Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Riešenie úloh strojového učenia využitím Kubernetes

Bakalárska práca

2022 **Štefan Hadbavný**

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Riešenie úloh strojového učenia využitím Kubernetes

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	9.2.1. Informatika
Školiace pracovisko:	Katedra počítačov a informatiky (KPI)
Školiteľ:	Marek Ružička
Konzultant:	Marcel Vološin

Košice 2022 **Štefan Hadbavný**

Abstrakt v SJ

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisissem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi necante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultriciesvel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero utmetus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit ametante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Kľúčové slová v SJ

LaTeX, programovanie, sadzba textu

Abstrakt v AJ

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisissem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi necante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultriciesvel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero utmetus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit ametante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Kľúčové slová v AJ

LaTeX, programming, typesetting

Bibliografická citácia

HADBAVNÝ, Štefan. *Riešenie úloh strojového učenia využitím Kubernetes*. Košice: Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky, 2022. 14s. Vedúci práce: Marek Ružička

Tu vložte zadávací list pomocou príkazu
`\thesisspec{cesta/k/suboru/so/zadavacim.listom}`
v preambule dokumentu.
Kópiu zadávacieho listu skenujte čiernobielo (v odtieňoch sivej) na 200 až 300 DPI! Nezabudnite do jednej práce vložiť originál zadávacieho listu!

Čestné vyhlásenie

Vyhlasujem, že som záverečnú prácu vypracoval(a) samostatne s použitím uvedenej odbornej literatúry.

Košice, 13.5.2022 .....

Vlastnoručný podpis

Poďakovanie

Na tomto mieste by som rád poďakoval svojmu vedúcemu práce za jeho čas a odborné vedenie počas riešenia mojej záverečnej práce.

Rovnako by som sa rád poďakoval svojim rodičom a priateľom za ich podporu a povzbudzovanie počas celého môjho štúdia.

V neposlednom rade by som sa rád poďakoval pánom *Donaldovi E. Knuthovi* a *Leslie Lamportovi* za typografický systém LaTeX, s ktorým som strávil množstvo nezabudnuteľných večerov.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Úvod práce stručne opisuje stanovený problém, kontext problému a motiváciu pre riešenie problému. Z úvodu by malo byť jasné, že stanovený problém doposiaľ nie je vyriešený a má zmysel ho riešiť. V úvode neuvádzajte štruktúru práce, t.j. o čom je ktorá kapitola. Rozsah úvodu je minimálne 2 celé strany (vrátane formulácie úlohy).

Ďalšie užitočné informácie môžete nájsť v Pokynoch pre vypracovanie záverečných prác[1].

Formulácia úlohy

Text záverečnej práce musí obsahovať sekciu s formuláciou úlohy resp. úloh riešených v rámci záverečnej práce. V tejto časti autor rozvedie spôsob, akým budú riešené úlohy a tézy formulované v zadaní práce. Taktiež uvedie prehľad podmienok riešenia.

[1] https://moodle.fei.tuke.sk/pluginfile.php/27971/mod_resource/content/16/Instructions_v15.pdf

1 Analytická časť

Pri strojovom učení a experimentoch je niekedy potreba na krátku dobu vyšší výkon. Pravidelne migrovanie medzi strojmi by bolo veľmi zdĺhavé. S tým nám pomôže platforma kubernetes.

Kubernetes je platforma, ktorá je veľmi robustná v nasadení, správe a orchestrácií kontajnerov. Dokáže rozdeliť súčasne záťaž medzi jednotlivými strojmi. Tato platforma v súčasnosti vytlačila predošle platformy a stala sa už štandardom. Nasadenie Kubernetes je najefektívnejšie, aj keď existuje dnes veľa podobných technológii, mnoho významných poskytovateľov ponuka klastre Kubernetes.

1.1 Kubernetes

Kubernetes je zložený z niekoľkých častí, ktoré sú potrebné pre jeho správne fungovanie a efektivitu. Pre správne pochopenie ako Kubernetes funguje je treba sa zoznámiť najmä s týmito termínmi.

Mikroslužby

Pre mikroslužby neexistuje presná definícia. Často sa používajú pri cloudových službách a aplikácii, ktoré sa nasadzujú pomocou kontajnerov.

Uzol

Uzol môže byt virtuálny stroj alebo fyzicky počítač, kde sú nasadené kontajnery. Za prijímanie a spúšťanie pracovných zaťažení a externých zdrojov sú zodpovedne uzly. Kubernetes spúšťa aplikácie a služby v kontajneroch na pomoc s izoláciou, správou a flexibilitou. Každý uzol musí mat modul tzv. runtime kontajnera. Uzol prijíma pracovne pokyny od hlavného uzla a podlá toho vytvára alebo ruší kontajnery a zároveň prispôsobuje pravidla sieťovej prevádzky.

Pod

Pod je skupina jedného alebo viacerých kontajnerov so zdieľaným úložiskom, sieťou a špecifikáciou ako majú kontajnery fungovať. Pri necloudových sú spúšťane na rovnakom fyzickom alebo virtuálnom stroji a pri cloudových na rovnakom logickom hostiteľovi. Všetky kontajnery v pode môžu medzi sebou komunikovať aj sú na samostatných uzloch. Sú vytvorené na základe pracovného zaťaženia nazývanými ovládače, ktoré riadia vytváranie, kopírovanie a stav podov v klastri. Ak napríklad zlyhá uzol v klastri, ovládač zisti, že moduly v tomto uzle nereagujú a vytvorí náhradne moduly na iných uzloch.

Klaster

V klastroch sa spúšťajú kontajnerové aplikácie, ktoré spravuje Kubernetes. Klaster je v podstate séria uzlov spojených dohromady. Spojením uzlov zhromažďujú svoje zdroje(CPU, RAM, atď.). Klaster je v tom prípade oveľa výkonnejší ako jednotlivé stroje. Kubernetes presúva pody po klastri, pri pridávaní alebo odstraňovaní uzlov.[1] Sú to navzájom prepojene počítače, ktoré vykonávajú určitú činnosť.

Kontajner

Kontajner je podobný virtuálnemu stroju ale kontajner je viac efektívnejší, podobne ako virtuálny stroj ma vlastný súborový systém, zdieľaný procesor, operačnú pamäť a ďalšie. Sú funkčne na rôznych operačných systémoch, pretože sa oddeľujú od základnej infraštruktúry.[2] Výrazne zvyšujú efektivitu, vyžadujú menej systémových prostriedkov, pretože neobsahujú obraz operačného systému. Zabezpečujú lepší vývoj aplikácii a lepšiu prenosnosť.

1.1.1 Architektúra

Pracovné uzly (worker), hlavný uzol (master) a spolu s API, tvoria architektúru Kubernetes. V tejto časti si povieme o architektúre ako tieto uzly fungujú a načo nám poslúži API.

V hlavnom uzle nájdeme komponenty, ktoré riadia klaster spolu s údajmi o stave a konfigurácii klastra. Tieto základne komponenty pracujú tak aby dokázali zabezpečiť prácu tak aby kontajnery fungovali v dostatočnom počte a spotrebnými zdrojmi. Hlavný uzol je neustále v kontakte s jednotlivými strojmi, ktoré

vykonávajú prácu. Hlavný uzol sa postará o klaster, tak ako sme ho nakonfigurovali.

Kube-apiserver

Tento komponent odhaľuje API hlavnému uzlu. V podstate funguje ako frontend k informáciám o stave klastra a premosťuje API s inými objektmi Kubernetes, napríklad modulmi, radičmi a službami. Server API určuje, ci je požiadavka platná, a ak áno, spracuje ju. K API sa pristupuje prostredníctvom Rest, cez rozhranie príkazového riadka kubectl alebo prostredníctvom iných nástrojov napríklad kubeadm. Zabezpečuje všetku interakciu medzi komponentmi.

Kube-controller-manager

Na hlavnom uzle riadi všetky ovládače. Každý ovládač je vlastne samostatný individuálny proces. Pre zjednodušenie správy klastrov sú však všetky skompilovane do jedného procesu, Za tuto kompiláciu je zodpovedný kube-controller-manager. Jeden ovládač konzultuje plánovač a uisťuje sa, že beží správny počet podov. Ak pod spadne, iný ovládač si to všimne a zareaguje. Ovládač pripája služby k podom, takže požiadavky smerujú do správneho koncového bodu. Existujú aj ovládače na vytváranie účtov a prístupových tokenov API.

Etcd

Ukladá informácie o konfigurácii pre veľké distribuovane systémy Kubernetes teda používa etcd ako úložisko hodnôt jednotlivých kľúčov. Etcd modul musí byt stále dostupný, aby sa zabezpečilo správne fungovanie služieb. Údaje Etcd sú veľmi doležíte a odporúča sa vytvoriť si zálohu.

Plánovač

dopisatt

Komponent, ktorý sa nasadzuje ako prvý je takzvaný runtime kontajnera. Zvyčajne sa inštaluje spustením Dockera, ale dostupné sú aj alternatívy ako rkt a runc. Runtime kontajnera je zodpovedný za spustenie a správu kontajnerov, aplikácií zapuzdrených v relatívne izolovanom, ale ľahkom ako keby operačnom prostredí.

Každá jednotka práce v klastri je na svojej základnej úrovni implementovaná ako jeden alebo viac kontajnerov, ktoré je potrebné nasadiť.

Kubelet

Kubelet je agent, ktorý je spustený na každom pracovnom uzle v klastri. Je to dôležitý komponent, pretože prijíma inštrukcie z hlavného uzla. Kubelet v podstate riadi pody. Zabezpečuje, že všetky kontajnery bežia v pode a že tieto pody sú v poriadku a bežia v správnych časových intervaloch. Číže vytvára a odstraňuje moduly, na základe pokynov od hlavného uzla, ktoré moduly je potrebné pridať alebo odstrániť. Keď riadiaci uzol potrebuje, aby sa niečo vykonalo v uzle, kubelet vykoná tuto akciu.

Kube-proxy

Na správu jednotlivých hostiteľských podsietí a sprístupnenie služieb iným komponentom je na každom uzlovom serveri spustená malá proxy služba s názvom kube-proxy . Tento proces preposiela požiadavky správnym kontajnerom, môže vykonávať primitívne vyvažovanie záťaže a je vo všeobecnosti zodpovedný za zabezpečenie toho, aby bolo sieťové prostredie predvídateľné a dostupné, ale v prípade potreby izolované. Používa proxy UDP, TCP a SCTP, ale nerozumie HTTP.

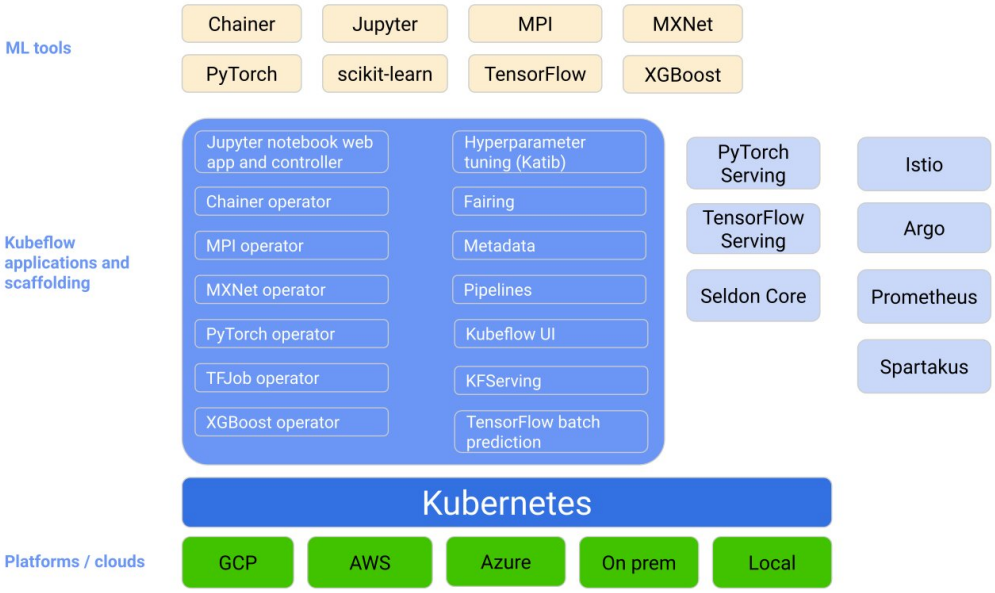
1.2 Kubeflow

Kubeflow ako platforma, je vhodná na nasadenie a vývoj strojového učenia. Primárne slúži pre inžinierov a vedcov, ktorý pracujú s dátami. Obsahuje viaceré komponenty, z ktorých si vývojári môžu vybrať, čo je pre ich používateľov najlepšie, čo znamená, že na nasadzovanie nie je potrebný každý jeden komponent.[3]

1.2.1 Architektúra

Stavia na platforme Kubernetes ako systéme na nasadenie, škálovanie a správu zložitých systémov. Pomocou konfiguračných rozhraní Kubeflow, môžeme špecifikovať nástroje strojového učenia, potrebné pre náš pracovný postup. Potom môžeme nasadiť pracovný postup do rôznych cloudov, miestnych platforiem na experimentovanie a na produkčné použitie.[3]

Na nasledujúcom obrázku môžte vidieť architektúru Kubeflow.



Obr. 1.1: Architektúra kubeflow

1.2.2 Pracovný postup

Postup pozostáva z viacerých krokov, taktiež z iterácie, ktorá je hlavným prvkom pri vyvíjaní sýtemu strojového učenia. Pri tomto postupe je potrebné vykonávať zmeny v parametroch aby sme dosiahli požadované výsledky.

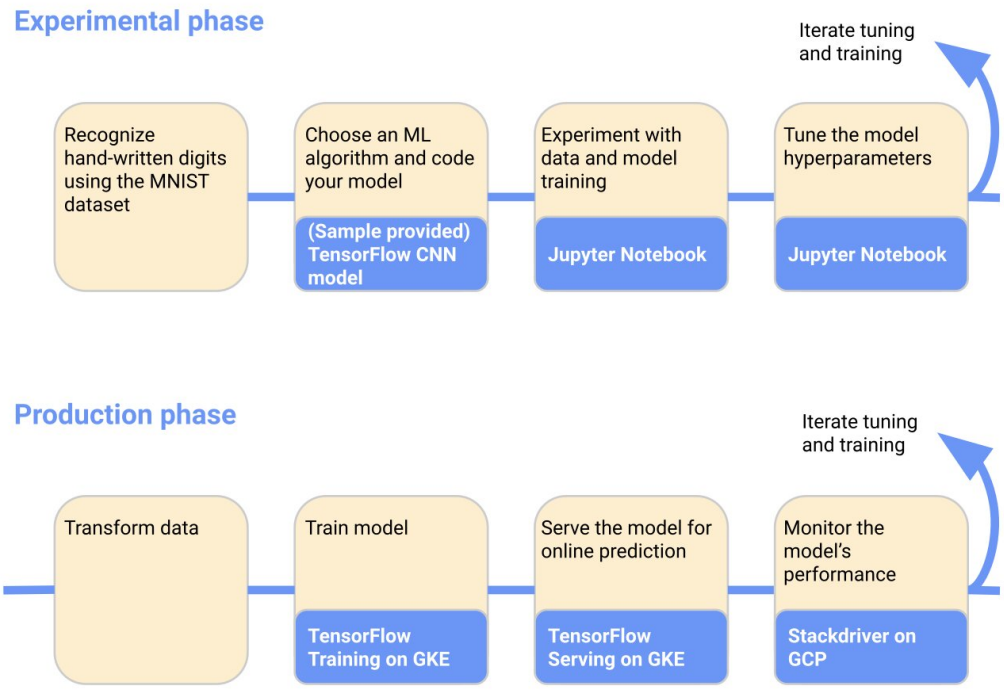
Následne si povieme niečo viac o týchto postupoch.

Ako prvé by sme mali vyvíjať model na základe predpokladov a testov. Môžeme ho opísať v nasledujúcich bodoch:[3]

- Identifikovanie problému, ktorý ma systém vyriešiť
- Zbieranie dát, ktoré potrebujeme na trénovanie modelu
- Vyberanie algoritmu a nakódovanie modelu
- Experiment s údajmi a trénovanie modelu
- Vyladenie parametrov

Ďalej môžeme nasadiť systém, ktorý bude vykonávať tieto procesy:

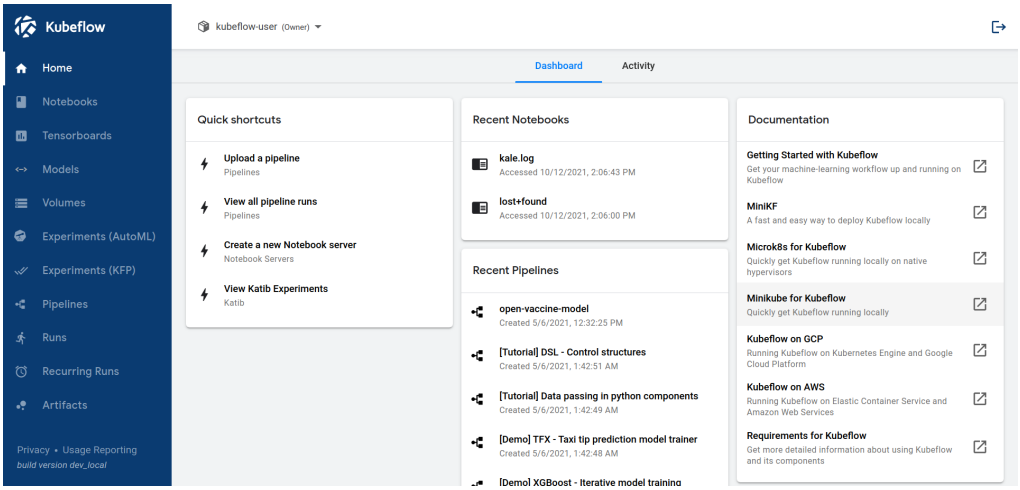
- Transformáciu údajov, ktoré náš systém potrebuje
- Trénovanie modelu
- Podanie modelu na online prevádzku
- Monitorovanie výsledkov na úpravu a zmenu modelu



Obr. 1.2: Pracovné postupy

1.2.3 Rozhrania

dopisaaat neskoor



Obr. 1.3: Rozhranie

- v knihe [4] autor prezentuje naozaj odvážne myšlienky
- nemenej zaujímavé výsledky publikuje ďalší autor v článku [5]
- v konferenčnom príspevku [6] sú uvedené tiež zaujímavé veci

- LaTeX[1] je typografický jazyk

Given a set of numbers, there are elementary methods to compute its Greatest Common Divisor, which is abbreviated GCD. This process is similar to that used for the Least Common Multiple (LCM).

1.2.4 Donec vehicula consequat

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisissem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi necante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultriciesvel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero utmetus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit ametante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

1.2.5 Nullam in mauris consectetur

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisissem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi necante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultriciesvel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero utmetus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit ametante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Zdrojový kód 1.1: Program, ktorý pozdraví celý svet

```
#include <stdio.h>
int main() {
    /* Print Hello, World! */
    printf("Hello, World!\n");
    return 0;
```

[1] https://www.latex-project.org/

```
}
```

1.2.6 Vestibulum tristique elementum varius

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisissem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi necante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultriciesvel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero utmetus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit ametante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Tabuľka 1.1: Country list

Country List			
Country Name or Area Name	ISO ALPHA 2 Code	ISO ALPHA 3 Code	ISO numeric Code
Afghanistan	AF	AFG	004
Aland Islands	AX	ALA	248
Albania	AL	ALB	008
Algeria	DZ	DZA	012
American Samoa	AS	ASM	016
Andorra	AD	AND	020
Angola	AO	AGO	024

1.3 Phasellus id pretium neque

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisissem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi necante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultriciesvel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero

utmetus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit ametante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisissem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi necante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultriciesvel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero utmetus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit ametante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

2 Syntetická časť

Syntetická časť opisuje metódy použité na syntézu riešenia a opisuje syntézu samotného riešenia (zvyčajne je to návrh/implementácia softvérového resp. hardvérového riešenia), pričom sa opiera o závery analytickej časti práce. Začína od toho, ako sa bude riešenie používať: najdôležitejšie scenáre používania a používateľské rozhranie, ktoré bude tieto scenáre efektívne podporovať. Až potom je na rade vnútorná architektúra alebo použité technológie. Syntetická časť tvorí zvyčajne ½ jadra práce.

Syntetickú časť práce vhodne rozdeľte do kapitol a pomenujte ich podľa toho, čomu sú venované.

3 Vyhodnotenie

Vyhodnocovacia časť je kľúčovou časťou záverečnej práce. Tato časť obsahuje vyhodnotenie navrhnutého (vytvoreného) riešenia. Uprednostňované je objektívne vyhodnotenie výsledkov práce, ktoré sa opiera o meranie a štatistické metódy, prípadne matematické dôkazy. V prípade nameraných hodnôt musí autor opísať metódu merania, priebeh merania, výsledky a interpretáciu výsledkov v kontexte riešeného problému a stanovených cieľov. Na základe vyhodnotenia riešenia autor opíše prínosy svojej práce. Vyhodnocovacia časť tvorí zvyčajne ¼ jadra práce.

4 Záver

Záver práce obsahuje zhrnutie výsledkov práce s jasným opisom prínosov a pôvodných (vlastných) výsledkov autora a vyhodnotenie splnenia stanovených cieľov. Je to stručné zhrnutie informácií uvedených v záverečnej práci. Záver by nemal obsahovať nové informácie.

V závere by mal tiež autor poukázať na prípadné otvorené otázky, ktoré sú nad rámec rozsahu práce a mal by odporučiť ďalšie aktivity na pokračovanie pri riešení problému. Rozsah záveru je minimálne 1 celá strana.

Literatúra

1. WWW.DOCS.BYTEMARK.CO.UK. *https://docs.bytemark.co.uk/article/kubernetes-terminology-glossary/the-basics*. 2021.
2. WWW.KUBERNETES.IO. *https://kubernetes.io/docs/concepts/overview/what-is-kubernetes*. 2021.
3. WWW.KUBEFLOW.ORG. *https://www.kubeflow.org/docs/started/kubeflow-overview*. 2021.
4. BABINGTON, Peter. *The title of the work*. 3. vyd. The address: The name of the publisher, 1993. 10. ISBN 3257227892. An optional note.
5. ADAMS, Peter. The title of the work. *The name of the journal*. 1993, roč. 4, č. 2, s. 201–213. An optional note.
6. DRAPER, Peter. The title of the work. In: EDITOR, The (ed.). *The title of the book*. The address of the publisher: The publisher, 1993, zv. 4, s. 213. 5. An optional note.

Zoznam skratiek

GCD Greatest Common Divisor.

LCM Least Common Multiple.

Zoznam príloh

Príloha A Karel Language Reference

Príloha B CD médium – záverečná práca v elektronickej podobe,

Príloha C Používateľská príručka

Príloha D Systémová príručka

A Karel Language Reference

Karel's Primitives

- `void movek()` - Moves *Karel* one intersection forward.
- `void turn_left()` - Pivots *Karel* 90 degrees left.
- `void pick_beeper()` - Takes a beeper from the current intersection and puts it in the beeper bag.
- `void put_beeper()` - Takes a beeper from the beeper bag and puts it at the current intersection.
- `void turn_on(char* path)` - Turns *Karel* on.
- `void turn_off()` - Turns *Karel* off.

Karel's Sensors

- `int front_is_clear()` - Returns `1` if there is no wall directly in front of *Karel*. `0` if there is.
- `int right_is_clear()` - Returns `1` if there is no wall immediately to *Karel*'s right. `0` if there is.
- `int beepers_present()` - Returns `1` if *Karel* is standing at an intersection that has a beeper. `0` otherwise.
- `int facing_north()` - Returns `1` if *Karel* is facing north. `0` otherwise.
- `int beepers_in_bag()` - Returns `1` if there is at least one beeper in *Karel*'s beeper bag. `0` if the beeper bag is empty.

Misc Functions

- `void set_step_delay(int)` - Sets delay of one *Karel*'s step in miliseconds.
- `loop(int)` - Repeats *Karel*'s instruction in a loop.