DAY 1

הפתרון האופטימלי הוא ליצור כמה מיקרוסרוויסים בשביל שרתי הbackend ולהשתמש בLoad Balancer (LB) על מנת ליצור שוויון בחלוקת העומס. יצירת כמה שרתי backend תעזור בהפחתת עומס המשתמשים על שרת אחד, ובכך תעזור בהימנעות מקריסת האתר עקב עומס. כאשר יש כמה שרתי backend, גם אם אחד מהשרתים קורס, הוא לא יקריס את כל האתר, מכיוון שה-LB יחוון את המשתמשים לשרת backend אחר פעיל.

מכיוון שאין הרבה זמן, וגם העומס על האתר שאבנה לא אמור להיות כבד, לא אעשה זאת באתר, ויהיה רק שרת backend אחד.

Horizontal Scaling – יצירת עוד nodes כדי לחלק את העומס (למשל, יצירת עוד שרתי backend)

Vertical Scaling – הוספת משאבים לשרת אחד. למשל, להגדיל את כוח ההתמודדות של השרת עם עומס. יותר קל לעשות אבל פחות חסכוני. בנוסף יש סיכון של Single Point Of Failure כי יש רק שרת אחד עם כוח גדול, ואם הוא נופל כל האתר קורס.

אני אבחר להשתמש בnode.js על מנת לבנות את שרת הbackend שלי כי יש לו יתרונות שיעזרו לי לעשות את האתר יותר סקלבילי כמו: התמודדות א-סינכרונית עם בקשות כך שניתן לקבל הרבה בקשות בו-זמנית ללא בעיה.

כדי להימנע מSPOF גם בשכבת הDB, הפתרון האידיאלי יהיה להשתמש ב-Data Replication. יהיה master DB ועותקים של המאסטר. במאסטר יהיה ניתן רק לכתוב, ואת העותקים שלו יהיה ניתן רק לקרוא. יתרונות השיטה: המודל מאפשר ליותר שאילתות להתבצע בו-זמנית ואם הDB נהרס פיזית, המידע נשמר, מכיוון שיש לו עותקים במקומות אחרים.

על מנת לשפר את זמן התגובה והטעינה:

האידיאל – להשתמש בcaching וCDN

Caching – מקום אחסון זמני לתוצאות של תגובות יקרות או של מידע שניגשים אליו בתדירות גבוהה, כדי שלבקשות בשבילן ייקח הרבה פחות זמן להחזיר תשובה ולהגיב. כאשר אנו מקבלים בקשה, שרת האינטרנט בודק אם המידע נמצא בcache, במידה וכן, הוא מחזיר את המידע משם, ובמידה ולא, הוא שולח בקשה לDB, שומר את התשובה בcache, ומחזיר את המידע ללקוח.

CDN – רשת שרתים מרוחקים שמשומשים על מנת לאחסן מידע סטטי, כמו תמונות, סרטונים, קבצי CSS, קבצי JS ועוד. כאשר הלקוח מבקש מידע סטטי, השרת האינטרנטי בודק אם הוא קיים בCDN, במידה וכן, שולח ללקוח, ובמידה ולא, מבקש מהשרת, מאחסן את התשובה בCDN, ומחזיר ללקוח.