

Architektury výpočetních systémů (AVS 2025)

Projekt č. 1: Vektorizace kódu

Marta Jaroš (martajaros@fit.vut.cz)

Termín odevzdání: 7. 11. 2025

1. ÚVOD

Cílem tohoto projektu je zrychlit výpočet tzv. Mandelbrotovy množiny. Její základní výpočet vektorizovatelný téměř není, ovšem šikovným přeskládáním smyček a přidáním dalších parametrů je možné dosáhnout rozumného zrychlení. Profilovacím nástrojem bude Intel Advisor.

Projekt lze přeložit a spustit prakticky kdekoliv (vyžaduje CMake a kompilátor), budeme se ale spoléhat na optimalizační reporty poskytované Intel kompilátorem. Je žádoucí, aby cílový procesor disponoval podporou vektorového rozšíření AVX-512. Referenčním strojem je **výpočetní klastr Barbora s kompilátorem Intel** (vizte A), na jehož **výpočetním uzlu** provádějte všechna měření do tohoto projektu. K ladění vašeho řešení můžete využít například i počítače v CVT (mimo serverů jako je Merlin, protože tam je omezená RAM na 1 GB) (vizte B) nebo vlastní počítač, jestliže si nainstalujete nástroje od Intelu (jsou pro studenty zdarma). Pro prvotní ladění funkcionality algoritmu je možné využít i překladač GCC.

2. MANDELBROTOVA MNOŽINA

Vlastní implementace Mandelbrotovy množiny není cílem vaší práce. Přesto však je důležité získat představu o jejím fungování. Mandelbrotova množina¹ je množina bodů komplexní roviny získané rekurzivním procesem. Je to jeden z nejznámějších fraktálů.

¹https://cs.wikipedia.org/wiki/Mandelbrotova_mno%C5%BEina

http://kmlinux.fjfi.cvut.cz/~pauspetr/html/skola/fraktaly/reserse.htm#_Toc73066115

K jejímu určení se používá zobrazení, které každému komplexnímu číslu c přiřazuje posloupnost komplexních čísel z_n . Tato posloupnost je dána rekurzivním přepisem

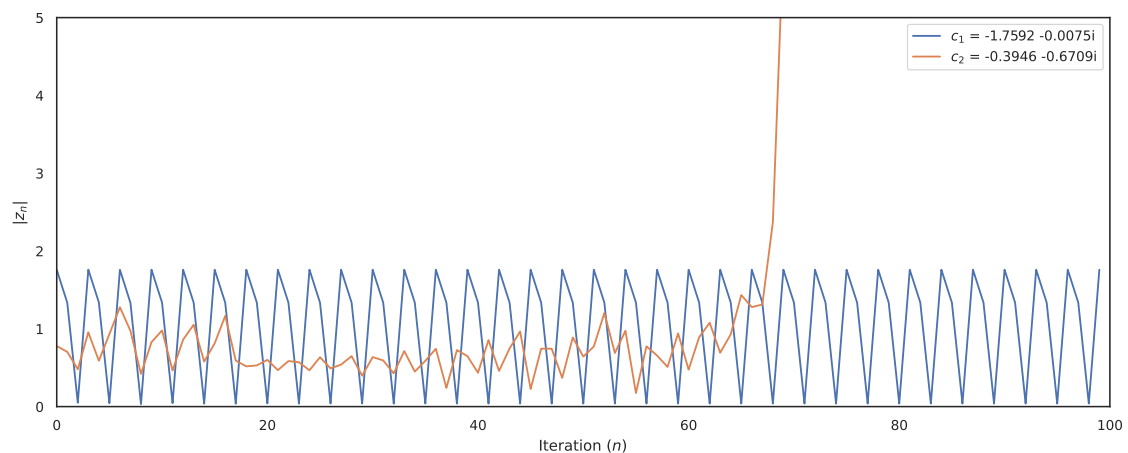
$$z_0 = 0, z_{n+1} = z_n^2 + c$$

Mandelbrotovu množinu lze potom definovat jako množinu

$$M = \{c \in \mathbb{C} | c \rightarrow c^2 + c \rightarrow (c^2 + c)^2 + c \cdots \text{je omezena}\}.$$

Bylo dokázáno, že posloupnost jde k nekonečnu právě tehdy, kdy velikost nějakého členu překročí hodnotu 2.

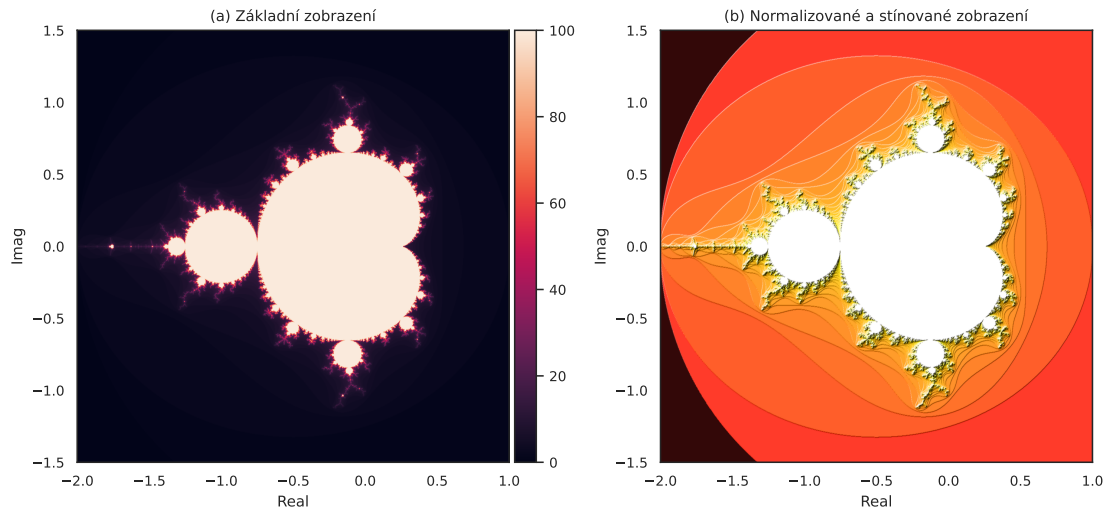
Na základě tohoto iteračního výpočtu pro každý bod v prostoru (v našem případě pro reálnou část mezi -2 a 1 a pro imaginární mezi -1.5 a 1.5) určíme jednotlivé členy z_n pro $n \in (0, 100)$. Vybrali jsem dvě počáteční hodnoty c a vypočítali pro ně absolutní hodnotu $|z_n|$ (obr. 2.1): hodnota c_1 , pro kterou se pohybujeme v intervalu do 2, a hodnotu c_2 , pro kterou dojde k překročení prahové hodnoty 2 kolem 70. iterace a v blízké době se hodnota z_n přiblíží k nekonečnu.



Obrázek 2.1: Průběh absolutní hodnoty $|z_n|$ pro dvě vybrané hodnoty c pro 100 iterací.

Pokud určíme, ve které iteraci došlo k překročení prahové hodnoty 2, dostáváme se k zobrazení na obrázku 2.2a. To můžeme pomocí logaritmické normalizace (power-norm) a stínování zdůraznit, jak je zobrazeno na obrázku 2.2b.

Vášim úkolem bude tedy vektorizovat algoritmus, který se skládá ze tří smyček (vizte kód níže, který je i v souboru *RefMandelCalculator.cc*). První dvě smyčky iterují přes všechny body v prostoru komplexních čísel, třetí (nejvíce zanořená, implementovaná ve funkci **mandelbrot**) počítá iterativně, zda nedošlo k překročení prahové hodnoty ($|z_n| > 2 \Leftrightarrow \text{im}(z_n)^2 + \text{re}(z_n)^2 > 4$). Vzorce uvedené ve výpočtu odpovídají násobení komplexních čísel.



Obrázek 2.2: Znázornění iterace, ve které došlo k překročení prahové hodnoty $|z_n| > 2$.

```

1  template <typename T>
2  static inline int mandelbrot(T real, T imag, int limit) {
3      T zReal = real;
4      T zImag = imag;
5
6      for (int i = 0; i < limit; ++i) {
7          T r2 = zReal * zReal;
8          T i2 = zImag * zImag;
9
10         if (r2 + i2 > 4.0f)
11             return i;
12
13         zImag = 2.0f * zReal * zImag + imag;
14         zReal = r2 - i2 + real;
15     }
16     return limit;
17 }
18
19 int * calculateMandelbrot() {
20     int *pdata = data;
21     for (int i = 0; i < height; i++) {
22         for (int j = 0; j < width; j++) {
23             float x = x_start + j * dx; // current real value
24             float y = y_start + i * dy; // current imaginary value
25             *(pdata++) = mandelbrot(x, y, limit);
26         }
27     }
28     return data;
29 }

```

3. STRUKTURA PROJEKTU, PŘEKLAD A SPUŠTĚNÍ

Archiv zadání obsahuje:

- `MB-xlogin00.txt`: šablonu otázek, na které je nutné odpovědět
- `scripts/`: pomocné skripty pro evaluaci
- `src/`: adresář se zdrojovými kódy
- `evaluate.sl`: skript pro cluster, který provede evaluaci
- `advisor.sl`: skript pro cluster, který naměří hodnoty pro Intel Advisor

Příložené **skripty neměňte!**

Zdrojový kód je rozdělen na pomocné soubory `main.cc`, který zpracovává argumenty příkazové řádky, soubory v `common`, které se starají o ukládání výsledků do NumPy pole, zpracování argumentů příkazové řádky a měření času. Jádro práce nalezneme ve složce `calculators`. Důležité jsou tři kalkulátory označené jako **Ref** (referenční bez vektorizace), **Line** (vektorizace po řádcích) a **Batch** (vektorizace po malých skupinách). Soubory `calculators/LineMandelCalculator.*` a `calculators/BatchMandelCalculator.*` jsou jádrem vaší práce. Do vyznačených míst tedy doplňte váš kód.

Pro kompilaci a spuštění je možné použít CMake následujícím způsobem (pro kompilátor Intel):

```
ml purge
ml intel-compilers/2024.2.0 CMake/3.23.1-GCCcore-11.3.0 # pouze na Barbore
cd Assignment
mkdir build && cd build
CC=icx CXX=icpx cmake -DUSE_O3=OFF ..

make -j
```

Pro překlad programu s parametry `-O0`, respektive s `-O3`, nastavte proměnnou `-DUSE_O3=OFF`, respektive `-DUSE_O3=ON`.

Vygenerované optimalizační reporty pro jednotlivé kalkulátory poznáte podle přípony `.optrpt` a najdete je ve složce `build/CMakeFiles/mandelbrot.dir/calculators`.

Poté je možné spustit vlastní aplikaci s různými parametry příkazové řádky:

```
$ ./mandelbrot -h
AVS Assignment 1 - Mandelbrot calculation using SIMD instructions
Usage:
  AVS: Mandelbrot [OPTION...] <OUTPUT>

  -s, --size arg           Base matrix size (default: 2048)
  -i, --iters arg          Number of iterations (default: 100)
  -c, --calculator arg     Calculator name [ref, batch, line] (default: ref)
      --batch              Run in silent/batch mode
  -h, --help               Print help
```

Můžete měnit základ velikosti matice s , kdy výsledná matice bude mít rozměr $3s \times 2s$, počet iterací algoritmu, kalkulátor (důležité pro vaše testování). Výsledek je možné uložit do komprimovaného NumPy pole (*.npz)². S tímto polem pak pracují skripty pro testování. Prvním přístupem je vizualizace pole uloženého z běhu. Je nutné načíst knihovny *Pandas*, *NumPy* např. načtením balíku *matplotlib*. Pozor ale, natažením tohoto balíku může dojít k přepsání již natažených balíků.

```
$ ml matplotlib # pro praci na clusteru v Ostrave
$ ./mandelbrot -c ref -s 1024 -i 100 ref.npz
===== Mandelbrot SIMD calculator =====
Calculator:      RefMandelCalculator
Base size:      1024
Matrix size:    3000x2000
Iteration limit: 100
Elapsed Time:   512 ms
$ python3 ../scripts/visualise.py ref.npz --show --save img.png
```

Nebo můžete pole navzájem porovnat — pozor, vzhledem k tomu, že kompilátor může měnit pořadí floatových operací, tak nemusí výsledky odpovídat úplně přesně, ale může docházet k drobným odchylkám.

```
$ ./mandelbrot -c batch -s 1024 -i 100 batch.npz
===== Mandelbrot SIMD calculator =====
Calculator:      BatchMandelCalculator
Base size:      1024
Matrix size:    3072x2048
Iteration limit: 100
Elapsed Time:   ??? ms
$ ./mandelbrot -c ref -s 1024 -i 100 ref.npz
===== Mandelbrot SIMD calculator =====
Calculator:      RefMandelCalculator
Base size:      1024
Matrix size:    3072x2048
Iteration limit: 100
Elapsed Time:   ??? ms
$ python3 ../scripts/compare.py batch.npz ref.npz
[ok] Results are very close (eps = 0.007% )
```

Nebo můžete pustit automatický test všech tří implementací (vhodné na konci před odevzdáním).

```
$ ml matplotlib # pro praci na clusteru v Ostrave
$ bash ../scripts/compare.sh
RefMandelCalculator;512;1536;1024;100;???
LineMandelCalculator;512;1536;1024;100;???
BatchMandelCalculator;512;1536;1024;100;???
Reference vs line
[ok] Results are very close (eps = 0.005% )
Reference vs batch
[ok] Results are very close (eps = 0.005% )
Batch vs line
[ok] Results are same
Test passed
```

²<https://numpy.org/doc/stable/reference/generated/numpy.load.html>

4. POSTUP PRÁCE

Váším úkolem budou čtyři základní kroky, které se budou podílet na hodnocení (max. 15 bodů).

1. Implementovat vektorizovanou verzi po řádcích a po menších skupinách.
2. Vyhodnotit obě implementace v Intel Advisor.
3. Vyhodnotit efektivitu implementací.
4. Odpovědět na otázky v dokumentu `MB-xlogin00.txt`. Odpovídejte prosím **stručně** ale **věcně**. Obecné odpovědi nemusí být hodnoceny kladně.

Pracujte však i s textovými optimalizačními reporty vygenerovanými kompilátorem. V referenční implementaci jsou tři smyčky `for`. Vzhledem k datovým závislostem budete muset změnit pořadí smyček tak, aby počítání iterací bylo místo nejnižší úrovně o úroveň výš. Proto bude vhodné průběžně ukládat data do nějaké pomocné paměti. Pro korektní měření **všechny alokace paměti provádějte v konstruktoru kalkulátoru**. Vektorizaci provádějte pomocí OpenMP SIMD pragem. Abyste se dostali na rozumnou efektivitu, je nutné nějakým způsobem pro celou vektorizovanou smyčku určit, zda výpočet ukončit ($\forall i : |z_n^i| > 2$ — pomůže vám redukce).

Při výpočtu je vhodné využít toho, že **malice je symetrická**, tzn. je možné vypočítat jen její polovinu.

Druhým krokem bude vyhodnotit vektorizaci v nástroji Intel Advisor. Doporučujeme s tímto nástrojem pracovat průběžně. Při spouštění na clusteru Barbora využijte Slurm skript `advisor.sl` (`sbatch advisor.sl`), který vám rychle spočítá výsledky, do složky `build_advisor/advisor_{calc}` uloží projekty, které si v Advisor GUI zobrazíte. Pro připojení na cluster s podporou grafického rozhraní využijte `ssh -X -i /id_rsa-dd-... username@barbora.it4i.cz`, tunelování VNC nebo službu Open On-Demand. Práce s nástrojem Advisor a tunelování přes VNC je detailně popsána v příloze A. V Advisoru `ml Advisor; advixe-gui` otevřete projekt, zvolte možnost *Show my results*. Připomínáme, že login uzal neslouží k náročným výpočtům. Proto na něm také nesmíte spustit příkazy *Collect* nebo *Start survey analysis*, abyste nezatěžovali přihlašovací stroje. Výsledky si můžete stáhnout a analyzovat v lokální instalaci Advisoru, případně v CVT (vizte přílohu B).

Po analýze kódu můžete vyhodnotit celkovou efektivitu skriptem `sbatch evaluate.sl`. Tento skript spustí několik běhů projektu (paralelně tak, aby nebyl překročen počet jader) a vytvoří soubor `datalog.csv`. Z tohoto souboru vám skript

```
python3 scripts/plot_evaluate.py datalog.csv --save eval.png
```

vytvoří boxploty pro různá nastavení velikosti mřížky a počtu iterací. Tento graf potom odevzdáváte a závěry z něj shrnete v odpovědním formuláři.

4.1. ODEVZDÁVANÉ SOUBORY

Ve výsledném archivu *xlogin00.zip* odevzdávejte pouze šest následujících souborů:

- *LineMandelCalculator.cc/h*
- *BatchMandelCalculator.cc/h*
- *eval.png*
- *MB-xlogin00.txt*

Do jiných souborů nesmíte zasahovat. Před odevzdáním doporučujeme ověřit funkčnost (vizte výše).

4.2. HODNOCENÍ

Pro hodnocení projektu platí následující podmínky:

- Hodnotí se pouze pro velikosti bloků (parametr $-s$) jako mocniny dvou,
- až 1 bod získáte za správnost implementace po řádcích, až 1 bod za správnost implementace po blocích,
- až 1 bod získáte za rychlost vektorizace po řádcích, až 3 body za rychlost po blocích,
- hranice pro rychlost budou určeny podle distribuce rychlostí vás a vašich kolegů (pro $s=4096$),
- a správnost odpovědí na otázky (až 9 bodů).

Chtěli bychom upozornit, že **nejde o soutěž** mezi vámi a vašimi kolegy v získání bodů za rychlost vašeho řešení. Rychlost vašeho řešení odpovídá tomu, jak dobře jste dokázali řešení napsat a využít vektorizace. Přirozeně tedy zapadnete do jiné "rychlostní skupiny", která se odvíjí od kvality implementace a optimalizace, ne od výsledků vašich kolegů!

Pracujete na sdíleném stroji, je tedy potřeba počítat s tím, že při práci na poslední chvíli nebudete mít čas správně vyhodnotit vaše řešení (a tím ztratit velkou část bodů). Případné dotazy k zadání řešte prosím přes fórum v IS VUT (ideální pro obecné dotazy, problémy s připojením, atd.) nebo přes mail s Martou Jaroš. Implementační dotazy můžete řešit také po přednáškách či cvičeních.

A. SUPERPOČÍTAČ BARBORA

Superpočítač Barbora umístěný na VŠB v Ostravě je složen z celkem 192 uzlů, každý uzel je osazen dvěma 18jádrovými procesory Intel Cascade Lake 6240 a 192 GB RAM. Tyto procesory podporují tedy vektorové instrukce AVX-512. Pro připojení na superpočítač

Po skončení práce musíte vašeho klienta ručně ukončit! Jinak dojdou porty na uzlu:
vncserver -kill :3.

Následně se je možné připojit pomocí VNC klienta. Ten se v příkladu výše musí připojit na `login1.barbora.it4i.cz` (vizte hostname v konzoli, na tom konkrétním login uzlu byl puštěn VNC server, není možné použít přímo round-robin DNS) na port 5900 + číslo displeje, v příkladě výše 3, tj. 5903. Připojení musí být prostřednictvím SSH tunelu, protože port 5903 není přístupný mimo síť it4i. Jestliže váš VNC klient nepodporuje tunelování přes SSH, vytvořte si jej dle zmíněného návodu (v GNU/Linuxu pomocí příkazu `ssh -L 5903:login1.barbora.it4i.cz:5903 username@barbora.it4i.cz`, v OS Windows např. prostřednictvím PuTTY). Pak VNC server připojíte k lokálnímu počítači na protunelovaný port.

Po připojení ke vzdálené ploše můžete otevřít emulátor terminálu (menu Aplikace → Systémové nástroje → Terminál) a vytvořit interaktivní úlohu následujícím způsobem:

```
[username@login1.barbora ~]$ sbatch advisor.sl
[username@login1.barbora ~]$ squeue --me -l
# pockat, az uloha dobehne
[username@login1.barbora ~]$ ml Advisor
[username@login1.barbora ~]$ advixe-gui

# nebo spustit interaktivni ulohu na uzlu
[username@login1.barbora ~]$ salloc -A DD-23-135 -p qcpu_exp \
-N 1 -t 1:00:00 --comment 'use:vtune=2022.2.0' --x11

salloc: Pending job allocation 44795
salloc: job 44795 queued and waiting for resources
salloc: Granted.

[username@cn49.barbora ~]$ ml Advisor
[username@cn49.barbora ~]$ advixe-gui
```

Příkaz `sbatch` zadá požadavek na spuštění úlohy do fronty; jakmile bude v systému dostatek volných uzlů, dojde ke spuštění úlohy. Parametr `-A` určuje projekt, v rámci kterého máme alokované výpočetní hodiny (neměnit), `-p` určuje frontu, do které bude úloha zařazena (pokud budete na spuštění úlohy čekat příliš dlouho, můžete zkusit frontu `qcpu`, ale preferujte `qcpu_exp`), parametr `-N` určuje počet uzlů, které budou úloze přiděleny, parametr `-t` určuje maximální čas alokovaný pro úlohu a další možnosti (v našem případě načtení modulu pro profiler). Interaktivní úlohu získáme použitím příkazu `salloc` namísto `sbatch`. Pomocí `--x++` tunelujeme X-windows protokol do login uzlu. Více o spouštění úloh na superpočítačích IT4I naleznete v dokumentaci⁶.

Nyní jste již v terminálu připojeni k výpočetnímu uzlu (vizte nový hostname v terminálu) s tím, že by jste měli být schopní spustit grafickou aplikaci:

```
[username@cn49.barbora ~]$ ml VTune Advisor
[username@cn49.barbora ~]$ amplxe-gui
[username@cn49.barbora ~]$ vtune-gui
```

Alternativně můžete využít i Open On Demand rozhraní⁷.

⁶<https://docs.it4i.cz/en/docs/general/pbs-job-submission-and-execution>

⁷<https://docs.it4i.cz/en/docs/general/access-services/graphical-user-interface/ood>

Software na superpočítači je dostupný pomocí tzv. *modulů*. Práci s nimi zajišťuje příkaz `ml` (*module load*). Tento příkaz bez parametrů vypíše aktuálně načtené moduly, a všechny moduly specifikované jako parametry se pokusí načíst. Příkaz `ml purge` je všechny odstraní.

Moduly je potřeba načíst po každém přihlášení nebo získání výpočetního uzlu (jsou implementovány proměnnými prostředí). V tomto projektu budou pro překlad (a spouštění) potřeba moduly `intel-compilers` (kompilátor), `CMake` (překladačový systém) a pro profilování modul `VTune Advisor` (jak bylo již demonstrováno výše). **Jestliže jej potřebujete, modul profileru načtete jako první!** Závisí na starších knihovnách, které mohou být přepsány novějšími po načtení kompilátoru. Pro profilování kvůli knihovnám musíte mít načtený i modul `intel-compilers`, jinak se vaše přeložená aplikace nespustí.

B. SPOUŠTĚNÍ NA POČÍTAČÍCH CVT FIT

Projekt je možné spustit na počítačích CVT, jako jsou počítače v učebnách nebo server *merlin* — na něm ale nespustíte Intel Advisor nebo VTune, můžete na něm ale ladit funkčnost (Intel kompilátor je k dispozici). V učebnách s Linuxem (a Intel procesorem) je možné použít nástroje Intel Advisor a VTune napřímo.

```
$ source /usr/local/share/intel/setvars.sh
:: initializing oneAPI environment ..
$ mkdir build
$ cd build/
$ CC=icx CXX=icpx cmake -DUSE_03=OFF ..
-- The C compiler identification is Intel ...
-- The CXX compiler identification is Intel ....
-- ...
$ make
...
$ bash ../scripts/compare.sh
$ advixe-gui # pouze na pocitacich v ucebnach kvuli limitu RAM
$ vtune-gui # take
```