท้ายรายงานสรุปผลการทดสอบ

Overview

ในการรายงานข้อมูล ชุดทดสอบของ Talos สามารถส่ง JSON ไปยัง Graph Server: เว็บแอปพลิเคชันกราฟภายในที่ยอมรับข้อมูลของ Talos ตราบใดที่ข้อมูลนั้นตรงตามรูปแบบที่กำหนดไว้ล่วงหน้าเฉพาะสำหรับการทดสอบ ค่า แพลตฟอร์ม และการกำหนดค่าแต่ละรายการ

Overview

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

Mozilla ได้เพิ่มแพลตฟอร์ม ผลิตภัณฑ์ และการทดสอบใหม่ๆ ด้วยการกำกับดูแลระบบทั้งหมดเพียงเล็กน้อยในฐานะโซลูชันแบบครบวงจร Talos ต้องการทำงานเข้มงวด

1

Noise

สคริปต์ที่เฝ้าดูข้อมูลที่เข้ามาทำให้เกิดสัญญาณรบกวนในการทดสอบมากเท่ากับการถดถอยที่เกิดขึ้นจริงและไม่สามารถเชื่อถือได้

2

To determine a regression

สคริปต์ได้เปรียบเทียบการเช็คอินแต่ละครั้งกับ Firefox กับค่าสำหรับการเช็คอินสามครั้งก่อนหน้าและสามครั้งหลังจากนั้น ซึ่งหมายความว่าผลลัพธ์ของ Talos สำหรับการเช็คอินของคุณอาจไม่สามารถใช้ได้เป็นเวลาหลายชั่วโมง

3

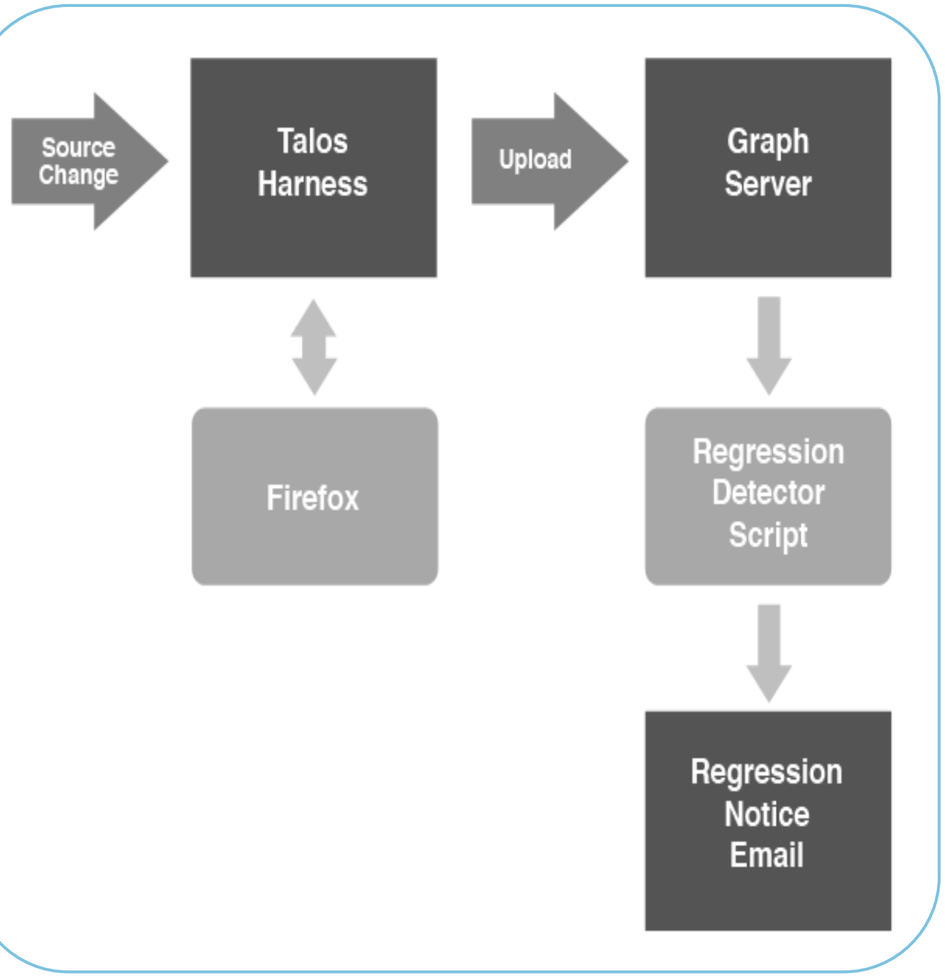
Graph Server

-ข้อกำหนดที่เข้มงวดว่าข้อมูลขาเข้าทั้งหมดจะเชื่อมโยงกับแพลตฟอร์ม สาขา ประเภทการทดสอบ และการกำหนดค่าที่กำหนดไว้ก่อนหน้านี้ ซึ่งหมายความว่าการเพิ่มการทดสอบใหม่ทำได้ยากเนื่องจากต้องใช้คำสั่ง SQL กับฐานข้อมูลสำหรับการทดสอบใหม่แต่ละครั้ง

4

Talos

นั้นใช้งานได้ยากโดย Talos เนื่องจากต้องใช้ข้อกำหนดทั่วไปที่จริงจังเกินไป มีขั้นตอน "กำหนดค่า" เพื่อสร้างสคริปต์การกำหนดค่าที่จะใช้เพื่อเรียกใช้การทดสอบในขั้นตอนต่อไป



Talos architecture

Overview

จากงานและความสนใจของพวกเขา เราเริ่มจัดทำแผนเพื่อขจัดหรือลดเสียงรบกวนในการทดสอบ Talos เรารวบรวมแฮ็กเกอร์เพื่อควบคุมตัวเอง นักพัฒนาเว็บเพื่ออัปเดต Graph Server และนักสถิติเพื่อกำหนดวิธีที่เหมาะสมที่สุดในการทดสอบแต่ละครั้งเพื่อให้ได้ผลลัพธ์ที่คาดการณ์ได้โดยมีสัญญาณรบกวนน้อยที่สุด

Understanding What You Are Measuring

การทดสอบหนึ่งโดยเฉพาะ เรารู้ว่าการทดสอบนี้นำเว็บไซต์ 100 อันดับแรกที่มีการสแนปชอตในเวลาและโหลดทีละหน้า โดยทำซ้ำ 10 ครั้ง Talos โหลดหน้าเว็บ รอเหตุการณ์ mozAfterPaint จากนั้นบันทึกเวลาตั้งแต่โหลดหน้าจนถึงรับกิจกรรมนี้

Understanding What You Are Measuring

เมื่อดูจุดข้อมูล 1,000 จุดที่เกิดจากการทดสอบครั้งเดียว ไม่มีรูปแบบที่ชัดเจน ลองนึกภาพการต้ม 10,000 คะแนน เหล่านั้นให้เป็นตัวเลขเดียวและติดตามตัวเลขนั้นเมื่อเวลาผ่านไป จะเกิดอะไรขึ้นถ้าเราทำให้ CSS แยกวิเคราะห์เร็วขึ้น แต่การโหลดรูปภาพช้าลง เราจะตรวจพบได้อย่างไร? เป็นไปได้ไหมที่หน้า 17 จะช้าลงหากหน้าอื่น ๆ ทั้งหมด 99 หน้ายังคงเหมือนเดิม? ในการแสดงวิธีคำนวณค่าใน Talos เวอร์ชันดั้งเดิม ให้พิจารณาตัวเลขต่อไปนี้

Understanding What You Are Measuring

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

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

Rewrite vs. Refactor

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

Rewrite vs. Refactor

พัฒนาทั้งกรอบงาน Talos และ Datazilla ระบบการรายงานแบบคู่ขนานกันตั้งแต่เริ่มต้น โดยทิ้งโค้ดเก่าทั้งหมดไว้เบื้องหลัง โดยเฉพาะอย่างยิ่งเมื่อพูดถึงการแสดงละคร มันจะง่ายกว่ามากในการกำหนดระบบใหม่โดยไม่ต้องพยายามเชื่อมโยงในการสร้างข้อมูลการพัฒนาสำหรับระบบ Datazilla ที่กำลังจะมีขึ้นในการรันอัตโนมัติ

Rewrite vs. Refactor

เราคิดว่าจำเป็นต้องทำเช่นนี้เพื่อให้เราสามารถสร้างข้อมูลการทดสอบด้วยบิลด์จริงและโหลดจริง เพื่อให้แน่ใจว่าการออกแบบของเราจะปรับขนาดอย่างเหมาะสม ในท้ายที่สุด ข้อมูลบิลด์นั้นไม่คุ้มกับความซับซ้อนในการปรับเปลี่ยนระบบการผลิต หากเรารู้ในตอนนั้นว่าเรากำลังเริ่มโครงการยาวหนึ่งปีแทนที่จะเป็นโครงการหกเดือนที่คาดการณ์ไว้ เราจะเขียน Talos และเฟรมเวิร์กผลลัพธ์ใหม่ตั้งแต่ต้น

Creating a Performance Culture

เนื่องจากเป็นโครงการโอเพ่นซอร์ส เราจึงต้องน้อมรับแนวคิดและการวิพากษ์วิจารณ์จากบุคคลและโครงการอื่นๆ ไม่มีผู้อำนวยการฝ่ายพัฒนาบอกว่าสิ่งต่างๆ จะทำงานอย่างไร เพื่อให้ได้ข้อมูลมากที่สุดและตัดสินใจได้ถูกต้อง จำเป็นต้องดึงผู้คนจำนวนมากจากหลายทีม

Creating a Performance Culture

การเปลี่ยนแปลงใน Firefox นี่คือสิ่งที่ Talos ทำ

- Talos รวบรวม 25 จุดข้อมูลสำหรับแต่ละหน้า
- ตัวเลขทั้งหมดเหล่านี้ถูกอัปโหลดไปยัง Datazilla
- Datazilla ดำเนินการวิเคราะห์ทางสถิติหลังจากวางจุดข้อมูลห้าจุดแรก (95% ของเสียงรบกวนพบได้ใน 5 จุดข้อมูลแรก)

Creating a Performance Culture

- จากนั้น การทดสอบ T-Test ของ Welch จะใช้ในการวิเคราะห์ตัวเลขและตรวจสอบว่ามีสิ่งผิดปกติใดๆ ในข้อมูลต่อหน้าหรือไม่ เมื่อเทียบกับแนวโน้มก่อนหน้าจากการกดครั้งก่อน1
- จากนั้นผลลัพธ์ทั้งหมดของการวิเคราะห์ T-Test จะถูกผลักผ่านตัวกรอง False Discovery Rate ซึ่งช่วยให้มั่นใจว่า Datazilla สามารถตรวจจับผลบวกที่ผิดพลาดที่เกิดจากเสียงรบกวน2

Creating a Performance Culture

- สุดท้าย หากผลลัพธ์อยู่ในเกณฑ์ความคลาดเคลื่อนของเรา Datazilla จะรันผลลัพธ์ผ่านอัลกอริธึมการปรับให้เรียบแบบเอ็กซ์โพเนนเชียลเพื่อสร้างเส้นแนวโน้มใหม่3 หากผลลัพธ์ไม่อยู่ในเกณฑ์ความคลาดเคลื่อนของเรา ผลลัพธ์จะไม่สร้างเส้นแนวโน้มใหม่และหน้าจะถูกทำเครื่องหมายเป็น ความล้มเหลว.
- เรากำหนดตัวชี้วัดว่าผ่าน/ไม่ผ่านโดยรวมตามเปอร์เซ็นต์ของหน้าที่ส่งผ่าน ผ่าน 95% คือ "ผ่าน"

Creating a Performance Culture

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

Creating a Performance Culture

การนำสิ่งนี้จากโซลูชันที่ใช้งานได้มาแทนที่โซลูชันที่มีอยู่ ต้องใช้ทั้งสองระบบเคียงข้างกันสำหรับ Firefox เวอร์ชันเต็ม กระบวนการนี้ช่วยให้แน่ใจว่าเราจะดูการถดถอยทั้งหมดที่รายงานโดย Graph Server ดั้งเดิม และตรวจสอบให้แน่ใจว่าเป็นจริงและรายงานโดย Datazilla ด้วย เนื่องจาก Datazilla รายงานแบบต่อหน้าแทนที่จะเป็นระดับชุดทดสอบ จึงจำเป็นต้องมีการปรับให้เข้ากับ UI ใหม่ และวิธีที่เรารายงานการถดถอย

Conclusion

เราได้เจาะลึกทุกส่วนของการทดสอบประสิทธิภาพอัตโนมัติที่ Mozilla เราได้วิเคราะห์ชุดทดสอบ เครื่องมือการรายงาน และความสมบูรณ์ทางสถิติของผลลัพธ์ที่สร้างขึ้น ตลอดปีนั้น เราใช้สิ่งที่เราเรียนรู้เพื่อทำให้เฟรมเวิร์กของ Talos ง่ายต่อการบำรุงรักษา เรียกใช้ได้ง่ายขึ้น ตั้งค่าได้ง่ายขึ้น ทดสอบแพตช์ทดลองได้ง่ายขึ้น และมีโอกาสเกิดข้อผิดพลาดน้อยลง เราได้สร้าง Datazilla เป็นระบบที่ขยายได้สำหรับการจัดเก็บและเรียกข้อมูลตัวชี้วัดประสิทธิภาพทั้งหมดของเราจาก Talos และการทำงานอัตโนมัติในอนาค

Conclusion

เราได้เริ่มต้นการวิเคราะห์ทางสถิติประสิทธิภาพการทำงานใหม่ และสร้างการตรวจจับการถดถอย/การปรับปรุงที่ปฏิบัติได้ทางสถิติ เราได้ทำให้ระบบทั้งหมดเหล่านี้ใช้งานง่ายขึ้นและเปิดกว้างมากขึ้น เพื่อให้ผู้มีส่วนร่วมจากทุกที่สามารถดูโค้ดของเราและแม้แต่ทดลองวิธีใหม่ๆ ในการวิเคราะห์ทางสถิติเกี่ยวกับข้อมูลประสิทธิภาพของเรา

Thank you!