Architektura a programování paralelních systémů
Projekt č. 2 - Programování s MPI

Radek Hrbáček, ihrbacek@fit.vutbr.cz
Jiří Jaroš, jarosjir@fit.vutbr.cz

Termín odevzdání: 10. května 2015 23:59
Hodnocení: až 15 bodů

1 Úvod

Cílem projektu je osvojit si základní principy programování se zasíláním zpráv s využitím knihovny MPI a ověřit škálování na několika uzlech superpočítače Anselm.

Jako testovací problém byla stejně jako v prvním projektu zvolena numerická simulace problému šíření tepla. Budeme tedy uvažovat jeden 2D řez procesorovým chladičem věžovité konstrukce, který má měděnou základnu a nosnou konstrukci, ke které je připojeno několik hliníkových žeber. Na styčné ploše chladiče s procesorem dochází k přenosu tepla z procesoru do chladiče. Toto teplo přechází „difuzí“ do žeber chladiče, kde je dále předáváno chladicímu mediu, jímž je zde vzduch. Pro realistickou simulaci uvažujeme, že vzduch proudí laminárně kolmo k ose řezu chladiče (teplo se tedy postupně odvádí pryč ze zkoumané oblasti).

Aby bylo možné analyzovat výsledky simulace, je průběh ohřívání chladiče ukládán do souboru. Tento soubor nám poslouží k vizualizaci výsledků a ověření správnosti výpočtu. Problém však nastává, pokud ukládáme data příliš často. Může tedy snadno nastat situace, kdy je I/O hlavní brzda výpočtu (nelze paralelizovat).

2 Popis numerické metody šíření tepla

Existuje několik metod, jak vyřešit problém šíření tepla numericky. Jmenujme například metody konečných diferencí, konečných elementů, metody hraničních prvků či spektrální metody. My se zaměříme na techniku nejjednodušší (bohužel i nejméně přesnou a nejpomalejší), kterou je metoda označovaná jako FDTD[1] (*Finite Difference Time Domain*), tedy metoda konečných diferencí v čase.

Zkoumanou doménu tedy diskretizujeme pomocí uniformní mřížky velikosti $N \times N$ bodů. Pro jednoduchost uvažujme mřížku čtvercového tvaru. V každém bodě mřížky definujeme potřebné parametry media. V našem případě postačuje tepelná vodivost pro měď, hliník a vzduch. Pokud daný bod domény připadá na vzduch, zavádíme rovněž koeficient perfuze (tedy jak rychle proudí vzduch okolo chladiče a odvádí teplo pryč). Dále pak v každém bodě udržujeme aktuální teplotu chladiče. Na počátku simulace je nastavena klidová teplota (výchozí hodnota je $20°C$). V místě styku chladiče s procesorem je nastavena konstantní teplota ohřívače (výchozí hodnota je $100°C$). Pro úplnost dodejme, že teplota na hranách domény je konstantní (tzv. *Dirichletova podmínka*). Protože simulujeme vývoj teploty na chladiči v čase, diskretizujeme rovněž i časovou osu a simulaci budeme provádět v předem daném počtu kroků.

Představme si nyní, že máme spočítat teplotu v čase $t+1$. K tomuto účelu využijeme jednak znalosti aktuální teploty v okolí zkoumaného bodu a jednak historii vývoje změny teploty v daném bodě v čase. Díky těmto údajům spočteme gradient teploty v prostoru a provedeme dopřednou integraci v čase. K tomuto účelu bylo vymyšlena spousta zajímavých metod (např. metoda Runge-Kutta 4. řádu, vícekrokové metody, ad.).

My opět zvolíme tu nejjednodušší techniku a budeme aproximovat gradient v prostoru pomocí jednoduchého 4-okolí. Následující rovnice definuje metodu FDTD 2. řádu v prostoru a 1. řádu v čase:

[1] `http://www.eecs.wsu.edu/~schneidj/ufdtd/ufdtd.pdf`

$$T_{t+1}[i][j] = \frac{T_t[i-1][j] + T_t[i+1][j] + T_t[i][j-1] + T_t[i][j+1] + T_t[i][j]}{5.0} \quad (1)$$

Tato rovnice zjednodušeně říká, že teplota bodu v čase $t+1$ ja dána průměrem teplot tohoto bodu a jeho 4 sousedů v čase t. Takto bychom však byli schopni simulovat šíření tepla pouze v homogenním mediu. My ale chceme použít měď, hliník a vzduch. Heterogenitu proto zavedeme pomocí koeficientu difuze, jenž definuje tepelnou vodivost. Každý člen rovnice je tedy vynásoben normalizovanou hodnotou tepelné vodivosti v daném bodě.

Abychom byli schopni simulovat odvod tepla proudícím vzduchem, je každý bod media odpovídající vzduchu upraven pomocí následujícího vztahu.

$$T_{t+1}[i][j] = (\alpha * T_0) + (1 - \alpha * T_{t+1}([i][j])) \quad (2)$$

kde α definuje relativní rychlost proudění vzduchu vzhledem k délce časového kroku a T_0 je teplota čerstvého vzduchu.

Výpočet nyní probíhá tak, že procházíme jednotlivé body a aktualizujeme teplotu. Při běžné (sekvenční i paralelní) implementaci je nutné využít dvě pole (zdrojové a cílové), tak aby se nové hodnoty počítaly vždy pouze ze starých hodnot a nedocházelo k tomu, že někteří sousedé daného bodu byli již aktualizováni a jiní ne. To by v případě paralelní verze vedlo na nedeterministické chování a každý nový běh by dával odlišné výsledky.

Abychom zjednodušili testování implementace, budeme v každém kroku počítat i průměrnou teplotu v oblasti dané jedním sloupcem bodů kolmým na základnu.

3 Knihovna HDF5 pro práci s velkými daty

HDF5[2] je knihovna (a zároveň označení pro její souborový formát) určená pro rychlé čtení/ukládání vědeckých dat na pevném disku. Jak název napovídá, data jsou v souboru uložena hierarchicky ve stromové struktuře tvořené tzv. skupinami (`groups`) a jednotlivými datasety (`datasets`) uchovávajícími uživatelská data. Datasety mají formát N-rozměrné matice určitého datového typu (`int`, `float`, `string`, ...). V rámci tohoto projektu budeme používat sériovou verzi knihovny HDF5, jenž je určená pro práci na jednom výpočetním uzlu.

HDF5 soubory mají zpravidla koncovku `.h5`. Pro zjištění obsahu takového souboru slouží utilita `h5dump`, např. příkaz

```
h5dump -H soubor.h5
```

vypíše základní informace o souboru a typu uložených dat. Více možných příkazů a pohledů na uložená data naleznete v nápovědě (`h5dump --help`).

4 Vizualizační nástroj VisIt

VisIt[3] je interaktivní (klient-server) vizualizační nástroj určený pro grafické zobrazování vědeckých dat. V tomto projektu slouží VisIt k offline zobrazení průběhu simulace uložené v hdf5 souboru, což může výrazně zjednodušit ladění programu a pomůže lépe pochopit řešený problém a dopady jednotlivých změn provedených ve zdrojovém kódu na vlastnosti výstupu. Jedna z dalších velmi používaných alternativ je např. nástroj *ParaView*[4].

5 High-level popis implementace

Po rozbalení balíčku z WISu se objeví tři hlavní složky - `DataGenerator`, `Sources` a `Scripts`. Ve složce `DataGenerator` je implementovaný generátor vstupních souborů s parametry zkoumané oblasti. Mezi tyto parametry patří např. tvar chladiče, tepelná vodivost mědi a hliníku tvořící samotný chladič, vodivost okolního vzduchu, velikost domény, teplota chladiče a procesoru ve výchozím stavu, atd. Tento program generuje soubor v hdf5 formátu, který jde následně na vstup simulátoru (složka `Sources`).

[2] `http://www.hdfgroup.org/HDF5/`
[3] https://wci.llnl.gov/simulation/computer-codes/visit/
[4] `http://www.paraview.org/`

Ve složce `Sources` je hlavní jádro projektu, simulátor šíření tepla v dané 2D doméně. Obsah tvoří tři hlavní zdrojové soubory: `BasicRoutines.cpp` obsahuje jednoduché pomocné funkce pro tisk nápovědy, zpracování parametrů příkazové řádky, atd. `MaterialProperties.cpp` obsahuje třídu, která načte data vytvořená generátorem z hdf5 souboru do paměti. Nejdůležitější je soubor `proj02.cpp`. Zde se nachází plně funkční sekvenční verze simulátoru `SequentialHeatDistribution()` a vašim úkolem je implementovat paralelní verzi ve funkci `ParallelHeatDistribution()`. Tento soubor se bude následně odevzdávat do WISu.

V poslední složce `Scripts` je sada jednoduchých PBS scriptů, pomocí kterých je možné spustit výpočet na Anselmu pro různé počty procesů. Každý script spustí simulaci pro různé velikosti domény a jeho výstupem je CSV soubor s informacemi o daných bězích a naměřenými časy. Vašim úkolem bude na základě tohoto CSV souboru popsat mj. škálovatelnost (zrychlení) a efektivitu vašeho řešení v souboru `exec.txt` (více v kapitole 10).

Jak pracovat s těmito programy a skripty je popsáno v následujících kapitolách.

5.1 Co je úkolem studenta?

- Student má povoleno editovat **pouze** zdrojový text v souboru `proj02.cpp`. Ten se bude následně odevzdávat do wisu a bude bodován.
- V souboru `proj02.cpp` je povoleno upravit, resp. doplnit implementaci **pouze** této funkce:

  `ParallelHeatDistribution()`

  Ostatní kód a funkce **musí** zůstat nedotčeny.
- Kromě samotné implementace je vaším úkolem mj. změřit zrychlení a efektivitu paralelizace na Anselmu a získanými výsledky vyplnit soubor `exec.txt`. Podrobnosti jsou popsány v následujících kapitolách.

Suma sumárum, úkolem je upravit, resp. doplnit kód paralelní funkce a vyplnit textový soubor naměřenými výsledky.

6 Popis sekvenční verze

Sekvenční verze algoritmu je implementovaná ve funkci `SequentialHeatDistribution`, kterou naleznete v souboru `proj02.cpp`. **Tuto verzi neupravujte.** Sekvenční verze slouží jako ukázka, jak korektně spočítat celou simulaci bez použití paralelizace, a zároveň je to odrazový můstek pro bodovanou implementaci paralelní verze popsané v následujících kapitolách. Pro lepší orientaci ve zdrojovém kódu jsou zde komentáře anotovány pomocí čísla nebo písmena v hranatých závorkách. Toto označení koresponduje s číslem, resp. písmenem odrážky v následujícím popisu. Výpočet sekvenční verze probíhá následovně:

1. Pokud je specifikován název výstupního hdf5 souboru parametrem `outputFileName`, k názvu se připojí suffix `_seq` (těsně před `.h5` koncovku) a dojde k vytvoření souboru. V opačném případě se průběh simulace neukládá.
2. Vytvoří se pomocné pole `tempArray`, které zaručí, že výpočet aktuální iterace bude vycházet pouze z hodnot té předchozí.
3. Hlavní pole `seqResult` i pomocné pole `tempArray` inicializujeme počátečními hodnotami (teplotami) z pole `initTemp` struktury `materialProperties`. Tyto hodnoty byly vytvořeny generátorem vlastností materiálu (složka `DataGenerator`).
4. Pro lepší přehlednost přejmenujeme pole `seqResult` na `newTemp` a `tempArray` na `oldTemp`. Na konci každé iterace budou v `oldTemp` o jednu iteraci starší výsledky oproti `newTemp` (`newTemp` tedy odpovídá času t a `oldTemp` času $t-1$.
5. Uložíme si aktuální čas pro pozdější výpočet celkové doby běhu simulace.
6. Spustíme samotnou simulaci s `nIterations` kroky (specifikováno parametrem spuštění simulátoru `-n`). V každé iteraci se po řádcích projdou všechny body domény (kromě okrajů, ty zůstávají pevné) a na základě teploty a vlastností akutálního bodu a jeho čtyř sousedů (popsáno v kapitole 2) se jeho teplota aktualizuje. V každé iteraci se tedy provedou tyto podkroky:

(a) Spočteme nové teploty všech bodů domény pomocí funkce `ComputePoint`:

i. Spočteme index aktuálního bodu a jeho čtyř sousedů (indexujeme 2D pole uložené po řádcích jako 1D pole).

ii. Normalizujeme `domainParams` (tepelnou vodivost materiálu) na hodnoty v intervalu $\langle 0; 1\rangle$ (nula značí dokonalý izolant, jednička supravodič). Tato úprava urychlí šíření tepla doménou, tudíž zmenší počet iterací celé simulace nutných pro dosažení stejného výsledku.

iii. Spočítáme novou teplotu bodu na základě teploty a tepelné vodivosti sousedů i jeho samotného. Zde je důležité uvědomit si potřebu dvou polí `oldTemp` a `newTemp`. Kdybychom používali pouze jedno pole, někteří sousedé aktuálně počítaného bodu už by měli aktualizované hodnoty z této iterace a dostávali bychom chybné výsledky. V paralelní verzi by to navíc vedlo na nedeterministické chování (pokud by souseda aktuálního bodu mělo na starosti jiné vlákno, mohlo by v prvním běhu programu aktualizovat hodnotu tohoto souseda před aktuálním bodem a podruhé třeba až po aktuálním bodu).

iv. Pokud aktuální bod reprezentuje vzduch, snížíme jeho teplotu. Tím simulujeme ochlazování chladiče aktivním větráním.

(b) Spočítáme průměrnou teplotu v prostředním sloupci domény. Tato část má větší význam až v paralelní verzi, kde bude možné porovnávat průběh této hodnoty oproti té sekvenční (musí být totožné - s odchylnou na úrovni numerické chyby *float*).

(c) Aktuální iteraci zapíšeme do souboru. Abychom omezili množství zapisovaných dat, ukládáme pouze ty iterace, jejichž index je delitelný parametrem `diskWriteIntensity`, tedy pokud má `diskWriteIntensity` hodnotu 10, uloží se každá desátá iterace.

(d) Prohodíme ukazatele polí `newTemp` a `oldTemp`.

(e) Tisk průběhu simulace v procentech a průměrné teploty v prostředním sloupci domény.

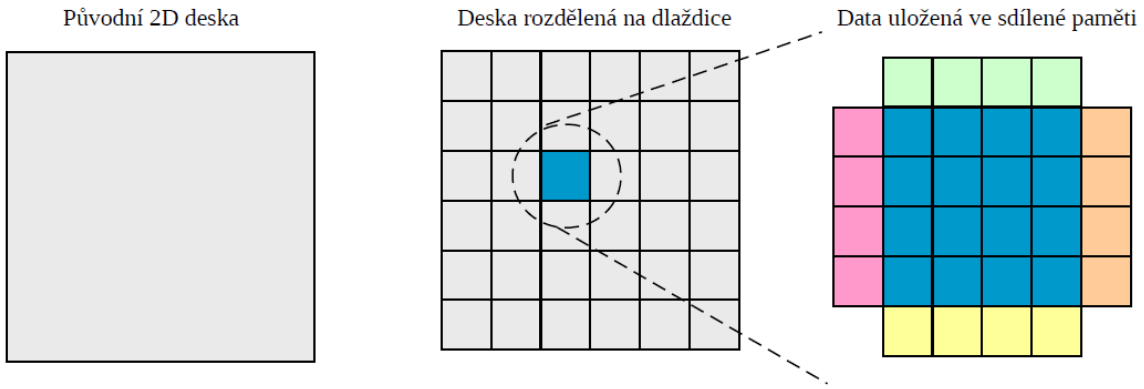
7. Tisk výsledného času simulace, jedné iterace atd. V *batch* módu (parametr programu `-b`) se vše tiskne v CSV formátu se středníkem jako oddělovačem.

8. Pokud je celkový počet iterací lichý, jsou poslední výsledky v pomocném poli `tempArray` (dřívější `oldTemp`), proto je překopírujeme do `seqResult` (dřívější `newTemp`).

7 Implementace paralelní verze [až 15 bodů]

Cílem projektu je paralelizovat sekvenční algoritmus. Každý proces bude mít přidělenu čtvercovou, resp. obdélníkovou, dlaždici. V každé iteraci bude počítat hodnoty bodů na této dlaždici. K tomu ovšem bude potřebovat i hodnoty z okolí této dlaždice (tzv. *Halo zóny*). Tyto oblasti si procesy musejí v každé iteraci vyměňovat.

Abychom dosáhli rozumného zrychlení, je potřeba co nejvíce překrýt výpočet a komunikaci. Proto bude Vašim úkolem nejprve spočítat body v těchto Halo zónách, tyto body odeslat sousedním procesům



Obrázek 1: Princip paralelizace s využitím distribuované paměti. Každý proces má jednu dlaždici, na které pracuje + body z hranic od souseda.

a až poté počítat vnitřní body dlaždice. Po spočtení vnitřních bodů dojde k přijetí Halo zón od okolních dlaždic. K tomuto účelu použijte neblokující volání MPI.

Paralelní varianta by měla vycházet ze sekvenční a tedy by měla odpovídat následující posloupnosti kroků:

1. Pokud je specifikován název výstupního hdf5 souboru parametrem `outputFileName`, k názvu se připojí suffix `_par` (těsně před `.h5` koncovku) a dojde k vytvoření souboru. V opačném případě se průběh simulace neukládá. **Soubor bude vytvářet pouze jeden (nultý) proces!**

2. Doménu o straně N je nutné rozdělit na stejně velké dlaždice. Budeme uvažovat pouze velikosti rovné mocninám dvojky. Rovněž počet procesů P bude vždy mocnina dvojky. Pokud bude velikost domény sudá mocnina dvojky, pak ji lze rozdělit na čtvercové dlaždice o straně $N/\sqrt{P}$. Pokud bude velikost domény lichou mocninou dvojky, pak je nutné vytvořit obdélníkové dlaždice (s šířkou dvakrát větší než výškou).

3. Každý proces si vytvoří pomocná pole pro svou dlaždici a okolí:

   - 2 pole pro data (opět je nutné zajistit, aby výpočet aktuální iterace vycházel pouze z hodnot té předchozí)
   - 1 pole pro materiál
   - 1 pole pro teplotní vlasnosti materiálu

4. Počáteční hodnoty pro tato pole má k dispozici pouze nultý proces v polích `initTemp`, `domainMap`, `domainParams` struktury `materialProperties`. Tyto hodnoty byly vytvořeny generátorem vlastností materiálu (složka `DataGenerator`). Nultý proces bude mít za úkol tato data distribuovat ostatním procesům.

5. Uložíme si aktuální čas pro pozdější výpočet celkové doby běhu simulace (stačí, aby toto provedl nultý proces).

6. Spustíme samotnou simulaci s `nIterations` kroky.

7. V každé iteraci se projdou všechny body dlaždice (kromě okrajů domény, ty zůstávají pevné) a jejich teploty se aktualizují.

   (a) Výpočet probíhá analogicky k sekvenční verzi.

   (b) Pro dosažení co nejvyššího zrychlení je nutné nejprve spočítat okrajové body dlaždice, tyto odeslat okolním procesům, následně spočítat vnitřní body. Tím dojde k překrytí výpočtu vnitřních bodů a komunikace.

   (c) Zápis do souboru provádějte pomocí nultého uzlu, ve kterém shromáždíte data ze všech dlaždic.

8. Tisk výsledného času simulace, jedné iterace atd. V *batch* módu (parametr programu `-b`) se vše tiskne v CSV formátu se středníkem jako oddělovačem.

9. Výsledné teploty z jednotlivých dlaždic je nutné shromáždit v nultém procesu a vrátit pomocí parametru `parResults`.

Při implementaci můžete využít následujících tipů:

- Pomocí `MPI_Type_vector` nebo `MPI_Type_create_subarray` si můžete vytvořit datové typy pro adresování např. dlaždice uvnitř domény nebo sloupce uvnitř dlaždice. Pozor, zřejmě bude nutné ošetřit tyto typy pomocí `MPI_Type_create_resized`.

- Data okolních bodů můžete držet buď v samostatných polích nebo můžete dlaždici vytvořit o něco větší.

- Pro výpočet průměrné hodnoty v prostředním sloupci můžete využít vlastní komunikátor.

8 Spuštění projektu

8.1 Spuštění na lokálním PC

Implementaci můžete testovat na libovolném stroji s Linuxem. Je nutné mít nainstalovaný C++ kompilátor, hdf5 knihovnu v sériové verzi a volitelně nástroj VisIt, který vizualizuje vypočtenou simulaci a může velmi usnadnit ladění aplikace. Postup je následující:

1. Ověřte, že máte nainstalován kompilátor jazyka C++.
2. Stáhněte si hdf5 knihovnu z této adresy: `http://www.hdfgroup.org/ftp/HDF5/current/src/hdf5-1.8.14.tar`
3. Tento archiv rozbalte a spusťte configure script:

   ```
   ./configure --prefix=CESTA
   ```

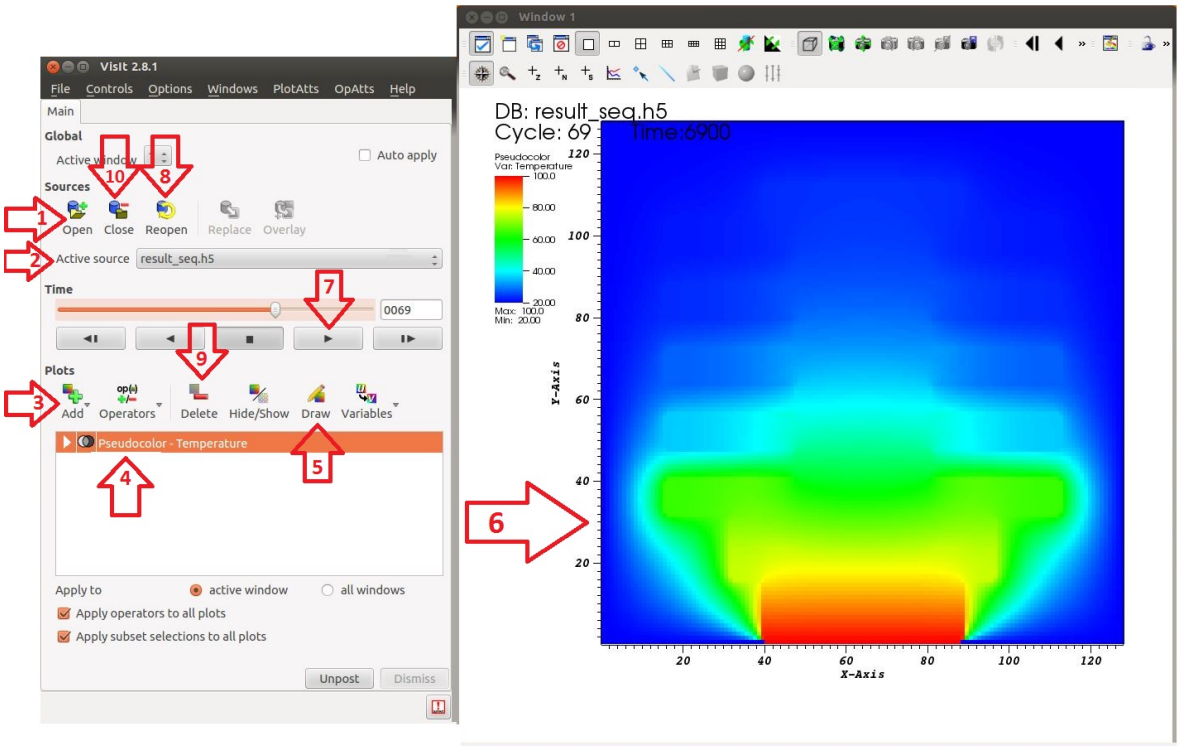
   kde CESTA by pro snažší linkování měla být v systémové proměnné `PATH`, nejlépe tedy `/usr/local/` nebo podobně. Následně přeložte knihovnu příkazem `make` a nainstaluje s `make install` (s příkazem `sudo`, pokud instalujete do systému).
4. Z WISu si stáhněte a rozbalte zdrojové soubory projektu.
5. Makefily ve složkách `Source` a `DataGenerator` jsou aktuálně nastaveny pro překlad na superpočítači Anselm s využitím Intel kompilátoru. Pro překlad na lokálním PC bude pravděpodobně nutné změnit cestu k hdf5 knihovně v proměnné `HDF5_DIR` (na `CESTA` z configure scriptu), použitý kompilátor (např. na `g++`), `-openmp` přepsat na `-fopenmp`, odstranit `-xhost` a `-xavx` nahradit za `-mavx`. Pokud pracujete ve VirtualBoxu, bude pravděpodobně nutné odstranit `-mavx`, popř. ho zkusit nahradit za `-msse`.
6. Spustťe příkaz `make`, jak ve složce `DataGenerator`, tak ve složce `Source`. Měly by vzniknout spustitelné soubory `arc_generator` a `arc_proj01`.
7. Nyní zkusíme spustit nějakou jednodušší simulaci. Ve složce `DataGenerators` zadejte `make test`. Vznikne soubor `material.h5`. Následně zadejte ve složce `Source` příkaz `make test`. Spustí se sériová verze simulace, jejímž výstupem je soubor `result_seq.h5`.
8. Výsledek simulace uložený v souboru `result_seq.h5` nyní zkusíme vizualizovat. Z domovské stránky[5] si stáhněte nástroj VisIt (doporučená verze je 2.8.1). Po rozbalení spusťte spustitelný soubor s názvem `visit`. Otevře se hlavní okno aplikace (obrázek 2, čísla v závorkách v následujícím popisu odpovídají číslům v šipkách na obrázku). Klikněte na tlačítko `Open` (1) v sekci `Sources` a vyberte soubor `result_seq.h5`. Název souboru by se měl objevit v `Active source` (2). Nyní můžeme začít vizualizovat. Klikněte na tlačítko `Add` (3), vyberte `Pseudocolor` a následně `Temperature`. V bílém okénku se objeví vybraný filtr (4). Nyní klikněte na tlačítko `Draw` (5) a v novém okně (6) se vykreslí první časový krok vizualizace, kdy ve spodní části je krátký proužek s teplotou $100°C$ a ve zbytku je teplota $20°C$ (na obrázku je zobrazený 69. časový krok). Celou simulaci lze spustit v sekci `Time` kliknutím na tlačítko `Play` (7), popř. ručně přecházet mezi jednotlivými časovými kroky pomocí táhla. Pro znovuotevření souboru (např. po novém běhu simulace) slouží tlačítko `Reopen` (8). To ale naneštěstí ne vždy funguje, takže je většinou nutné smazat všechny filtry tlačítkem `Delete` (9), soubor zavřít tlačítkem `Close` (10) a znovu otevřít.

8.2 Spuštění na Anselmu

Pro vývoj a ladění aplikace bude pravděpodobně stačit lokální PC/merlin, měření výsledků a časů ale provádějte na Anselmu, který má 209 výpočetních uzlů, každý s 2×8 jádrovými procesory Intel Xeon. Ve WISu ve složce `Ostatní` je prezentace `Anselm_intro.pdf`, ve které jsou shrnuty základní informace o tomto stroji. Více podrobností o Anselmu lze najít v dokumentaci[6].

Na Anselm se přihlásíte přes konzoli v Linuxu nebo Putty ve Windows příkazem

[5] `https://wci.llnl.gov/simulation/computer-codes/visit/executables`
[6] `https://docs.it4i.cz/anselm-cluster-documentation`



Obrázek 2: VisIt - Popis grafického uživatelského rozhraní.

```
ssh login@anselm.it4i.cz
```

kde login je přihlasovací jméno, které dostanete na lístečku od Jirky Jaroše (pozn.: od 13. dubna bude možné se přihlásit na Anselm pouze pomocí SSH klíče, bude proto nutné vygenerovat si svůj privátní a veřejný klíč[7]). Po zadání hesla jste přihlášeni na login uzlu. Login uzel slouží pouze pro nenáročné operace, jako je přenos souborů přes internet, kompilace programů, atd. Nesmí se zde spouštět žádné dlouhotrvající náročné výpočty (pod pohrůžkou BANu od administrátorů Anselmu), na to slouží výpočetní uzly.

Zdrojové soubory projektu můžete přenést na Anselm např. protokolem SCP. Příkaz

```
scp myfile login@anselm.it4i.cz:/home/login/target-folder
```

přenese soubor `myfile` z lokálního PC na Anselm do složky `target-folder` vašeho domovského adresáře. Pro přenos celé složky je nutné doplnit parametr `-r`

```
scp -r myfolder login@anselm.it4i.cz:/home/login/target-folder
```

Opět místo loginu zadejte svoje přihlašovací jméno z lístečku. Alternativně je možné namountovat vzdálený disk ke svému lokálnímu PC. Na svém lokálním PC spusťte příkazy

```
mkdir some-folder
sshfs login@anselm.it4i.cz:. some-folder
```

a do složky `some-folder` se zobrazí celý váš home adresář z Anselmu. Soubory poté můžete jednoduše kopírovat příkazem `cp` nebo přetahováním myši.

Nyní jsme připojeni a máme překopírovány zdrojáky projektu na Anselmu. Další krok je překlad. Pro překlad projektu je nutné natáhnou některé moduly, v našem případě Intel kompilátor, MPI a knihovnu hdf5, příkazem

```
module load intel/15.2.164 impi hdf5/1.8.13
```

Nyní lze přeložit projekt příkazem `make` ve složkách `Source` a `DataGenerator`. Na login uzlu můžete zkusit jednoduchý testovací běh příkazem `make test`.

[7] `https://docs.it4i.cz/anselm-cluster-documentation/accessing-the-cluster/shell-and-data-access`

Máme přeloženo a můžeme zkusit něco spustit. Pro testování paralelní verze je nutné připojit se na výpočetní uzel. Ze začátku je vhodné pracovat v interaktivním módu, kdy máme k dispozici konzoli a je možné ručně spouštět jednotlivé běhy s různými nastaveními. Z login uzlu se na výpočetní uzel připojíme příkazem

```
qsub -q qexp -A it4i-8-3 -l select=1:ncpus=16:mpiprocs=16:ompthreads=1 -I
```

Připojení by mělo proběhnout v řádu jednotek až desítek sekund, někdy déle. Parametr `-q` specifikuje frontu (v tomto případě expresní `qexp`), `-A` specifikuje název interního it4i projektu, `-l` žádá o výpočetní zdroje, v tomto případě žádáme o 1 uzel a všech 16 jader tohoto uzlu. Nakonec `-I` značí interaktivní mód. Nyní máte po dobu jedné hodiny k dispozici všech 16 jader a 64GB operační paměti daného uzlu. Po uplynutí této doby vám bude plánovačem uzel odebrán (neuložená práce bude ztracena!) a je třeba znovu zažádat o uzel přízem `qsub...` uvedeným výše a po připojení opět natáhnout požadované moduly. Pro krátké běhy (např. 1 iterace) v debug modu lze využít i login uzel. Snažte se využívat frontu `qexp`, popřípadě frontu `qprod`.

Přestože Anselm umožňuje spustit VisIt vzdáleně s grafickým uživatelským rozhraním (přes VNC), bude jednoduší provádět vizualizaci na svém lokálním PC, kde máte VisIt nainstalovaný. Využijte namoutování disku příkazem `sshfs` popsaným výše pro přístup k výstupním hdf5 souborům simulace spočítané na Anselmu. Proces vizualizace pak probíhá totožným způsobem popsaným v části práce na lokálním PC.

9 Testování implementace

Projekt má dvě části, generátor vlastností materiálu v doméně (složka `DataGenerator`) a samotný simulátor šíření tepla (složka `Sources`). V kapitole 8 byly popsány kroky nutné k rozchození celého projektu, v této kapitole si popíšeme různá nastavení a možnosti spuštění a testování.

Generátor má následující parametry spuštění:

```
  -o <string>  - název výstupního hdf5 souboru
  -N <integer> - velikost domény (pouze mocnina dvou až do 1024,
                 více by se počítalo příliš dlouho)
  -H <float>   - teplota procesoru
  -C <float>   - teplota chladiče a okolního vzduchu v počátečním stavu
  -h           - zobrazí nápovědu
```

Hodnoty parametrů si můžete zvolit v rámci testování prakticky libovolné, je však doporučené mít dostatečný rozdíl mezi teplotou procesoru a teplotou chladiče a zároveň počítat na doméně rozumné velikosti mocniny dvou, maximálně 1024^2. Výstupní soubory simulace by jinak mohly nabobtnat na stovky megabajtů až gigabajty, a to po síti nikdo přenášet nechce.

Simulátor šíření tepla má tyto parametry:

```
Povinné argumenty:
  -m <0-2>     - mód simulace (mód 0 - sekvenční verze, mód 1 - paralelní verze)
  -n <integer> - počet iterací (časových kroků) simulace, více == delší běh
  -w <integer> - hustota zápisu na disk (např. hodnota 10 znamená, že se do souboru
                 zapíše výsledek každé desáté iterace)
  -i <string>  - název souboru s vlastnostmi materiálu v doméně (generuje DataGenerator)

Volitelné argumenty:
  -o <string>  - název výstupního souboru; pokud není zadán, nic se nezapisuje
  -a <float>   - air flow rate (rychlost proudění vzduchu žebry chladiče), rozumné
                 hodnoty jsou v intervalu <0.5, 0.0001> (0.5 značí odebrání 50% tepla
                 po každém časovém kroku)
  -d           - debug mode - vypsání výsledných teplot v doméně na stdout
  -v           - verification mode - porovná výsledky sekvenční a paralelní verze
                 a vypíše OK nebo FAILED
  -b           - batch mode - informace o běhu jsou na stdout vypisovány v CSV formátu
```

Počet iterací a hlavně hustotu zápisu do souboru se snažte držet na rozumné hodnotě tak, aby výpočet netrval více než minutu a zároveň jste si s danou velikostí výstupního souboru nezaplnili celý hard disk.

Výsledky obou paralelních verzí musí být shodné se sekvenční verzí (výsledky je myšlena průměrná teplota v prostředním sloupci a rozložení teplot v doméně). Shodu lze testovat parametrem `-v`, vypsání teplot celé domény na stdout provádí parametr `-d`.

Dejte pozor: některé chyby se projeví jen za určitého počtu procesů a velikosti domény.

10 Ověření škálovaní, zrychlení a efektivity

V rámci projektu je požadováno mimo korektního chování dosáhnout i rozumné zrychlení. Parametry a nastavení aplikace, které se budou hodnotit, jsou uvedeny ve skriptech `run_X.pbs` a souboru `exec.txt` ve složce `Scripts`. Vstupní data je možné vygenerovat pomocí skriptu `gen.sh`. Po spuštění všech skriptů na Anselmu příkazem `qsub run_X.pbs` získate jako výstup soubory `out_X.csv`. Z těchto naměřených údajů se vyplní soubor `exec.txt`. Do souboru **exec.txt** uveďte:

- Výpočetní čas pro sekvenční verzi a paralelní verzi (2–8 uzlů) na mřížkách 256^2 až 4096^2 v násobcích dvou, a to pro zapnuté ukládání do souboru a vypnuté ukládání.
- Dosažená zrychlení přepočteno na 1 iteraci vůči sekvenční verzi.
- Dosaženou efektivitu přepočteno na 1 iteraci.

11 Odevzdání a bodování

Odevzdávají se soubory **proj02.cpp** a **exec.txt**, které zabalíte pomocí tar.gz či zip, pojmenujete vaším loginem a odevzdáte do WISu. Pokud budete mít libovolný dotaz, připomínku nebo návrh, neváhejte se ozvat na email, případně přispějte do fóra.

Za úplné vypracování projektu je možné dostat 15 bodů, je však třeba splnit následující požadavky:

- Základní verze bez překrytí komunikace, správné předávání Halo zón, správné výsledky, výpočet průměrné teploty v prostředním sloupci (7 bodů)
- Překrytí komunikace a výpočtu (5 bodů)
- Efektivita (3 body)