SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Fakulta informatiky a informačných technológií

Evidenčné číslo: FIIT-12345-67890

Názov záverečnej práce

Bakalárska práca

2026 **Meno Priezvisko**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Fakulta informatiky a informačných technológií

Evidenčné číslo: FIIT-12345-67890

Názov záverečnej práce

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Ústav informatiky, informačných systémov a softvérového inžinierstva
Vedúci záverečnej práce:	Ing. Meno Priezvisko, PhD.

2026 **Meno Priezvisko**

Sem vložte zadanie záverečnej práce.

Poďakovanie

Ďakujem vedúcemu práce za odborné vedenie, cenné rady a pripomienky.

Abstrakt

Abstrakt výstižne informuje o cieľoch práce, jej obsahu, výsledkoch a význame.

Kľúčové slová: kľúčové slovo, kľúčové slovo, kľúčové slovo

Abstract

The abstract summarises the aims of the thesis, its content, results and significance.

Keywords: keyword, keyword, keyword

Obsah

Úvod **8**

1 Spoločná editácia dokumentov **9**

1.1 Požiadavky na systém 9

2 Prístupy k synchronizácii **11**

2.1 Operačná transformácia 11

2.1.1 Transformačná funkcia 11

2.2 Bezkonfliktné replikované dátové typy 11

3 Návrh a implementácia **12**

3.1 Ukladanie histórie 12

4 Vyhodnotenie **13**

4.1 Diskusia výsledkov 14

Záver **15**

Literatúra **16**

Zoznam skratiek a značiek

STU	Slovenská technická univerzita v Bratislave
ISO	Medzinárodná organizácia pre normalizáciu

Úvod

Textové dokumenty dnes len výnimočne vznikajú v rukách jediného autora. Študenti píšu seminárne práce vo dvojiciach, vývojári spoločne udržiavajú technickú dokumentáciu a výskumné tímy dokončujú články na poslednú chvíľu z niekoľkých miest naraz. Aby takéto písanie fungovalo, musí editor zabezpečiť, že všetci účastníci vidia rovnaký text a že sa žiadna úprava nestratí.

Cieľom tejto práce je porovnať dva hlavné prístupy k synchronizácii dokumentov pri spoločnej editácii v reálnom čase, operačnú transformáciu a bezkonfliktné replikované dátové typy, a overiť ich na prototype webového editora. Prvá kapitola vymedzuje požiadavky na takýto systém, druhá opisuje oba prístupy, tretia návrh prototypu a štvrtá výsledky meraní.

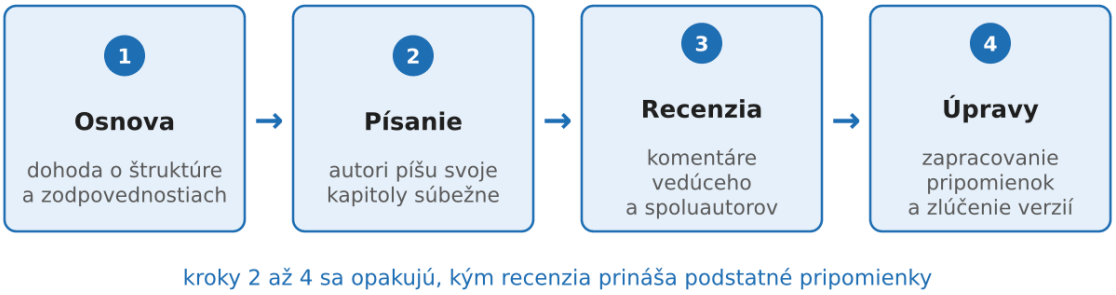
1 Spoločná editácia dokumentov

Spoločnou editáciou rozumieme situáciu, keď viacerí používatelia upravujú ten istý dokument a ich zmeny sa navzájom prenášajú bez ručného zlučovania. Podľa toho, ako rýchlo sa zmeny šíria, rozlišujeme tri základné režimy práce:

- **asynchrónna editácia**, pri ktorej si autori odovzdávajú celé verzie dokumentu a zlučujú ich až dodatočne,
- **editácia so zámkami**, keď smie v jednom okamihu upravovať určitú časť dokumentu len jeden autor,
- **editácia v reálnom čase**, pri ktorej sa každá zmena okamžite prejaví ostatným používateľom.

Táto práca sa zaoberá výlučne posledným režimom. Ten je pre používateľov najpohodlnejší, ale kladie najväčšie nároky na návrh systému, pretože zmeny vznikajú súbežne na mnohých miestach a prichádzajú v rôznom poradí.

Editácia v reálnom čase pritom nemení len techniku, ale aj priebeh celej spolupráce. Typický postup spoločného písania, ako ho opísali opýtaní používatelia, zachytáva Obrázok 1.



Obrázok 1: Typický priebeh spoločného písania dokumentu

1.1 Požiadavky na systém

Z rozhovorov s používateľmi existujúcich editorov a z odbornej literatúry sme odvodili štyri požiadavky, ktoré musí systém na spoločnú editáciu splniť:

1. konvergencia, teda po doručení všetkých zmien majú všetky kópie dokumentu rovnaký obsah,
2. zachovanie zámeru, teda výsledok úpravy zodpovedá tomu, čo autor v okamihu úpravy videl,
3. odozva, teda lokálna úprava sa prejaví okamžite bez čakania na server,

4. odolnosť, teda krátkodobý výpadok spojenia nevedie k strate rozpísaného textu.

Prvé dve požiadavky sa týkajú správnosti, zvyšné dve použiteľnosti. V praxi sa ukazuje, že práve kombinácia okamžitej odozvy a zaručenej konvergencie je najťažšou časťou celého návrhu.

2 Prístupy k synchronizácii

Systémy na spoločnú editáciu v reálnom čase vykonajú každú zmenu najprv lokálne a až potom ju rozošlú ostatným účastníkom. Líšia sa predovšetkým tým, ako riešia situáciu, keď dve zmeny vzniknú súbežne a každý účastník ich dostane v inom poradí.

2.1 Operačná transformácia

Operačná transformácia vychádza z práce Ellisa a Gibbsa (Ellis; Gibbs, 1989), ktorí ako prví opísali, ako upraviť súbežnú operáciu tak, aby ju bolo možné vykonať aj po inej operácii, o ktorej jej autor nevedel. Každá úprava sa zapisuje ako operácia vloženia alebo zmazania na určitej pozícii v texte.

Ak napríklad jeden používateľ vloží slovo na začiatok odseku a druhý súčasne zmaže slovo na jeho konci, musí sa pozícia mazania posunúť o dĺžku vloženého slova. Tento posun počíta transformačná funkcia.

2.1.1 Transformačná funkcia

Transformačná funkcia dostane dvojicu súbežných operácií a vráti ich upravené verzie. Aby všetky kópie dokumentu konvergovali, musí funkcia spĺňať takzvané transformačné vlastnosti, ktorých dodržanie je pre všeobecné operácie ťažké dokázať. Väčšina nasadených systémov sa preto spolieha na centrálny server, ktorý určuje jediné poradie operácií.

2.2 Bezkonfliktné replikované dátové typy

Druhý prístup predstavujú bezkonfliktné replikované dátové typy, súhrnne označované skratkou CRDT (Shapiro; Preguiça; Baquero; Zawirski, 2011). Namiesto dodatočnej úpravy operácií volia takú reprezentáciu dokumentu, v ktorej súbežné operácie komutujú a ich poradie nemá na výsledok vplyv.

„Replikovaný dátový typ je bezkonfliktný, ak ľubovoľné dve repliky, ktoré dostali rovnakú množinu aktualizácií, majú rovnaký stav bez ohľadu na poradie ich doručenia.“

Každý znak dokumentu dostane jednoznačný identifikátor, ktorý zostáva rovnaký počas celej existencie znaku. Pozícia vloženia sa potom neurčuje číslom, ale odkazom na identifikátor susedného znaku, takže ju súbežné úpravy inde v texte nemôžu posunúť.

3 Návrh a implementácia

Prototyp editora sa skladá z webového klienta, synchronizačného servera a databázy, do ktorej server priebežne ukladá stav dokumentov. Klient aj server sú napísané tak, aby bolo možné synchronizačnú vrstvu vymeniť a porovnať oba opísané prístupy na rovnakom používateľskom rozhraní.

Klient udržiava lokálnu repliku dokumentu a každú úpravu používateľa prevedie na operáciu nad replikovaným dátovým typom. Operácie sa odosielajú serveru po dávkach, aby sa pri rýchlom písaní nezahltila sieť. Zjednodušené spracovanie prichádzajúcej dávky na serveri ukazuje nasledujúci výpis:

```
def spracuj_davku(dokument, davka):
    for operacia in davka.operacie:
        if operacia.id in dokument.pouzite:
            continue
        dokument.aplikuj(operacia)
        dokument.pouzite.add(operacia.id)
    rozosli_ostatnym(dokument.id, davka)
```

Keďže operácie komutujú, server nemusí dávky zoraďovať ani transformovať. Stačí, keď zabezpečí, že sa žiadna operácia nevykoná dvakrát a že každá dávka dorazí ku všetkým pripojeným klientom.

3.1 Ukladanie histórie

Server neukladá len výsledný text, ale celý log operácií. Z logu možno obnoviť ľubovoľnú skoršiu verziu dokumentu a znova zostaviť stav repliky klienta, ktorý bol dlhší čas odpojený. Tento prístup zodpovedá architektúre založenej na nemennom zázname udalostí, ako ju opisuje Kleppmann (Kleppmann, 2017). Aby log nerástol neobmedzene, server ho v pravidelných intervaloch zhusťuje do snímok.

4 Vyhodnotenie

Prototyp sme porovnali s referenčnou implementáciou operačnej transformácie na súbore záznamov skutočných editačných relácií. Pre každú reláciu sme sledovali latenciu, teda čas medzi úpravou u autora a jej zobrazením u ostatných účastníkov, a priemernú latenciu relácie sme vypočítali tak, ako ukazuje (1):

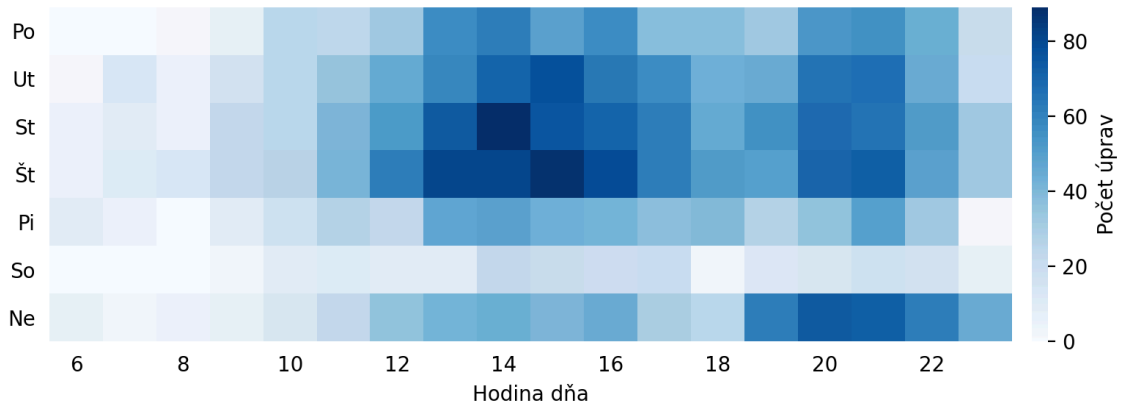
$$\bar{t} = \frac{1}{n} \sum_{i=1}^{n} \left(t_i^{\text{doručenie}} - t_i^{\text{úprava}} \right) \tag{1}$$

kde n je počet operácií v relácii. Namerané hodnoty pre rôzny počet súbežne pracujúcich používateľov zhŕňa Tabuľka 1.

Tabuľka 1: Priemerná latencia a veľkosť uložených dát podľa počtu používateľov

Používatelia	**Latencia OT (ms)**	**Latencia CRDT (ms)**	**Dáta (kB)**
2	38	35	12
5	52	41	19
10	87	48	31
20	164	63	58

S rastúcim počtom používateľov rastie latencia operačnej transformácie výrazne rýchlejšie, pretože server musí každú operáciu transformovať voči všetkým súbežným operáciám. Cenou za nižšiu latenciu CRDT je väčší objem uložených dát, keďže každý znak nesie svoj identifikátor. Rozloženie úprav v čase, z ktorého vychádza záťažový test, ukazuje Obrázok 2.



Obrázok 2: Počet úprav podľa dňa v týždni a hodiny v sledovaných reláciách

4.1 Diskusia výsledkov

Výsledky potvrdzujú, že pre menšie tímy sú oba prístupy rovnocenné a rozhoduje skôr jednoduchosť implementácie. Pri väčšom počte súbežných používateľov má CRDT jasnú výhodu v latencii aj v odolnosti voči výpadkom spojenia. Doplňujúce rozhovory s používateľmi, vedené podľa zásad kvalitatívneho výskumu (Hendl, 2005), ukázali, že rozdiel v latencii pod sto milisekúnd používatelia nevnímajú, no stratu rozpísaného textu po výpadku spojenia považujú za zásadný problém.

Záver

Práca porovnala operačnú transformáciu a bezkonfliktné replikované dátové typy ako dva prístupy k synchronizácii dokumentov pri spoločnej editácii v reálnom čase. Na prototype webového editora sme ukázali, že CRDT umožňujú jednoduchší server bez centrálneho zoraďovania operácií a pri väčšom počte používateľov ponúkajú nižšiu latenciu.

Hlavnou nevýhodou CRDT zostáva väčší objem ukladaných dát. Ďalším krokom preto bude overiť, ako veľkosť dát ovplyvňuje pravidelné zhusťovanie logu do snímok a ako sa prototyp správa pri dlhodobej prevádzke s desiatkami tisíc dokumentov.

Literatúra

ELLIS, C. A.; GIBBS, S. J., 1989. Concurrency control in groupware systems. In: *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*. New York: ACM, s. 399–407. Dostupné z doi: 10.1145/67544.66963.

HENDL, J., 2005. *Kvalitativní výzkum: základní metody a aplikace*. Praha: Portál.

KLEPPMANN, M., 2017. *Designing data-intensive applications*. Sebastopol: O'Reilly Media. isbn 978-1-4493-7332-0.

SHAPIRO, M.; PREGUIÇA, N.; BAQUERO, C.; ZAWIRSKI, M., 2011. Conflict-free replicated data types. In: *Stabilization, Safety, and Security of Distributed Systems*. Berlin: Springer. Vol. 6976, s. 386–400. Lecture Notes in Computer Science. Dostupné z doi: 10.1007/978-3-642-24550-3_29.