DIS - Skalerbarhed 2 - Giant-scale services, Vertical vs horisontal, GFS

Vertical vs Horisontal

Vertikal skalering

At skalere noget er et spil om procenter. Jo større et system, jo mindre chance for nedbrud.
Et spil om procenter hvor vi generelt set at jo flere servere vi har jo flere servere vi har, jo mindre chance er der for at noget går galt, når noget går ned.
Hvis 2/3 af et distribueret system skal være enige – så skal de kunne stemme.

når man opgraderer en computer ved at give den en større processor, mere hukommelse og så videre

Man kan ikke blive ved med at skalere vertikalt. Kun hvis der er fysisk plads. Og så findes der også arkitekturer der gør, at computere fx ikke kan have 150gb RAM fx. Det betyder **at man SKAL på et tidspunkt skalere horisontalt**

Horisontal skalering

når man opgraderer en service ved at sætte to eller flere servere sammen i et cluster.

Man kan altid skalere horisontalt

Brewer's / CAP theorem

Consistency
Alle data er ens

Availability
Alle data er tilgængelige

Partition tolerance
Alle data kan overleve nedbrud

Man kan ikke få alt, man skal vælge.

Se billede

High Availability

High availability betyder at et distribueret system er bygget til at fejle.

Det er klogt at tage højde for at verden ikke er perfekt. Noget er stærkt tilgængeligt - omvendt - det er ikke altid tilgængeligt

Giant-scale services

Google & facebook

Et godt eksempel. De stiller en masse services til rådighed, men laver også alt muligt andet - det kan ikke klares på én mainframe

Når vi taler om giant-scale services, så består de ofte af klienter, et netværk, load balancers, servere og fysiske datalag.
Fysiske datalag - det er bare harddiske - står et sted man kan gemme ting

Hvordan undgår vi, at de ikke går ned?

Skalering går blandt andet ud på at fordele forespørgsler, så der ikke kun er en enkelt server, der laver det hele.

Round-robin-algoritmen blev for alvor populær med DNS-systemet, som gjorde sit indtog i 1995.
algoritme - en måde at gennemløbe data på.
Det er en algoritme til at lave en slags cirkel – en cirkulær fordeling af

forespørgsler.

Google File System

Det er et problem når der er flere, der gerne vil gemme noget data på samme tid det samme sted.

Her bruger man **låsemekanismer** i mindre systemer, men i større…

GFS provides an atomic append operation called record append. In a traditional write, the client specifies the offset at which data is to be written.

In a record append, however, the client specifies only the data. GFS appends it to the file at least once atomically (i.e., as one continuous sequence of bytes) at an offset of GFS's choosing

This is similar to writing to a file opened in O_APPEND mode in Unix without the **race conditions** when multiple writers do so concurrently.
#

Vi har en stor mængde data (Append) – hvis de ved noget – det er billigere for google at skrive den samme data igen (hvis der er noget nyt), i stedet for at kigge ned i den samme data for at se om de ved noget data eller ej.

Google File System bygger på en lang række koncepter, der kombineres med hinanden for at skabe en løsning. Fx *master* og *chunk operations*.
Det er bygget på noSQL

Fast Recovery

Bemærk at de nævner fast recovery som noget, der sker på baggrund af data, der er gemt et sted.

Det svarer fuldstændig til at I mister hukommelsen og genfinder den ved at læse jeres dagbog.

Race Conditions

Race conditions opstår når der går kuk i hvilke værdier, der er tilgængelige hvornår.

Oplever når vi koder – når vi sidder og koder i et programmeringssprog som er asynkront –

Eksekverer ved RPC fx og går videre i koden. Her forekommer der race conditions.