



Bakalářská práce

Název práce

Studijní program:

Autor práce:

Vedoucí práce:

B0613A140005 – Informační technologie

Jméno Příjmení

doc. Ing. Jméno Příjmení, Ph.D.

Liberec 2026

Tento list nahrad'te
originálem zadání.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

1. 1. 2026

Jméno Příjmení

NÁZEV PRÁCE

ABSTRAKT

Abstrakt shrnuje cíl práce, použité metody, hlavní výsledky a závěry.

Klíčová slova: klíčové slovo, klíčové slovo, klíčové slovo

THESIS TITLE

ABSTRACT

The abstract summarises the aim of the thesis, the methods, the main results and the conclusions.

Keywords: keyword, keyword, keyword

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu práce za cenné rady a připomínky.

OBSAH

Seznam zkratk	7
Úvod	8
1 Společná editace dokumentů	9
1.1 Požadavky na systém	10
2 Přístupy k synchronizaci	11
2.1 Operační transformace	11
2.1.1 Transformační funkce	11
2.2 Bezkonfliktní replikované datové typy	11
3 Návrh a implementace	13
3.1 Ukládání historie	13
4 Vyhodnocení	14
4.1 Diskuse výsledků	15
Závěr	16
Literatura	17

SEZNAM ZKRATEK

TUL Technická univerzita v Liberci

ÚVOD

Textové dokumenty dnes jen výjimečně vznikají v rukou jediného autora. Studenti píší seminární práce ve dvojicích, vývojáři společně udržují technickou dokumentaci a výzkumné týmy dokončují články na poslední chvíli z několika míst najednou. Aby takové psaní fungovalo, musí editor zajistit, že všichni účastníci vidí stejný text a že se žádná úprava neztratí.

Cílem této práce je porovnat dva hlavní přístupy k synchronizaci dokumentů při společné editaci v reálném čase, operační transformaci a bezkonfliktní replikované datové typy, a ověřit je na prototypu webového editoru. První kapitola vymezuje požadavky na takový systém, druhá popisuje oba přístupy, třetí návrh prototypu a čtvrtá výsledky měření.

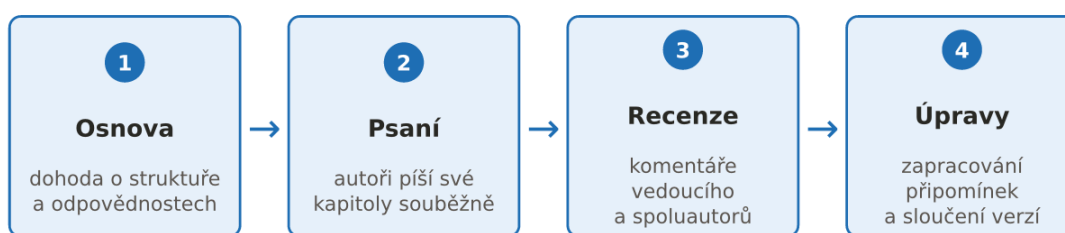
1 SPOLEČNÁ EDITACE DOKUMENTŮ

Společnou editací rozumíme situaci, kdy více uživatelů upravuje tentýž dokument a jejich změny se navzájem propisují bez ručního slučování. Podle toho, jak rychle se změny šíří, rozlišujeme tři základní režimy práce:

- **asynchronní editace**, při níž si autoři předávají celé verze dokumentu a slučují je až dodatečně,
- **editace se zámky**, kdy smí v jednu chvíli upravovat určitou část dokumentu jen jeden autor,
- **editace v reálném čase**, při níž se každá změna okamžitě promítne ostatním uživatelům.

Tato práce se zabývá výhradně posledním režimem. Ten je pro uživatele nej pohodlnější, ale klade největší nároky na návrh systému, protože změny vznikají souběžně na mnoha místech a přicházejí v různém pořadí.

Editace v reálném čase přitom nemění jen techniku, ale i průběh celé spolupráce. Typický postup společného psaní, jak jej popsali dotazovaní uživatelé, zachycuje Obrázek 1.1.



kroky 2 až 4 se opakují, dokud recenze přináší podstatné připomínky

Obrázek 1.1: Typický průběh společného psaní dokumentu

1.1 POŽADAVKY NA SYSTÉM

Z rozhovorů s uživateli existujících editorů a z odborné literatury jsme odvodili čtyři požadavky, které musí systém pro společnou editaci splnit:

1. konvergence, tedy po doručení všech změn mají všechny kopie dokumentu stejný obsah,
2. zachování záměru, tedy výsledek úpravy odpovídá tomu, co autor v okamžiku úpravy viděl,
3. odezva, tedy lokální úprava se projeví okamžitě bez čekání na server,
4. odolnost, tedy krátkodobý výpadek spojení nevede ke ztrátě rozepsaného textu.

První dva požadavky se týkají správnosti, zbylé dva použitelnosti. V praxi se ukazuje, že právě kombinace okamžité odezvy a zaručené konvergence je nejtěžší částí celého návrhu.

2 PŘÍSTUPY K SYNCHRONIZACI

Systémy pro společnou editaci v reálném čase si každou změnu nejprve provedou lokálně a teprve potom ji rozešlou ostatním účastníkům. Liší se především tím, jak řeší situaci, kdy dvě změny vzniknou souběžně a každý účastník je obdrží v jiném pořadí.

2.1 OPERAČNÍ TRANSFORMACE

Operační transformace vychází z práce Ellise a Gibbse [1], kteří jako první popsali, jak upravit souběžnou operaci tak, aby ji bylo možné provést i po jiné operaci, o které její autor nevěděl. Každá úprava se zapisuje jako operace vložení nebo smazání na určité pozici v textu.

Pokud například jeden uživatel vloží slovo na začátek odstavce a druhý současně smaže slovo na jeho konci, musí se pozice mazání posunout o délku vloženého slova. Tento posun počítá transformační funkce.

2.1.1 Transformační funkce

Transformační funkce dostane dvojici souběžných operací a vrátí jejich upravené verze. Aby všechny kopie dokumentu konvergovaly, musí funkce splňovat takzvané transformační vlastnosti, jejichž dodržení je pro obecné operace obtížné dokázat. Většina nasazených systémů proto spoléhá na centrální server, který určuje jediné pořadí operací.

2.2 BEZKONFLIKTNÍ REPLIKOVANÉ DATOVÉ TYPY

Druhý přístup představují bezkonfliktní replikované datové typy, souhrnně označované zkratkou CRDT [2]. Místo dodatečné úpravy operací volí takovou reprezentaci dokumentu, v níž souběžné operace komutují a jejich pořadí nemá na výsledek vliv.

Replikovaný datový typ je bezkonfliktní, pokud libovolné dvě repliky, které obdržely stejnou množinu aktualizací, mají stejný stav bez ohledu na pořadí jejich doručení.

Každý znak dokumentu dostane jednoznačný identifikátor, který zůstává stejný po celou dobu existence znaku. Pozice vložení se pak neurčuje číslem, ale odkazem na identifikátor sousedního znaku, takže ji souběžné úpravy jinde v textu nemohou posunout.

3 NÁVRH A IMPLEMENTACE

Prototyp editoru se skládá z webového klienta, synchronizačního serveru a databáze, do níž server průběžně ukládá stav dokumentů. Klient i server jsou napsány tak, aby bylo možné synchronizační vrstvu vyměnit a porovnat oba popsané přístupy na stejném uživatelském rozhraní.

Klient udržuje lokální repliku dokumentu a každou úpravu uživatele převede na operaci nad replikovaným datovým typem. Operace se odesílají serveru po dávkách, aby se při rychlém psaní nezahltla síť. Zjednodušené zpracování příchozí dávky na serveru ukazuje následující výpis:

```
def zpracuj_davku(dokument, davka):
    for operace in davka.operace:
        if operace.id in dokument.pouzite:
            continue
        dokument.aplikuj(operace)
        dokument.pouzite.add(operace.id)
    rozesli_ostatnim(dokument.id, davka)
```

Protože operace komutují, nemusí server dávky řadit ani transformovat. Stačí, když zajistí, že se žádná operace neprovede dvakrát a že každá dávka dorazí ke všem připojeným klientům.

3.1 UKLÁDÁNÍ HISTORIE

Server neukládá jen výsledný text, ale celý log operací. Z logu lze obnovit libovolnou dřívější verzi dokumentu a znovu sestavit stav repliky klienta, který byl delší dobu odpojen. Tento přístup odpovídá architektuře založené na neměnném záznamu událostí, jak ji popisuje Kleppmann [3]. Aby log nerostl neomezeně, server jej v pravidelných intervalech zhušťuje do snímků.

4 VYHODNOCENÍ

Prototyp jsme porovnali s referenční implementací operační transformace na sadě záznamů skutečných editačních relací. Pro každou relaci jsme sledovali latenci, tedy dobu mezi úpravou u autora a jejím zobrazením u ostatních účastníků, a průměrnou latenci relace jsme spočítali podle (4.1):

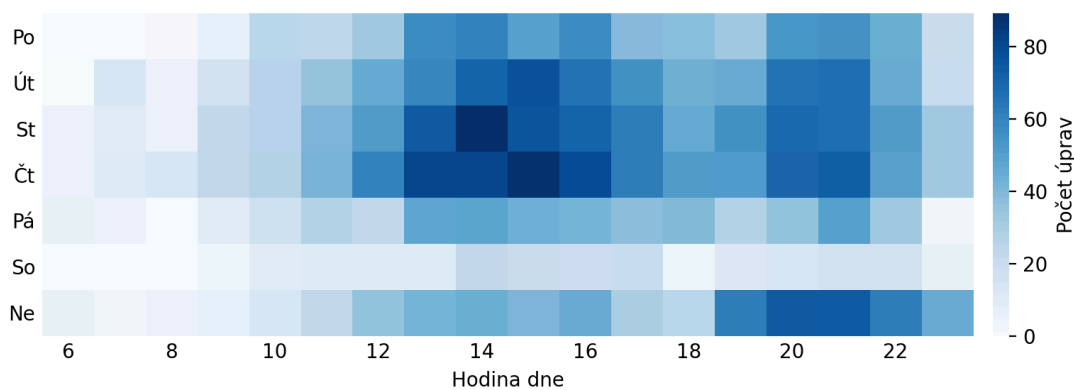
$$\bar{t} = \frac{1}{n} \sum_{i=1}^n (t_i^{\text{doručení}} - t_i^{\text{úprava}}) \quad (4.1)$$

kde n je počet operací v relaci. Naměřené hodnoty pro různý počet souběžně pracujících uživatelů shrnuje Tabulka 4.1.

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

Tabulka 4.1: Průměrná latence a velikost uložených dat podle počtu uživatelů

S rostoucím počtem uživatelů roste latence operační transformace výrazně rychleji, protože server musí každou operaci transformovat proti všem souběžným operacím. Cenou za nižší latenci CRDT je větší objem uložených dat, neboť každý znak nese svůj identifikátor. Rozložení úprav v čase, z něhož vychází zátěžový test, ukazuje Obrázek 4.1.



Obrázek 4.1: Počet úprav podle dne v týdnu a hodiny ve sledovaných relacích

4.1 DISKUSE VÝSLEDKŮ

Výsledky potvrzují, že pro menší týmy jsou oba přístupy rovnocenné a rozhoduje spíše jednoduchost implementace. Při větším počtu souběžných uživatelů má CRDT jasnou výhodu v latenci i v odolnosti vůči výpadkům spojení. Doplnující rozhovory s uživateli, vedené podle zásad kvalitativního výzkumu [4], ukázaly, že rozdíl v latenci pod sto milisekund uživatelé nevnímají, zato ztrátu rozepsaného textu po výpadku spojení považují za zásadní problém.

ZÁVĚR

Práce porovnala operační transformaci a bezkonfliktní replikované datové typy jako dva přístupy k synchronizaci dokumentů při společné editaci v reálném čase. Na prototypu webového editoru jsme ukázali, že CRDT umožňují jednodušší server bez centrálního řazení operací a při větším počtu uživatelů nabízejí nižší latenci.

Hlavní nevýhodou CRDT zůstává větší objem ukládaných dat. Dalším krokem proto bude ověřit, jak velikost dat ovlivňuje pravidelné zhušťování logu do snímků a jak se prototyp chová při dlouhodobém provozu s desítkami tisíc dokumentů.

LITERATURA

1. ELLIS, Clarence A.; GIBBS, Simon J. 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. Available from DOI: [10.1145/67544.66963](https://doi.org/10.1145/67544.66963).
2. SHAPIRO, Marc; PREGUIÇA, Nuno; BAQUERO, Carlos; ZAWIRSKI, Marek. Conflict-free replicated data types. In: *Stabilization, Safety, and Security of Distributed Systems*. Berlin: Springer, 2011, vol. 6976, pp. 386–400. Lecture Notes in Computer Science. Available from DOI: [10.1007/978-3-642-24550-3_29](https://doi.org/10.1007/978-3-642-24550-3_29).
3. KLEPPMANN, Martin. *Designing data-intensive applications*. Sebastopol: O'Reilly Media, 2017. ISBN 978-1-4493-7332-0.
4. HENDL, Jan. *Kvalitativní výzkum: základní metody a aplikace*. Praha: Portál, 2005.