FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

KIV/PPR

Semestrální Práce

Jméno a přijmení: Martin Forejt
Osobní číslo: A20N0079P

Datum: 28. 11. 2021

1 Zadání

Program semestrální práce dostane, jako jeden z parametrů, zadaný souboru, přístupný pouze pro čtení. Bude ho interpretovat jako čísla v plovoucí čárce - 64-bitový double. Program najde číslo na arbitrárně zadaném percentilu, další z parametrů, a vypíše první a poslední pozici v souboru, na které se toto číslo nachází.

Program se bude spouštět následovně:

```
pprsolver.exe soubor percentil procesor
```

- soubor - cesta k souboru, může být relativní k program.exe, ale i absolutní
- percentil - číslo 1 – 100
- procesor - řetězec určujíící, na jakém procesoru a jak výpočet proběhne
    - single - jednovláknový výpočet na CPU
    - SMP - vícevláknový výpočet na CPU
    - anebo název OpenCL zařízení - pozor, v systému může být několik OpenCL platforem
- Součástí programu bude watchdog vlákno, které bude hlídat správnou funkci programu

Další požadavky:

- Součástí programu bude watchdog vlákno, které bude hlídat správnou funkci pro-gramu.
- Testovaný soubor bude velký několik GB, ale paměť bude omezená na 250 MB. Tozařídí validátor.
- Program musí skončit do 15 minut na iCore7 Skylake.
- Program nebude mít povoleno vytvářet soubory na disku.
- Jako čísla budete uvažovat pouze ty 8-bytové sekvence, pro které `std::fpclassify` vrátí `FP_NORMAL` nebo `FP_ZERO`. Jiné sekvence budete ignorovat.
- Všechny nuly jsou si stejně rovné.
- Pozice v souboru vypisujte v bytech, tj. indexováno od nuly.
- Nalezené číslo číslo vypište v hexadecimálním formátu, např. pomocí `std::hexfloat`

2 Úvod

Cílem této práce je vytvoření programu, který nalezne hodnotu zadaného percentilu v souboru, který bude interpretovat jako sekvenci 64 bitových čísel v plovoucí čárce (double) a dále nalezne první a poslední výskyt této hodnoty v daném souboru. Prostředí pro běh programu má omezenou paměť na 250 MB a běh výpočtu nesmí běžet déle něž 15 minut. Algoritmus bude implementován třetmi různými způsoby: sériově, paralelně s použitím SMP a paralelně s použitím knihovny OpenCL.

3 Algoritmus

V literatuře lze najít více definicí toho co je to percentil, v této práci se za P-tý percentil ($1 \leq P \leq 100$) z množiny N seřazených hodnot považuje taková nejmenší hodnota z množiny N, kde ne více než P procent všech hodnot z množiny N je menší než tato hodnota a alespoň P procent hodnot je menší nebo rovno této hodnotě. Index i této hodnoty v seřazené množině získáme takto:

$$i = \frac{P}{100} \times N$$

Naivní algoritmus pro nalezení čísla ze souboru na daném percentilu P je jednoduchý. Stačí načíst všechny čísla (vyfiltrovat ta nevalidní), seřadit je a za výsledek prohlásit číslo na pozici i dle předchozí rovnice. Problémem tohoto řešení je omezená paměť pro načtení všech čísel ze souboru.

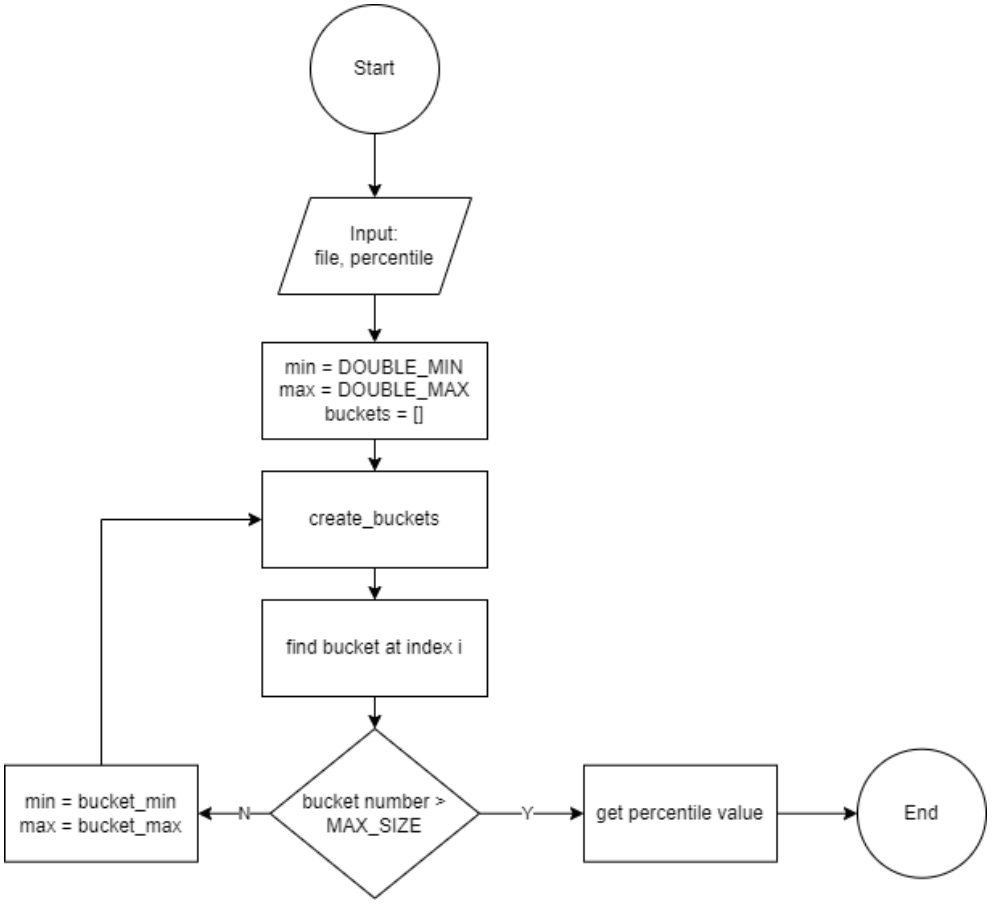
Implementovaný algoritmus je založen na tvorbě histogramu. V hlavní smyčce algoritmus provádí tři kroky:

1. Vytvoření histogramu (naplnění hodnotami)
2. Spočítání indexu i a nalezení bucketu histogramu na tomto indexu
3. Vytvoření definice „pod-histogramu“ z tohoto bucketu (min, max hodnoty)

Pokud je v kroku 3 v novém histogramu dostatečně málo hodnot (tak aby se všechny vešli do paměti), stačí je již všechny načíst do paměti, seřadit a získat hodnotu na zadaném percentilu. V opačném případě se pokračuje na krok 1 a tento nový histogram je naplněn hodnotami. Tato hlavní smyčka je také vidět na obrázku 1.

Pro reprezentaci histogramu a hodnot jednotlivých bucketů je použito pole, kde index pole představuje minimální hodnotu daného bucketu získanou operací bitového posunu viz dále a hodnota na daném indexu představuje počet hodnot v tomto bucketu.

Získání indexu (a nejmenší hodnoty daného bucketu) z 64 bitové doublu získáme pomocí bitového posunu vpravo. Počet posunutých bitů následně určuje „hustotu“ histogramu a počet a velikost jeho bucketů. Musíme si ovšem dát pozor na to, že první bit doublu určuje jeho znaménko a v první polovině histogramu budou kladné hodnoty a ve druhé polovině záporné. Ty budou ještě převráceně (od nejvyšší po nejnižší).



Obrázek 1 - Algoritmus

4 Implementace

Program je implementován v jazyce C++17, pro paralelní výpočet na SMP je použita knihovna Intel TBB, pro výpočet na GPU je použita knihovna OpenCL.

Po spuštění programu se nejprve zkontrolují parametry z příkazové řádky a spustí se výpočet pomocí funkce `run` ze souboru `percentile_finder.h`. Tato funkce obsahuje zmíněnou hlavní smyčku algoritmu a volá funkce ze souboru `bucketing.h` pro jeho jednotlivé kroky:

```
1. buckets = create_buckets(file, histogram)
2. bucket = find_bucket(buckets, histogram)
3. histogram.shrink(buckets, bucket_index, percentile_position)
```

Po ukončení hlavní smyčky algoritmu se již nejde hodnota na daném percentilu pomocí funkce `get_percentile_value(file, histogram)` a první a poslední pozice v souboru kde se tato hodnota nachází pomocí `get_value_positions(file, histogram, value)`.

5 Paralelizace

Funkce `create_buckets` je implementována třemi způsoby tzn. sériově, parallelně s použitím knihovny Intel TBB a paralelně s použitím OpenCL. Pomocí knihovny TBB jsou kromě jejich sériové verze implementovány i funkce `get_percentile_value` a `get_value_positions`. U těchto funkcí by nebyla implementace pomocí OpenCL efektivní.

5.1 SMP

V případě SMP paralelizace se využívá `tbb::parallel_pipeline`, která má vždy tři etapy. Při první se náčítají čísla ze souboru do bufferu, který je následně předán druhé etapě. Druhá etapa tento soubor zpracuje, např. v případě funkce `create_buckets` z nich vytvoří část histogramu, kterou předá poslední etapě. Tato etapa z jednotlivých částí složí kompletní histogram. První a poslední etapa běží sériově, druhá běží parallelně.

5.2 OpenCL

Jak již bylo zmíněno, pomocí OpenCL byla paralelizována pouze funkce na vytvoření histogramu `create_buckets`, ostatní části nemá smysl pomocí OpenCL paralelizovat a v tomto případě je použita sériová verze.

Program v OpenCL dostane pole čísel a pro každé zjistí, zde se jedná o validní číslo a zda leží uvnitř histogramu. V případě, že jsou obě podmínky splněny zjistí index bucketu pro dané číslo.

6 Testování

Pro testování správnosti (nalezení správné hodnoty a její pozice) programu byly použity vygenerované soubory se známými hodnotami percentilů a všechny verze algoritmu fungují správně.

Pro měření rychlosti jednotlivých verzí algoritmu byl použit volně dostupný soubor *debian-11.1.0-amd64-DVD-1.iso* o velikosti 3,8 GB. Pro každou variantu výpočtu byl program spuštěň 5x, výsledky jednotlivých měření (v sekundách) jsou vidět v tabulce níže.

Tabulka 1: Měření času běhu programu (s)

Typ výpočtu	1.	2.	3.	4.	5.	Průměr
Single	23,73	22,11	22,35	21,59	22,63	22,48
SMP	17,97	17,59	17,54	17,72	17,49	17,66
OpenCL	20,13	19,91	19,92	20,25	19,82	20,00

Ze změřených dat můžeme spočítat pro oba paralelní výpočty urychlení oproti sériové verzi podle vztahu:

$$S = \frac{T_s}{T_p}$$

kde T_s je doba běhu sériového výpočtu a T_p doba běhu výpočtu paralelního. Urychlení výpočtu pomocí SMP je 1,3 a pro OpenCL 1,1. Nízké urychlení OpenCL verze je způsobeno nízkou paralelizací, která při použití našeho algoritmu nelze efektivně rozšířit.

Testování probíhalo na CPU *Intel® Core™ i7-7500U @ 2.70GHz* a OpenCL výpočty na GPU *Intel® HD Graphics 620* pod operačním systémem Windows 10 Pro.

7 Uživatelská dokumentace

Program má tři parametry:

- cesta k souboru
- pecentil – číslo od 0 do 100
- způsob výpočtu - řetězec:
    - "single" - sériový výpočet
    - "SMP" - paralelní výpočet s použitím SMP
    - "název Opencl zařízení" - paralelní výpočet na OpenCL zařízení

8 Závěr

V rámci této semestrální práce jsem vytvořil program, který přečte soubor, interpretuje ho jako sekvenci 64 bitových doublů a nalezne hodnotu zadaného percentilu společně s pozicemi jeho prvního a posledního výskytu. Program má implementován tři různé způsoby výpočtu: sériově, paralelně s použitím SMP a paralelně s použitím OpenCL.

SMP paralelní verze dosáhla oproti sériové urychlení 1,3, OpenCL 1,1. Všechny verze zvládnout zpracovat soubor velikosti 3,8GB s omezenou pamětí do zadaného časového limitu.