Jízdní řády hromadné dopravy

Lukáš Riedel, 2019
Matematicko-fyzikální fakulta, Univerzita Karlova

Úvod

Systém aplikací zahrnuje **serverovou**, **počítačovou** a **mobilní** aplikaci.

Klientské aplikace poskytují uživateli možnost **vyhledávat spojení v jízdních řádech**, odjezdy ze stanice a informace o lince. Dále je možné zobrazit si mapu dopravní sítě, informace o výlukách či mimořádnosti v dopravě. Nalezené výsledky je možné si uložit mezi **oblíbené položky**.

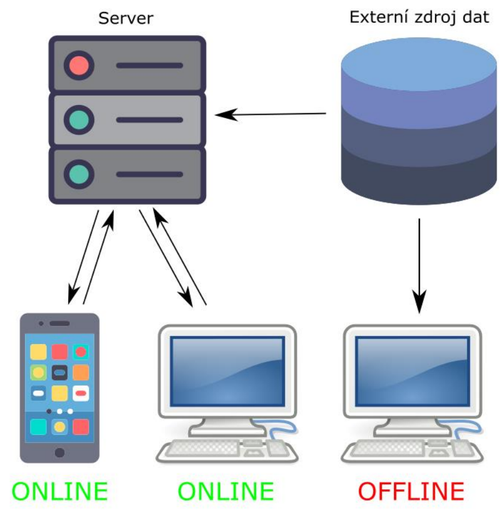
Serverová aplikace obstarává samotné vyhledávání, řešení využívá inovativní **RAPTOR algoritmus**, který pracuje v **lineární časové složitosti**.

Počítačová aplikace podporuje možnost **offline** módu. Aplikace běžící v tomto módu ke své činnosti nepotřebuje server.

Existující řešení se omezují na jednu konkrétní oblast, což může uživatele limitovat.

Prezentovaný systém je plně **přizpůsobitelný**. Dokáže zpracovat **libovolná data** v GTFS formátu, popřípadě **spojit** více datových sad a vytvořit tak vyhledávač pro libovolně velkou oblast, potažmo i celý svět.

Architektura



Ostatní

- Aplikace běží nad **platformou .NET**.
- Jádro je psáno v **nativním kódu** za využití **C++/CLI**.
- Klientské aplikace jsou **lokalizované**, lokalizační soubory se dodávají **zvnějšku**.
- Mobilní aplikace je psána **multiplatformně.**
- Způsob **zobrazování výsledků** lze modifikovat pomocí externích **XSLT**, **CSS** a **JavaScript** skriptů.

Zpracování dat

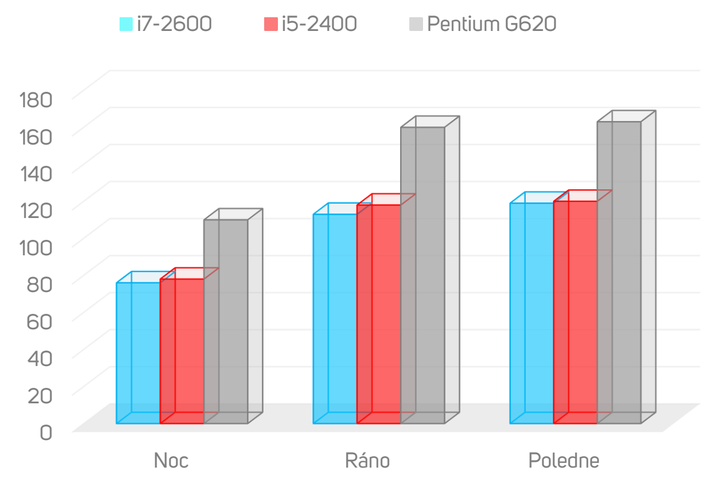
- Zpracování dat spočívá ve **stažení** dat z externího zdroje a **vytvoření** dat ve formátech **TFD** a **TFB**.
- Na serveru jsou implementovány **automatické aktualizace**, aby bylo možné provozovat server 24/7.
- **Klientská data** jsou nutná pro základní funkcionalitu.
- Modul pro zpracování dat lze snadno rozšířit o **parsování jiného formátu**.

Oblíbené položky

- **Přidání** oblíbené položky je ekvivalentní hledání dané položky dle **maximálního času příjezdu**.
- Po přijetí odpovědi ze serveru se položka serializuje do **XML** formátu a **uloží se** na disk.
- **Vyhledávání** oblíbené položky se uskutečňuje offline nad daným XML souborem, odpověď na uživatelův požadavek se nalezne pomocí **dotazovacího jazyku**.

Výsledky

Implementovaný algoritmus nalezne spojení v rozmezí **10-100 ms**, v závislosti na denní době a parametrech nalezeného spojení. Dijkstrův algoritmus může být až několikanásobně pomalejší.



Graf udává čas (v milisekundách) potřebný k nalezení spojení mezi stanicemi Horčičkova a Letiště v závislosti na denní době a výkonu procesoru.

Algoritmus na vyhledávání spojení

Algoritmus 1 RAPTOR

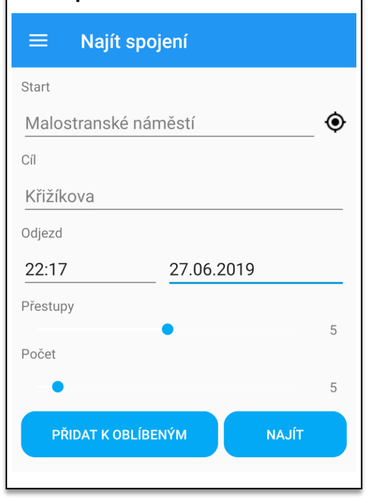
Vstup: výchozí a cílová zastávka $s_s, s_t \in \mathcal{S}$, čas odjezdu $\tau \in \Pi$ a maximální počet přestupů $n \in \mathbb{N}_0$
Výstup: nejlepší možný příjezdový čas do s_t uložený v $\tau^*(s_t)$, zrekonstruované spojení z s_s do s_t

1: **for all** i **do**
2: $\tau_i(\_) \leftarrow \infty$
3: $\tau^*(\_) \leftarrow \infty$
4: $\tau_0(s_s) \leftarrow \tau$
5: Označ s_s
6: **for** $k \leftarrow 1, \ldots, n+1$ **do** ▷ Jedna iterace cyklu je jedno kolo.
7: $Q = \emptyset$
8: **for all** označená zastávka s **do** ▷ První fáze kola.
9: **for all** linka r obsahující zastávku s **do**
10: **if** $(r,s') \in Q$ pro nějakou zastávku s' **then**
11: Nahraď (r,s') dvojicí (r,s) v Q, pokud s předchází s' v lince r
12: **else**
13: Přidej (r,s) do Q
14: Odznač s
15: **for all** $(r,s) \in Q$ **do** ▷ Druhá fáze kola.
16: $(t,r) \leftarrow \perp$ ▷ Aktuální jízda linky.
17: **for all** zastávka $s_i \in s, \ldots, s_{|r|}$ (části) trasy linky r **do**
18: **if** $(t,r) \neq \perp \wedge \tau_{arr}(t,s_i) < \min\{\tau^*(s_i), \tau^*(s_t)\}$ **then**
19: $\tau_k(s_i) \leftarrow \tau_{arr}(t,s_i)$
20: $\tau^*(s_i) \leftarrow \tau_{arr}(t,s_i)$
21: Označ s_i
22: **if** $(t,r) = \perp \vee \tau_{k-1}(s_i) \leq \tau_{dep}(t,s_i)$ **then**
23: $(t,r) \leftarrow et(r,s_i,\tau_{k-1}(s_i))$
24: **for all** označená zastávka s **do** ▷ Třetí fáze kola.
25: **for all** definovaný přestup $\mathcal{F}(s,s')$ **do**
26: $\tau_k(s') \leftarrow \min\{\tau_k(s'), \tau_k(s) + \mathcal{F}(s,s')\}$
27: Označ s'
28: **if** žádná zastávka není označená **then return**

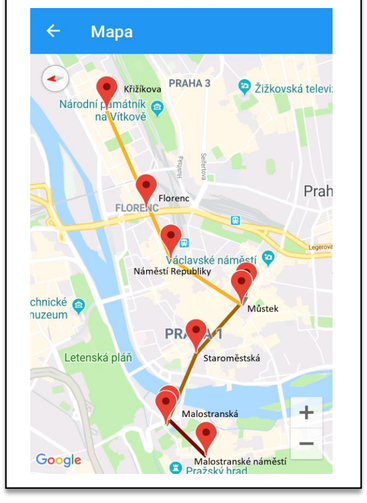
Zpracování požadavku na vyhledání spojení

FRONT-END

1. Uživatel v klientské aplikaci **zadá požadavek**.

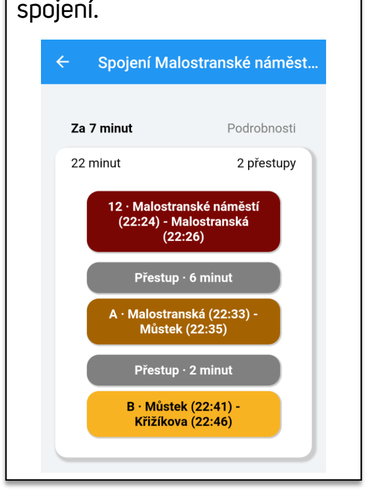


9. Z detailu spojení lze dále zobrazit **mapa spojení**.



8. U jednotlivých spojení lze zobrazit jejich **detail**.

7. Výsledky se **zobrazí** uživateli. Jedná se o **seznam** spojení.



BACK-END

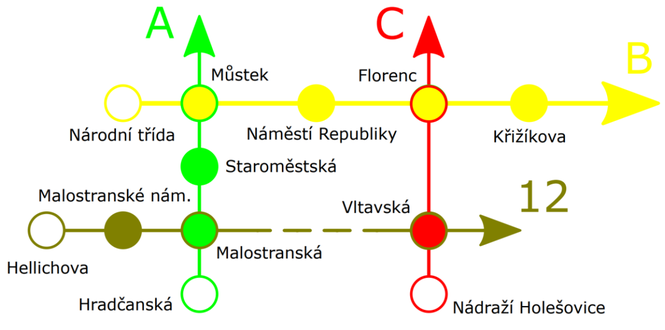
2. Klientská aplikace požadavek zpracuje, prohledá **oblíbené položky** a popřípadě ho **odešle** na server.

6. Klientská aplikace odpověď **přijme**. Pokud se jednalo o aktualizaci **oblíbené položky**, pak ji **uloží** na disk.

SERVER

3. Server požadavek **přijme** a předá ho k řešení na **threadpool**.

4. **Algoritmus** na vyhledávání spojení pracuje v **kolech**. V prvním kole se zpracuje trasa linky 12. Ve druhém kole se zpracuje trasa linek A a C. Ve třetí kole se zpracuje trasa linky B.



5. Server odpověď **odešle** zpět klientovi v rámci téhož spojení.