AVS: Projekt 1

Vyhodnocení prvního projektu AVS

Vyhodnotili jsme řešení vašeho prvního projektu do předmětu AVS. Cílem bylo vektorizovat výpočet Mandelbrotovy množiny po “řádcích” a po menších “batchích”.

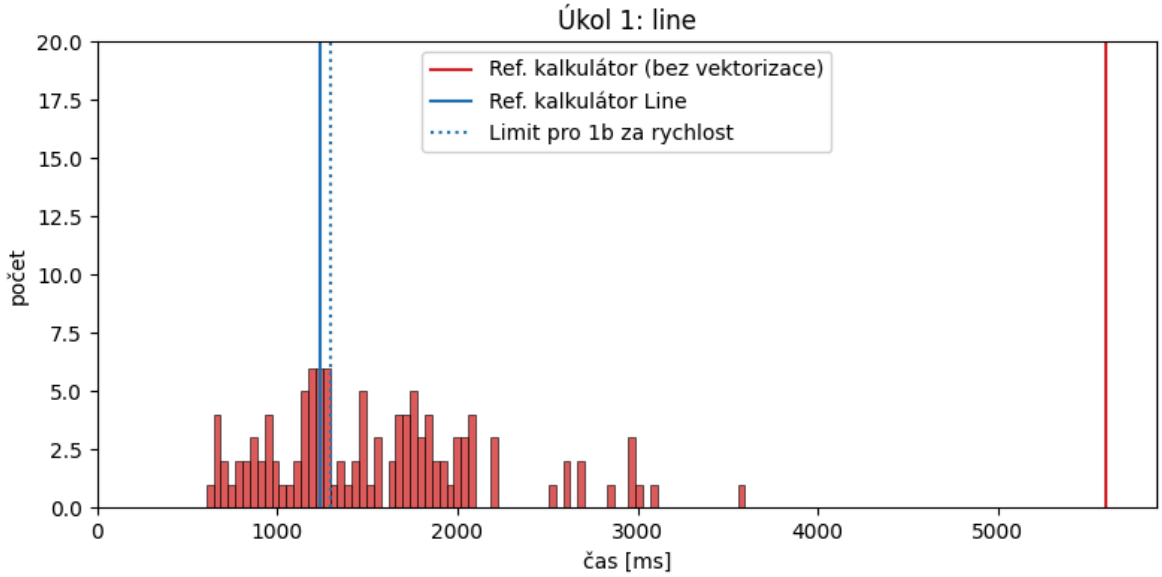
Postup hodnocení

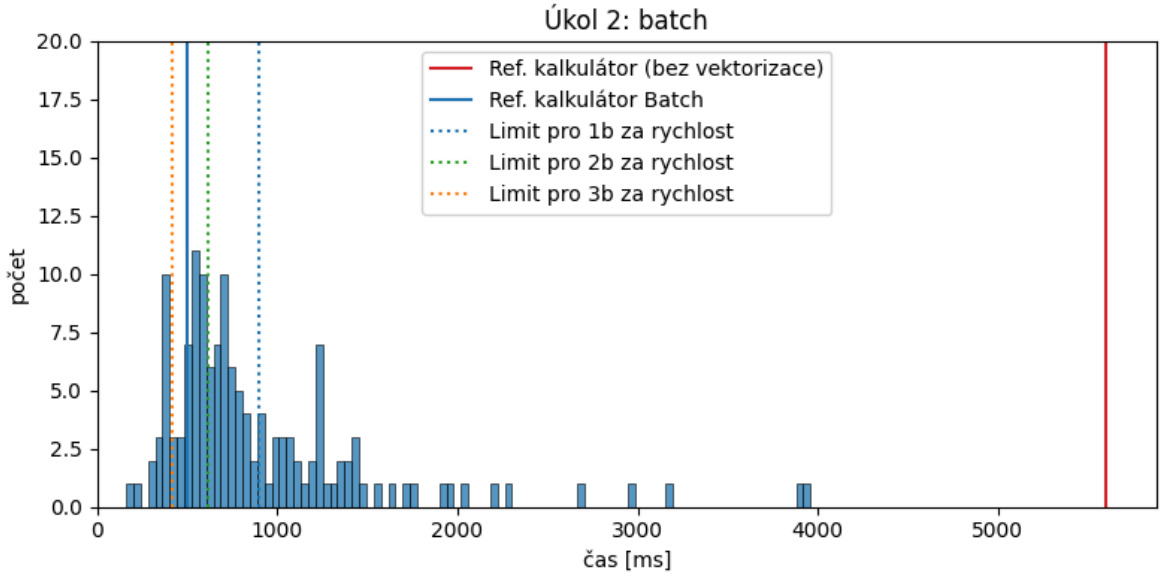
Při vyhodnocení bylo vaše řešení zkompilováno a spuštěn pro obě dvě implementace výpočet (se základem 4096) a porovnán výstup s výstupem referenční implementace. Můžete si všimnout toho, že porovnání výsledků odpovídá skriptům, které jste dostali. Nebyla tedy testována žádná *okrajová podmínka* atd.

```
cmake .
make
for calc in "line" "batch"; do
    for i in `seq 3` ; do
        ./mandelbrot -s 4096 -i 100 -c $calc --batch cmp_$calc.npz
    done
    python3 compare.py cmp_ref.npz cmp_$calc.npz
done
```

Na základě funkčnosti a rychlosti jste mohli získat až 2 body za *Line* implementaci (1 za funkčnost a 1 za rychlost); až 4 body za *Batch* implementaci (1 za funkčnost a až 3 za rychlost). Ve výstupu v ISu se můžete podívat, jak jste se umístili (kolik procent lidí před vámi mělo efektivnější kód).

Celkovou distribuci časů (ořízlou o výjimky, kde výpočet trval několik sekund) a prahové hodnoty můžete vidět na následujícím obrázku





Prahové hodnoty byly zvoleny tak, že od těchto časů se řešení moc nelišilo. Nejvyšší výkonnost jste mohli získat

- prohozením smyček,
- správným ukončením výpočtu, pokud bylo vše dopočítáno,
- odladěním podmínek a jejich správného pořadí,
- odladěním parametrů `#pragma omp simd ...`, často např. `simdlen`,
- využitím symetrie - nejrychlejší implementace pak použily prosté `memcpy` nebo `std::copy` - ideálně ve chvíli, kdy máme ještě data v cache,
- předpočítáním hodnot pro x a y.

Dost často však rozdíl mezi velmi efektivní implementací a méně efektivní byl pouze “kosmetický”.

Nejčastější chyby a poznámky

Neprohození smyček

Tímto nebyla implementována “Line” a tudíž nedodrženo zadání (za tuto skutečnost byly odebrány 2 body).

Parametry OpenMP

Není vhodné pouze uvodit iteraci `#pragma omp simd` bez jakýchkoliv dalších parametrů. Je třeba určit hlavně zarovnání paměti.

Implementace zarovnané paměti

Někteří z vás použili dovětek `aligned(x: 64)` pro paměť, která ovšem nebyla alokována jako zarovnaná. Pro alokaci zarovnané paměti je nutné použít funkce jako je např. `aligned_alloc` nebo `_mm_malloc`.

O1: Jaká byla dosažena výkonnost v Intel Advisoru pro implementace ref, line a batch (v GFLOPS)?

Odpovědi se neshodovali s referenčním řešením naměřeným na výpočetním uzlu Barbory. Je podezření, že byl použit login uzel pro měření.

O2: Porovnejte implementaci referenčního rešení oproti “Line” a “Batch” implementaci. Jakým způsobem jsou načítána data, proč, a jaký to má vliv na výkon.

Velká část z vás se nevyjádřila k tomu, jak byla načítána data. Očekávaný byl vzor načítání dat z paměti (řádky, skupiny). Dále se velmi často vyskytovalo vyjádření ohledně referenční verze o velkém množství výpadků při práci s pamětí. Toto tvrzení však nebylo ničím podložené. Referenční řešení téměř s pamětí nepracuje, tudíž výpadky jsou ojedinělé.

O3: Porovnejte Roofline všech modelů. Jsou implementace memory bound? Jaká je vaše aritmetická intenzita? Dokázali byste ji přibližně spočítat i ručně a jak?

Zde bylo častou chybou tvrzení, že referenční řešení je memory-bound a chybějící pokus o ruční výpočet AI. V Advisoru bylo možné pozorovat, že v roofline modelu chyběl “bod” pro nejvnitřnější smyčku. Ze survey jste mohli vyčíst, že tato smyčka vůbec nepracuje s pamětí (vše je v registrech, vizte assembler). Pro tuto smyčku je tedy teoreticky AI nekonečno. V summary se poté nacházela správná hodnota AI získaná statisticky pro celý program. Implementace je tedy core-bound. Doufali jsem, že manuální výpočet vám pomůže kriticky se zamyslet nad hodnotami v Advisoru. Bohužel, o manuální výpočet se vás ovšem pokusilo pouze málo.

O4: Komentujte vliv velikosti cache na výkonnost a chování na větších velikostech domény.

Většina odpovědí na tuto otázku byla příliš obecná a neopírala se o vaši “Line” a “Batch” implementaci.

O5: Jaký vliv na výkonnost má symetrie? Lze tento rozdíl pozorovat v Intel Advisor? (nápověda: instrukce)

Někteří z vás ignorovali nápovědu, což nemuselo vést k dostatečné odpovědi. Při použití symetrie se zmenší výpočetní čas, avšak nenaroste výkon.

Závěr

Doufáme, že jste si z projektu odnesli to, co může rychlost a výkonnost vašeho algoritmu ovlivňovat. Že se vám podařilo nahlédnout pod pokličku paralelního zpracování pomocí vektorizace a zjistili jste, že napsat vektorizovaný kód nemusí být tak složité, ale i to, že vektorizace přináší nové problémy. Je potřeba se dívat na implementaci jako celek a nakonec dostatečně implementaci odladit.

Marta Jaroš a Oliver Kuník