ISA

Sa porastom broja korisnika javila bi se i potreba za promenom strukture srvera i načinom njihovog opsluživanja. Prva promena koja se obično nameće jeste poboljšanje trenutne konfiguracije kupovinom boljih i jačih komponenata (vertikalno skaliranje). Sa daljim porastom korisnika kada trenutno postojeće komponente ne mogu da podrže promet prelazi se na horizontalno skaliranje. Nabavljaju se novi serveri,a sa nabavljenim serverima javlja se potreba rasporedjivanja resursa.

CLUSTER

Računarski cluster predstavlja skup povezanih računara-servera koji rade zajedno i čine jedan sistem. Svaki čvor(povezani računar) ima isti zadatak. Komponente cluster-a su najčešće povezane LAN mrežom i sastoje se od istog hardware-a. Uloga im je da poboljšaju performanse i dostupnost sistema. Zadaci koje sistem obavlja se dele izmedju cluster-a i ideja je da sistem ne prestane da radi ako otkaže neki njegov deo.
Alat koji bi nam koristio za ovaj deo mogao bi biti Kubernetes koji bi služio za upravljanje klasterima i njihovo skaliranje.

LOAD BALANCING

Load balancer je softver koji osluškuje zahteve od korisnika koji žele da koriste usluge neke web aplikacije. Njegova uloga je da prihvati zahtev, odredi kojem backend serveru može da prosledi zahtev i da ga prosledi tom serveru. Izbor servera se najčešće vrši na osnovu trenutnog opterećenja tog servera. Takođe, sprečava korisnike da direktno kontaktiraju backend servere i sakriva njihovu internu strukturu.

Neki balanseri pružaju mehanizam reagovanja na događaje kada su svi backend serveri zauzeti/nedostupni. U tom slučaju se korisnik prosleđuje na rezervni load balancer ili mu se prikazuje poruka o trenutnom stanju.

Load balancer mora da čuva podatke o sesiji ako je aplikacija bazirana na upotrebi sesija, da korisnik ne bi primetio cluster strukturu web aplikacije. Svaki put jednog usera treba da šalje na isti backend server, da bi konzistentnost podataka bila očuvana. To se rešava kada kod prvog zahteva korisnika balancer napravi cookie kom je serveru prosledjen korisnik i dalje se nastavlja komunikacija uz upotrebu cookie-ja.

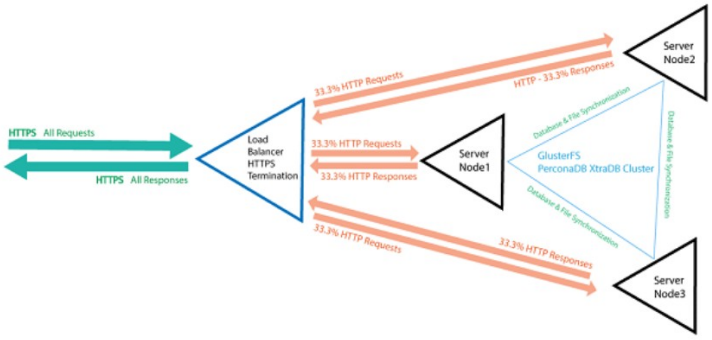
Načini balansiranja:
- Round-robin DNS: upotrebom više dns podešavanja za isti hosting domen.
- Reverse-proxy: koristi se frontend server ili namenski hardware uredjaj.

Primer: NGINX i round robin pristup:

```
http {
    upstream myapp1 {
    ip_hash;                                    - čuva sesiju
    server srv1.example.com  weight=3;          - težine za servere
    server srv2.example.com;
    server srv3.example.com;
    }

    server {
        listen 80;

        location / {
            proxy_pass http://myapp1;
        }
    }
}
```



ORKESTRACIJA

Orkestracija ja proces integracije 2 ili više aplikacija/servisa u jedan proces ili sinhronizovanje podataka u realnom vremenu. Glavni ciljevi orkestracije su:
1) razdvajanje aplikacija zaduženih za različite delove sistema
2) povezivanej različitih aplikacija, medjusobna razmena poruka između njih
3) centralizovano upravljanje i praćenje integracija

REDUNDANCY ENGENEERING
U inzenjerstvu redundantnost predstavla duplikaciju komponenti koje su vazne za funkcionisanje sistema, načešće u obliku fail-safe pristupa.

Fail-safe pristup podrazumeva to da je sistem u stanju da prepozna i otkloni određene tipove otkaza bez da se funkcionisanje sistema naruši. Ovim pristupom nije moguće u potpunosti otkloniti otkaze, ali je moguće reagovati na otkaze koji se često dešavaju ili predstavljaju kritične delove sistema.

Primer u kojem bismo mogli iskoristiti redundancy engeneering pristup je tako što bismo pokretali aplikaciju na više servera. U slučaju otkaza apliakcije ili samog servera, ostali serveri bi nastavili normalno da rade.

Primer u kojem mi ovo radimo je restartovanje docker cantainera u kojem je pokrenuta aplikacija u slučaju bilo kakvog otkaza, kako u aplikaciji, tako i na samom serveru na kojem je pokrenut kontejner. Ovo znači da će aplikacija sama pokušati da se pokrene čim server bude u mogućnosti. Tim pristupom sprečavamo da aplikacija ostane nepokrenuta sve dok je neko od nas ručno ne pokrene.

RECOVERY AND BACKUP
Ovaj pristup podrazumeva lako vraćanje aplikacije/servisa u radno stanje u slučaju da dođe do otkaza. Potrebno je što pre uvideti otkaz i otkloniti ga. Da bi se ovo uradilo potrebno je imati backup sistema.

Backup predstavlja kopiju sistema ili stanja sistema. Ovo se odnosi kako na podatke, tako i na hardver koji je potreban za rad sistema.

Primer u kojem bismo mi mogli iskoristiti backup pristup je backup stanja baze u određenim intervalima. Backup je potrebno čuvati na lokaciji koja je fizicki i logicki odvojena od servera na kojem je baza pokrenuta.

CACHE
Mehanizam keširanja se koristi da bi se stvari koje se često koriste čuvale na mestu sa kojeg se lako dobavljaju.
Na front-end strani bismo keširanje mogli koristiti za statičke fajlove, kao što su css i js fajlovi.
Na back-end strani bismo keširanje mogli koristiti za čuvanje proračuna koje neke metode rade.
Ono što bi nama najviše koristilo je proxy keširanje koje čuva podatke izmedju korisnika i load balancera.

BAZE PODATAKA
Baze same po sebi su verovato najsporiji deo web arhitekture. Da bi se ovo ubrzalo koristio bi se SQL query cache (pakuje SQL upite, vraca kesirane podatke ako baza kaze da oni nisu prljavi (nisu menjani). Upite treba modifikovati tako da vraćaju manji broj rezulta od interesa, samo onih koji bi se u jednom trenutku prikazivali. Za ovakav pristup se koristi koncept paginacije. Za česte upite potrebno je kreirati indekse koji imaju ulogu da ubrzaju pretragu u bazi.

Drugi način ubrzavanja baze podataka je prelazak sa pesimističnog na optimističan režim zaključavanja. To podrazumeva da se tabele ne zaključavaju pri pristupanju, već se čuva brojač izmena na osnovu kojeg se detektuje da li je bilo promena za vreme obavljanja nekog upita. Ovaj mehanizam je pogodan kada kolizije nisu previše česte.

Treći način bi bio prelazak sa Sql na NoSql baze koje bolje rade sa većom kolicinom podataka i imaju sve potrebne informacije na jednom mestu bez potreba za join-ovima i merge-ovanjem tabela da bi se dobili podaci.
Nad njima je lakse vrši horizontalno skaliranje u odnosu na Sql baze, mada su problematične sa stanovništva striktne strukture podataka. Koriste se kada se ne vrši puno upisa u bazu, već je akcenat na brzom i optimalnom pribavljanju podataka.

Ako ni jedan od ovih načina nije dovoljan, potrebno je uposliti više servera da opslužuju bazu. Ideja bi se realiyovala tako što bi se napravila jedna ili više master baza koja bi služila da opslužuje write zahteve i veci broj drugih replikacionih baza koje bi sluzile za read operacije. Master baze treba mdjusobno povezati tako da razmenjuju poruke o izmenama, dok bi replikacione baze bile vezane za po jednu od master baza koje bi njima slale promene ako dodje do njih. Takodje, pojavom više baza ponovo se javlja potreba za Load Balancer-om. Rešenje sa vise baza nudi i dobru mogućnost očuvanja podataka u slučaju kvara na nekoj od baza.