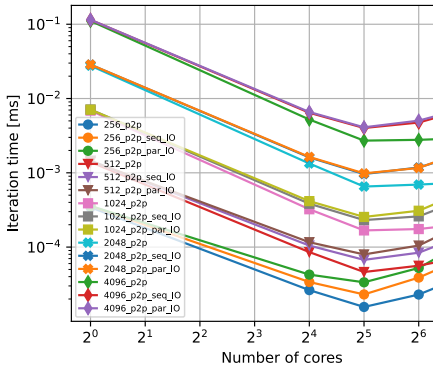
Dokumentace k projektu do kurzu PPP

Kateřina Fořtová (xforto00)

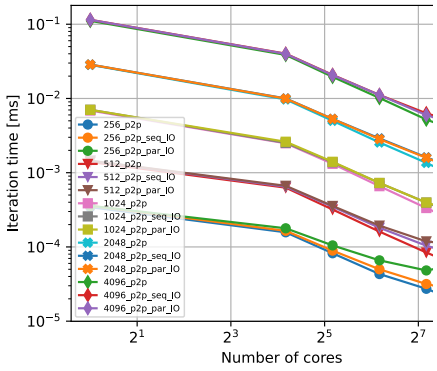
Zhodnocení výsledků projektu

Vliv 1D a 2D dekompozice

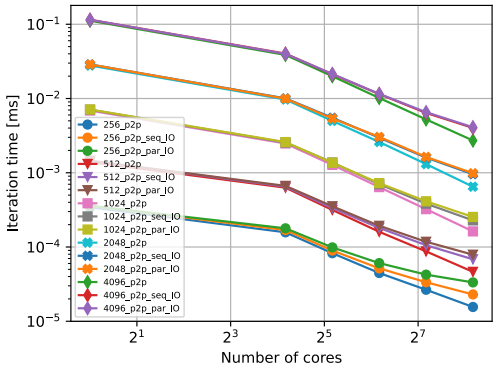
Z grafů silného škálování vyplývá, že naměřené výpočetní časy pro 1D a 2D dekompozici byly velmi podobné. Využití OpenMP SIMD při počítání funkce `ComputePoint` se velmi pozitivně projevilo na celkovém času běhu. V rámci experimentů byla změřena doba simulace pro doménu 16×16 po 50 000 iterací s jedním vláknem pro každý proces, aby byly získány podrobnější znalosti o rozdílu mezi 1D a 2D dekompozicí.



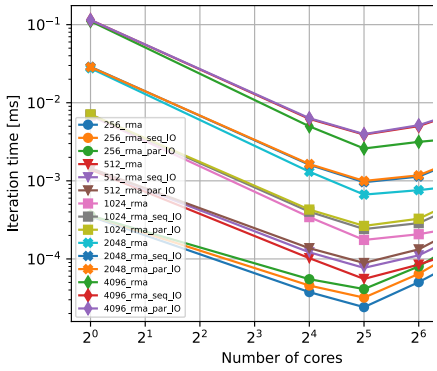
Obrázek 1: Silné škálování, 2D dekompozice (1 – 128 procesů, P2P)



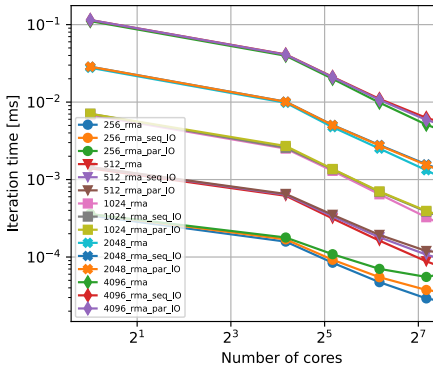
Obrázek 2: Silné škálování, 1D dekompozice (1 – 32 procesů, P2P)



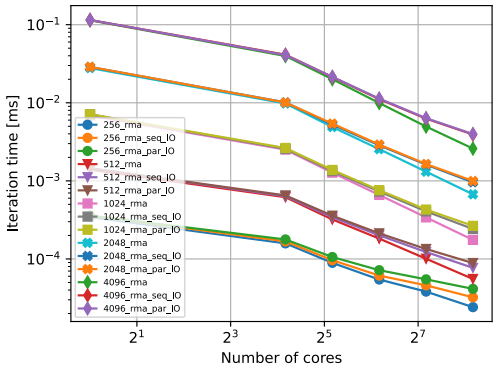
Obrázek 3: Silné škálování, 2D dekompozice (1 – 32 procesů, P2P)



Obrázek 4: Silné škálování, 2D dekompozice (1 – 128 procesů, RMA)



Obrázek 5: Silné škálování, 1D dekompozice (1 – 32 procesů, RMA)



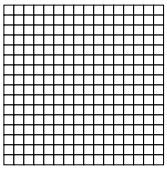
Obrázek 6: Silné škálování, 2D dekompozice (1 – 32 procesů, RMA)

Počet procesů	P2P 1D	P2P 2D	RMA 1D	RMA 2D
2	0,120439	0,129546	0,21667	0,218116
4	0,144353	0,167122	0,319009	0,325851
8	0,157083	0,185093	0,379824	0,394661
16	0,231228	0,22164	0,826918	0,463391
32	0,248989	0,242793	0,921975	0,550269

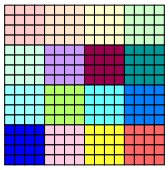
Tabulka 1: Naměřené časy simulace v sekundách pro doménu 16×16 po 50 000 iterací

Uvažujeme-li 1D dekompozici, kdy počet procesů je větší nebo roven velikosti strany domény, v našem případě při počtu 16 nebo 32 procesů, pak je výhodnější využití 2D dekompozice. Uvažujeme diskretizaci simulační domény pouze do uniformní čtvercové mřížky. Při komunikaci s využitím 2D dekompozice jsou předávány okraje o šířce hodnoty

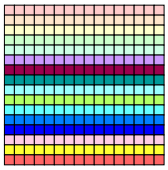
2 pouze levému, pravému, dolnímu nebo hornímu procesu. Pokud využijeme 1D dekompozici, kdy každý proces vlastní jeden řádek domény, nebo pouze část řádku ($16 \times 16 = 256$ / 16 procesů = 16 (viz obrázek 9) nebo 256 / 32 procesů = 8), pak okraje překrývají nejen sousední procesy jako u 2D dekompozice, ale i sousedy těchto procesů. V tomto případě probíhá náročnější komunikace. Pokud je však počet procesů menší než velikost strany domény, pak jeden proces vlastní několik řádků domény (např. pro 8 procesů každý proces vlastní 2 řádky domény). V tomto případě je výhodnější 1D dekompozice, protože okraje překrývají pouze sousední procesy stejně jako u 2D dekompozice, ale navíc se posílají jen horní a dolní okraje, nikoliv také levé a pravé okraje jako u 2D dekompozice. V tomto případě je tedy pro 1D dekompozici sníženo množství komunikace a urychlení simulace oproti 2D dekompozici se projevuje znatelněji pro mód RMA. V obou případech je pak na konci simulačního kroku spočítána průměrná teplota na základě prostředního sloupce čtvercové domény.



Obrázek 7: Příklad čtvercové domény 16 x 16



Obrázek 8: Příklad 2D dekompozice pro 16 procesů



Obrázek 9: Příklad 1D dekompozice pro 16 procesů

Paralelní I/O

Na základě grafů silného škálování můžeme zjistit, že paralelní zápis do souboru je obecně výhodnějším hlavně při výpočtu větších velikostí domény a vyšším počtu výpočetních uzlů. Pokud využíváme menší velikost domény, např. o velikosti 256, pak je výhodnější využít sekvenční zápis. Při paralelním zápisu do souboru je využito tzv. funkce hyperslab, kdy je pro každý proces spočítán index počáteční a koncové pozice pro zápis do finálního pole a výsledky tedy nemusí být dodatečně přeskládávány. Po zápisu všech procesů tak vznikne finální snapshot teplot celé domény. Každá dlaždice procesu však musí být před zápisem do souboru zbavena okrajů o velikosti 2, které se překrývají se sousedy procesu.

Jedním z možných pokročilých ladění paralelního zápisu do souboru je využití kolektivního přístupu k metadatům výstupního souboru. Pokud každý proces čte nezávisle metadata při otevírání výstupního souboru, pak může být otevírání HDF5 souboru při vysokém počtu procesů pomalé – je vytvořeno mnoho žádostí o čtení stejných metadat. Nastavení kolektivního přístupu k metadatům může být provedeno s užitím `H5Pset_coll_metadata_write` a `H5Pset_all_coll_metadata_ops`. Při kolektivním nastavení nultý proces čte jako jediný záznam metadat a informace posílá pomocí broadcastu ostatním procesům. Toto ladění je vhodné zejména pro vyšší počet procesů [1].

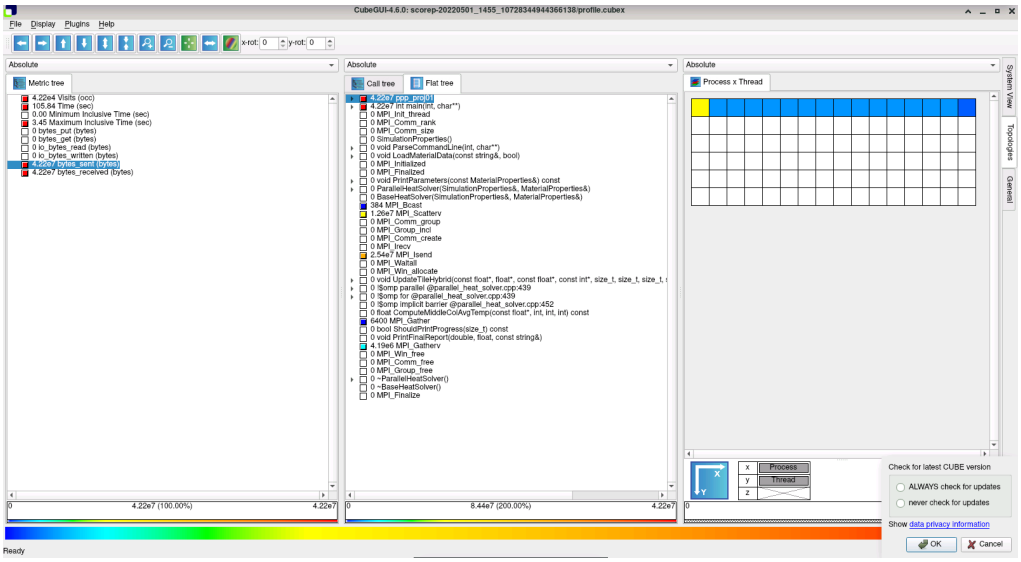
Analýza pomocí Score-P, Cube a Vampir

Z hlediska odeslané velikosti zpráv je nejvíce zatížen nultý proces. Tento proces se stará o broadcast dat ostatním procesům, pomocí kolektivních operací rozesílá vnitřky dlaždic a následně získává informace o finální teplotě i výsledku celé domény. Rozesílání sdílených okrajů však už je řešeno pomocí komunikace jednotlivé dvojice sousedních procesů. Množství komunikace ostatních procesů je již vyrovnanou. Při každé z neblokujících komunikací jsou procesy nuceny čekat, dokud všechny žádosti o odeslání nebo obdržení zpráv nejsou vykonány. Zde je tedy zátěž neuniformní, protože probíhá čekání na nejpozdější procesy. Při využití více vláken (zde 5 vláken na 1 proces) je upravená funkce `UpdateTileHybrid` časově náročnou. Tato funkce je převzatá implementačně z metody `UpdateTile`, avšak lépe ošetřuje situaci, kdy dlaždice daného procesu obsahuje část statického okraje domény.

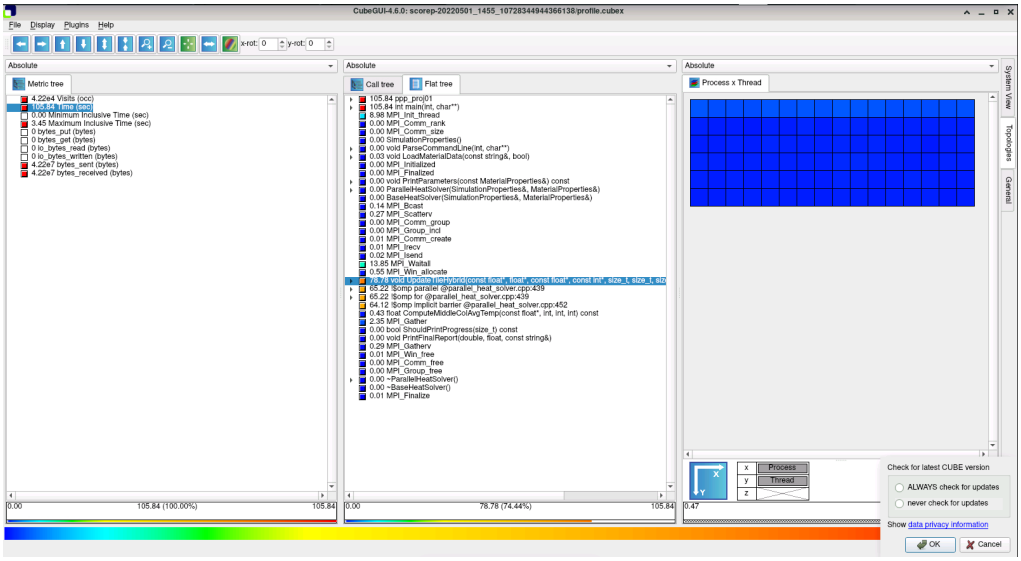
Analýza programu Vampir nám umožňuje zobrazit posloupnost událostí mezi procesy. Implicitní bariéra sloužící pro garanci ukončení paralelních výpočtů všech vláken procesu je časově nejnáročnější částí programu – jedná se tedy o největší problém při užití více vláken pro 1 proces, při užití pouze 1 vlákna na 1 proces je čas strávený ve funkci mnohem nižší. Nejvíce časově náročnou MPI funkcí je pak `MPI_Waitall`, kdy se musí čekat na dokončení neblokující komunikace.

Reference

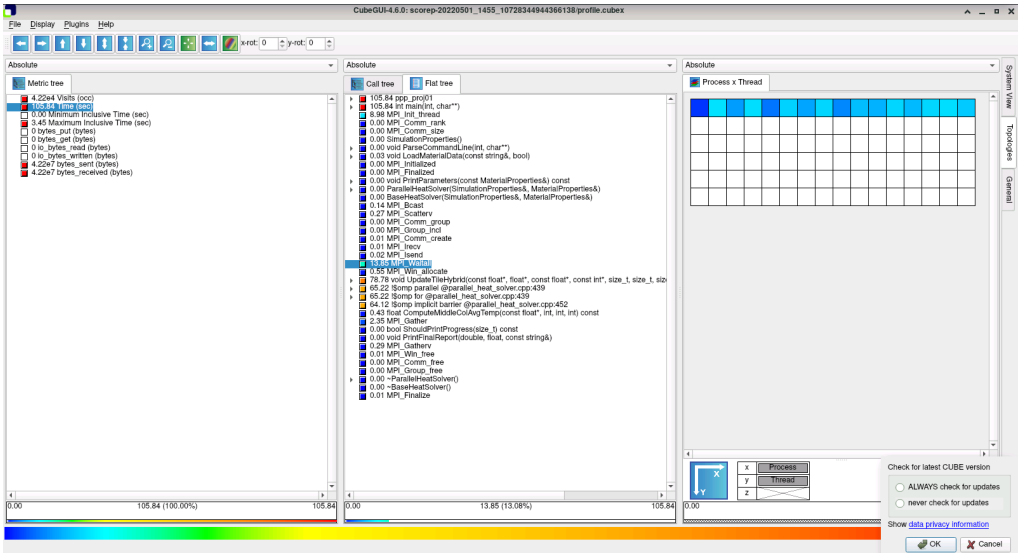
[1] BREITENFELD, M. S., POURMALK, E., BYNA, S. et al. *Achieving High Performance I/O with HDF5* [online]. Poslední změna 6. února 2020 [cit. 20. dubna 2022]. Dostupné na: <https://bit.ly/3xFUaeE>.



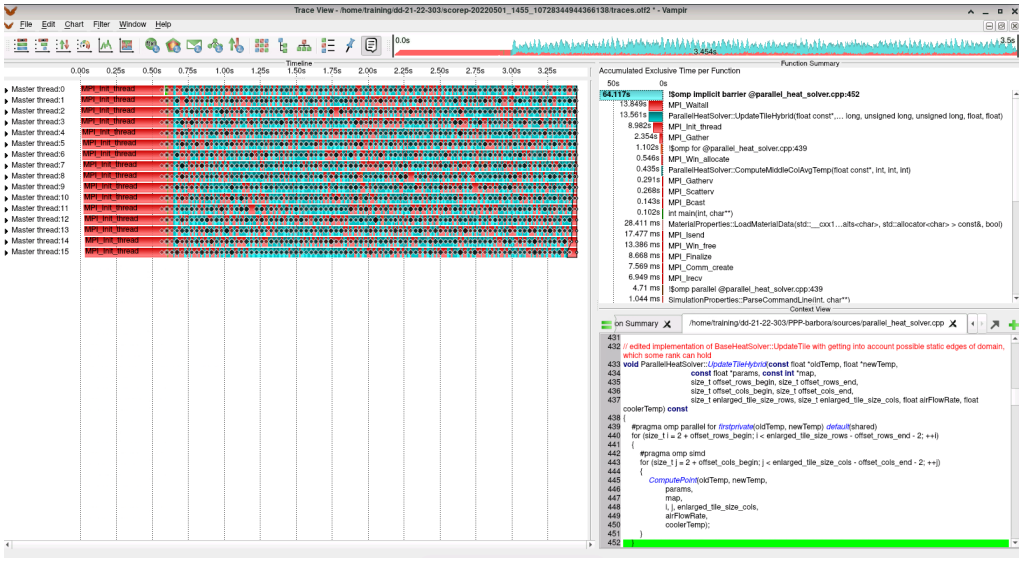
Obrázek 10: Cube – Nevyvážení zátěže nultého procesu vzhledem k ostatním pro velikost odeslaných zpráv



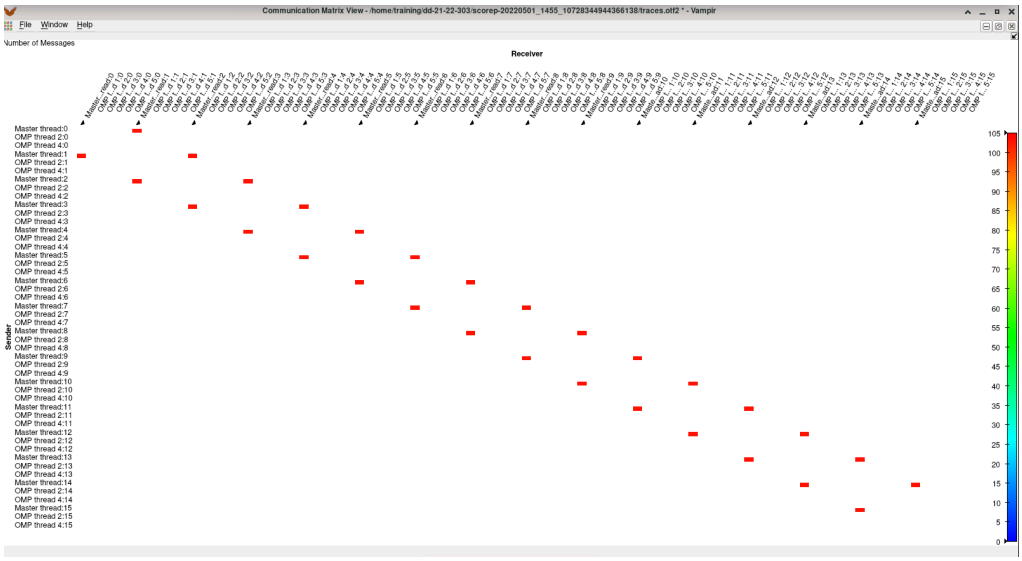
Obrázek 11: Cube – Časová náročnost jednotlivých funkcí a rozložení časové zátěže procesů a vláken při funkci Update-TileHybrid



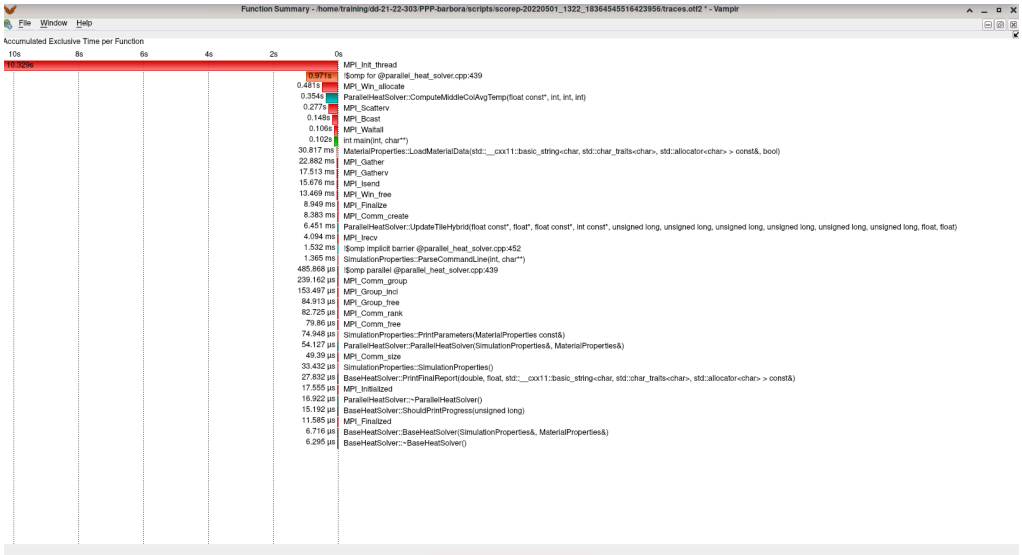
Obrázek 12: Cube – Časy čekání procesů na dokončení neblokující komunikace



Obrázek 13: Vampir – 1D dekompozice s využitím 6 vláken na 1 proces



Obrázek 14: Vampir – Komunikační matice pro 1D dekompozici



Obrázek 15: Vampir – Časy běhů funkcí pro 1D dekompozici s využitím 1 vlákna na 1 proces