

**Meno: Róbert Šišaj**

## **Ročníkový projekt II – špecifikácia**

Program na využitie voľnej výpočtovej kapacity LAN pri paralelne spracovateľných úlohách.

**Popis:** Program je určený pre menší počet počítačov, predpoklad cca do 20. Malo by to byť klient-server aplikácia. Užívateľ zadá serveru úlohu na spracovanie, ten úlohu rozdrobí a rozošle klientom na spracovanie, prijme dielčie výsledky a skompletizuje finálny výsledok. Úloha nutne musí byť paralelizovateľná, inak spracovanie nemá žiadny pozitívny efekt. Predpokladá sa úloha skladajúca sa z menších podúloh. Ideálne by tieto podúlohy mali byť také, že na menší objem dát pripadá viac výpočtových nárokov, aby program nezdegeneroval na niečo, čo viac-menej nezmyselne posieľa údaje po sieti sem a tam. (Príklad: výpočtovo náročnejšia úprava sady fotiek, rendrovanie sekvencie obrázkov...)

OS: Windows (XP, 2000) | (Linux?)

Programovací jazyk: C++

Snaha o UML popis pomocou Use Case:

### **Server program:**

Meno: **Nastavenie aplikácie**

Číslo Use Case: UC1

Cieľ: Užívateľ si môže nastaviť konfiguráciu aplikácie podľa jeho prania.

Aktér: Užívateľ

Kontext: Užívateľ si praje zmeniť niektoré z nastavení aplikácie (napr. číslo portu, na ktorom bude server prijímať údaje od klienta, prípadne použitý algoritmus na využívanie okolitých počítačov, ...)

Štandardný postup:

1. Užívateľ sa rozhodne zmeniť konfiguráciu aplikácie.
2. Užívateľ nájde v menu príslušnú položku, ktorú treba zmeniť, a prevedie zmenu.
3. Užívateľ potvrdí zmenu tlačítkom, alebo ok s následným zatvorením preferences.
4. Systém podľa nových hodnôt nastaví príslušné parametre aplikácie.
5. Užívateľ ukončí prácu v preferences ich uzatvorením.

Meno: **Edituj úlohy**

Číslo Use Case: UC2

Cieľ: Užívateľ zadáva a odoberá úlohy, ktoré chce vykonať.

Aktér: Užívateľ

Kontext: Užívateľ si vytvorí buď nový zoznam úloh alebo načíta zoznam úloh zo súboru. Ďalej užívateľ pridáva, upravuje, alebo maže úlohy, ktoré majú byť spracované. Po dokončení týchto úprav môže spustiť samotný výpočet a obdržať výsledok alebo si zoznam úloh uložiť do súboru.

Štandardný postup:

1. Užívateľ si vytvorí nový projekt alebo si otvorí nejaký už existujúci.
2. Systém načíta data potrebné pre nový, príp. otvorený projekt
3. Užívateľ **nastavuje premenné** (UC21) potrebné pre ďalšie výpočty, ako použitý algoritmus a pod.
4. Užívateľ pracuje na **editácii úloh** (UC22) ktoré chce spracovať, napr. aká úprava obrázkov sa má previesť a pod.
5. Užívateľ spustí výpočet na pripravené data z bodu 4. a čaká na výsledok. Stav úloh a okolitých počítačov je priebežne zobrazovaný, užívateľ má plnú kontrolu nad celou aplikáciou a môže kedykoľvek spracovanie prerušiť.
6. Systém prevedie výpočet a dôjde k **oznámeniu výsledkov** (UC23).

Alternatíva 1:

3. Ak užívateľ nenastaví premenné, budú použité defaultné hodnoty.

Meno: **Nastavenie premenných**

Číslo Use Case: UC21

Cieľ: Nastavenie premenných, ktoré ovplyvňujú výpočet pre daný projekt.

Aktér: Užívateľ

Kontext: Užívateľ po začatí práce s projektom môže nastaviť niektoré premenné, ktoré ovplyvňujú výpočet. Ak ich nenastaví, budú sa brať do úvahy defaultne nastavené hodnoty.

Štandardný postup:

1. Užívateľ nastaví podľa potrieb premenné pre editovaný projekt.

Poznámka:

Zmeny niektorých premenných nie sú na prvý pohľad viditeľné, ale pri výpočte výsledkov môžu hrať svoju rolu. (napr. premenná určujúca použitý algoritmus na výber počítača a pod).

Meno: **Editácia úloh**

Číslo Use Case: UC22

Cieľ: Užívateľ vyberá úlohy, ktoré chce spracovať, a tiež, akú operáciu chce s datami previesť.

Aktér: Užívateľ

Kontext: Užívateľ v tomto kroku špecifikuje data, s ktorými sa bude prevádzkať výpočet. Môže pridávať nové úlohy, zmeniť, čo sa má vykonať s už pridanými úlohami, alebo mazať niektoré úlohy. Vždy má pred sebou kompletný zoznam úloh, ktoré projekt obsahuje.

Štandardný postup:

Pre pridávanie úloh:

1. Užívateľ vyberie voľbu pridať ďalšiu úlohu a vyplní potrebné údaje.
2. Systém zaznamená nové data.

Pre editáciu úloh:

1. Užívateľ vyberie danú úlohu v zozname, pre ktorú chce previesť zmenu a zmení data alebo operáciu, ktorú chce s nimi uplatniť.
2. Systém zaznamená nové hodnoty.

Pre mazanie úloh:

1. Užívateľ vyberie úlohu zo zoznamu, ktorú chce odstrániť.
2. Užívateľ vyberie voľbu odstrániť a potvrdí zmenu.
3. Systém odstráni príslušný riadok zo zoznamu a k nemu odpovedajúce data.

Meno: **Oznámenie výsledkov**

Číslo Use Case: UC23

Cieľ: Užívateľ spustí vypracovanie úloh a obdrží výsledky.

Aktér: Užívateľ

Kontext: Užívateľ po dokončení editácie projektu pošle pripravené data na spracovanie.

Vtedy sa spustí algoritmus, ktorý posiela okolitým počítačom v sieti úlohy a prijíma od nich výsledky. Na konci upozorní užívateľa, že práca skončila. Užívateľ má počas výpočtu možnosť kedykoľvek prerušiť výpočet.

Štandardný postup:

1. Užívateľ spustí vypracovanie úloh, systém
2. Systém vezme data, ktoré boli zadané pri editácii úloh a spracuje ich.  
**(výpočet výsledku (UC231))**
3. Systém po skončení práce zobrazí údaje o priebehu spracovania a stav výsledku pre užívateľa.

Meno: **Výpočet výsledkov**

Číslo Use Case: UC231

Cieľ: Je spustený algoritmus, ktorý postupne rozosiela úlohy spolu s potrebnými dátami klientom a prijíma od nich výsledky.

Akter: Systém

Kontext: Algoritmus je spustený na zadané data, jeho cieľ je čo možno najoptimálnejšie a časovo najefektívnejšie získať požadovaný výsledok. Jednotlivé úlohy posiela okolitým počítačom v sieti a prijíma od nich výsledky. Aplikácia priebežne informuje užívateľa o celkovom stave výpočtu, aktualizuje si údaje o sieti a vyrovnáva sa so zmenami.

Štandardný postup:

1. Systém vezme data a pošle ich na spracovanie algoritmu.

Poznámka:

Nie je možné zaručiť, že výpočet prebehne (napríklad ak v sieti nebude aktívny žiadny klient, prípadne pri úplnom výpadku siete). Jediný spôsob, ako zaručiť spracovanie úloh aj pri týchto extrémnych podmienkach, je implementovať spracovanie úloh aj do Server programu. Alternatívne by sa mohol pri výpadku posledného klienta výpočet zastaviť alebo by výpočet začal bežať priamo na serveri až do doby, kým sa ohlási nový klient alebo kým výpočet neskončí. Algoritmus by nemal viesť k zbytočnému zaťažovaniu siete, v ktorej má prebiehať výpočet.

## **Klient program:**

Meno: **Nastavenie aplikácie**

Číslo Use Case: UC1

Cieľ: Užívateľ si môže nastaviť konfiguráciu aplikácie podľa jeho prania.

Aktér: Užívateľ

Kontext: Užívateľ si praje zmeniť niektoré z nastavení aplikácie (napr. číslo portu, na ktorom bude prijímať údaje od serveru, prioritu klientského programu, ...)

Štandardný postup:

1. Užívateľ sa rozhodne zmeniť konfiguráciu aplikácie.
2. Užívateľ nájde v menu príslušnú položku, ktorú treba zmeniť, a prevedie zmenu.
3. Užívateľ potvrdí zmenu tlačítkom, alebo ok s následným zatvorením preferencí.
4. Systém podľa nových hodnôt nastaví príslušné parametre aplikácie.
5. Užívateľ ukončí prácu v preferencí ich uzatvorením.

Meno: **Zobrazenie informácií**

Číslo Use Case: UC2

Cieľ: Informovať užívateľa o stave aplikácie.

Akter: Užívateľ

Kontext: Užívateľ požaduje zobraziť informácie o stave aplikácie, aplikácia zobrazí údaje (počet spracovaných úloh, čas práce, ...).

Štandardný postup:

1. Užívateľ vyberie možnosť zobraziť informácie o stave aplikácie.
2. Aplikácia spracuje údaje a zobrazí výstup.

Poznámka:

Užívateľ má možnosť kedykoľvek ukončiť spoluprácu so serverom a prestať sa podieľať na výpočte. Aplikácia poskytuje aspoň 2 spôsoby ukončenia: okamžité prerušenie výpočtu a ukončenie aplikácie alebo ukončenie aktuálnej úlohy a následné ukončenie.

Klientský program beží viac-menej samostatne, užívateľ sa oň nemusí starať. Naivná predstava: Užívateľ pri odchode z pracoviska spustí klientský program a odíde. Klientský program teda musí sám komunikovať so serverom a nemal by k tomu potrebovať užívateľa.

## **Komunikácia medzi Klientom a Serverom:**

Obidva programy hneď po spustení otvoria svoj socket na príjem. Klientský program priebežne oslovuje server (po nejakom dlhšom časovom intervale) správou, že chce spolupracovať (musí byť, nemožno predpokladať, ako dlho potrvá klientovi spracovanie 1 úlohy). Server si upraví príslušný záznam. Klient oslovuje buď známy server (načíta údaj z konfiguračného súboru) alebo použije broadcast (iba kým nedostane odpoveď od serveru). Použitie broadcast vysielania obmedzuje počet serverov v broadcast doméne najviac na 1 bežiaci server.

Server po spustení algoritmu na rozdeľovanie úloh vyberie nejaký stroj, pošle mu úlohu a zároveň si poznačí, že klient dostal pridelenú prácu. Klient prijme úlohu, spracuje a odošle serveru výsledok. Server prijme výsledok a upraví si záznam pri danom klientovi, že už nepracuje.

Užívateľ pri vypínaní klienta má na výber, či chce dokončiť aktuálnu úlohu a skončiť až po odoslaní výsledku (klient vtedy pracuje stále rovnako, akurát si poznačí, že spolu s odoslaním výsledku musí serveru odoslať aj príkaz o ukončení, aby už nedostal novú úlohu) alebo či skončí hneď (pošle sa iba oznámenie o ukončení, žiadny výsledok sa neposiela). Server sa musí vedieť vysporiadať aj s neštandardným výpadkom klienta.

### **Rozhranie Klient-aplikácia:**

Jednoduchšia varianta: Projekt bude podporovať len úlohy pre 1 aplikáciu, nepovolí miešať úlohy určené rôznym aplikáciám. Projekt sa vytvorí vlastne v aplikácii, potom sa zavolá server (Príklad - predstava: V aplikácii vyberiem 20 obrázkov, ktorým chcem zmeniť veľkosť, zavolám funkciou server a následne mu ich pošlem ako data. Potom vyberiem ďalších 15 obrázkov, ktoré chcem previesť do čiernobieleho formátu, takisto odošlem serveru ako dáta. Server by mal teraz obsahovať projekt zložený z 35 úloh.). Užívateľ môže prípadne niektoré úlohy odstrániť a potom spustiť vykonávanie. Server rozdeľuje úlohy z projektu jednotlivým klientom. Klient obdrží úlohu a zavolá aplikáciu, že má pre ňu dáta. Aplikácia si môže zistiť informáciu o dátach, prípadne vyzdvihnúť dáta od klienta a pracovať s nimi. Potom opäť zavolá klienta a predá mu výsledok. Klient vráti výsledok serveru a ten ho uloží.

Zložitejšia varianta: Projekt bude podporovať ľubovoľné úlohy. Projekt sa vytvorí postupne v aplikácii/aplikáciách, potom sa zavolá server, ďalej všetko obdobne ako pri jednoduchšej variante.

Otázka: Čo ak klient nemá nainštalovaný program, ktorým boli vytvorené dáta pre server?

- Možnosti:
1. Klient pri prihlasovaní pošle svoj zoznam aplikácií, ktoré má nainštalované.
  2. Adaptívne: Server pošle klientovi úlohu určitého typu, klient ju môže odmietnuť s tým, že ju nedokáže spracovať. Server si poznačí ku klientovi, že nepodporuje tento typ úloh.

Otázka: Má klient predpokladať, že je aplikácia spustená?

- Možnosti:
1. Áno - vhodné pre veľmi zmiešané úlohy. V pamäti je natiiahnutých viac programov, klient vždy vyberie jeden, ktorému pošle dáta, počká na odpoveď a tú odošle späť na server.
  2. Nie:
    1. Klient sa pozrie, či je otvorená aplikácia, ktorú potrebuje, ak nie je, tak ju otvorí a pošle jej úlohu.
    2. Klient sa pozrie, či je otvorená aplikácia, ktorú potrebuje, ak nie je, zatvorí súčasnú aplikáciu (ak bola otvorená) a otvorí správnu. Vhodnejšie pre systémy, kde nie je dostatok voľnej pamäte.

3. Klienti sa postupne špecializujú. Server si nejako rozdelí úlohy podľa toho, aké aplikácie vyžadujú. Pošle prvú úlohu prvému klientovi, ak ten prijme a vráti výsledok, server sa mu snaží posielat' len úlohy rovnakého typu. V prípade, že má úlohu, na ktorú nemá špecialistu, sa snaží vyčerpať ostatné úlohy. Ak už stoja nejakí „špecialisti“, skúša im posielat' úlohy nového typu - klienti menia špecializáciu.

Preferovaná bude jednoduchšia varianta, predpoklad je, že počítače v sieti sú určené na kancelársku činnosť a teda nemajú nadbytok voľných prostriedkov. Otváranie a zatváranie aplikácií môže byť zdlhavesšie ako samotné spracovanie úlohy a to je nežiadúce.

Funkcie rozhrania:

volané z aplikácie:

- na posielanie dát z aplikácie
- na výber dát pre aplikáciu
- na zisťovanie informácií o dátach, či sú prítomné, od akej aplikácie sú a

podobne

volané z klienta:

- riadiace funkcie? oznám tej a tej aplikácii, že tu má dáta