Vysoké učení technické
Fakulta informačních technologií

Praktické paralelní programování

Projekt č. 1 – MPI a paralelní I/O

Alena Tesařová (xtesar36) 30. dubna 2021

1 Implementace

Projekt se nachází v přiložených souborech parallel_heat_solver.cpp a parallel_heat_solver.h. Byly implementovány všechny části podle zadání. V konstruktoru dojde k inicializace všech proměnných, které jsou k vykonání programu potřeba jako je

- rozměry x,y dlaždice (bez okrajů, i s okraji),
- rozměry celé matice (bez okrajů, i s okraji),
- zapnuté módy (RMA, sekvenční IO),
- typy proměnných pro sloupce, řádky a dlaždice,
- vytvoření okna pro práci v RMA módu,
- offsety a informace k rankům.

Dále dojde v konstruktoru k otevření souboru pro zápis, pokud je zadán název výstupního souboru. Hlavním úkolem *RunSolveru* je iterace přes zadaný počet simulací a postupný výpočet teploty.

V mé implementaci si na začátku rozšiřuji ze všech stran vstupní matici o okraje, kvůli jednodušší a jednotné práci u rozesílání dat mezi ranky. Dále je třeba upozornit na to, že si vstupní matici ukládám do dlaždic transponovaně (ukládám sloupce matice jako řádky). Ve výpočtu to ničemu nevadí. Jediný problém této implementace bylo ukládání do souboru, jelikož zde bylo potřeba data přetransponovat. Tento problém byl vyřešen postupným zápisem do souboru po jednotlivých řádcích.

2 Profiling

Řešení projektu bylo testovány scripty poskytnutými v rámci projekty nacházející se ve složce *scripts*. Pro překlad byly použity moduly: *CMake intel/2020a HDF5/1.10.6-intel-2020a-parallel Score-P/6.0-intel-2020a*. Přechod na doporučenou verzi modulů (po obdržení emailu) mi způsoboval chyby, proto jsem zůstala při těchto modulech.

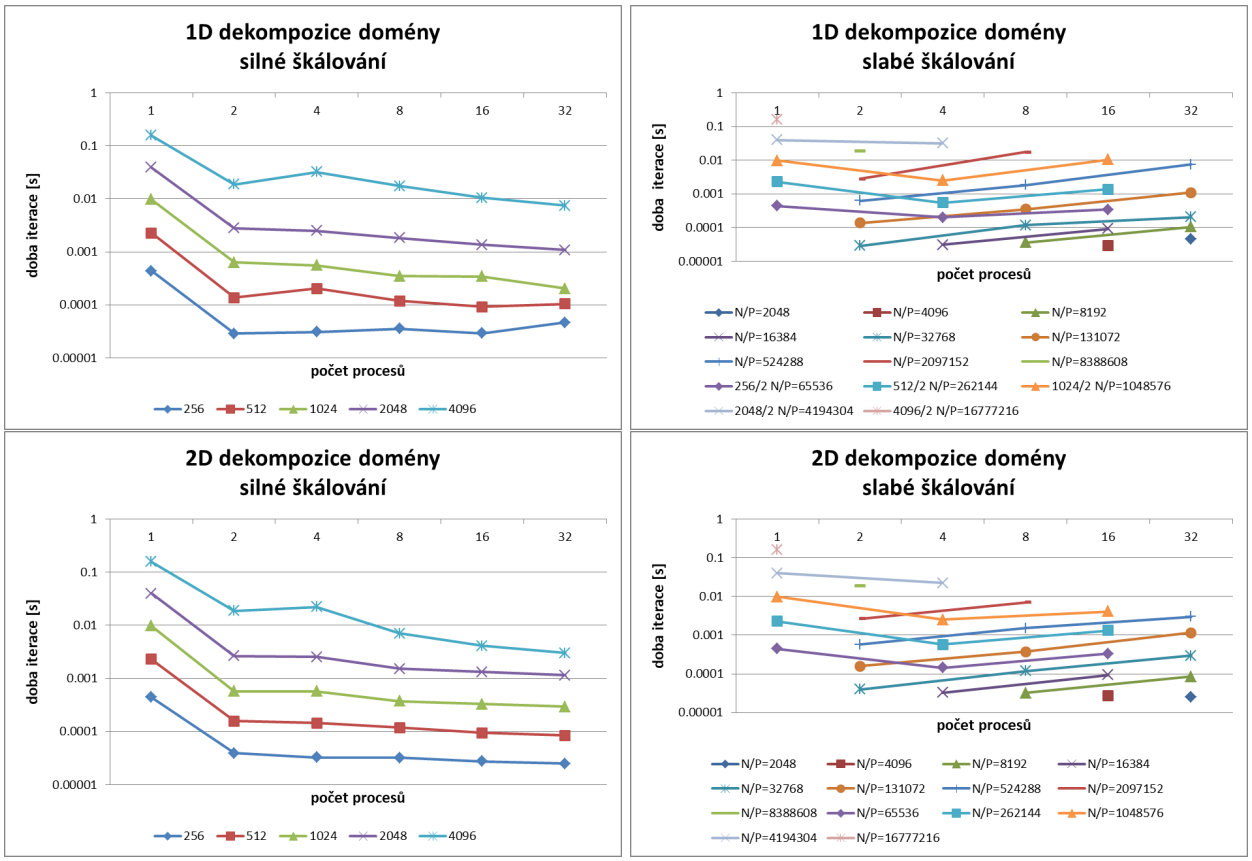
2.1 Grafy silného a slabého škálování

Na obrázku 1 je vidět porovnání grafů škálování (slabého i silného) u 1D a 2D dekompozice. Hodnoty byly naměřeny pomocí scriptů run_full_hybrid_1d.pbs a run_full_hybrid_2d.pbs. Jak jde vidět, výsledky jsou 1D a 2D dekompozice jsou téměř totožné. Výkon je velmi podobný, je možné, že je to způsobeno použitím funkce UpdateTile.

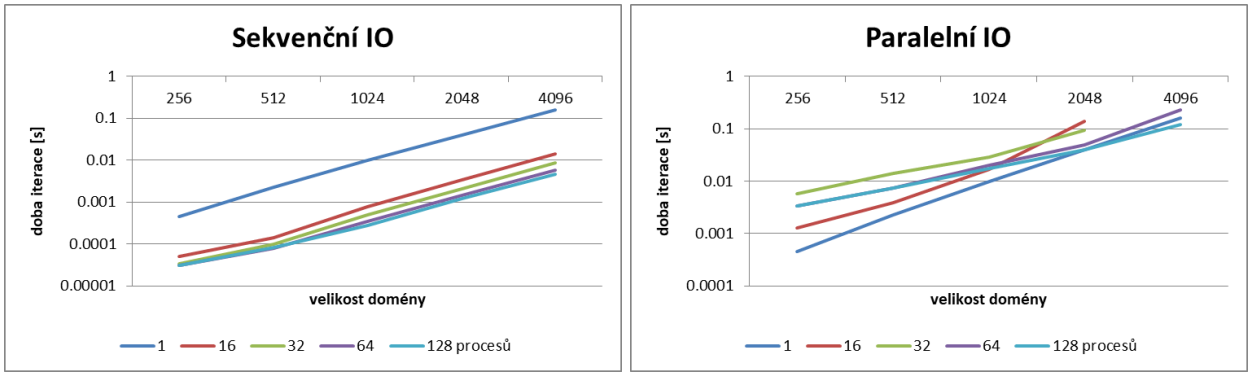
2.2 Testování I/O

K ověření funkce paralelního zápisu do souboru byl použit nástroj VisIt. Výsledek lze vidět na obrázku 3. Testovány byly vstupy velikost domény 16 a 4096. Na obrázku 2 je ukázáno porovnání sekvenční a paralelní verze. U sekvenční varianty se čas pouze velmi pomalu zmenšuje se zvyšujícím se počtem procesů. Paralelní verze se dá na první pohled méně efektivnější, jelikož se zvyšujícím se počtem procesů čas roste i čas. Začíná být efektivnější až u použití větší domény. Důvodem je velké množství komunikace, které se vyplatí až při větších doménách.

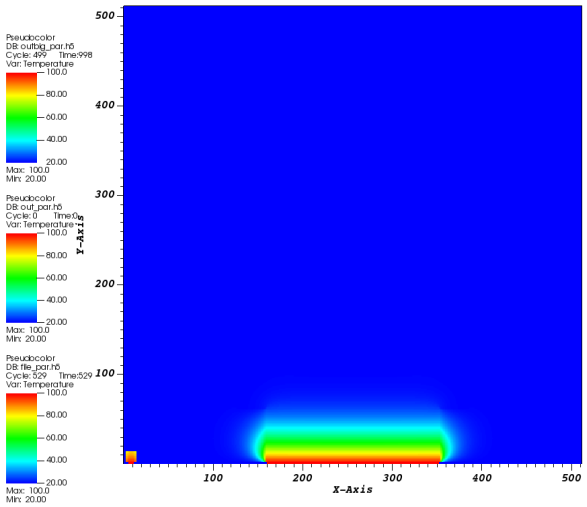
Paralelní I/O by šlo zefektivnit použitím sekvenčního zápisu do souboru do velikost domény 2048. Při větších doménách bych použila paralelní verzi.



Obrázek 1: Grafy 1D a 2D dekompozice



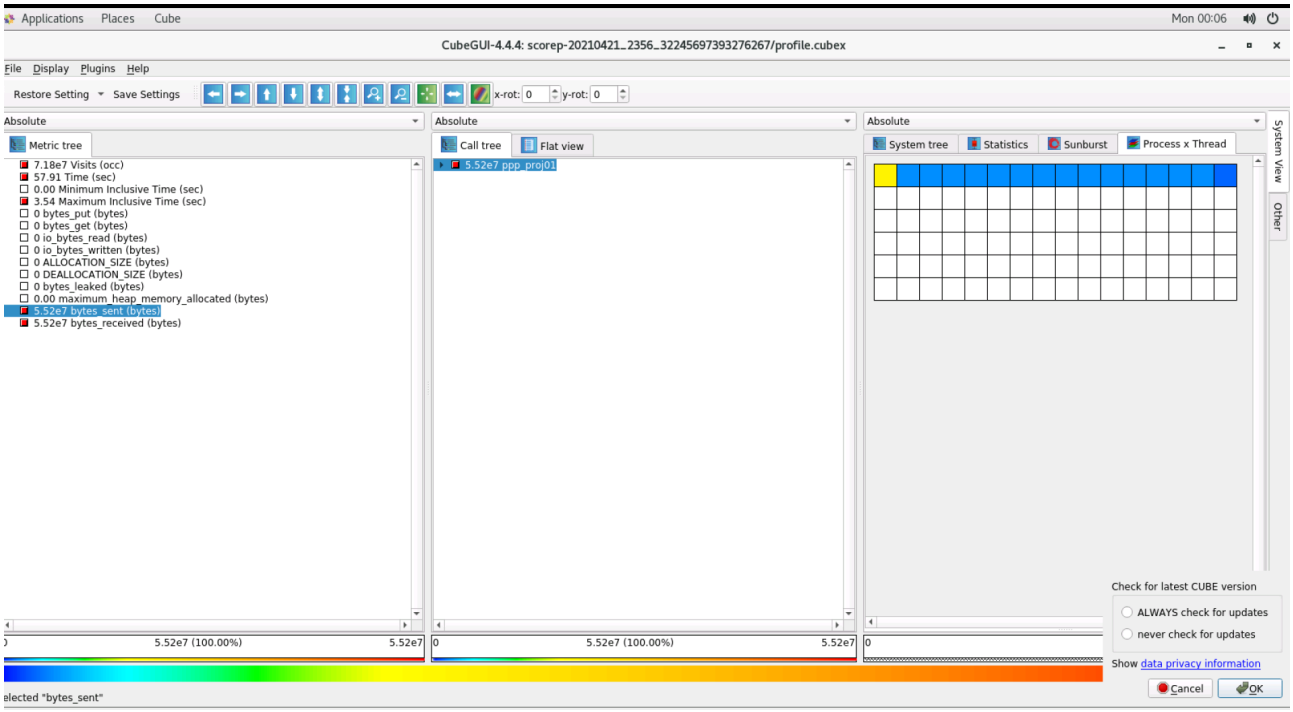
Obrázek 2: Porovnání paralelního a sekvenčního zápisu do souboru



Obrázek 3: Vizualizace nárůstu teploty v programu VisIt

2.3 Cube

Obrázek 4 vizualizuje vytížení komunikátoru pomocí nástroje Cube. Podle nástroje je nejvytíženější rank 0, který se stará o zvětšení vstupní matice, distribuci hodnot, počítání průměrné teploty a řídí celou komunikaci. Zátěž tedy není vyrovnaná.

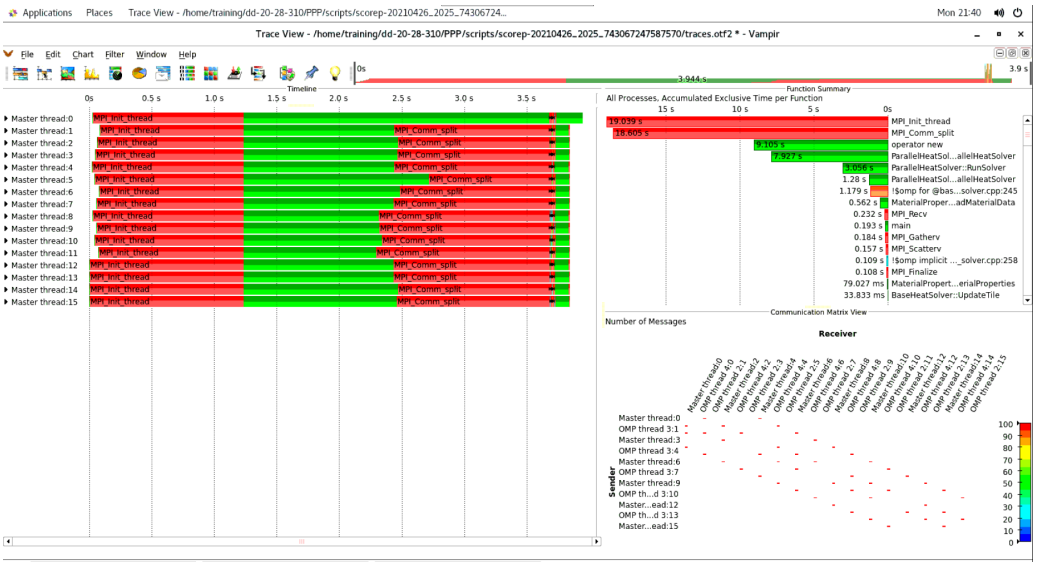


Obrázek 4: Profilování 2D dekompozice s nástrojem cube na 16 procesech

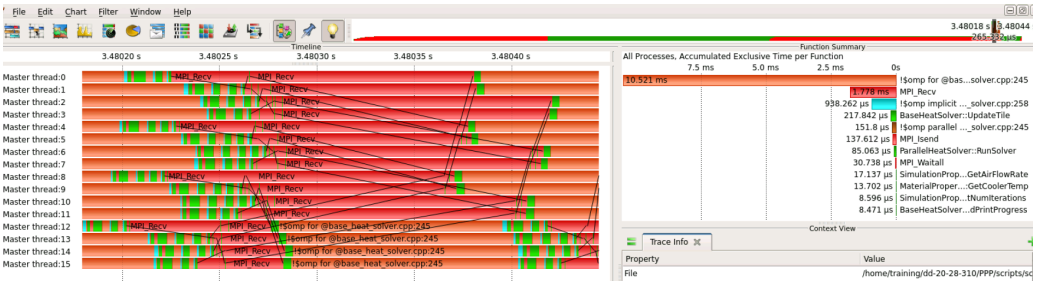
2.4 Vampir

Na obrázku 5 lze vidět výsledek profilování pomocí nástroje *Vampir*. Celkově trvá program 3.5 sec, z čehož nejvíce času zabírá MPI_Comm_split, jelikož ostatní ranky musí čekat na rank 0, který dělá předzpracování vstupních dat (rozšiřuje matice o okraje ze všech stran a tato operace je náročná). V pravém dolním rohu je zobrazena komunikační matice, kde lze vidět, jak spolu komunikují ranky, které se nachází vedle sebe (což nastává při výměně okrajů).

Při přiblížení na část výpočtu si lze všimnout překrytí komunikace s výpočtem, jak jde vidět na obrázku 6. Díky této optimalizaci je možné počítat, zatímco procesy komunikují a dochází k urychlení celého programu.



Obrázek 5: Profilování 2D dekompozice s nástrojem Vampir na 16 procesech



Obrázek 6: Zobrazení překrytí komunikace a výpočtu

3 Závěr

Díky profilování pomocí nástroje Vampir bylo zjištěno, že nejvíce času zabírá předzpracování dat, proto není program efektivní. Zlepšením by bylo pracovat s daty, tak jak jsou a soustředit se více na správnou distribuci dat procesům. Samotný výpočet teploty je obecně velmi rychlý a efektivní, jelikož se zde překrývá komunikace MPI a výpočet. Zhoršuje se doba výpočtu při zápisu do souboru nebo při častém zjišťování teploty ve středním sloupci.

V projektu byly splněny všechny body zadání, správnost výstupů byla ověřena porovnáním výsledků se sekvenční verzí, výstupy souborů byly otestovány pomocí nástroje VisIt a efektivita byla naměřena pomocí profilovacích nástrojů Vampir a Cube.