



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

Dipartimento di Informatica - Scienza e Ingegneria
Corso di Laurea Magistrale in Informatica

Editing collaborativo di documenti in tempo reale

Tesi di Laurea in Sistemi Distribuiti

Relatore:
Chiar.mo Prof.
Nome Cognome

Presentata da:
Mario Rossi

Correlatore:
Chiar.mo Prof.
Nome Cognome

Sessione marzo 2026
Anno Accademico 2025/2026

A chi mi ha sostenuto

Indice

Introduzione	1
1 Editing collaborativo di documenti	3
1.1 Requisiti del sistema	4
2 Approcci alla sincronizzazione	5
2.1 Trasformazione operativa	5
2.1.1 Funzione di trasformazione	6
2.2 Tipi di dati replicati privi di conflitti	6
3 Progettazione e implementazione	7
3.1 Archiviazione della cronologia	8
4 Valutazione	9
4.1 Discussione dei risultati	10
Conclusioni	11
A Materiale supplementare	13
Bibliografia	15

Introduzione

Oggi i documenti di testo sono solo raramente opera di un unico autore. Gli studenti scrivono le tesine in coppia, gli sviluppatori mantengono insieme la documentazione tecnica e i gruppi di ricerca completano gli articoli all'ultimo momento da più sedi contemporaneamente. Perché una scrittura di questo tipo funzioni, l'editor deve garantire che tutti i partecipanti vedano lo stesso testo e che nessuna modifica vada persa.

Obiettivo di questo lavoro è confrontare i due principali approcci alla sincronizzazione dei documenti nell'editing collaborativo in tempo reale, la trasformazione operazionale e i tipi di dati replicati privi di conflitti, e verificarli su un prototipo di editor web. Il primo capitolo definisce i requisiti di un sistema di questo tipo, il secondo descrive entrambi gli approcci, il terzo la progettazione del prototipo e il quarto i risultati delle misurazioni.

Capitolo 1

Editing collaborativo di documenti

Per editing collaborativo intendiamo la situazione in cui più utenti modificano lo stesso documento e le loro modifiche si propagano reciprocamente senza unione manuale. In base alla rapidità con cui le modifiche si propagano, distinguiamo tre modalità di lavoro fondamentali:

- **editing asincrono**, in cui gli autori si scambiano versioni complete del documento e le uniscono solo in un secondo momento;
- **editing con blocchi**, in cui una determinata parte del documento può essere modificata da un solo autore alla volta;
- **editing in tempo reale**, in cui ogni modifica si riflette immediatamente sugli altri utenti.

Il presente lavoro si occupa esclusivamente dell'ultima modalità. È la più comoda per gli utenti, ma pone le maggiori esigenze alla progettazione del sistema, poiché le modifiche nascono contemporaneamente in molti punti e arrivano in ordine diverso.

L'editing in tempo reale, peraltro, non cambia soltanto la tecnica, ma anche lo svolgimento dell'intera collaborazione. Il processo tipico di scrittura

collaborativa, così come descritto dagli utenti intervistati, è illustrato nella Figura 1.1.



Figura 1.1: Svolgimento tipico della scrittura collaborativa di un documento

1.1 Requisiti del sistema

Dalle interviste con utenti di editor esistenti e dalla letteratura specialistica abbiamo ricavato quattro requisiti che un sistema di editing collaborativo deve soddisfare:

1. convergenza, ossia, una volta consegnate tutte le modifiche, tutte le copie del documento hanno lo stesso contenuto;
2. preservazione dell'intenzione, ossia il risultato di una modifica corrisponde a ciò che l'autore vedeva al momento della modifica;
3. reattività, ossia una modifica locale ha effetto immediatamente, senza attendere il server;
4. robustezza, ossia una breve interruzione della connessione non comporta la perdita del testo in corso di stesura.

I primi due requisiti riguardano la correttezza, gli altri due l'usabilità. Nella pratica si osserva che proprio la combinazione di reattività immediata e convergenza garantita costituisce la parte più difficile dell'intera progettazione.

Capitolo 2

Approcci alla sincronizzazione

I sistemi di editing collaborativo in tempo reale eseguono ogni modifica prima localmente e solo dopo la distribuiscono agli altri partecipanti. Si differenziano soprattutto per il modo in cui gestiscono la situazione in cui due modifiche nascono contemporaneamente e ciascun partecipante le riceve in un ordine diverso.

2.1 Trasformazione operativa

La trasformazione operativa prende le mosse dal lavoro di Ellis e Gibbs [1], che per primi descrissero come adattare un'operazione concorrente in modo che possa essere eseguita anche dopo un'altra operazione di cui il suo autore non era a conoscenza. Ogni modifica viene registrata come un'operazione di inserimento o di cancellazione in una determinata posizione del testo.

Se, per esempio, un utente inserisce una parola all'inizio di un paragrafo e un altro cancella contemporaneamente una parola alla fine dello stesso, la posizione della cancellazione deve essere spostata della lunghezza della parola inserita. Questo spostamento è calcolato dalla funzione di trasformazione.

2.1.1 Funzione di trasformazione

La funzione di trasformazione riceve una coppia di operazioni concorrenti e ne restituisce le versioni adattate. Affinché tutte le copie del documento convergano, la funzione deve soddisfare le cosiddette proprietà di trasformazione, il cui rispetto è difficile da dimostrare per operazioni generiche. La maggior parte dei sistemi in uso si affida pertanto a un server centrale che stabilisce un ordine univoco delle operazioni.

2.2 Tipi di dati replicati privi di conflitti

Il secondo approccio è rappresentato dai tipi di dati replicati privi di conflitti, indicati complessivamente con la sigla CRDT [2]. Anziché adattare le operazioni a posteriori, sceglie una rappresentazione del documento in cui le operazioni concorrenti commutano e il loro ordine non influisce sul risultato.

Un tipo di dati replicato è privo di conflitti se due repliche qualsiasi che hanno ricevuto lo stesso insieme di aggiornamenti hanno lo stesso stato, indipendentemente dall'ordine in cui questi sono stati loro consegnati.

Ogni carattere del documento riceve un identificatore univoco che rimane invariato per tutta la durata dell'esistenza del carattere. La posizione di inserimento non è quindi determinata da un numero, ma da un riferimento all'identificatore del carattere adiacente, cosicché le modifiche concorrenti in altri punti del testo non possono spostarla.

Capitolo 3

Progettazione e implementazione

Il prototipo dell'editor è composto da un client web, da un server di sincronizzazione e da un database in cui il server salva costantemente lo stato dei documenti. Client e server sono scritti in modo da poter sostituire lo strato di sincronizzazione e confrontare entrambi gli approcci descritti sulla stessa interfaccia utente.

Il client mantiene una replica locale del documento e converte ogni modifica dell'utente in un'operazione sul tipo di dati replicato. Le operazioni vengono inviate al server in lotti, per non sovraccaricare la rete durante una digitazione rapida. Il listato seguente mostra l'elaborazione semplificata di un lotto in arrivo sul server:

```
def elabora_lotto(documento, lotto):  
    for operazione in lotto.operazioni:  
        if operazione.id in documento.apply:  
            continue  
        documento.apply(operation)  
        documento.apply.add(operation.id)  
    inoltra_agli_altri(documento.id, lotto)
```

Poiché le operazioni commutano, il server non deve né ordinare né trasformare i lotti. È sufficiente che garantisca che nessuna operazione venga eseguita due volte e che ogni lotto raggiunga tutti i client connessi.

3.1 Archiviazione della cronologia

Il server non salva soltanto il testo risultante, ma l'intero log delle operazioni. Dal log è possibile ripristinare qualsiasi versione precedente del documento e ricostruire lo stato della replica di un client rimasto a lungo disconnesso. Questo approccio corrisponde a un'architettura basata su una registrazione immutabile degli eventi, come la descrive Kleppmann [3]. Affinché il log non cresca indefinitamente, il server lo compatta a intervalli regolari in snapshot.

Capitolo 4

Valutazione

Abbiamo confrontato il prototipo con un'implementazione di riferimento della trasformazione operazionale su un insieme di registrazioni di sessioni di editing reali. Per ogni sessione abbiamo misurato la latenza, ossia il tempo che intercorre tra una modifica presso l'autore e la sua visualizzazione presso gli altri partecipanti, e abbiamo calcolato la latenza media della sessione secondo (4.1):

$$\bar{t} = \frac{1}{n} \sum_{i=1}^n (t_i^{\text{consegna}} - t_i^{\text{modifica}}) \quad (4.1)$$

dove n è il numero di operazioni nella sessione. I valori misurati per diversi numeri di utenti che lavorano contemporaneamente sono riassunti nella Tabella 4.1.

Utenti	Latenza OT (ms)	Latenza CRDT (ms)	Dati (kB)
2	38	35	12
5	52	41	19
10	87	48	31
20	164	63	58

Tabella 4.1: Latenza media e dimensione dei dati salvati in base al numero di utenti

Con l'aumentare del numero di utenti, la latenza della trasformazione operativa cresce molto più rapidamente, poiché il server deve trasformare ogni operazione rispetto a tutte le operazioni concorrenti. Il prezzo della minore latenza dei CRDT è un maggiore volume di dati salvati, dato che ogni carattere porta con sé il proprio identificatore. La distribuzione temporale delle modifiche, su cui si basa il test di carico, è illustrata nella Figura 4.1.

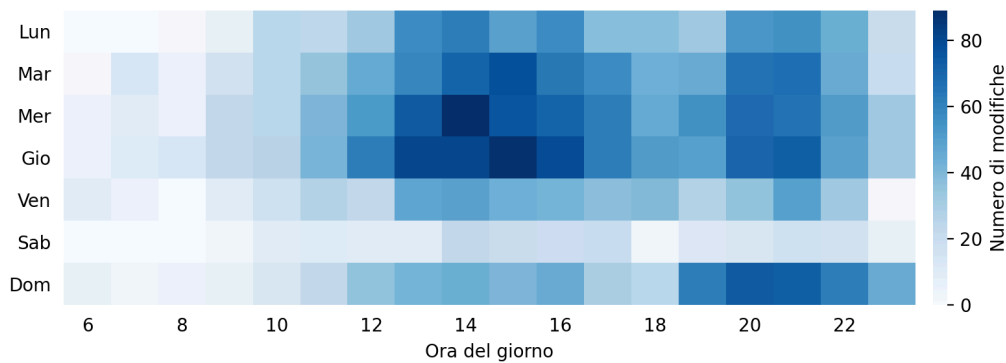


Figura 4.1: Numero di modifiche per giorno della settimana e ora nelle sessioni osservate

4.1 Discussione dei risultati

I risultati confermano che per i team più piccoli i due approcci si equivalgono e che a decidere è piuttosto la semplicità di implementazione. Con un numero maggiore di utenti simultanei, i CRDT offrono un netto vantaggio sia in termini di latenza sia di robustezza rispetto alle interruzioni della connessione. Le interviste integrative con gli utenti, condotte secondo i principi della ricerca qualitativa [4], hanno mostrato che gli utenti non percepiscono una differenza di latenza inferiore a cento millisecondi, mentre considerano un problema fondamentale la perdita del testo in corso di stesura dopo un'interruzione della connessione.

Conclusioni

Il lavoro ha confrontato la trasformazione operazionale e i tipi di dati replicati privi di conflitti come due approcci alla sincronizzazione dei documenti nell'editing collaborativo in tempo reale. Su un prototipo di editor web abbiamo mostrato che i CRDT consentono un server più semplice, privo di ordinamento centrale delle operazioni, e offrono una latenza inferiore con un numero maggiore di utenti.

Il principale svantaggio dei CRDT rimane il maggiore volume di dati salvati. Il passo successivo sarà quindi verificare in che modo la compattazione periodica del log in snapshot influisca sulla dimensione dei dati e come si comporti il prototipo in un funzionamento prolungato con decine di migliaia di documenti.

Appendice A

Materiale supplementare

Un'appendice contiene materiale che sostiene il testo ma non serve per seguirlo, come il questionario di un'indagine, i dati delle misure o un listato più lungo.

Bibliografia

- [1] C. A. Ellis e S. J. Gibbs, «Concurrency control in groupware systems», in *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, New York: ACM, 1989, pp. 399–407. doi: 10.1145/67544.66963.
- [2] M. Shapiro, N. Preguiça, C. Baquero, e M. Zawirski, «Conflict-free replicated data types», in *Stabilization, Safety, and Security of Distributed Systems*, Berlin: Springer, 2011, pp. 386–400. doi: 10.1007/978-3-642-24550-3_29.
- [3] M. Kleppmann, *Designing data-intensive applications*. Sebastopol: O'Reilly Media, 2017.
- [4] J. Hendl, *Kvalitativní výzkum: základní metody a aplikace*. Praha: Portál, 2005.

Ringraziamenti

Ringrazio il mio relatore per la guida durante tutto il lavoro e i colleghi del dipartimento per le molte discussioni che lo hanno reso possibile.