

**Česká zemědělská univerzita v Praze**

**Fakulta agrobiologie, potravinových a přírodních zdrojů**

**Katedra agroekologie a rostlinné produkce**



**Fakulta agrobiologie,  
potravinových a přírodních zdrojů**

**Název práce**

**Bakalářská práce**

**Jméno Příjmení**

**Agroekologie**

**doc. Ing. Jméno Příjmení, Ph.D.**

**© 2027 ČZU v Praze**

## **Čestné prohlášení**

Prohlašuji, že svou bakalářskou práci "Název práce" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 30. 4. 2027

---

**Poděkování**

Rád(a) bych touto cestou poděkoval(a) vedoucímu práce za odborné vedení a cenné rady.

# Název práce

## Souhrn

Sem napište souhrn práce (1 strana textu).

**Klíčová slova:** klíčové slovo, klíčové slovo, klíčové slovo

# Title of the Thesis

## Summary

Write the summary of your thesis here (1 page of text).

**Keywords:** keyword, keyword, keyword

# Obsah

|                                                |    |
|------------------------------------------------|----|
| Úvod.....                                      | 7  |
| 1 Společná editace dokumentů.....              | 8  |
| 1.1 Požadavky na systém.....                   | 8  |
| 2 Přístupy k synchronizaci.....                | 9  |
| 2.1 Operační transformace.....                 | 9  |
| 2.1.1 Transformační funkce.....                | 9  |
| 2.2 Bezkonfliktní replikované datové typy..... | 9  |
| 3 Návrh a implementace.....                    | 10 |
| 3.1 Ukládání historie.....                     | 10 |
| 4 Vyhodnocení.....                             | 11 |
| 4.1 Diskuse výsledků.....                      | 11 |
| Závěr.....                                     | 12 |
| 5 Literatura.....                              | 13 |
| 6 Seznam použitých zkratek a symbolů.....      | 14 |

## 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.

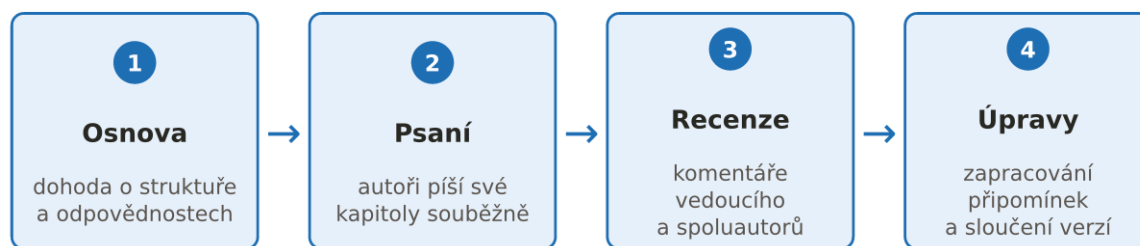
# 1 Společná editace dokumentů

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 nejpohodlnější, 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.



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

Obrázek 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 (Ellis & Gibbs 1989), 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 (Shapiro et al. 2011). 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 (Kleppmann 2017). 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 Rovnice 1:

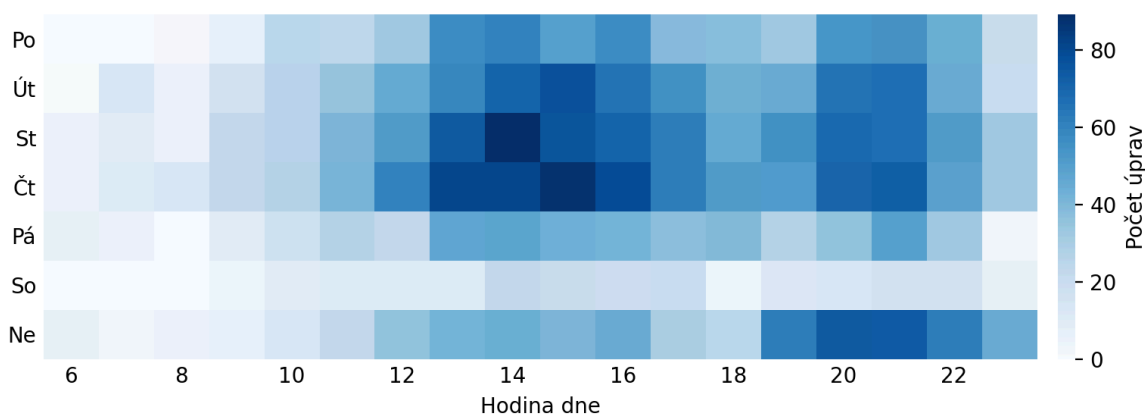
$$\bar{t} = \frac{1}{n} \sum_{i=1}^n (t_i^{\text{doručení}} - t_i^{\text{úprava}}) \quad (1)$$

kde  $n$  je počet operací v relaci. Naměřené hodnoty pro různý počet souběžně pracujících uživatelů shrnuje Tabulka 1.

Tabulka 1: Průměrná latence a velikost uložených dat podle počtu uživatelů

| 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        |

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 2.



Obrázek 2: 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 (Hendl 2005), ukázaly, že rozdíl v latenci pod sto milisekund uživatelé nevnímají, zato ztrátu rozepsaného textu po výpadku spojení považují za zásadní problém.

## Závěr

Práce porovnála 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.

## 5 Literatura

- Ellis CA, Gibbs SJ. 1989. Concurrency control in groupware systems. Pages 399–407 in Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data. ACM, New York.
- Hendl J. 2005. Kvalitativní výzkum: základní metody a aplikace. Portál, Praha.
- Kleppmann M. 2017. Designing data-intensive applications. O'Reilly Media, Sebastopol.
- Shapiro M, Preguiça N, Baquero C, Zawirski M. 2011. Conflict-free replicated data types. Pages 386–400 in Stabilization, Safety, and Security of Distributed Systems. Springer, Berlin.

## **6 Seznam použitých zkratk a symbolů**

ČZU Česká zemědělská univerzita v Praze

FAPPZ Fakulta agrobiologie, potravinových a přírodních zdrojů