Scalability of NetFlix

โดย

นายปวัณ ปั้นพินิจ

เลขประจำตัวนักศึกษา 5809610107

วิชาระบบปฏิบัติการ 2 คพ.447

คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยธรรมศาสตร์

NetFlix คืออะไร

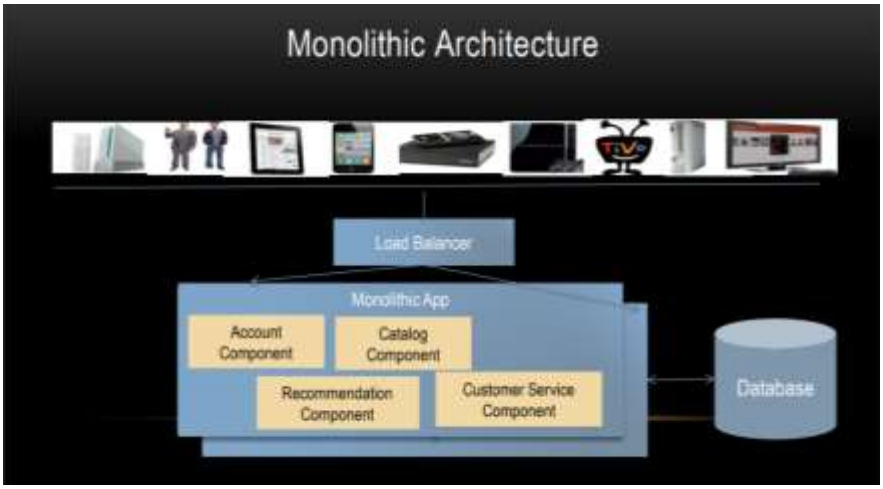
ต้องยอมรับว่าทุกวันนี้ การเจริญเติบโตของเทคโนโลยีนั้นเร็วมาก ทั้งในด้านซอฟต์แวร์และฮาร์ดแวร์ อินเตอร์เน็ตก็เป็นอีกหนึ่งเทคโนโลยีที่เติบโตเร็วและพัฒนาเร็วมาก โดยในปัจจุบันอินเตอร์เน็ตได้เข้ามามีบทบาทในการดำรงชีวิตของเรา เพราะทำให้การใช้ชีวิตประจำวันของเราง่ายมากขึ้น โดยเห็นได้จากมีบริการต่างๆมากมายที่ให้ได้ใช้ผ่านอินเตอร์เน็ต ทั้งทางสังคม เช่น (Facebook, LINE และ twitter) ซื้อของออนไลน์ รวมไปถึงการทำธุรกรรมทางธนาคาร ก็สามารถทำได้ง่ายผ่านทางอินเตอร์เน็ตแล้วเช่นกัน และแน่นอนว่าโลกของอินเตอร์เน็ตเป็นโลกที่กว้างมาก จึงทำให้เกิดบริการที่ให้ความบันเทิงด้วยเช่นกัน นั่นคือ Music, Video streaming ซึ่งแต่ล่ะบริการก็จะมีการให้บริการที่แตกต่างกันไป แต่วันนี้เราจะมาพูดถึงบริการตัวหนึ่งที่ชื่อว่า NetFlix

NetFlix เป็นบริการ Video Streaming ที่ให้บริการทางด้านความบันเทิง โดยให้บริการคอนเท้นต์ประเภท ภาพยนตร์ ซีรีส์ รายการโทรทัศน์ ผ่านทาง Video Streaming ซึ่งเสียค่าใช้บริการ NetFlix ก่อตั้งในปี ค.ศ. 1997 โดย แรดดอฟ (Randolph) และ รีด แฮสติ้งส์ (Reed Hathings) ในเมือง สก็ตวาเลย์ รัฐแคลิฟฟอเนีย ประเทศสหรัฐอเมริกา

ปัจจุบันมีผู้ใช้บริการของ NetFlix มากกว่า 69 ล้านคน ในกว่า 50 ประเทศ โดยตัวเลขของการทำ Video Streaming ทั้งรายการโทรทัศน์ ภาพยนตร์ และซีรีส์ ในหนึ่งเดือนนั้น มากกว่า 10 พันล้านชั่วโมงเลยทีเดียว ซึ่งถือเป็นตัวเลขที่เยอะมาก ในบทความนี้ เราจะมาพูดถึงการจัดการและการขยาย (Scalability) ระบบของ NetFlix กัน

ปัญหาที่เกิดขึ้นกับ NetFlix

ปัญหาหลักที่เกิดขึ้นกับ NetFlix นั้น คือจำนวนผู้ใช้เพิ่มมากขึ้นในระดับที่สูงมาก ทำให้สิ่งที่ NetFlix ใช้อยู่นั้นไม่สามารถรองรับจำนวนผู้ใช้ที่เติบโตขึ้นมากขนาดนี้ได้ ปัญหาดังกกล่าวเกิดจากปัจจัยดังต่อไปนี้



รูปที่ 1 แสดงสถาปัตยกรรมแบบ Monolithic Architecture ของ NetFlix
(ที่มา https://www.infoq.com/presentations/netflix-ipc)

สิ่งแรกคือสถาปัตยกรรมที่ NetFlix ใช้เป็นแบบ Monolithic Architecture โดยความหมายของสถาปัตยกรรมแบบ Monolithic ก็คือ ทุกอย่างจะรวมเป็นก้อนเดียวกัน ทุกฟังค์ชั่น ทุกความสามารถจะรวมเป็นโปรแกรมเดียว การพัฒนาในสถาปัตยกรรมนี้ ค่อนข้างที่จะไมมีความหยืดหยุ่น เนื่องจากต้องทำงานในสภาพแวดล้อมแบบเดียว นั่นคือทุกส่วนจะถูกพัฒนาโดยภาษาเดียวกัน ด้วยความที่โปรแกรมทั้งโปรแกรมเป็นก้อนเดียวกันหมด จะทำให้ตัวโปรแกรมใหญ่มาก มีโค้ดหลายพันบรรทัด นำไปสู่การพัฒนาและบำรุงรักษาระบบที่ยาก และปัญหาที่สำคัญที่สุดของสถาปัตยกรรมแบบนี้คือ เมื่อมีส่วนใดส่วนหนึ่งไม่สามารถทำงานได้ จะส่งผลกระทบกับทุกส่วนที่อยู่ในโปรแกรมเดียว ไม่สามารถทำงานต่อไปได้หรือทำให้มีประสิทธิภาพลดลง จากรูปที่ 1 เระจะเห็นว่าทุกช่องทางการใช้งาน (Edge of API) NetFilx ไม่ว่าจะเป็นจาก Mobile Application, Web Application และชื่องทางอื่นๆ เมื่อเข้ามาใช้ NetFlix จะถูกกระจาย request ด้วย Load Balancer และเข้ามาทำงานในโปรแกรมที่เป็นสถาปัตยกรรมแบบ Monolithic ถ้าเหตุการนี้เกิดขึ้นในช่วงแรกๆ ที่ผู้ใช้งานยังมีน้อย ก็จะสามารถใช้งานได้มีประสิทธิภาพ แต่เมื่อถึงจุดที่มีผู้ใช้งาน request เข้ามาเยอะมากจะทำให้ระบบช้ามาก ในด้านการขยายระบบในส่วนของฮาร์ดแวร์นั้น สามารถทำ Horizontal Scaling ได้ แต่ด้วยความที่เป็นความเป็นสถาปัตยกรรมแบบ Monolithic ทำให้ไม่เห็นผลเนื่องจากทั้งโปรแกรมยังเป็นส่วนเดียวกันอยู่

ปัญหาที่สองคือ Traffic NetFlix มี Datacenter เป็นของตัวเอง (ปัจจุบันไม่แน่ใจว่ายังมีไหม) เมื่อจำนวนผู้ใช้งานเพิ่มขึ้น traffic ที่ส่งมายัง Datacenter แห่งนี้ก็มีเพิ่มมากขึ้น นั้นทำให้การปัญหาถัดมาคือ ปัญหาด้าน traffic ที่เข้ามายัง Datacenter

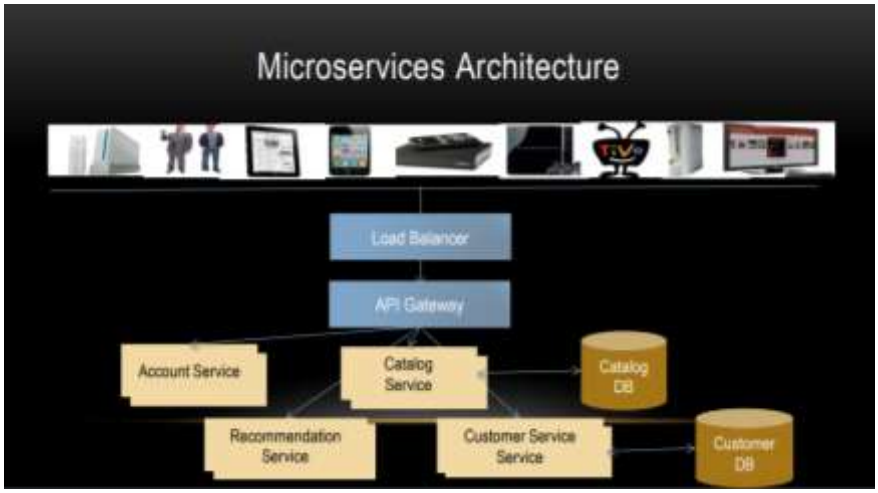
ปัญหาที่สามเป็นปัญหาที่สอคล้องกับปัญหาที่หนึ่งคือ ด้วยสถาปัตยกรรมหลักของระบบเป็น Monolithic เมื่อยุคสมัยผ่านไปช่องทางในการเชื่อมต่อมายังระบบก็เพิ่มมากขึ้น ด้วยความที่ยังเป็น Monolithic อยู่จึงทำให้การทำงานโดยการเรียกใช้ผ่านหลายช่องทางนั้นไม่สามารถทำได้ดีนัก

เทคนิควิธี หรือสถาปัตยกรรมที่ใช้ในการแก้ปัญหาของ NetFlix

ในการแก้ไขปัญหานั้น จะขอแยกวิธีการแก้ปัญหาเป็นสามหัวข้อตามปัญหาที่เกิดขึ้น

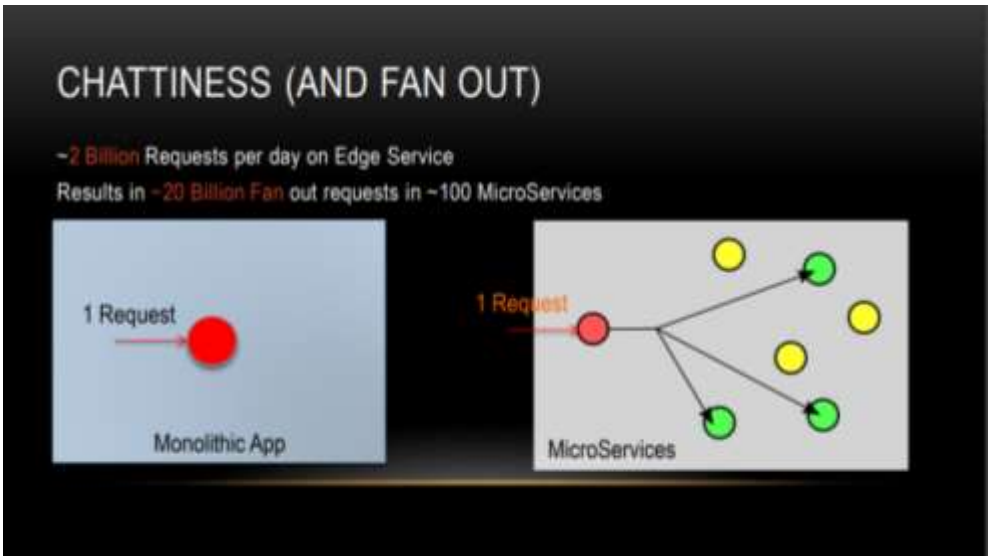
ปัญหาแรกคือ สถาปัตยกรรมแบบ Monolithic โดย Netflix ได้ทำการแก้ปัญหานี้โดยการแบ่งแยกฟังค์ชั่นต่างๆให้ออกเป็นส่วนๆ หรือเรียกอีกอย่างหนึ่งว่าสถาปัตยกรรมแบบ MicroServices โดยสถาปัตยกรรมแบบ MicroServices นั้น เป็นการออกแบบให้ทุกบริการหรือทุกฟังค์ชั่นนั้น เป็นอิสระต่อกัน ซึ่งสถาปัตยกรรมแบบนี้เหมาะกับองค์กรที่มีขนาดใหญ่ มีหลายบริการ คือแต่ละบริการหรือฟังค์ชั่นจะไม่ขึ้นต่อกัน ทุกตัวทำงานอิสระโดยเรียกใช้ซึ่งกันและกัน อีกทั้งยังดูแลบำรุงรักษาง่าย เนื่องจากแต่ละบริการสามารถใช้ภาษาในการพัฒนาที่

ต่างกันได้ ทำให้มีความยืดหยุ่นในการพัฒนา และมีประสิทธิภาพในด้านการใช้งาน นอกจากนี้การขยายระบบในด้าน Hardware ก็สามารถทำได้ง่ายขึ้นและมีประสิทธิภาพมาขึ้นทั้งในด้าน Vertical และ Horizontal scaling



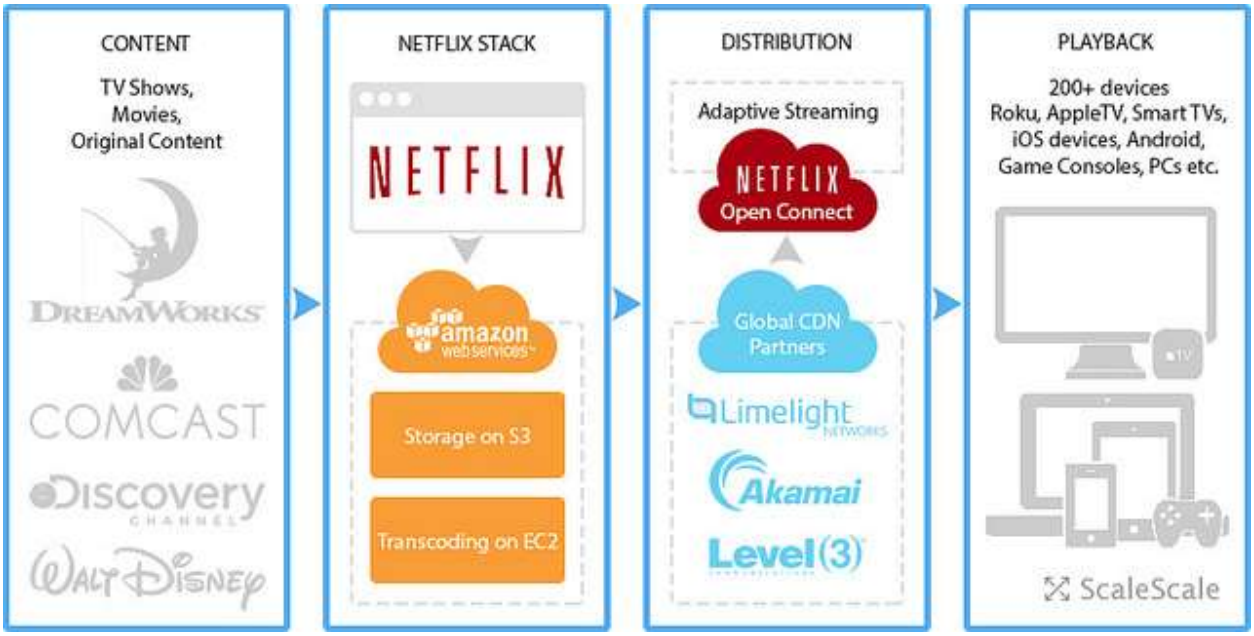
รูปที่ 2 แสดงสถาปัตยกรรมแบบ MicroServices Architecture ของ NetFlix
(ที่มา https://www.infoq.com/presentations/netflix-ipc)

อีกความแตกต่างกันระหว่างสถาปัตยกรรมแบบ Monolithic และ MicroServices คือเรื่องของการรองรับ Request ที่เข้ามา ถ้าเป็นแบบ Monolithic นั้น Request ใดๆที่เข้ามาจาเรียกใช้ที่ตัวระบบใหญ่ระบบเดียวเลย แล้วตัวระบบใหญ่จะจัดการการทำงานว่าต้องเรียกใช้ฟังค์ชั่นอะไร แต่ใน Microservice ต่างออกไป โดย MicroServices นั้น Request ที่เข้ามาจะระบุเลยว่าจะใช้บริการอะไร และบริการนั้นๆจะเป็นตัวรับ Request และดำเนินการด้วยตัวเอง ซึ่งถ้าหากบริการต่างๆนั้นอยู่ต่างเครื่องกัน ก็จะสามารถลดอัตราการใช้ทรัพยากรในเครื่องๆหนึ่งได้อย่างมีประสิทธิภาพ แต่แลกมาด้วยการเชื่อมต่อที่มากขึ้น



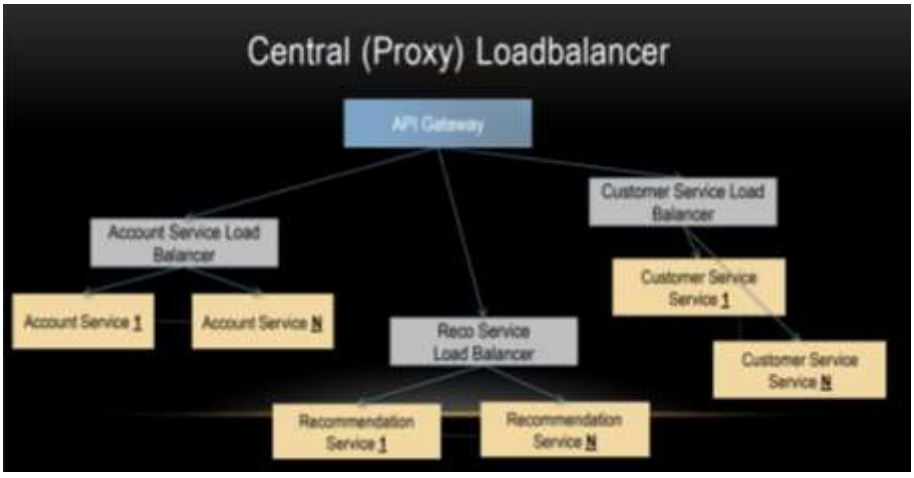
รูปที่ 3 แสดงความแตกต่างการรับ Request ในสถาปัตยกรรมแบบ Monolithic และ MicroServices
(ที่มา https://www.infoq.com/presentations/netflix-ipc)

ในส่วนของปัญหาที่สองที่เกี่ยวกับ Traffic Netflix ได้เปลี่ยนจากการทำ Datacenter ของตัวเองมาใช้ระบบแบบ Cloud services โดยใช้บริการจาก Amazon Web Services โดยระบบโครงสร้างพื้นฐานนั้น จะอยู่บนบริการ Amazon EC2 การเก็บไฟล์คอนเท้นต์ต่างๆ เช่น ภาพยตร์ ซีรีส์ รวมไปถึงรายการโทรทัศน์ จะถูกเก็บในบริการ Amazon S3 นอกจากนี้ ในด้านของ Static Content ต่างๆ Netflix ยังใช้บริการ CDN(Content Delivery Network) ด้วย ซึ่งทำให้ช่วยลด traffic ได้เป็นอย่างดี



รูปที่ 4 แสดงส่วนประกอบระบบโครงสร้างพื้นฐานของ NetFlix
(ที่มา http://highscalability.com/blog/2015/11/9/a-360-degree-view-of-the-entire-netflix-stack.html)

และสุดท้ายปัญหาเกี่ยวกับช่องทางที่เชื่อมต่อเข้ามา เนื่องจากทาง NetFlix ได้พัฒนาระบบใหม่เป็นสถาปัตยกรรมแบบ MicroServices ทำให้เกิดการพัฒนาเป็น API ขึ้นมา ซึ่งจะรองรับการเชื่อมต่อจากช่องทางต่างได้ง่ายขึ้น และมีประสิทธิภาพมากขึ้น



รูปที่ 5 แสดงการเรียกใช้บริการต่างๆผ่าน APIs ของ NetFlix
(ที่มา https://www.infoq.com/presentations/netflix-ipc)

ผลที่ได้เมื่อใช้เทคนิควิธีดังกล่าว

ผลจากการใช้เทคนิคดังกล่าวในหัวข้อที่แล้ว สามารถอภิปรายเป็นสามหัวข้อดังนี้

หัวข้อแรกคือการเปลี่ยนมาเป็นสถาปัตยกรรมแบบ MicroServices ผลที่ได้คือทำให้การพัฒนาและดูแลรักษาระบบง่ายขึ้น เนื่องจากแต่ละบริการไม่ขึ้นต่อกัน พัฒนาได้ในสภาพแวดล้อมที่แตกต่าง ทำให้สามารถมีทีมพัฒนาที่หลายหลายได้ และสามารถใช้ลักษณะเฉพาะของสภาพแวดล้อมนั้นๆในการพัฒนาได้อีกด้วย ในการพัฒนาหนึ่งบริการจะไม่กระทบต่อบริการอื่นๆ ที่สำคัญการลดการใช้ทรัพยากรต่อเครื่อง เนื่องจากแต่ละบริการอาจจะอยู่คนละที่กันก็ได้ รวมถึงยังช่วยจัดการ Request มากๆที่เข้ามาได้

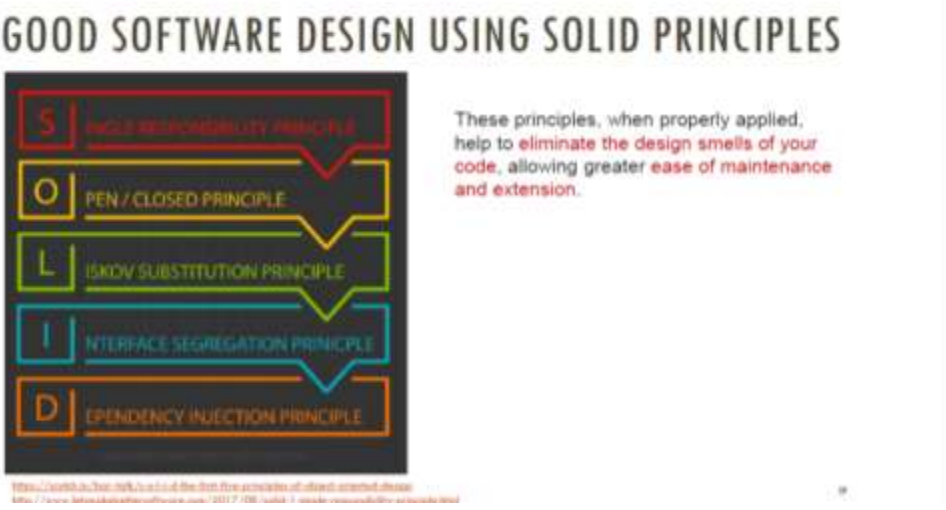
หัวข้อที่สองคือการเปลี่ยนมาใช้ Cloud service ซึ่งผลที่ได้จากการเปลี่ยนในครั้งนี้คือได้บริการที่ยืดหยุ่นด้วยทาง Amazon Web Service ก็รองรับบริการในหลายพื้นที่ ทำให้กระจายแหล่งข้อมูลไปทั่วทุกที่ และการใช้บริการ CDN นั้น ทำให้การโหลดข้อมูลพวกที่เป็น Static Content มีความรวดเร็วยิ่งขึ้น ซึ่งช่วยลด Traffic ที่จะเข้ามายังเครื่อข่ายหลักได้

และหัวข้อสุดท้ายคือการจัดกากปัญหาช่องทางที่เชื่อมต่อเข้ามา ทำให้การพึ่งพิงกันระหว่างบริการที่ไม่จำเป็นลดลง ซึ่งทำให้เพิ่มประสิทธิภาพแก่ทรัพยากรที่มีอยู่ได้เป็นอย่างดี

อภิปรายความสัมพันธ์กับเนื้อหาที่เรียน

ความเกี่ยวข้องจากเนื้อหาที่เรียนกับการปรับปรุงระบบของ NetFlix มีดังนี้

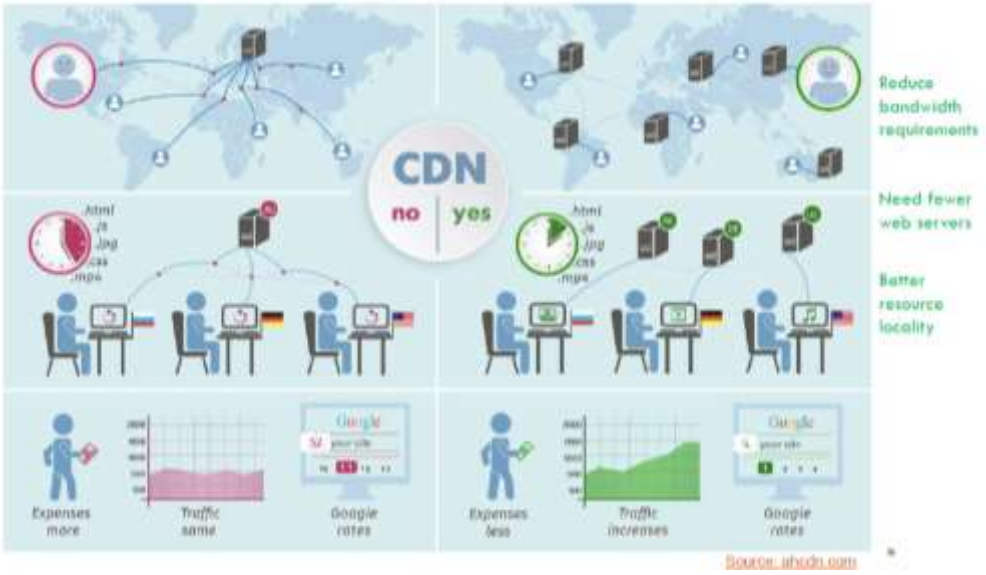
1. Good Software Design



จะเห็นได้ว่า ในการออกแบบระบบของ NetFlix ในช่วงแรกที่ทำให้เกิดปัญหาคือสถาปัตยกรรมแบบ Monolithic ซึ่งพูดให้เข้าใจง่ายๆก็คือทุกอย่างอยู่ในโปรแกรมเดียวกันหมดเลย และไม่นึกถึงการเติบโต

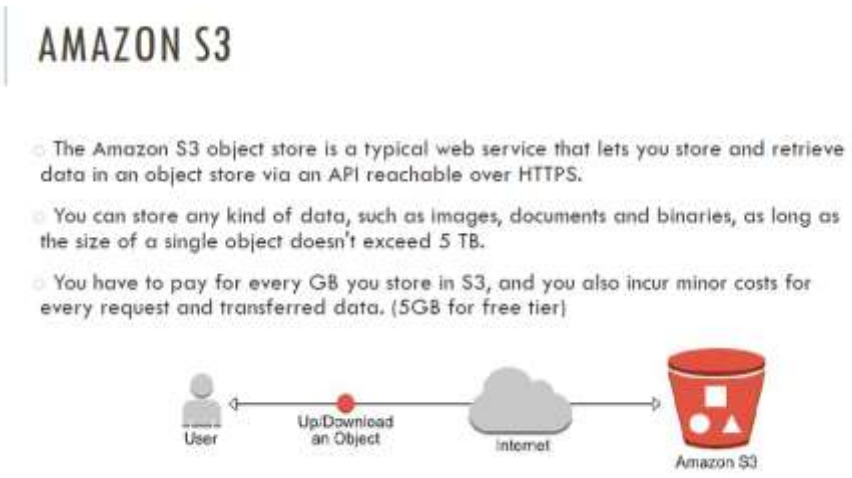
ของระบบในภายภาคหน้า ซึ่งถ้าหากใช้หลักการออกแบบซอฟต์แวร์ตามที่เรียนมา เช่น Sigle Responsibility Principle มาออกแบบระบบ จะทำให้ระบบมีระบบย่อย ไม่ใหญ่เกินไป บำรุงรักษาง่าย

2. CDN: Content Delivery Network



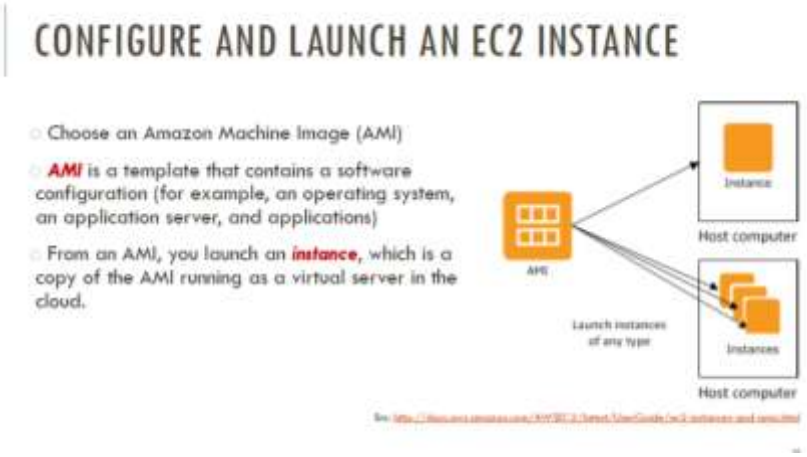
CDN เป็นส่วนหนึ่งที่ NetFlix ใช้ในการ Scale ระบบ ซึ่งถือว่ามีประโยชน์มาก ซึ่งช่วยลด traffic ที่จะเข้ามายังระบบหลักได้ ซึ่งในชั่วโมงเรียนได้ชี้ให้เห็นถึงความสำคัญของ CDN

3. Amazon Web Service S3



เป็นหนึ่งในบริการของ Amazon Web Service ซึ่ง NetFlix ได้นำมาใช้ในการ Scale ระบบ ในส่วนของการจัดเก็บไฟล์คอนเท้นต์ต่างๆ ซึ่งมีบรรยายในชั่วโมงเรียน

4. Amazon Web Service: EC2



ระบบโครงสร้างพื้นฐานของ NetFlix อยู่บน E2 ซึ่งมีบรรยายในชั่วโมงเรียน

5. Web APIs

NetFlix ได้สร้าง API ขึ้นมาเพื่อให้การเรียกใช้มายังบริการต่างมีประสิทธิภาพมากขึ้น ซึ่งมีการอธิบายให้เข้าใจในชั่วโมงเรียน

แหล่งข้อมูล

Scalable Microservices at Netflix. Challenges and Tools of the Trade

https://www.infoq.com/presentations/netflix-ipc

NetFlix

https://en.wikipedia.org/wiki/Netflix

A 360 Degree View Of The Entire Netflix Stack

http://highscalability.com/blog/2015/11/9/a-360-degree-view-of-the-entire-netflix-stack.html

The NetFlix Tech Blog

https://medium.com/netflix-techblog