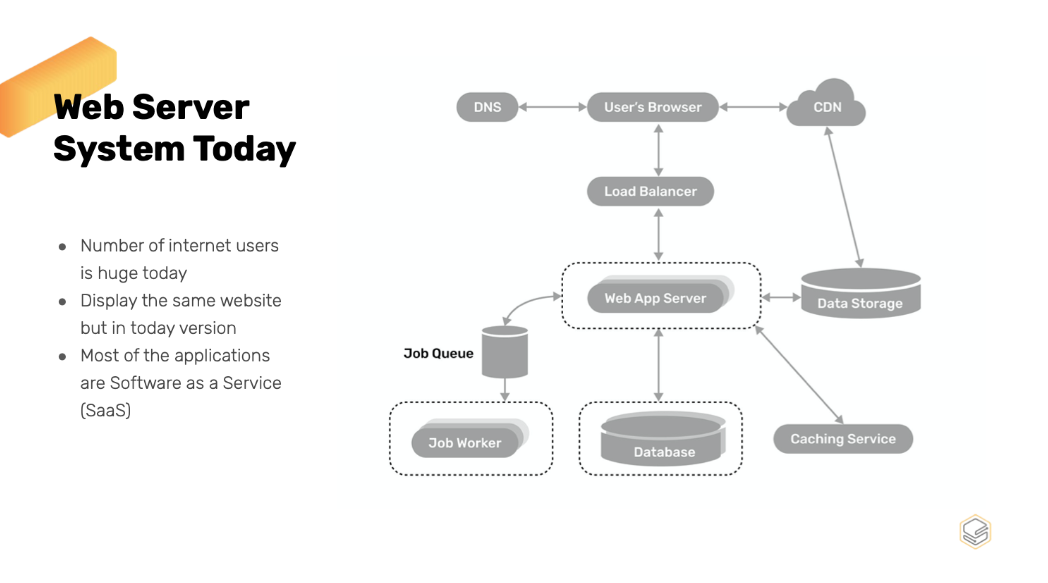
**System Design**

**1.Web Server System Today**

การออกแบบเว็บไซต์เพื่อรองรับผู้ใช้จำนวนมาก

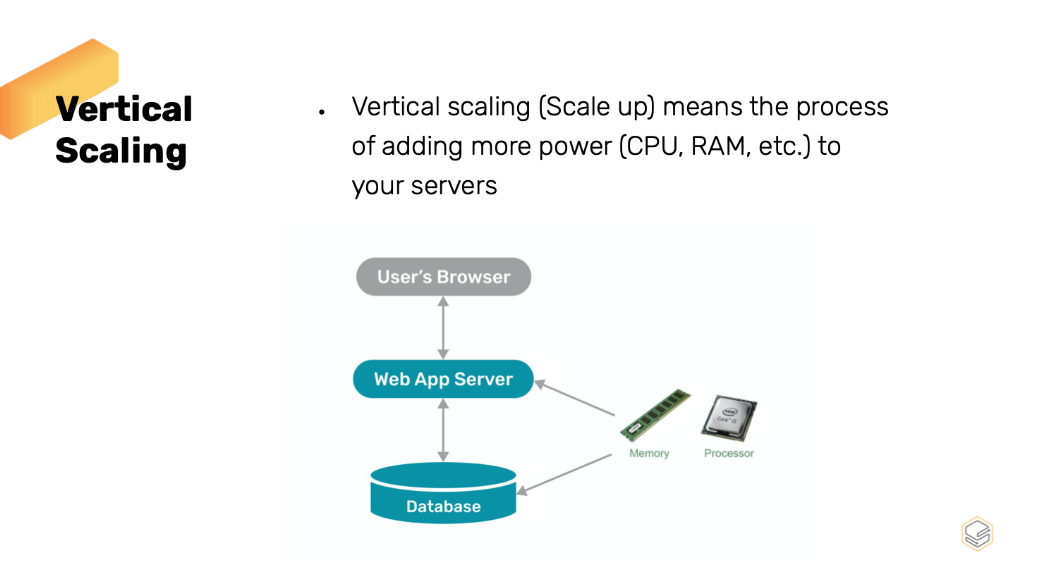


**2.Scaling**

**Vertical Scaling** **(การติดตั้งแนวตั้ง)** = เพิ่มความแรง CPU, RAM, etc ของ Server

ข้อดี - ง่ายในการติดตั้ง ไม่ต้อง Config

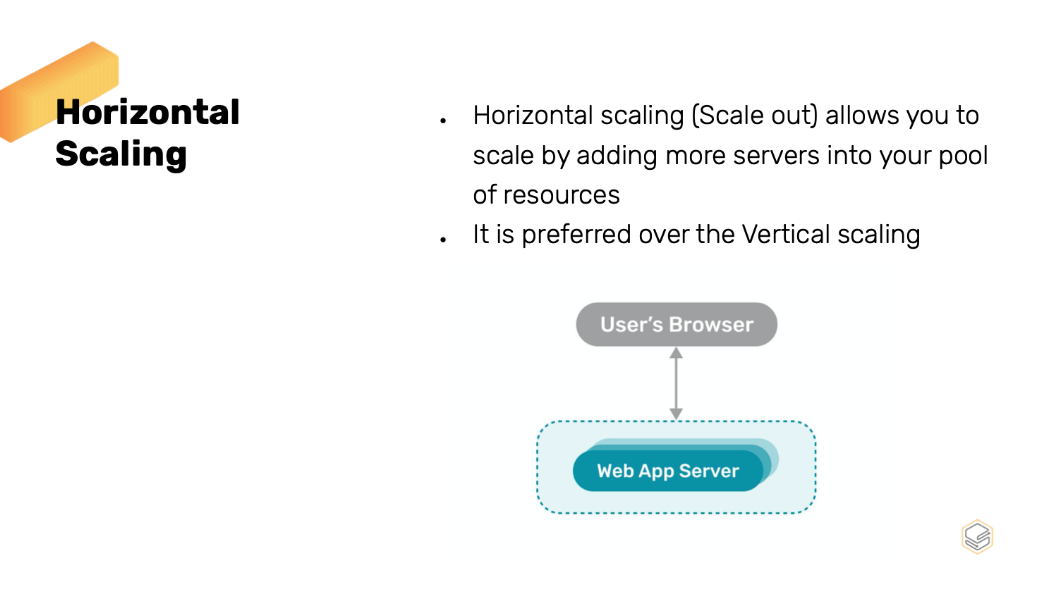
ข้อเสีย - เมื่อเทียบกับขนาดข้อมูล ราคาแพงกว่า



**Horizontal Scaling** **(การติดตั้งแนวราบ)** = เพิ่มจำนวน Server ตัวใหม่เลย

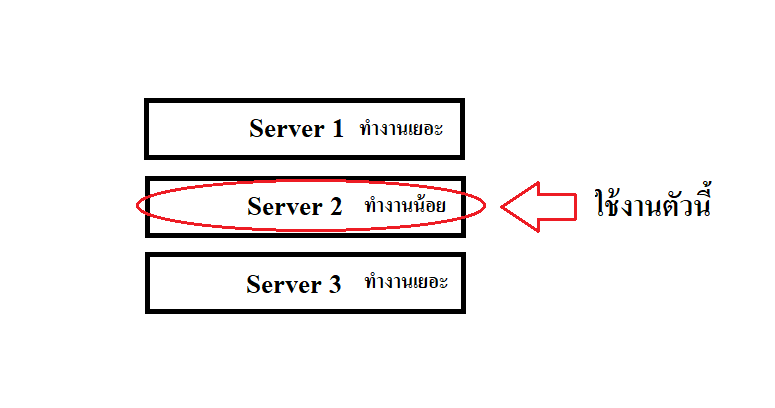
ข้อดี - เมื่อเทียบกับขนาดข้อมูล ราคาถูกกว่า, รองรับ User มากกว่า

ข้อเสีย - ต้อง config ใหม่ทุกตัว ยุ่งยาก



**3.Load Balancer**

- **Low latency** มีหน้าที่เป็นตัวกลางในการตรวจสอบ CPU Server ที่มีการทำงานน้อยสุด เพื่อตอบสนองการทำงานของของผู้ใช้เร็วที่สุด



- **High Availability** มีหน้าที่ตรวจสอบว่า Server1 มีปัญหาก็จะทำการย้ายการทำงานไปหา Server2 หรือ Server3,4,5 ระบบเราจะไม่ล่มง่ายๆ ถ้ากลับมาใช้งานได้ก็จะทำงานปกติ

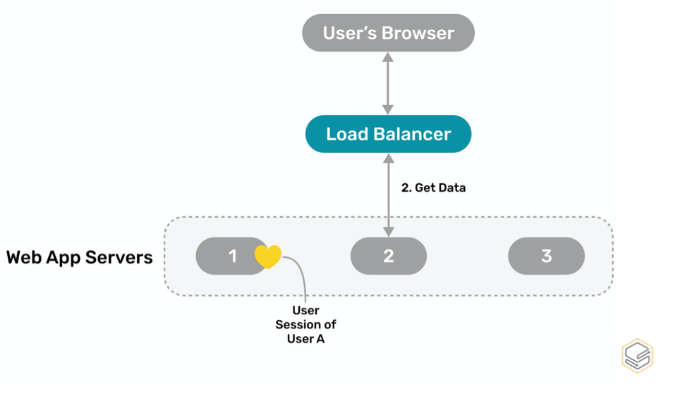
A picture containing text, screenshot, font, line

Description automatically generated

- **Horizontal Scaling** มีปัญหาเรื่อง State เช่น Login ค่าไว้ที่ Server1 แต่พอวิ่งไป Server2 ระบบไม่รู้จักค่าที่ Login มา ทำให้ต้องกลับไป login ใหม่อีกครั้ง เหตุที่ Server เปลี่ยนเพราะ Low latency ตรวจสอบความเร็วอยู่เสมอ ทำให้เปลี่ยน Server ไป – มา

A picture containing text, screenshot, diagram, font

Description automatically generated



A picture containing text, screenshot, diagram, design

Description automatically generated

**วิธีแก้คือ** นำข้อมูล Login ไปเก็บข้างนอกที่ไม่ใช่ Server เช่น คุกกี้,บราวเซอร์ เพื่อให้การเรียกใช้งานแต่ละครั้งเป็นการดึงข้อมูลจากภายนอกมาเช็คการ Login ได้ ไม่ต้อง Login ซ้ำ

หากจะเก็บข้อมูล Database ให้สร้าง Server แยก แล้วให้ Server ตัวอื่นๆวิ่งเข้าไปในตัวเดียวกัน

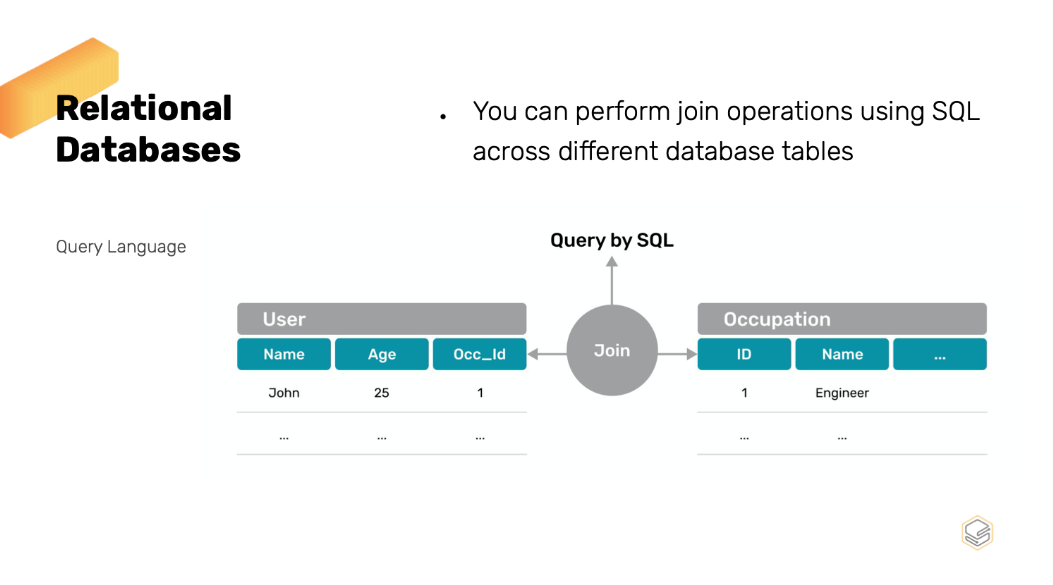
เรียก Server Database ประเภทนี้ว่า **Stateless Web Servers**

A picture containing text, screenshot, design

Description automatically generated

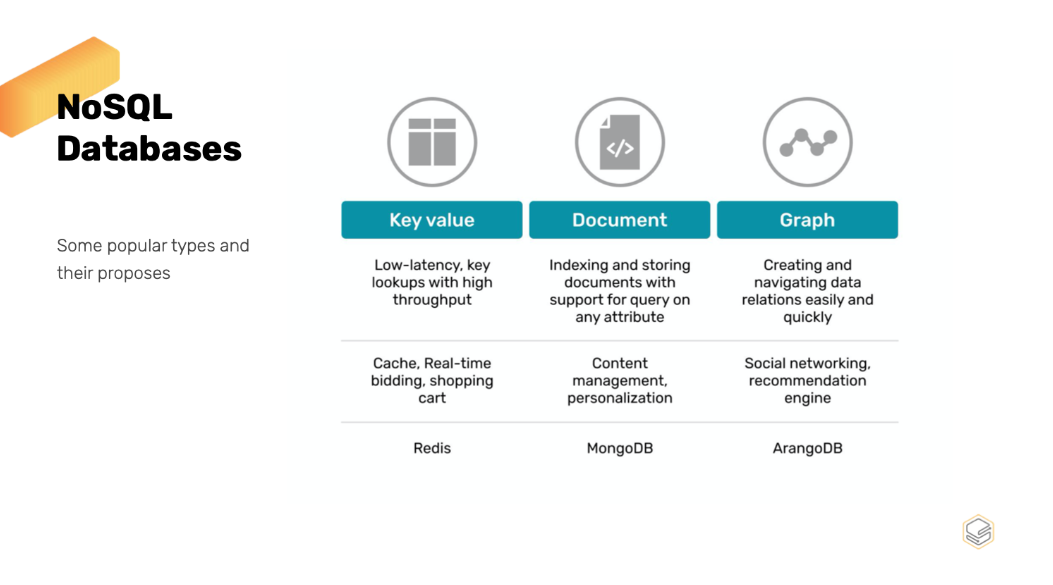
**4.Database**

- **Relational Database** สร้างหลายๆ table สามารถ Join ข้อมูลได้ และ Query by SQL ได้ ได้แก่ Microsoft SQL Server , PostgreSQL , MySQL



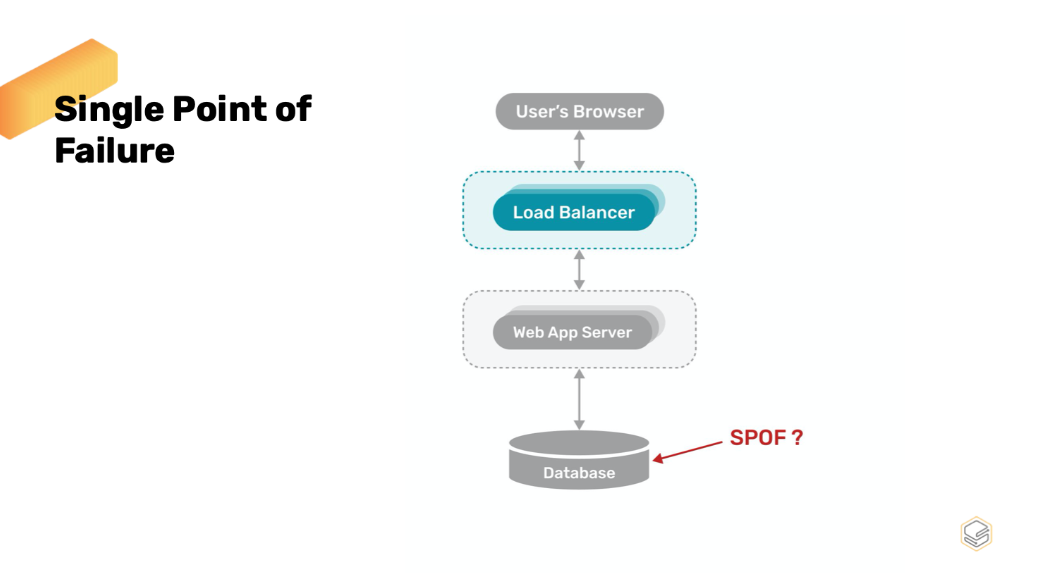
- **NoSQL** เป็นการเก็บข้อมูลจำนวนมากได้ แบบไม่ Structure ส่วนใหญ่ User เป็นคน Generate เอง ไม่แบ่งเป็น Table ยัดลง Database เลย และยังสามารถดึงข้อมูลออกอย่างรวดเร็วไม่ต้อง Join table

ได้แก่ Redis (Key value), MongoDB (Document), Arango DB (Graph)



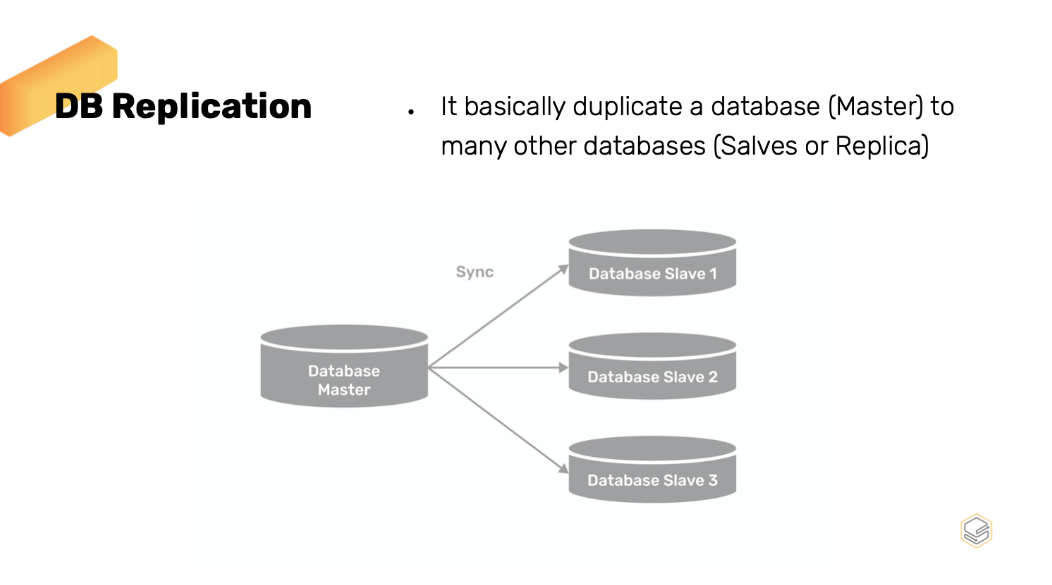
**5.Single Point of Failure (SPOF)** = ในระบบจุดใดจุดนึงตาย ระบบจะล่มไปเลย

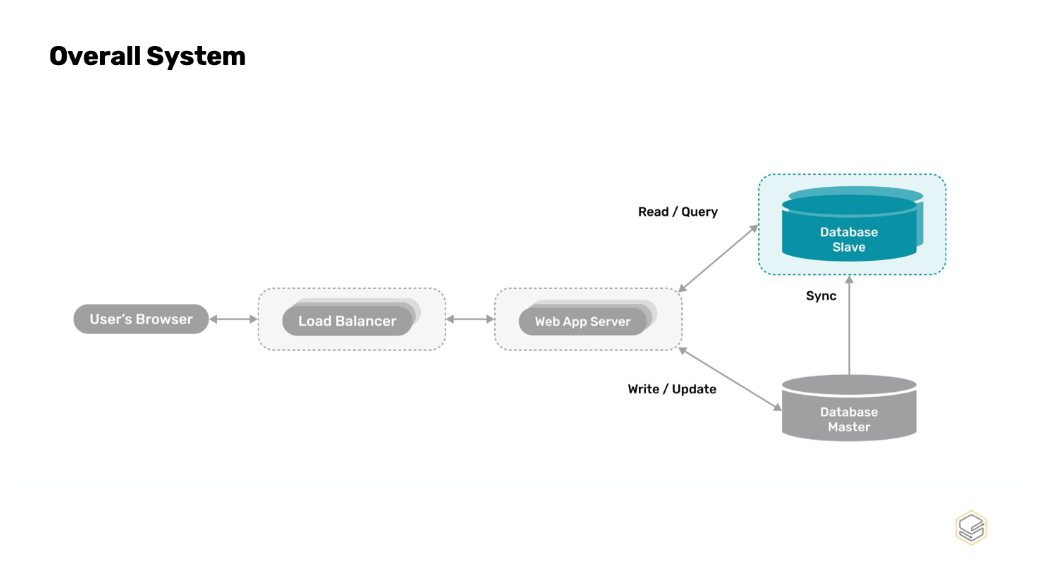
เป็นศัตรูกับ **High Availability** ต้องมี **Load Balancer** มากกว่า 1 ตัว



วิธีทำให้ **Database** ไม่เป็น **Single Point of Failure (SPOF)**

**- DB Replication** ทำการทำซ้ำ Duplicate ออกไป หลายๆตัว เรียกว่า (Salves or Replica) หากตัว Database master ตายไป มันก็จะไปหาตัวที่ Duplicate (Database Slave) ตัวอื่นให้เป็น master แทน





A picture containing text, screenshot, diagram, design

Description automatically generated

A picture containing text, screenshot, diagram, line

Description automatically generated

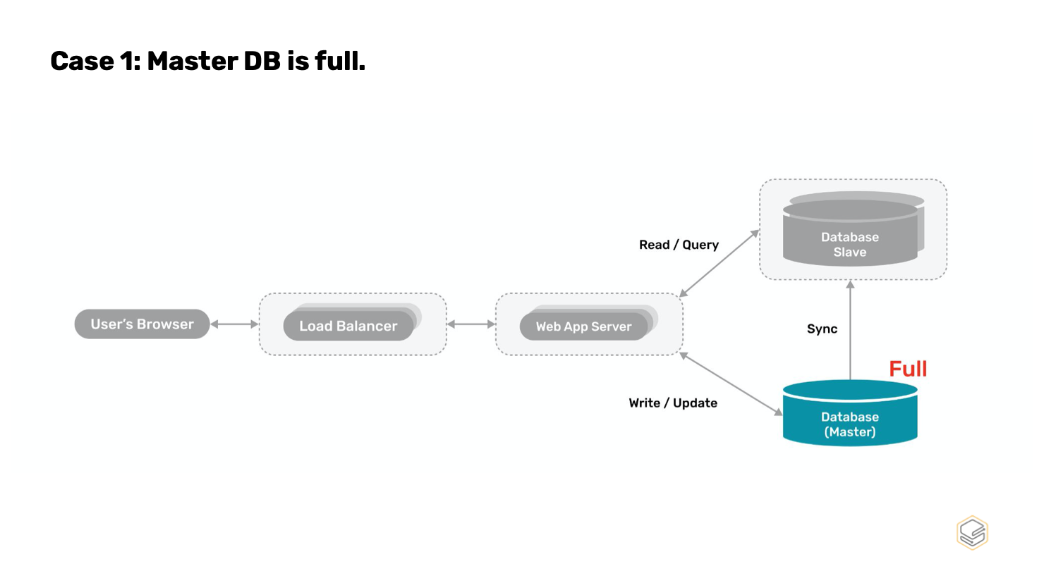
A picture containing text, screenshot, diagram, design

Description automatically generated

**6.DB Sharding**

- Master DB is full

ใช้ Sharding Key ในการแบ่งข้อมูล เช่น Date, Type, สามารถเลือกข้อมูลหรือแบ่งข้อมูลตามตารางต่างๆได้



A picture containing text, screenshot, font, design

Description automatically generated

**7.Storing Files**

- **Storing in a Cloud Object Storage** เอาข้อมูลที่เป็นไฟล์ไปจัดเก็บบน Data Storage

เช่น amazon S3, Google Cloud Storage เก็บข้อมูลฝั่ง front end

ส่วน back end ไว้เก็บข้อมูลใน Database

- ข้อเสีย ต้องดูแลจัดการ

A diagram of a cloud object storage

Description automatically generated

**8.DNS**

คือ **domain name system** ใส่ชื่อ Website จากนั้น DNS จะไปหา IP มาให้ จากนั้นจะส่งค่า Reques Data เข้าไป Acesst ตัว Website

A picture containing text, screenshot, diagram, font

Description automatically generated

**9.CDN**

คือ **Content Delivery Network** คือตัว Server ที่กระจายอยู่ทั่วโลก คือจะเชื่อมต่อกับ Storing Files ฝั่ง front end

- **ข้อดี** ตัว Server ที่กระจายอยู่ทั่วโลก คือจะเชื่อมต่อกับ Storing Files ฝั่ง front end

- **ข้อเสีย** ต้องอัพเดตไฟล์อยู่เสมอ หากมีการแก้ไขที่ตัว Server เช่น จะให้อัพเดตทุกๆกี่วัน กี่เดือน

แต่ห้ามสั้นเกินไป หากสั้นเกินไป ก็ไม่ต่างจากไม่มี CDN เพราะต้องอัพโหลดตลอดเวลา ยาวเกินก็ไม่ดีหาดเราเพิ่มข้อมูลอะไรไป ข้อมูลฝั่ง CDN ก็จะมองไม่เห็น ต้องให้เหมาะสม

A picture containing map, text

Description automatically generated

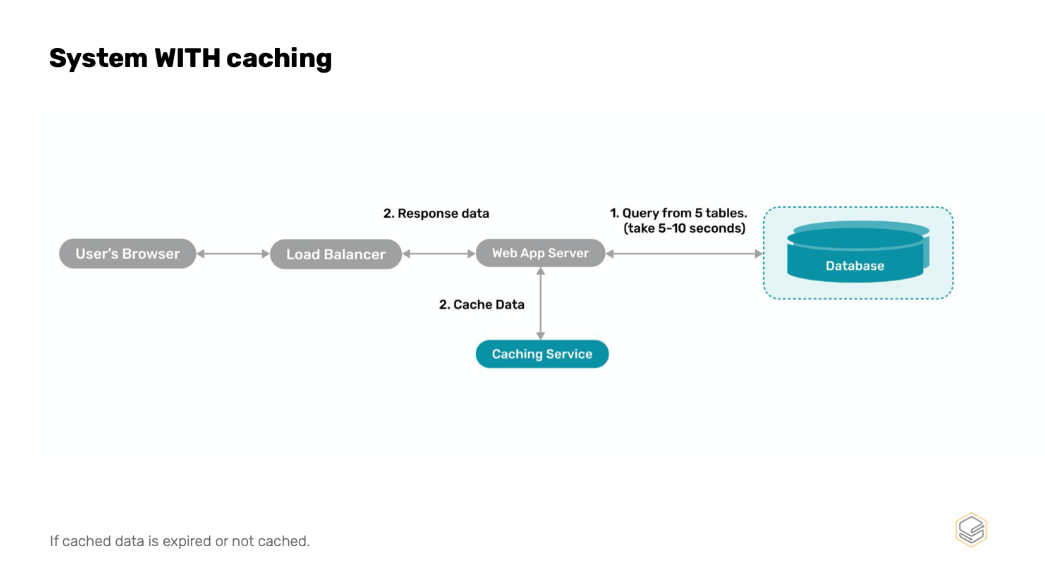
**10.Cache**

ข้อมูลจะถูกดึงมาจาก **Cache** เช่น Redis, Memcached เลย ไม่ต้องเข้าไปที่ Database

หากมีการค้นหาข้อมูลเหมือนกันกับคนก่อนหน้า ให้ไปเอาจาก Cache มาเลย ไม่ต้องเสียเวลาคำนวน Database

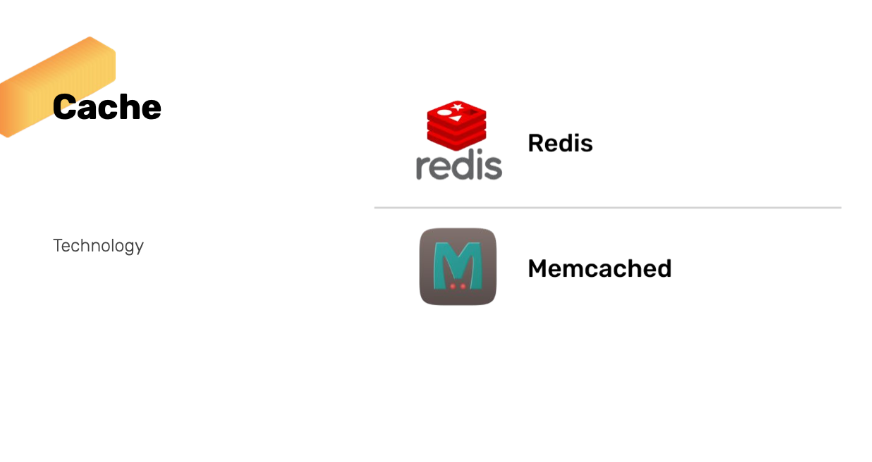
- **ข้อเสีย** ต้องอัพเดตไฟล์อยู่เสมอ หากมีการแก้ไขที่ตัว Server เช่น จะให้อัพเดตทุกๆกี่วัน กี่เดือน

แต่ห้ามสั้นเกินไป หากสั้นเกินไป ก็ไม่ต่างจากไม่มี Cache เพราะต้องอัพโหลดตลอดเวลา ยาวเกินก็ไม่ดีหาดเราเพิ่มข้อมูลอะไรไป ข้อมูลฝั่ง Cache ก็จะมองไม่เห็น ต้องให้เหมาะสม



A picture containing text, screenshot, diagram, line

Description automatically generated



**11.Queue**

- หากต้องใช้เวลานาน ให้เอาข้อมูลไปเข้า Queue ไว้ จะมี **Server Job Worker** หลายๆตัวให้ใช้วิธีแบบ **Horizontal Scaling** ไปดึงงานมาทำ เมื่อทำเสร็จให้ไปอัพเดต Database master

และแจ้งเตือนให้ User รู้ด้วย

A diagram of a work flow

Description automatically generated with low confidence

สำหรับวิธีการในการแจ้งผลการทำงานหลังจากที่ **Job Worker** ทำงานเสร็จ อาจแบ่งได้เป็น 2 แบบ

คือ Pull และ Push

Pull คือ การออกแบบให้ client มาคอยถามว่างานเสร็จหรือยัง เช่น การเขียน client-side

ให้เว็บ reload ตัวเองเป็นช่วง ๆ หรือ ใช้ Ajax ยิง request ไปถาม server เป็นช่วงก็ได้

Push คือ การออกแบบให้ server แจ้งผลการทำงานไปให้ client ทราบเอง เช่น การส่งอีเมล หรือ SMS หรือ ใช้วิธีที่ซับซ้อนแบบ WebSocket ก็ได้

**12.Data Center**

ทำการ Deploy หลายๆที่ ที่ไม่ใช่ที่เดียวกัน

A diagram of a building

Description automatically generated with low confidence

A diagram of data center

Description automatically generated with low confidence

A picture containing text, screenshot, diagram, design

Description automatically generated