Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Semestrální práce z KIV/OS

Semestrální práce - 1. Single-task výpočet

Jakub Schenk

A21N0069P

schenkj@students.zcu.cz

17. prosince 2022

Obsah

1 Zadání

V rámci jednoho tasku implementujte výpočet, který bude prováděn na nízkopříkonovém zařízení, konkrétně Raspberry Pi Zero. Po spuštění jsou veškeré požadované výstupy vypisovány na UART a veškeré vstupy jsou z UARTu čteny. Task přijme dva parametry delta a prediction, následně jsou přijímána desetinná čísla, která jsou využita při výpočtu, který by měl vypadat následně:

- Task provede po každém přijatém čísle fitting modelu $y(t + t\_pred) = A * b(t) + B * b(t) * (b(t) - y(t)) + C$, kde b(t) spočítáte jako: $b(t) = D/E * dy(t)/dt + 1/E * y(t)$
- A,B,C,D,E jsou parametry modelu (modelů)
- dy(t)/dt je v sice derivace, vzhledem k povaze hodnot stačí aproximovat diferencí dělenou číslem 1.0/(24.0*60.0*60,0), je-li t v sekundách
- fakticky jde o zjednodušené modely pro predikci hladin glukózy v intersticiální tekutině (zde poslouží pouze jako aplikační výpočet pro demonstraci)
- můžete použít metodu nejmenších čtverců, nebo za bonusové body implementovat jednoduchý genetický/evoluční algoritmus
- chcete-li minimální přesnost G/EA, inicializujte až polovinu jeho populace s předchozími výsledky získanými pomocí nejmenších čtverců (proč, viz příští rok na KIV/PPR)
- volitelně můžete na UART vypisovat průběh výpočtu
- jako odpověď na přijaté číslo task vypíše novou predikci y(t + t_pred) dle nově spočtených parametrů, popř. řetězec NaN, pokud ještě není možné predikovat
- UART terminál musí být během výpočtu rozumně responzivní
- kdykoliv probíhá výpočet, musí ho být možné zastavit zadáním příkazu "stop"(bez uvozovek)
- pokud je výpočet ukončen, predikce je spočtena dle starých parametrů
- pokud použijete metodu, která by byla výrazně rychlá, uměle ji zpomalte a zpomalení vhodně zdokumentujte - abychom mohli testovat responzivitu UART kanálu během výpočtu

2 Analytická část

Jelikož jsme dostali k dispozici nemalou část systému, abychom nezačínali úplně od začátku, stala se úloha sice velice náročnou, ale řešitelnou. Za nejkritičtější body práce bych označil "práci s cizím kódem" a "low level programování".

Důležité části systému, které potřebovaly implementovat byly hlavně halda pro uživatelský proces, čtení z UARTU a samotný uživatelský proces (a ten přidat do systému).

Možná ještě náročnější bylo pochopení a analýza celého systému.

2.1 Halda

Pro implementaci haldy je možné využt mechanismů v obdržené kostře, která spravuje kernel haldu. Tím se můžeme vyhnout implementace celého mechanismu. Co by mělo být změněno je alokace stránek, resp. mechanismus mapování paměti, aby byl y odděleny adresní prostory pro ukládání uživatelských procesů a procesů jádra. Pro tento typ úlohy se ale asi můžeme spokojit s jednodušší variantou, kdy budeme vše fyzicky mapovat do jednoho prostoru a tedy nemusíme vymýšlet nějaký složitější mapování nebo alokaci.

2.2 Čtení

Pro čtení z UARTu je potřeba obstarat nějakou paměť kam se budou přečtené hodnoty ukládat. Touto pamětí by měl nejlépe být nějaký jednoduchý mechanismus, který bude mít omezenou velikost, ale nebude hrozit jeho přetečení. Z tohoto důvodu bylo na cvičeních doporučováno využití kruhového bufferu, jakožto nejjednodušší a nejpraktičtější komponenty, která se pro tento typ úlohy hodí.

Samotné čtení pak může být implementováno podobně jako je zápis v kostře systému. Jedinou překážkou by pak bylo blokované čtení kvůli kolizím v paměti.

Pro praktické účely je vhodné implementovat i nějaké podpůrné funkce, pro snadnější čtení než aktivní čekání a čtení po znaku. Sem spadá například načítání do bufferu dokud není nalezen znak ukončující řetězec, nebo čtení pouze po stisknutí klávesy pomocí přerušení.

2.3 Uživatelský proces

Pro práci s uživatelským vstupem bude potřeba nejen jeho ošetření, ale hlavně metody pro jeho manipulaci. Tím je myšlen převod řetězce na číslo

a obráceně, získání délky řetězce, čtení po řádcích apod. Část těchto metod bude pravděpodobně využita už dříve v kódu systému, ale část bude nutná implementovat.

Samotný výpočet může být prováděn dvěma způsoby. Jedná se o výpočet pomocí nejmenších čtverců a regresi nebo genetický algoritmus. Pro výpočet pomocí regrese funkce a nejmenších čtverců je potřeba mechanismus pro práci s maticemi a jedná se o náročnější, ale rychlejší výpočet. V případě genetického algoritmu je potřeba generovat pseudonáhodná čísla a implementovat mechanismus pro hodnocení populace a evoluci generací. Tento přístup je sice pomalejší, ale není tak algoritmicky náročný.

3 Implementace systému

3.1 Halda

Při implementaci uživatelské haldy jsem se inspiroval implementací haldy pro kernel. Hlavně jsem se zaměřil na poznámky ze cvičení, na kterém otázka ohledně implementace tohoto problému byla podrobně zodpovězena. Halda tedy využívá sbrk a jinak podobné mechanismy jako halda pro kernel. Do tohoto jsou zahrnuty i funkce volané kernel haldou. Výsledkem je, že je uživatelská halda fyzicky alokovaná stejným stránkovačem jako kernelová a tedy spolu sdílejí fyzický prostor. Stránky jsou tedy sice alokovány ve stejné fyzické oblasti, ale kvůli povaze úlohy jsem to nepovažoval za omezující. Co se liší je přiřazená virtuální adresa, ale to už není tak implementačně zajímavé.

3.2 Kruhový zásobník

Jako buffer jsem zvolil kruhový zásobník a při jeho implementaci jsem vycházel z několika internetových zdrojů a povědomí o jeho obecné funkčnosti jsem získal absolvováním jiných předmětů. Součástí jeho implementace je několik funkcí pro čtení a několik pro zápis. Sem patří přečtení celého bufferu, přečtení jeho části dle zadané délky, nebo přečtení části končící daným znakem. Pro zápis se jedná o zápis jednoho znaku a zápis sekvence znaků dané délky.

3.3 Čtení z UART

Čtení z UARTu jsem implementoval velice podobně tomu, jak byl implementován zápis, s tím rozdílem že se jedná o blokující operaci.

Součástí implementace je i readUtils, což je obalová třída pro metodu, která přečte celý řádek - readLine. Ta do obdrženého bufferu nahraje přečtený

text až po znak značící jeho konec.

Pro práci s obdrženým vstupem je součástí systému stdstring, který obsahuje základní zpracování uživatelského vstupu a textu obecně, jako například převody na číslo apod.

4 Uživatelský proces

Uživatelský proces začne vypsáním základních informací a instrukcí do UARTu. Po validaci a přijmutí parametrů a několika hodnot, započne výpočet pomocí genetického algoritmu.

4.1 Inicializace a nahrazení slabší části populace

Pro první výpočet je vygenerována množina náhodných členů populace. Po každém dalším výpočtu zůstane jen lepší část populace a horší je opět inicializována náhodně. Jak velká část je také rozhodnuto náhodně zvolením náhodného člena populace. Vhodnost tohoto člena je brána jako krajní a horší členové populace jsou nahrazeny náhodně inicializovanými novými členy. Během tohoto procesu je vybrán člen populace s nejlepšími výsledky.

4.2 Křížení a mutace

Po tomto následuje křížení populace a to tak, že je náhodně pro každého člena populace vybráno jak velká část bude křížena s náhodným jiným členem populace.

Posledním krokem je mutace. S určitou pravděpodobností člen zmutuje a to tak, že je nahrazen náhodný z jeho parametrů náhodnou hodnotou.

4.3 Vyhodnocení a ukončení

Na konci je vyhodnocen nejlepší člen ze všech generací (jeho podoba před křížením a mutací) a je vypsán jako výsledek výpočtu. K tomu je využita funkce popsána v zadání. Výsledek je porovnán s jeho predikcí a podle odlišnosti obou výsledků je získána hodnota reprezentující jak moc se od sebe predikce a výsledek liší. Člen populace s takto vypočtenou nejnižší hodnotou je označen za výsledek.

Pokud procesu přijde hodnota jiná než "stop", "parameters"nebo číslo, program vstup ignoruje. Pokud obdrží "stop"je program ukončen. Tato záležitost nebyla dostatečně dobře definována, takže jsem nevěděl, jestli se má program ukončit, nebo jen pozastavit výpočet. Rozhodl jsem se pro ukončení

celého procesu, protože by jinak program nemohl být nenásilně ukončen. Aby "stop" pouze pozastavilo výpočet je nutné zasáhnout do kódu (poměrně malá změna).

5 Problémy

Krom analýzy problému a dostupného řešení, se většina problémů vyskytla mou nepozorností, kdy jsem zapomněl vyměnit magickou konstantu za parametr, nebo během debugování zakomentoval část kódu a zapomněl ji odkomentovat. Kdybych pak měl vybrat jeden z nich, tak by se jednalo o zdlouhavé debugování čtení z UART, kdy první čtení fungovalo a druhé už ne, i když byly části kódu identické. Chyba byla v readLine, kde zůstala magická konstanta v podmínce z testování funkce.

Jedním z větších nějak rozumně popsatelných problémů bylo pochopení algoritmu pro uživatelský proces. To naštěstí bylo lépe vysvětleno na cvičení, takže problém pak už nebylo tak těžké opravit.

Druhý z větších problémů nastal při implementaci čtení z UART, kdy jsem se snažil reimplementovat i část pro čtení na blokující verzi, ale nakonec jsem tuto implementaci zavrhl.

6 Testování

Naneštěstí nezbýval dostatek času pro rozsáhlejší testování, nebo automatické testy. Testování tedy proběhlo pouze ruční.

Během testování byla objevena chyba, při které program přestane fungovat, pokud je napsána první část klíčového slova "stop"před získáním výsledků, a druhá část je napsána a odeslána až poté. V tuto chvíli je v bufferu klíčové slovo "stop"a program se dostal do neočekávaného stavu ze kterého byl ukončen. Naštěstí výsledek této chyby byl stejný jako očekávaný výsledek programu, takže mu z důvodu časové tísně nebyla věnována větší pozornost.

Program byl testován pouze na osobním počítači v emulátoru, nikoliv na fyzickém zařízení.

7 Závěr

Věřím že odevzdaná práce splňuje požadovanou funkcionalitu, ale že by nebylo příliš složité běh aplikace narušit a rozbít. Aplikace sice nebyla vyzkoušena na fyzickém zařízení ale na emulátoru (alespoň u mě) funguje bez větších problémů.

Samotné zadání mi přijde poměrně náročné (a to i v porovnání s ostatními pracemi, které jsou považovány za přehnaně náročné). Zjednodušením by však došlo k degradaci úlohy pouze na jedno - doplnění systému a ozkoušení na "Hello world"programu, nebo vyvinutí pouze uživatelského procesu a jeho zprovoznění na již hotovém systému.