|  |  |
| --- | --- |
|  | tud_logo |
|  | Hochleistungssimulation im Ingenieurwesen 2. Hausübung |
|  |  |
|  |  |
|  | Xiangjun Wei 2594637 Bauingenieurwesen M.Sc  Guanlin Wang 2872498 Bauingenieurwesen M.Sc  Wintersemester 2018/19  17. Jan 2019 |
|  |  |

Inhaltsverzeichnis

[1. Einleitung 3](#_Toc535527108)

[2. MPI Programmen 3](#_Toc535527109)

[2.1. Sequentiell 3](#_Toc535527110)

[2.2. Parallel 3](#_Toc535527111)

[3. Ergebnisse auf eigene PC 3](#_Toc535527112)

[3.1. Sequentielles Ergebnis 3](#_Toc535527113)

[3.2. Parallelisiertes Ergebnis 4](#_Toc535527114)

[4. Ergebnisse auf Lichtenberg HLR 4](#_Toc535527115)

[4.1. Vergleich zwischen Parallel und Sequentiell 4](#_Toc535527116)

[4.2. Vergleich zwischen verschiedene p 4](#_Toc535527117)

[4.3. Vergleich zwischen verschiedene unter bestimmte m 5](#_Toc535527118)

[5. Aufgetretene Probleme 5](#_Toc535527119)

# Einleitung

In diese Übung machen wir die parallelisierte Matrixmultiplikation für quadratische Matrizen. Wir schreiben den parallelisierten Code mit MPJ nach Cannon-Algorithmus. Darüber hinaus rechnen wir die gr0ße Matrizen im HLR.

# MPI Programmen

## Sequentiell

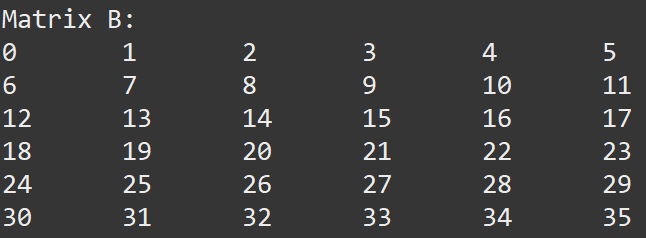
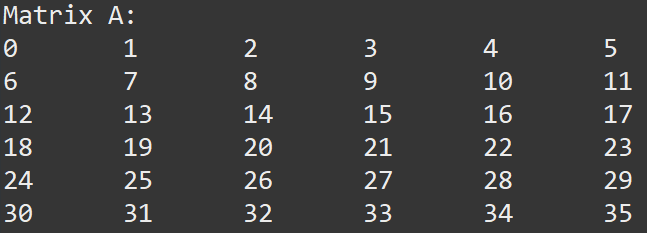
Wir schreiben den sequentiellen Code in Java mit den for-Schleife.

## Parallel

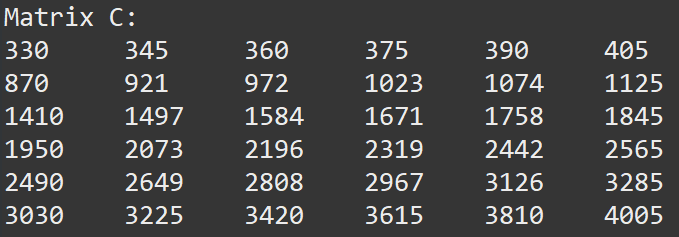
Für die parallelisierte MPI Code erstellen wir zuerst den „Cartcommunicator“. Dann machen wir die blockweise Verteilung der Matrizen für jeden Prozessor. Nachfolgend definieren wir „Shiftparameter“ für Cannon-Algorithmus. Dann verschieben wir die Matrizen und berechnen gleichzeitig. Zum Schluss setzen wir die Ergebnisse von allen Prozessoren zusammen.

# Ergebnisse auf eigene PC

Wir berechnen die Matrixmultiplikation: C = A\*B+C. A, B und C sind 6\*6 Matrizen und die Prozessoren sind 3\*3.



Ergebnis:



## Sequentielles Ergebnis

Rechnungszeit: 6 millisecond

## Parallelisiertes Ergebnis

Rechnungszeit: 174 millisecond.

# Ergebnisse auf Lichtenberg HLR

## Vergleich zwischen Parallel und Sequentiell

## Vergleich zwischen verschiedene p

## Vergleich zwischen verschiedene unter bestimmte m

# Aufgetretene Probleme

Wir treffen viele Probleme, wenn wir die Codes schreiben. Das Erstes ist die Zusammensetzung der Ergebnissematrix von allen Prozessoren. Wir lösen diese Probleme mit „MPI.Allreduce“ Methode. Wir benutzen die 1-Dimension Vektor von Ergebnis. Dann plus einen neuen Vektor, in dem alle Elemente gleich 0 sind.

Das Zweites Problem ist Benutzung des HLRs. Wir sind nicht vertraut mit Linux Befehle, so dass wir viele Hilfe brauchen. Wir müssen mich hier bei der Frau Möller bedanken.