

WhatsApp

จัดทำโดย

ชื่อ นายดวงเทียน พวงมาลัย 5809610198

เสนอ

ดร. ประภาพร รัตนธำรง

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา CS447

สาขาวิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยธรรมศาสตร์

ภาคเรียนที่ 1 ปีการศึกษา 2560

WhatsApp

          "WhatsApp" คือ แอพพลิเคชั่นสำหรับมือถือใช้สำหรับส่งข้อความแบบไม่จำกัด รวมไปถึงส่งรูปภาพ, เสียง และวิดีโอ ข้ามระบบปฏิบัติการ (Platform) ไม่ว่าจะเป็น iPhone, BlackBerry, Android และ Nokia ช่วยให้สามารถแชทกับเพื่อน ๆ ที่ใช้โทรศัพท์มือถือต่างระบบปฏิบัติการ เช่น นาย A ใช้งาน BlackBerry แต่นาย B ใช้งาน Android ทั้งสองคนสามารถแชทกันได้ง่ายโดยไม่ต้องโทรคุยกัน โดยการใช้งาน WhatsApp ซึ่งจะต้องเชื่อมต่อผ่าน GPRS/EDGE, Wi-Fi และ 3G

WhatsApp ใช้รุ่นที่กำหนดเองของ Open Standard Extensible Messaging และ Presence Protocol (XMPP) มาตรฐาน เมื่อติดตั้งแล้วจะสร้างบัญชีผู้ใช้โดยใช้หมายเลขโทรศัพท์ของผู้ใช้เป็นชื่อผู้ใช้ (Jabber ID: [phone number] @ s.whatsapp.net)

ซอฟต์แวร์ WhatsApp จะเปรียบเทียบหมายเลขโทรศัพท์ทั้งหมดจากสมุดที่อยู่ของอุปกรณ์โดยใช้ฐานข้อมูลกลางของผู้ใช้ WhatsApp เพื่อเพิ่มรายชื่อติดต่อลงในรายการที่ติดต่อ WhatsApp ของผู้ใช้โดยอัตโนมัติ ก่อนหน้านี้รุ่น Android และ Nokia Series 40 ใช้ IMEI แบบย้อนกลับของ IMEI เป็นรหัสผ่านของ MD5 ในขณะที่เวอร์ชัน iOS ใช้ที่อยู่ MAC ของ Wi-Fi ของโทรศัพท์แทน IMEI การปรับปรุง 2012 ในขณะนี้สร้างรหัสผ่านแบบสุ่มในฝั่งเซิร์ฟเวอร์ อุปกรณ์ Dual SIM บางอย่างอาจไม่สามารถใช้งานร่วมกับ WhatsApp ได้แม้ว่าจะมีวิธีแก้ไขปัญหานี้อยู่บ้าง

ในเดือนกุมภาพันธ์ปี 2015 WhatsApp ได้แนะนำคุณลักษณะการโทรด้วยเสียง ช่วยให้ WhatsApp สามารถดึงดูดกลุ่มผู้ใช้ที่แตกต่างไปจากเดิมอย่างสิ้นเชิงได้ เมื่อวันที่ 14 พฤศจิกายน 2016 WhatsApp ได้เพิ่มคุณลักษณะการโทรวิดีโอสำหรับผู้ใช้ทั่วทั้งอุปกรณ์ Android, iPhone และ Windows Phone ข้อความมัลติมีเดียจะถูกส่งโดยอัปโหลดภาพเสียงหรือวิดีโอที่จะส่งไปยังเซิร์ฟเวอร์ HTTP จากนั้นจะส่งลิงก์ไปยังเนื้อหาพร้อมกับภาพขนาดย่อที่เข้ารหัส Base64 (ถ้ามี) WhatsApp ทำตามกลไก "Store and forward" เพื่อแลกเปลี่ยนข้อความระหว่างผู้ใช้สองคน เมื่อผู้ใช้ส่งข้อความก่อนอื่นจะเดินทางไปยังเซิร์ฟเวอร์ WhatsApp ที่เก็บไว้ จากนั้นเซิร์ฟเวอร์จะขอให้ผู้รับยอมรับการรับข้อความอีกครั้ง ทันทีที่ข้อความได้รับการยอมรับเซิร์ฟเวอร์จะลบข้อความ มันไม่ได้มีอยู่ในฐานข้อมูลของเซิร์ฟเวอร์ เซิร์ฟเวอร์ WhatsApp เก็บข้อความไว้ในฐานข้อมูลของ 30 วันในกรณีที่ไม่ได้รับการส่งมอบ (เมื่อผู้รับไม่ได้ใช้งาน WhatsApp เป็นเวลา 30 วัน

**WhatsApp’s Functionality & Architecture**

* Server - Ejabberd
* Programming Language is ERLANG
* Protocol used is XMPP
* For Database management, here comes Mnesia & MySQL
* The Operating System is FreeBSD
* In order to store Multimedia files its uses YAWS storage server

**Platform**

Backend

Erlang

FreeBSD

Yaws, lighttpd

PHP

Custom patches to BEAM (BEAM is like Java’s JVM, but for Erlang)

Custom XMPP

Hosting may be in Soft layer

Frontend

Seven client platforms: iPhone, Android, Blackberry, Nokia Symbian S60, Nokia S40, Windows Phone

SQLite

Hardware

Standard user facing server:

Dual Westmere Hex-core (24 logical CPUs);

100GB RAM, SSD;

Dual NIC (public user-facing network, private back-end/distribution);

Tools and Techniques Used To Increase Scalability

* Wrote system activity reporter tool (wsar)

(เครื่องมือทำรายงานกิจกรรมในระบบ)

บันทึกสถิติของระบบในระบบรวมถึงสถิติ OS ประวัติฮาร์ดแวร์สถิติ BEAM สร้างขึ้นเพื่อให้เมตริกปลั๊กอินจากระบบอื่น ๆ สามารถใช้งานได้ง่ายเช่นหน่วยความจำเสมือน การใช้ CPU โดยรวมการใช้เวลาโดยรวมเวลาของผู้ใช้เวลาของระบบเวลาขัดจังหวะสวิทช์บริบทการเรียกระบบดักข้อมูลที่ส่ง / รับจำนวนการนับรวมข้อความทั้งหมดในคิวกระบวนการทั้งหมดเหตุการณ์พอร์ตที่ไม่ต่อเนื่องอัตราการเข้าชมไบต์เข้า / , สถิติการจัดตารางเวลา, สถิติการรวบรวมขยะ, การเก็บรวบรวมคำ ฯลฯ

* Hardware performance counters in CPU (pmcstat)

(นับประสิทธิภาพของ CPU แต่ละตัว)

ดูว่า CPU แต่ละตัวทำงานกี่เปอร์เซนในช่วงเวลาหนึ่ง ในการรัน Source code แล้วตัดสินใจว่าควรลดเวลาในการประมาลผลของ code หรือไม่ หากเวลารันน้อยการลดเวลาก็ไมทำให้สามารถทำงานเร็วขึ้น ควรมุ่งเน้นไปที่การปรับปรุงด้านอื่นๆแทน

* dtrace, kernel lock-counting, fprof

Dtrace เป็นส่วนใหญ่สำหรับการแก้จุดบกพร่องไม่ใช่ประสิทธิภาพ

Patched BEAM บน FreeBSD เพื่อรวมเวลาCPU time stamp

เขียนสคริปต์เพื่อสร้างมุมมองรวมของทุกกระบวนการเพื่อดูว่ามีการใช้จ่ายเป็นประจำอยู่ตลอดเวลา

Biggest win was compiling the emulator with lock counting turned on

* Some Issues:

ก่อนหน้านี้ได้เห็นการใช้เวลาในการจัดเก็บขยะเป็นอย่างมาก พบปัญหาบางอย่างเกี่ยวกับ network stack ที่ถูกปรับออกไป ปัญหาส่วนใหญ่เกิดจากความขัดแย้งของล็อคในโปรแกรมจำลองซึ่งแสดงให้เห็นอย่างมากในผลลัพธ์ของการนับล็อค

* Measurement:

Synthetic workloads หมายถึงการสร้างการเข้าชมจากสคริปต์ทดสอบของคุณเองมีค่าเพียงน้อยนิดสำหรับการปรับระบบที่หันหน้าไปทางผู้ใช้ในระดับสุดขีด

ทำงานได้ดีสำหรับอินเทอร์เฟซแบบง่ายๆเช่นตารางผู้ใช้สร้างแทรกและอ่านได้โดยเร็วที่สุด

หากสนับสนุนล้านการเชื่อมต่อบนเซิร์ฟเวอร์จะใช้เวลา 30 โฮสต์เพื่อเปิดพอร์ต IP เพียงพอเพื่อสร้างการเชื่อมต่อที่เพียงพอเพื่อทดสอบเซิร์ฟเวอร์เพียงหนึ่งเครื่อง สำหรับเซิร์ฟเวอร์สองล้านเครื่องที่จะใช้โฮสต์ 60 เครื่อง ยากที่จะสร้างแบบจำลองดังกล่าว

ประเภทของการเข้าชมที่เห็นในระหว่างการผลิตเป็นเรื่องยากที่จะสร้าง สามารถเดาได้ที่ปริมาณงานปกติ แต่ในความเป็นจริงให้ดูกิจกรรมเครือข่ายเหตุการณ์ในโลกเนื่องจากแพลตฟอร์มหลายแพลตฟอร์มเห็นพฤติกรรมที่แตกต่างกันระหว่างลูกค้าและประเทศที่แตกต่างกัน

* Tee’d workload:

Take normal production traffic and pipe it off to a separate system.

มีประโยชน์มากสำหรับระบบที่อาจทำให้เกิดผลข้างเคียง ไม่ต้องการตีเส้นการจราจรและทำสิ่งต่างๆที่ส่งผลต่อสถานะถาวรของผู้ใช้หรือทำให้เกิดข้อความหลายข้อความแก่ผู้ใช้

Erlang สนับสนุน hot loading, ดังนั้น ภายใต้ full production load, มีแนวคิดว่า , compile, load การเปลี่ยนแปลงแก้ไขแล้วดูว่าสามารถทำงานได้ดีหรือไม่ ใส่เซิร์ฟเวอร์ใน DNS สองสามครั้งเพื่อให้ได้รับการเข้าชมตามปกติเป็นสองเท่าหรือสามครั้ง สร้างปัญหาเกี่ยวกับ TTL เนื่องจากไคลเอ็นต์ไม่เคารพ TTL ของ DNS และมีความล่าช้าดังนั้นจึงไม่สามารถตอบสนองได้อย่างรวดเร็วเพื่อรับการเข้าชมมากกว่าที่จะสามารถจัดการได้

IPFW ส่งต่อการรับส่งข้อมูลจากเซิร์ฟเวอร์เครื่องหนึ่งไปยังอีกเครื่องหนึ่งเพื่อให้โฮสต์สามารถระบุจำนวนการเชื่อมต่อลูกค้าที่ต้องการได้ ข้อผิดพลาดทำให้เกิด kernel panic ซึ่งไม่ได้ผลดีนัก

* Results:

เริ่มต้นที่การเชื่อมต่อพร้อมกัน 200K ต่อเซิร์ฟเวอร์

First bottleneck แสดงที่ 425K. ระบบจะหยุดการทำงานแล้ว จัดตารางเวลาในการวัดว่ามีการทำงานที่เป็นประโยชน์มากหรือนอนหลับหรือปั่น ภายใต้โหลดมันเริ่มตีล็อคนอนเพื่อ CPU 35-45% ถูกใช้ในระบบ แต่ schedulers ที่ใช้ 95%

รอบแรกของการแก้ไขมีการเชื่อมต่อมากกว่าหนึ่งล้านครั้ง

การใช้ VM อยู่ที่ 76% CPU อยู่ที่ 73% BEAM ทำงานโดยใช้อัตรา 45% ซึ่งตรงกับเปอร์เซ็นต์ผู้ใช้ซึ่งดีมากเนื่องจากโปรแกรมจำลองทำงานเป็นผู้ใช้

การใช้ CPU ปกติไม่ได้เป็นการวัดว่าระบบทำงานที่วุ่นวายเนื่องจากตารางเวลาใช้ CPU อย่างไร

เดือนต่อมาได้มีการแก้ปัญหา Bottleneck 2 ล้านการเชื่อมต่อต่อเซิร์ฟเวอร์

อัตราการใช้ BEAM อยู่ที่ 80% ซึ่งใกล้เคียงกับที่ FreeBSD อาจเริ่มส่งเพจ CPU เป็นเรื่องเดียวกันกับการเชื่อมต่อสองครั้ง ตัวกำหนดเวลากำลังกดดันการต่อสู้ แต่ทำงานได้ดี

ดูเหมือนจะเป็นสถานที่ที่ดีในการหยุดการเขียนโค้ด Erlang

เดิมมีสองกระบวนการ Erlang ต่อการเชื่อมต่อ ตัดที่หนึ่ง

มีบางสิ่งที่มีตัวจับเวลา

มีการเชื่อมต่อสูงสุดที่ 2.8 M ต่อเซิร์ฟเวอร์ 571 k / pkts วินาที > 200k ms/msวินาที เพิ่มประสิทธิภาพหน่วยความจำบางอย่างเพื่อให้โหลด VM ลดลงเหลือ 70% พยายามเชื่อมต่อ 3 ล้าน แต่ล้มเหลว

ดูคิวข้อความแบบยาวเมื่อระบบประสบปัญหา ทั้งคิวข้อความเดียวหรือชุดคิวข้อความ

เพิ่มเครื่องมือ BEAM ในสถิติคิวข้อความสำหรับแต่ละกระบวนการ มีการส่ง / รับข้อความมากน้อยเพียงใด

การสุ่มตัวอย่างทุกๆ 10 วินาทีจะเห็นกระบวนการมีข้อความ 600K ในคิวข้อความโดยมีอัตราการชะล้าง 40K โดยมีความล่าช้า 15 วินาที เวลาระบายน้ำที่คาดการณ์ไว้คือ 41 วินาที

**เอกสารอ้างอิง**

The WhatsApp Architecture Facebook Bought For $19 Billion : <http://highscalability.com/blog/2014/2/26/the-whatsapp-architecture-facebook-bought-for-19-billion.html> (12/17/2017)

WhatsApp: <https://en.wikipedia.org/wiki/WhatsApp#User_statistics> (12/17/2017)

WhatsApp Store data : <https://www.quora.com/How-does-WhatsApp-store-data> (12/17/2017)