Vorgehensweise

Ziel der Aufgabe war es, die Wärmeleitungsgleichung zu visualisieren. Dafür sollten die in der Vorlesung eingeführten Stencil Berechnungen für Parallelität genutzt werden. Der bereitgestellte Code macht es möglich, die Rechnungen in `heat_update.cpp` zu visualisieren. Die Simulation wird durch die Funktionen in `heat_init.cpp` mit den in `constants.h` festgelegten Konstanten gestartet. Um den nächsten Schritt der Simulation zu berechnen, mussten die Funktionen in `heat_update.cpp` implementiert werden:

1. `start_halo_exchange`: Hier findet die non-blocking Kommunikation zwischen den Prozessen statt. Dafür wurde in `heat.h` ein Struct `ParallelData` definiert, die einen kartesischen Kommunikator nutzt. Außerdem hat jeder Prozess ein Datenfeld vom Typ `Field`, in dem die Temperaturwerte gespeichert sind. Das Feld hat auch an jeder Seite ein Ghostlayer, in dem die entsprechenden Daten, die mit `MPI_Irecv` von den anderen Prozessen empfangen werden. Für die Kommunikation wird mit `MPI_Isend` und `MPI_Irecv` non-blocking Kommunikation genutzt, damit die Stencil-Updates, die in ihrer Berechnung unabhängig von Daten aus anderen Prozessen sind, bereits ausgeführt werden können. Für die Implementierung von `MPI_Isend` und `MPI_Irecv` wurden jeweils Datentypen definiert, die eine gesamte Zeile bzw. Spalte repräsentieren. Der Stand der Sende- & Empfangsfunktionen wird in einem `requests` Array gespeichert.

2. `update_interior_temperature`: Wie bereits erwähnt, können die Rechnungen innerhalb des Datenfelds bereits ausgeführt werden, bevor gesichert ist, dass die Kommunikation zwischen den Prozessen fertig ist. Um die Temperaturwerte aus den Werten des Feldes aus dem vorigen Simulationsschritt (`prev`) zu berechnen, nutzt man die gegebene Formel. Da mit einem 5-point stencil gearbeitet wird, werden die Werte rund um den zu berechnenden Punkt benötigt. Also werden zunächst die Indizes dieser umliegenden Punkte mit jedem Berechnungsschritt initialisiert. Mit einem nested for loop kann nun für jeden Punkt innerhalb des Datenfelds (nicht am Rand) den Temperaturwert updaten.

3. `complete_halo_exchange`: Da nun noch die Berechnung der Temperaturwerte am Rand des Datenfeldes aussteht, muss erst die Kommunikation zwischen den Prozessen fertiggestellt sein. Daher wird in dieser Funktion einfach die MPI Funktion `MPI_Waitall` aufgerufen, die mit dem `requests` Array als Argument erst MPI_SUCCESS zurückgibt, wenn alle `MPI_Isend` und `MPI_Irecv` Aufrufe fertig sind.

4. `update_boundary_temperature`: Nun muss die Berechnung der Temperaturwerte in der jeweils zweiten und vorletzten Zeile bzw. Spalte ausgeführt werden (erste und letzte sind das Ghostlayer). Dafür wird genauso vorgegangen, wie in `update_interior_temperature`.

In dieser Reihenfolge werden die obigen Funktionen in `main.cpp` aufgerufen, um den nächsten Simulationsschritt zu berechnen. Als letztes musste der `timeStep` berechnet werden. Die Formel dafür wurde ebenfalls gegeben.
Die Laufzeit des Programms wurde durch das Parallelisieren der Temperaturberechnung innerhalb der Datenfelder beschleunigt. Allerdings sind rechenaufwendige Teile des Programms nicht parallelisierbar, da das gesamte Datenfeld bei jedem Checkpoint für das Sichern der .png files zusammengesetzt werden muss. Die Parallelisierung, die hier durchgeführt wurde, lohnt sich trotzdem, da das Berechnen der Randwerte, das sequenziell ausgeführt werden muss, wesentlich schneller ist als das Berechnen der inneren Werte. Das ist natürlich nur der Fall, wenn das Datenfeld für jeden Prozess groß genug ist.

Ändern der Konstanten

Mit den gegebenen Kontanten ist die Wärmeleitung rund um den Kreis in der Mitte und am Rand zu beobachten. Durch das Ändern der Konstanten kann der Einfluss von diesen auf die Simulation bestimmt werden. In den beigefügten .png files ist jeweils der Stand am Ende der Simulation zu sehen.

Simulation Steps

Erhöht man die Anzahl der Simulationsschritte, verschwimmt der Rand des Kreises umso mehr, denn je länger die Simulation läuft, desto mehr Wärmeaustausch findet statt.

Grid Size

Verändert man die Grid Size, also `DEFAULT_ROWS` bzw. `DEFAULT_COLS`, ist zu beobachten, dass bei einem kleineren Grid größere Unterschiede nach 1000 Simulationsschritten bemerkbar sind, da das .png file mit weniger Pixeln kleinere Differenzen nicht mehr so gut darstellen kann. Führt man die Simulation mit einem größeren Radius aus, kann man auch den Wärmeaustausch des Kreises mit dem Rand beobachten.

Δx, Δy

Verkleinert man Δx und Δy, sind größere Effekte zu beobachten, das folgt dadurch, dass die beiden Werte quadratische Teiler in der Wärmeleitungsgleichung sind. Wenn beispielsweise nur Δx geändert wird, sind Unterschiede in x-Richtung zu beobachten und die Simulation wird dadurch verzerrt.

Initialized Temperatures

Je näher die Temperaturen in und außerhalb aneinanderliegen, desto schneller findet der Wärmeausgleich statt.

Diffusion Constant

Je größer */alpha* ist, desto schneller findet sind Änderungen im Wärmeaustausch zu beobachten.

Zusammenarbeit

Es haben alle Gruppenmitglieder in gleichen Anteilen zur Ausarbeitung beigetragen.