รายงาน
เรื่อง Google Scalability

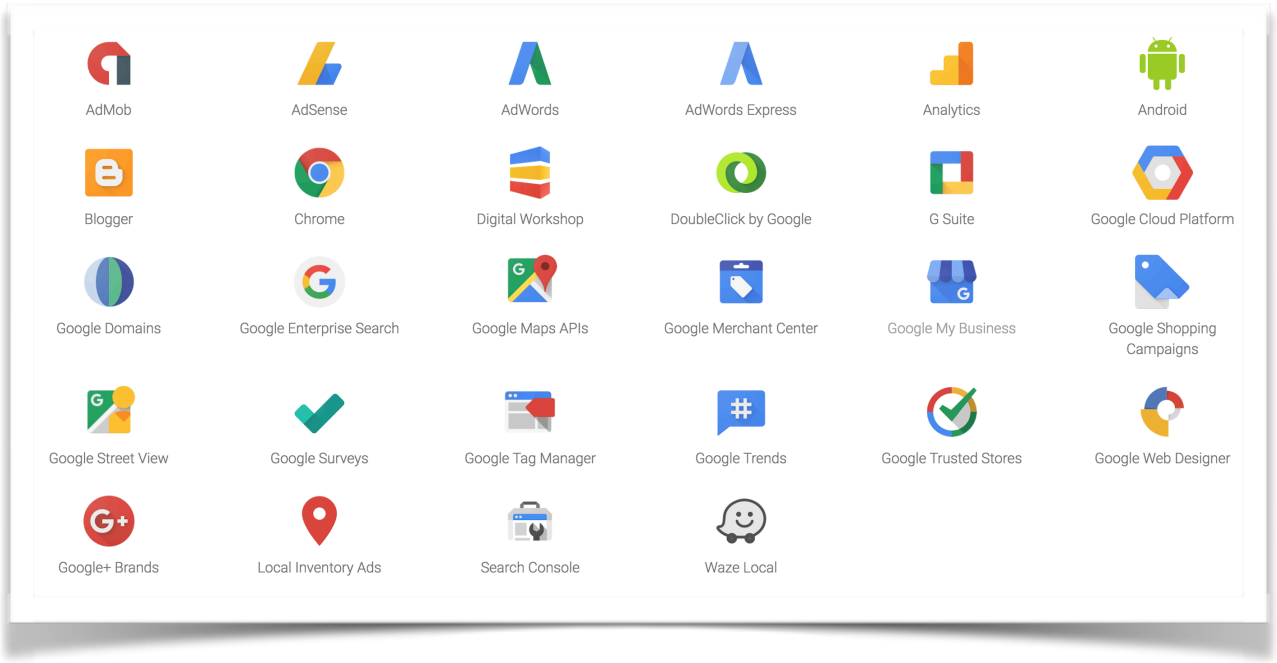
จัดทำโดย
นายอายุตม์ มั่งมีทรัพย์ 5809610347

เสนอ
ดร. ประภาพร รัตนธำรง

รายงานนี้เป็นส่วนหนึ่งของวิชา คพ. 447 ระบบปฏิบัติการ 2
ภาคเรียนที่ 1 ปีการศึกษา 2560

Google Scalability

บริษัท Google ให้บริการอะไร



ภาพที่ 1 ตัวอย่างผลิตภัณฑ์ต่าง ๆ ของ Google

บริษัท Google เป็นบริษัทสัญชาติอเมริกันที่เป็นบริษัทย่อยของบริษัท Alphabet ซึ่งปัจจุบันให้บริการให้บริการหลากหลายไม่ว่าจะเป็นการให้บริการซอฟต์แวร์สำหรับ Bussiness, Developers หรือผลิตภัณฑ์สำหรับทุกคนที่สามารถใช้ได้โดยไม่เสียเงินเช่น Gmail, Google Drive, Google Translate หรือแม้แต่ Browser ที่เราใช้กันเป็นประจำทุก ๆ วันเช่น Google Chrome แต่ในวันนี้เราจะมาพูดถึงประเด็นเกี่ยวกับผลิตภัณฑ์หนึ่งที่ชื่อว่า Google Search

Google Search เป็นบริการที่เป็นที่นิยมมากที่สุดในหมู่ผลิตภัณฑ์ของ Google ซึ่งแต่ละวันจะมีผู้ใช้บริการถึงประมาณ 4,464,000,000 คนต่อวันซึ่งน่าสนใจมากว่า Google มีการรองรับจำนวนผู้คนที่ใช้บริการมากมายขนาดนั้นได้อย่างไร เราจึงนำเราไปสู่ในหัวข้อ **ประเด็นปัญหาเกี่ยวกับ Scalability ที่ประสบปัญหาในการให้บริการ**

Search Engine	Searches per day
Google	*4,464,000,000*
Bing	*873,964,000*
Baidu	*583,520,803*
Yahoo	*536,101,505*
Other (AOL, Ask etc)	*128,427,264*

ภาพที่ 2 สถิติการใช้งาน Search Engine ในปี 2017

ประเด็นปัญหาเกี่ยวกับ Scalability ที่ประสบในการให้บริการ

1. File System ที่มีอยู่ไม่ตอบสนองความต้องการของตน

Google พบว่ามีตนเองมีความต้องการที่จะจัดเก็บข้อมูลจำนวนมหาศาลด้วย File System ที่เชื่อถือได้ และพบว่า File System ที่มีในปัจจุบันไม่สามารถตอบสนองความต้องการของตนเองได้เช่น

- การขยายไปใช้ในเครื่อง nodes เป็นพัน ๆ เครื่อง
- การ read/write ข้อมูลเป็นจำนวนมาก
- ความสามารถในการจัดการบล็อคของข้อมูลเป็นจำนวนหลาย ๆ Gigabytes
- ต้องการการประสิทธิภาพจำนวนมากในการจัดการ node เพื่อลดการเกิดคอขวด (bottlenecks)

2. ปัญหาการจัดเรียงข้อมูลจำนวนมาก

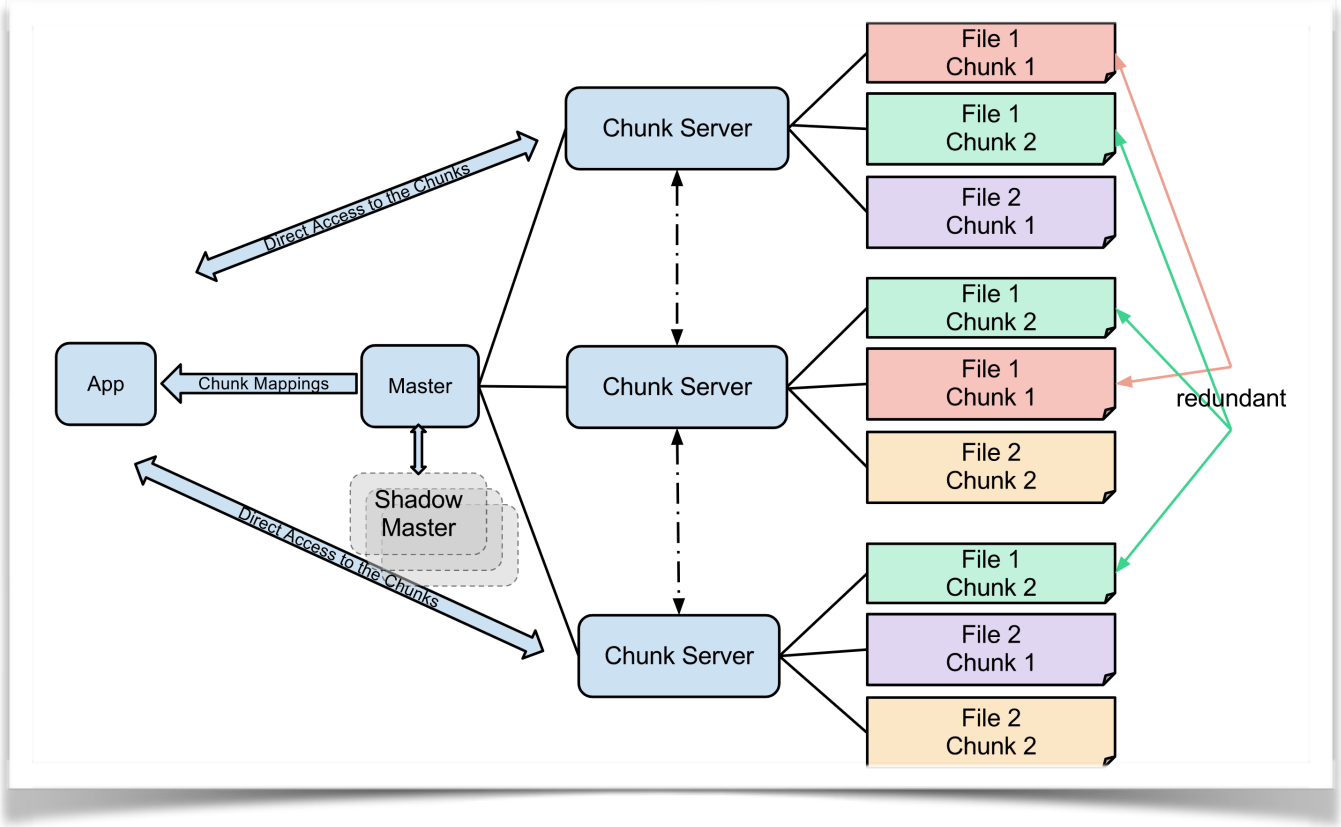
ในแต่ละวัน Google ต้องทำการจัดเรียงข้อมูลจำนวนมากเช่น 1 Petabytes / day หรือประมาณ 1,000,000 Gigabytes ต่อวันซึ่งนับว่าเป็นข้อมูลมหาศาลมากสำหรับการจะเรียงข้อมูลด้วยเครื่อง dual processor ด้วย ram เพียง 4 - 8 Gigabytes ต่อเครื่อง

3. ปัญหาการจัดเก็บข้อมูลจำนวนมาก

Google ต้องการจัดเก็บข้อมูลจำนวนมากเช่น ภาพดาวเทียมขนาดมากกว่า 100 Terabytes หรือจะเป็น URL, keywords การค้นหาเป็นจำนวนพัน ๆ ล้านลงใน database แต่ database ณ ขณะนั้นไม่ได้ตอบสนองการเก็บข้อมูลเป็นพัน ๆ ล้าน records แต่ถึงจะรองรับการจัดเก็บข้อมูลจำนวนมหาศาลแน่นอนว่าค่าใช้จ่ายที่จะตามมาก็จะสูงมากด้วยเช่นกัน

เทคนิควิธี หรือสถาปัตยกรรมที่ใช้ในการแก้ปัญหาในการให้บริการ

1. การแก้ปัญหา File System ที่ไม่ตอบสนองความต้องการด้วย Google File System (GFS)



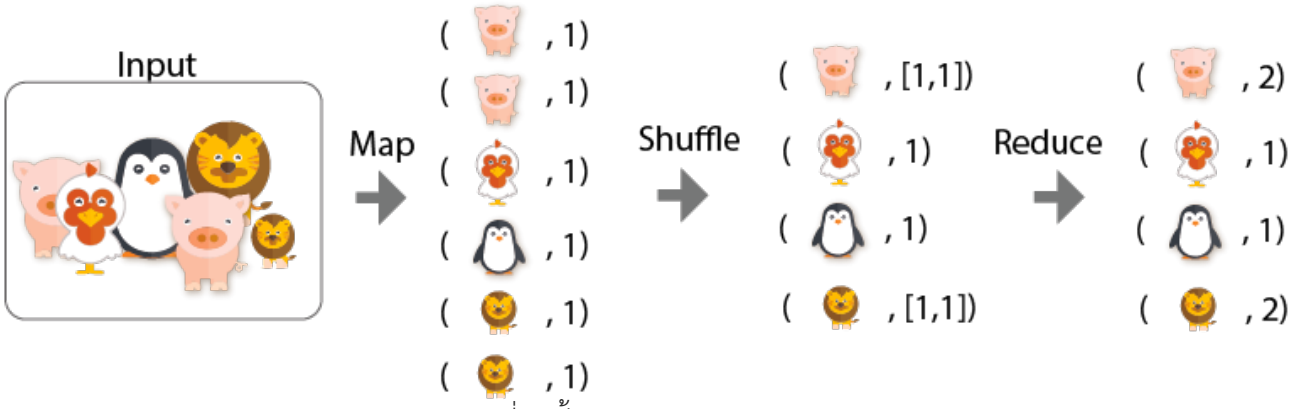
ภาพที่ 3 Google File System (GFS)

นักพัฒนาของ Google ได้ตัดสินใจที่จะสร้าง File System ของตนเองที่ชื่อว่า Google File System (GFS) เพื่อตอบสนองความต้องการที่แตกต่างของตนในด้านที่ได้กล่าวไปข้างต้น

- การสเกลเครื่องในเครือข่ายเดียวกัน
- การจัดการ read/write bandwidth
- การจัดการ blocks of data
- การจัดการที่มีประสิทธิภาพระหว่าง nodes เพื่อลด bottlenecks

ซึ่งจากภาพโครงสร้างของระบบ Google File System จะเห็นได้ว่ามีการกระจายข้อมูลไปแต่ละ Chunk Server จึงทำให้การอ่านข้อมูลเป็นไปได้ด้วยความรวดเร็วมากขึ้นโดยจะแบ่งไฟล์เป็น fixed size ขนาด 64 Megabytes และกระจายไปให้กับ Chunk Server หลาย ๆ ตัว ถ้ามี Application ต้องการใช้งานก็จะร้องขอไฟล์ไปที่ Master และ Master จะเป็นตัวกลางที่คอยบอกว่าไฟล์นี้ถูกแบ่งไว้ที่ Chunk Server ที่ไหนบ้าง และจะมีการสำรองข้อมูลของไฟล์ส่วนของตัวเองกระจายไปยังแต่ละ Chunk Server เพื่อเป็นการสำรองข้อมูลและช่วยในการเข้าถึงข้อมูลได้รวเร็วยิ่งขึ้น

2. การแก้ปัญหาการจัดเรียงข้อมูลจำนวนมากด้วย MapReduce



ภาพที่ 4 ขั้นตอนการ MapReduce

MapReduce เป็นรูปแบบการเขียนโปรแกรมที่เกี่ยวข้องกับการดำเนินและสร้าง data sets ที่ user โดยจะใช้ user ระบุ map function ที่จะต้องนำไปประมวลผลกับ key, value pair เพื่อที่จะสร้างตัวกลางของ key, value pair และ reduce function ที่จะคอยทำการรวบรวมข้อมูลที่มี key กลางเดียวกันมาไว้ด้วยกันโดยจะมี 3 ขั้นตอนหลัก ๆ คือ Map, Shuffle และ Reduce

Map - ทำหน้าที่จับคู่ข้อมูลที่เป็น key, value เดียวกัน โดยผลลัพธ์ที่ได้คือ (Intermediate Key หรือคีย์ที่ได้หลังจาก function Map)

Shuffle - ทำหน้าที่รวบรวมข้อมูลที่มี key เหมือนกันมาไว้ด้วยกัน

Reduce - เป็นขั้นตอนที่นำเอา Intermediate Key มาแล้วดึงเอาข้อมูลที่ key ตรงกันมาดำเนินการร่วมกันเพื่อให้ได้ผลลัพธ์ที่ขนาดเล็กสุดเท่าที่จะเป็นไปได้

3. การแก้ปัญหาการจัดเก็บข้อมูลจำนวนมากด้วย BigTable

ตามที่ได้กล่าวไปข้างต้นว่าข้อมูลของ Google นั้นมีมากมายและ File System ปัจจุบันไม่ตอบโจทย์ตนเองมากนักจนถึงกับต้องสร้าง File System ของตนเองที่มีชื่อว่า Google File System ขึ้นมา และแน่นอนว่าการเก็บข้อมูลด้วยเช่นกัน BigTable เป็นการเก็บข้อมูลแบบบีบอัด ประสิทธิภาพสูง ถูกสร้างและถูกใช้อยู่บน Google File System อีกทีหนึ่ง BigTable เป็นการเก็บข้อมูล (data storage) ที่ใช้ในผลิตภัณฑ์ของ Google อยู่ผลิตภัณฑ์เช่น Google Maps, Youtube, Gmail, Google Earth และอื่น ๆ

ผลที่ได้เมื่อใช้เทคนิควิธีดังกล่าว

1. การแก้ปัญหา File System ที่ไม่ตอบสนองความต้องการด้วย Google File System (GFS)

ทำให้ Google สามารถทำการ Horizontal Scaling ด้วยการเพิ่ม node ของเครื่องคอมพิวเตอร์ได้โดยไม่ต้องกังวลกับการจัดเก็บไฟล์ และ Google File System นั้นเป็น Reliable access สำหรับการเข้าถึงข้อมูลที่กลุ่มของ cluster มีขนาดใหญ่มาก มีการกระจายข้อมูลไปให้แต่ละ Chunk Server ซึ่งแต่ละ Server นั้นจะมีชิ้นส่วนของอีก Server หนึ่งอยู่ด้วยจึงไม่ต้องตกเป็น Single Point of Failure ในกรณีที่โดนโจมตีหรือเกิดข้อมูลสูญหายขึ้น เราสามารถเรียกข้อมูลจากเครื่องอื่นแทนได้

ประสิทธิภาพในการทำงานการอ่านข้อมูลจะสามารถทำได้อย่างรวดเร็วมากเพราะสามารถดึงชิ้นส่วนข้อมูลที่ถูกแบ่งไว้ใน Chunk Server จากหลาย ๆ Chunk Server ได้ทำให้การอ่านข้อมูลรวดเร็วมากเมื่อมีจำนวน node ของเครื่องเยอะขึ้น แต่ก็ต้องแลกด้วยประสิทธิภาพในการเขียนข้อมูลด้วยเช่นกันในการ update ข้อมูลจำเป็นต้องคอยอัพเดทชิ้นส่วนจาก Chunk Server ในแต่ละตัวถ้าหากมีกาอัพเดทข้อมูลเพิ่มเติม

2. การแก้ปัญหาการจัดเรียงข้อมูลจำนวนมากด้วย MapReduce

MapReduce จะทำการประมวลผลข้อมูลและสร้าง big data sets ด้วย parallel algorithm ซึ่งการประมวลผลแบบ parallel จะทำได้รวดเร็วมากยิ่งขึ้นยิ่งมี node ของเครื่องคอมพิวเตอร์เยอะยิ่งขึ้นซึ่งประสิทธิภาพก็จะยิ่งดีขึ้นแต่ก็ต้องแลกด้วยค่าใช้จ่ายที่เยอะขึ้นด้วยเช่นกัน

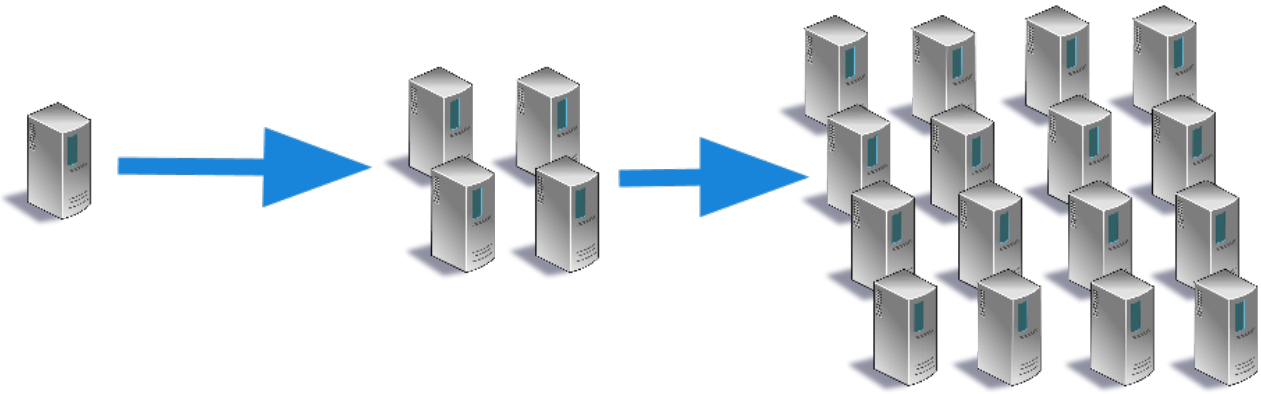
3. การแก้ปัญหาการจัดเก็บข้อมูลจำนวนมากด้วย BigTable

BigTable เป็นการจัดเก็บข้อมูลแบบ NoSQL ซึ่งข้อดีคือเก็บข้อมูลแบบ Unstructured ซึ่งไม่ต้อง concern กับ column ของตารางนั้น ๆ โดยสามารถเพิ่มข้อมูลฟิลด์ใดที่ในอนาคตจำเป็นต้องเพิ่มฟิลด์นั้น แต่เราไม่ต้อง reconfiguration ฐานข้อมูลใหม่ทั้งหมด

BigTable ถูกออกแบบโดย Google ที่ถูกใช้งานบน Google File System อีกทีหนึ่งซึ่งผลดีก็คือไม่ต้องเสียเวลากับการทำงานอย่างอื่นที่ไม่ใช่สิ่งที่ต้องทำ และสามารถแสดงประสิทธิภาพการทำงานและการประมวลผลได้ดียิ่งกว่าการใช้ฐานข้อมูลที่ไม่ได้ออกแบบและการดำเนินการขึ้นเอง

อภิปรายความสัมพันธ์กับเนื้อหาที่เรียนในชม.บรรยาย

1. มีการทำ Horizontal Scaling



Google ได้มีการทำ cluster กล่าวคือ การนำกลุ่มของระบบคอมพิวเตอร์ ซึ่งเรียกว่าโหนด (nodes) มาทำงานร่วมกันโดยหลัก ๆ เพื่อใช้ในการอ่าน/เขียนข้อมูล/ประมวลผลข้อมูลซึ่งเราสามารถเพิ่มขีดความสามารถของระบบได้มากยิ่งขึ้นหากเราทำการเพิ่ม node จำนวนมากขึ้น

2. มีการใช้ NoSQL เพื่อทำการ Scaling

เนื่องจาก Google Search เป็นบริการที่ค้นหา content ในเว็บและจำเป็นต้องเก็บ URL, keywords รวมถึงข้อมูลการเปลี่ยนแปลงต่าง ๆ บ่อย การใช้ NoSQL เพื่อทำการเก็บข้อมูลจึงเหมาะสมที่จะทำ Scalability มากกว่าและรองรับการ read/write กับข้อมูลจำนวนมาก ๆ ได้เร็วกว่า SQL ในกรณีที่ข้อมูลมหาศาลและเหมาะสำหรับข้อมูลที่เปลี่ยนแปลงบ่อย

3. มีการใช้ RAID (Redundant Array of Independent Disks) และ Data Partitioning (Sharding)

เนื่องจากการเก็บไฟล์รูปแบบของ Google File System มีลักษณะการกระจายไฟล์ออกไปเป็นแต่ละส่วนใน Chunk Server จึงทำให้เมื่อเกิดเหตุการณ์ไม่คาดฝันขึ้นเช่น Hard disk เสียหรือเซิฟเวอร์เปิดไม่ติด เรายังจะสามารถเรียกดูข้อมูลไฟล์ส่วนต่อไปจาก Server เครื่องอื่น ๆ ได้โดยไม่ต้องขึ้นอยู่กับเครื่องเดียว

4. มี Availability ที่สูงมาก

จะเห็นได้ว่ามีการทำทั้ง RAID, Data Partitioning ที่ช่วยในการเข้าถึงข้อมูลและสำรองข้อมูลในเวลาเดียวกันจึงทำให้ Server ไม่ Down หรือ failed ง่าย ๆ

5. มี Reliability ที่สูงมาก

Google จะมีการทำ MapReduce ของข้อมูลที่เก็บมาได้ในแต่ละวันทำให้ข้อมูลที่ได้น่าเชื่อถือและมีความใหม่อยู่ตลอดเวลา

แหล่งอ้างอิงที่น่าเชื่อถือ

[1] http://highscalability.com/google-architecture
[2] http://www.25hoursaday.com/weblog/2007/06/25/GoogleScalabilityConferenceTripReportMapReduceBigTableAndOtherDistributedSystemAbstractionsForHandlingLargeDatasets.aspx
[3] https://en.wikipedia.org/wiki/Google_File_System
[4] https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf
[5] https://medium.com/@aorjoa/mapreduce-%E0%B8%84%E0%B8%B7%E0%B8%AD%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3-62c33f8d9923