

UiT Norges arktiske universitet

Fakultet for naturvitenskap og teknologi

Sanntids samskriving av dokumenter

Operasjonell transformasjon og CRDT i en nettbasert editor

Fornavn Etternavn

Masteroppgave i informatikk, INF-3990, mai 2027

Innholdsfortegnelse

Innledning.....	1
1 Samarbeidsredigering av dokumenter.....	2
1.1 Krav til systemet.....	2
2 Tilnærminger til synkronisering.....	4
2.1 Operasjonell transformasjon.....	4
2.1.1 Transformasjonsfunksjonen.....	4
2.2 Konfliktfrie replikerte datatyper.....	4
3 Utforming og implementering.....	6
3.1 Lagring av historikken.....	6
4 Evaluering.....	7
4.1 Diskusjon av resultatene.....	8
Konklusjon.....	9
Referanser.....	10
Vedlegg A: Spørreskjema.....	11

Tabelliste

Tabell 1 – Gjennomsnittlig latens og størrelse på lagrede data etter antall brukere.....	7
--	---

Figurliste

Figur 1 – Typisk forløp ved felles skriving av et dokument.....	2
Figur 2 – Antall endringer etter ukedag og klokkeslett i de undersøkte øktene.....	8

Forord

Her kan du takke veilederen din og andre som har hjulpet deg med oppgaven.

Innledning

I dag blir tekstdokumenter bare unntaksvis skrevet av én enkelt forfatter. Studenter skriver semesteroppgaver i par, utviklere vedlikeholder teknisk dokumentasjon sammen, og forskergrupper ferdigstiller artikler i siste liten fra flere steder samtidig. For at slik skriving skal fungere, må editoren sørge for at alle deltakerne ser den samme teksten, og at ingen endring går tapt.

Målet med denne oppgaven er å sammenligne de to viktigste tilnærmingene til synkronisering av dokumenter ved samarbeidsredigering i sanntid, operasjonell transformasjon og konfliktfrie replikerte datatyper, og å prøve dem ut på en prototyp av en nettbasert editor. Det første kapittelet fastsetter kravene til et slikt system, det andre beskriver begge tilnærmingene, det tredje utformingen av prototypen og det fjerde måleresultatene.

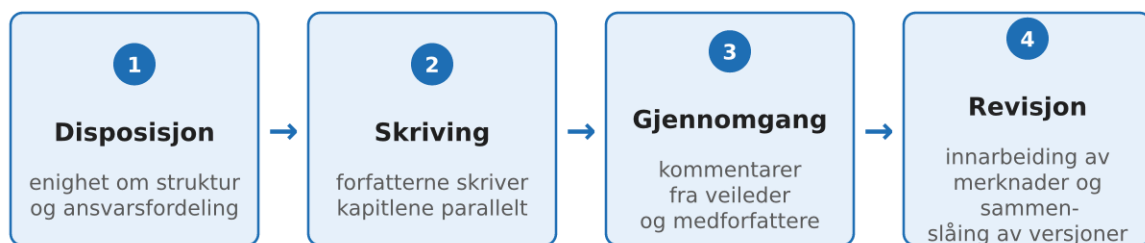
1 Samarbeidsredigering av dokumenter

Med samarbeidsredigering mener vi en situasjon der flere brukere redigerer det samme dokumentet og endringene deres overføres til hverandre uten manuell sammenslåing. Ut fra hvor raskt endringene spres, skiller vi mellom tre grunnleggende arbeidsformer:

- **asynkron redigering**, der forfatterne utveksler hele versjoner av dokumentet og slår dem sammen først i etterkant,
- **redigering med låser**, der bare én forfatter om gangen kan redigere en bestemt del av dokumentet,
- **redigering i sanntid**, der hver endring umiddelbart blir synlig for de andre brukerne.

Denne oppgaven tar utelukkende for seg den siste arbeidsformen. Den er den mest bekvemme for brukerne, men stiller de største kravene til utformingen av systemet, fordi endringer oppstår samtidig mange steder og ankommer i ulik rekkefølge.

Sanntidsredigering endrer dessuten ikke bare teknikken, men også forløpet av hele samarbeidet. Den typiske arbeidsgangen ved felles skrijving, slik de intervjuede brukerne beskrev den, er vist i Figur 1.



trinn 2 til 4 gjentas så lenge gjennomgangen gir vesentlige merknader

Figur 1 – Typisk forløp ved felles skrijving av et dokument

1.1 Krav til systemet

Ut fra intervjuer med brukere av eksisterende editorer og fra faglitteraturen har vi utledet fire krav som et system for samarbeidsredigering må oppfylle:

1. konvergens, det vil si at når alle endringer er levert, har alle kopier av dokumentet det samme innholdet,

2. bevaring av intensjon, det vil si at resultatet av en endring svarer til det forfatteren så i øyeblikket endringen ble gjort,
3. responstid, det vil si at en lokal endring trer i kraft umiddelbart uten å vente på serveren,
4. robusthet, det vil si at et kortvarig brudd i forbindelsen ikke fører til tap av påbegynt tekst.

De to første kravene gjelder korrekthet, de to siste brukbarhet. I praksis viser det seg at nettopp kombinasjonen av umiddelbar respons og garantert konvergens er den vanskeligste delen av hele utforming.

2 Tilnærminger til synkronisering

Systemer for samarbeidsredigering i sanntid utfører først hver endring lokalt og sender den deretter videre til de andre deltakerne. De skiller seg først og fremst i hvordan de håndterer situasjonen der to endringer oppstår samtidig og hver deltaker mottar dem i en annen rekkefølge.

2.1 Operasjonell transformasjon

Operasjonell transformasjon bygger på arbeidet til Ellis og Gibbs (Ellis & Gibbs, 1989), som var de første til å beskrive hvordan en samtidig operasjon kan justeres slik at den også kan utføres etter en annen operasjon som forfatteren ikke kjente til. Hver endring registreres som en innsettings- eller sletteoperasjon på en bestemt posisjon i teksten.

Dersom for eksempel én bruker setter inn et ord i begynnelsen av et avsnitt og en annen samtidig sletter et ord på slutten av det, må posisjonen for slettingen forskyves med lengden av det innsatte ordet. Denne forskyvningen beregnes av transformasjonsfunksjonen.

2.1.1 Transformasjonsfunksjonen

Transformasjonsfunksjonen tar imot et par samtidige operasjoner og returnerer justerte versjoner av dem. For at alle kopier av dokumentet skal konvergere, må funksjonen oppfylle de såkalte transformasjonsegenskapene, som det er vanskelig å bevise at holder for generelle operasjoner. De fleste systemer i bruk baserer seg derfor på en sentral server som fastsetter én enkelt rekkefølge av operasjonene.

2.2 Konfliktfrie replikerte datatyper

Den andre tilnærmingen er konfliktfrie replikerte datatyper, samlet betegnet med forkortelsen CRDT (Shapiro et al., 2011). I stedet for å justere operasjonene i etterkant velger de en representasjon av dokumentet der samtidige operasjoner kommuterer og rekkefølgen deres ikke påvirker resultatet.

«En replikert datatype er konfliktfri dersom to vilkårlige replikaer som har mottatt den samme mengden oppdateringer, har samme tilstand uavhengig av rekkefølgen oppdateringene ble levert i.»

Hvert tegn i dokumentet får en entydig identifikator som forblir den samme så lenge tegnet eksisterer. Innsettingsposisjonen angis da ikke med et tall, men med en henvisning til

identifikatoren for nabotegnet, slik at samtidige endringer andre steder i teksten ikke kan forskyve den.

3 Utforming og implementering

Prototypen av editoren består av en nettklient, en synkroniseringsserver og en database der serveren fortløpende lagrer tilstanden til dokumentene. Både klienten og serveren er skrevet slik at synkroniseringslaget kan byttes ut og de to beskrevne tilnærmingene kan sammenlignes med det samme brukergrensesnittet.

Klienten holder en lokal replika av dokumentet og omformer hver endring brukeren gjør, til en operasjon på den replikerte datatypen. Operasjonene sendes til serveren i bunker, slik at nettverket ikke overbelastes ved rask skriving. En forenklet behandling av en innkommende bunke på serveren er vist i følgende kodeutdrag:

```
def behandle_bunke(dokument, bunke):
    for operasjon in bunke.operasjoner:
        if operasjon.id in dokument.anvendte:
            continue
        dokument.anvend(operasjon)
        dokument.anvendte.add(operasjon.id)
    send_til_andre(dokument.id, bunke)
```

Siden operasjonene kommuterer, trenger ikke serveren å ordne eller transformere bunkene. Det er tilstrekkelig at den sørger for at ingen operasjon utføres to ganger, og at hver bunke når frem til alle tilkoblede klienter.

3.1 Lagring av historikken

Serveren lagrer ikke bare den endelige teksten, men hele operasjonsloggen. Fra loggen kan en hvilken som helst tidligere versjon av dokumentet gjenopprettes, og replikatilstanden til en klient som har vært frakoblet over lengre tid, kan bygges opp på nytt. Denne tilnærmingen svarer til en arkitektur basert på en uforanderlig hendelseslogg, slik Kleppmann beskriver den (Kleppmann, 2017). For at loggen ikke skal vokse ubegrenset, komprimerer serveren den med jevne mellomrom til øyeblikksbilder.

4 Evaluering

Vi sammenlignet prototypen med en referanseimplementering av operasjonell transformasjon på et sett opptak av reelle redigeringsøkter. For hver økt målte vi latensen, det vil si tiden fra en endring hos forfatteren til den vises hos de andre deltakerne, og beregnet gjennomsnittlig latens for økten etter (1):

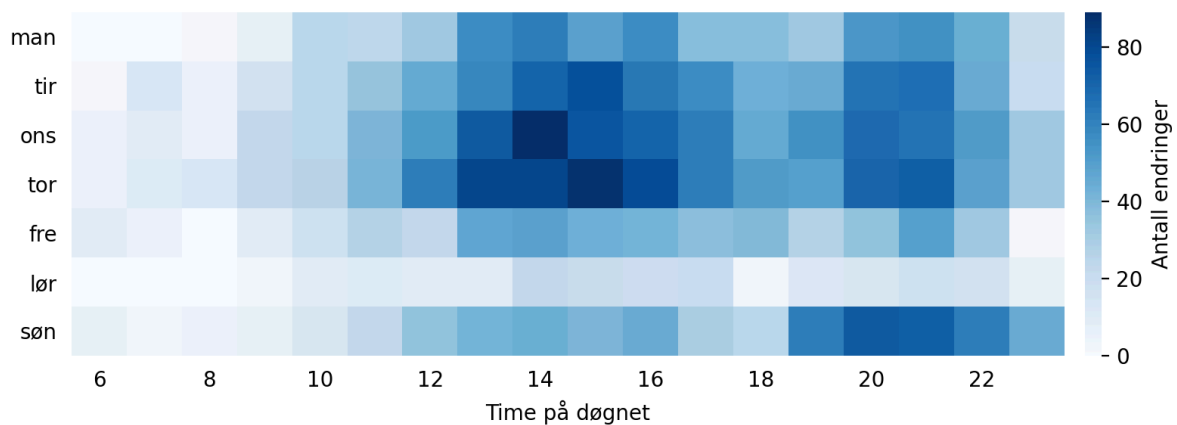
$$\bar{t} = \frac{1}{n} \sum_{i=1}^n (t_i^{\text{levering}} - t_i^{\text{endring}}) \quad (1)$$

der n er antall operasjoner i økten. De målte verdiene for ulike antall samtidig arbeidende brukere er oppsummert i Tabell 1.

Tabell 1 – Gjennomsnittlig latens og størrelse på lagrede data etter antall brukere

Brukere	Latens OT (ms)	Latens CRDT (ms)	Data (kB)
2	38	35	12
5	52	41	19
10	87	48	31
20	164	63	58

Etter hvert som antallet brukere øker, vokser latensen for operasjonell transformasjon betydelig raskere, fordi serveren må transformere hver operasjon mot alle samtidige operasjoner. Prisen for den lavere latensen med CRDT er et større volum av lagrede data, ettersom hvert tegn bærer sin egen identifikator. Fordelingen av endringer over tid, som belastningstesten bygger på, er vist i Figur 2.



Figur 2 – Antall endringer etter ukedag og klokkeslett i de undersøkte øktene

4.1 Diskusjon av resultatene

Resultatene bekrefter at begge tilnærmingene er likeverdige for mindre team, og at det heller er enkel implementering som avgjør. Ved et større antall samtidige brukere har CRDT en klar fordel både når det gjelder latens og robusthet mot brudd i forbindelsen. Supplerende intervjuer med brukere, gjennomført etter prinsippene for kvalitativ forskning (Hendl, 2005), viste at brukerne ikke merker en forskjell i latens på under hundre millisekunder, mens de anser tap av påbegynt tekst etter et brudd i forbindelsen som et alvorlig problem.

Konklusjon

Oppgaven har sammenlignet operasjonell transformasjon og konfliktfrie replikerte datatyper som to tilnærminger til synkronisering av dokumenter ved samarbeidsredigering i sanntid.

Med en prototyp av en nettbasert editor har vi vist at CRDT muliggjør en enklere server uten sentral ordning av operasjonene og gir lavere latens ved et større antall brukere.

Den største ulempen med CRDT er fortsatt det større volumet av lagrede data. Neste steg blir derfor å undersøke hvordan den jevnlige komprimeringen av loggen til øyeblikksbilder påvirker datamengden, og hvordan prototypen oppfører seg ved langvarig drift med titusenvis av dokumenter.

Referanser

Ellis, C. A., & Gibbs, S. J. (1989). Concurrency control in groupware systems. *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, 399–407.

<https://doi.org/10.1145/67544.66963>

Hendl, J. (2005). *Kvalitativní výzkum: základní metody a aplikace*. Portál.

Kleppmann, M. (2017). *Designing data-intensive applications*. O'Reilly Media.

Shapiro, M., Preguiça, N., Baquero, C., & Zawirski, M. (2011). Conflict-free replicated data types. *Stabilization, Safety, And Security of Distributed Systems*, 6976, 386–400. https://doi.org/10.1007/978-3-642-24550-3_29

Vedlegg A: Spørreskjema

Et vedlegg inneholder materiale som støtter teksten, men som ikke er nødvendig for å følge den, for eksempel et spørreskjema, måledata eller en lengre kodeliste.