

Programmierung von CAx-Systemen

Optimierung von C++-Programmen

1. Aufgabenstellung

2. Tipps zur C++-Programmierung

3. Internet-Links, Quellenangaben

Aufgabe

- Implementieren Sie eine einfache C++-Klasse zur Verarbeitung von quadratischen Matrizen.
- Programmieren Sie eine Funktion zur Matrix-Matrix-Multiplikation.
- Überprüfen Sie, ob Ihre Funktion korrekte Ergebnisse liefert: Multiplizieren Sie große Matrizen, die mit Zufallszahlen gefüllt sind. Führen Sie dieselben Multiplikationen unter Nutzung der Lineare-Algebra-Bibliothek „Armadillo“ aus. Vergleichen Sie die Ergebnisse.
- Vergleichen Sie die Laufzeit Ihrer Multiplikationsfunktion mit Armadillo. *vergl. [1]*

```
09:42:05: Starting U:\CAx\build-MatrixSpeed-Desktop_Qt_6_3_1_MinGW_64_bit-Release\release\MatrixSpeed.exe...
 100;    0.001;    0.001;    0.000;    0.001;    0.096
 125;    0.002;    0.001;    0.001;    0.000;    0.001
 156;    0.004;    0.003;    0.001;    0.001;    0.000
 195;    0.007;    0.007;    0.002;    0.000;    0.001
 243;    0.014;    0.013;    0.004;    0.001;    0.001
 303;    0.026;    0.026;    0.007;    0.003;    0.001
 378;    0.050;    0.050;    0.012;    0.003;    0.000
 472;    0.099;    0.098;    0.024;    0.006;    0.002
 590;    0.193;    0.191;    0.048;    0.008;    0.002
 737;    0.387;    0.375;    0.091;    0.015;    0.003
 921;    0.937;    0.730;    0.190;    0.027;    0.007
1151;    2.262;    1.492;    0.450;    0.061;    0.011
1438;    5.519;    3.078;    1.003;    0.157;    0.026
```

1. Aufgabenstellung

2. Tipps

3. Internet-Links, Quellenangaben

Compiler-Optionen, Optimizer

MatrixSpeed.pro @ MatrixSpeed - Qt Creator

Projekte

- MatrixSpeed
 - MatrixSpeed.pro
 - Quelldateien
 - main.cpp

Projekt: MatrixSpeed
Kit: Desktop Qt 5.12.12 MinGW 64-bit
Deployment: Deployment-Konfiguration
Ausführung: MatrixSpeed

Erstellen

- Debug
- Profile
- Release

```
1 TEMPLATE = app
2 CONFIG += console c++20
3 CONFIG -= app_bundle
4 CONFIG -= qt
5
6 SOURCES += main.cpp
7 INCLUDEPATH += armadillo-12.8.4/include
8
9 ##### Windows #####
10 LIBS += -L$$PWD/armadillo-12.8.4/examples/lib_win64 -lopenblas
11 #QMAKE_CXXFLAGS+= -Wall -Ofast -ffast-math -ftree-vectorize -march=native -fopenmp
12 #QMAKE_LFLAGS += -Wall -Ofast -ffast-math -ftree-vectorize -march=native -fopenmp
13
14 ##### Apple M1 #####
15 #QMAKE_CXXFLAGS+= -Wall -Ofast -ffast-math -ftree-vectorize -march=native -framework Accelerate
16 #QMAKE_LFLAGS += -Wall -Ofast -ffast-math -ftree-vectorize -march=native -framework Accelerate
17
```

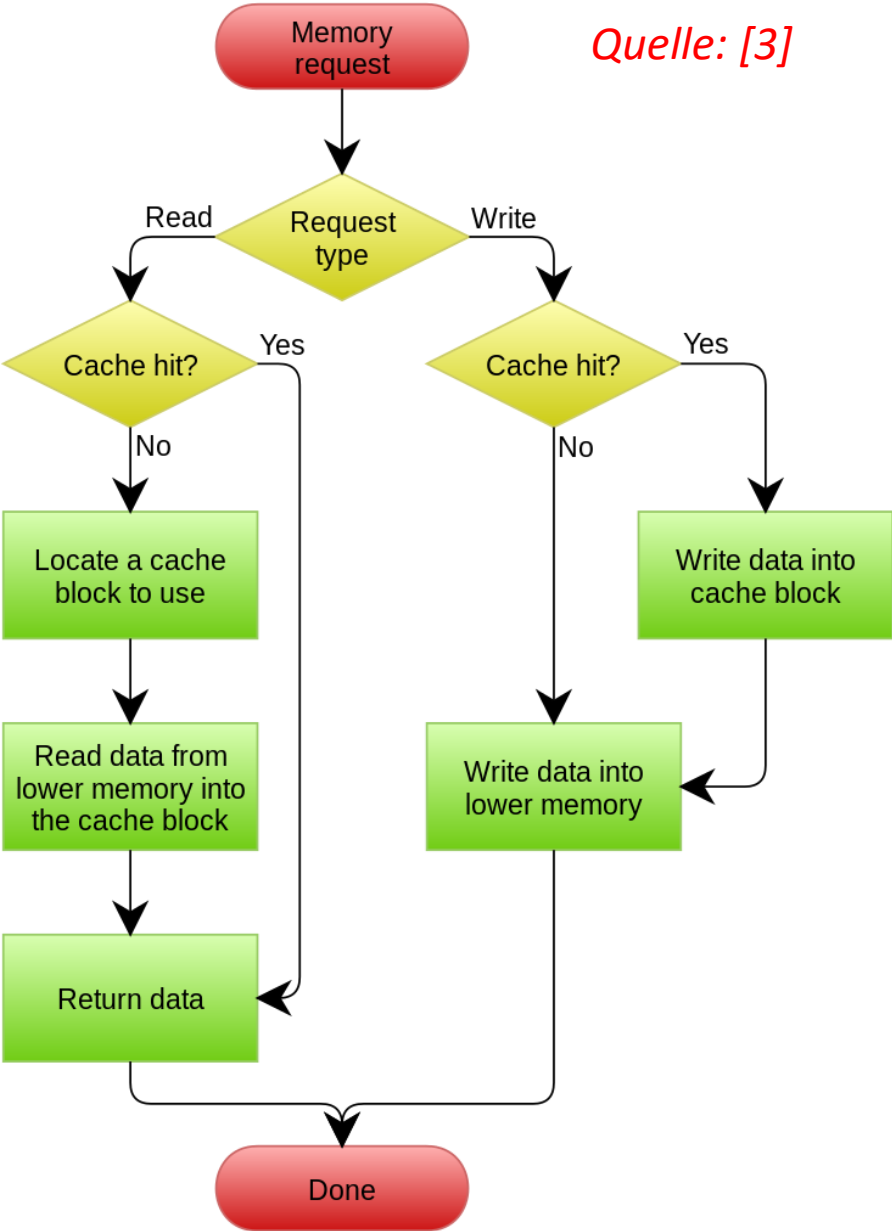
vergl. [2]

Suchmuster (Strg+K) 1 Probleme 2 Suchergebnisse 3 Ausgabe der Anwendung 4 Kompilierung 5 QML-Debuggerkonsole 6 Allgemeine Ausgaben 8 Testergebnisse

Cache-Speicher

Quelle: [3]

„Cache Miss“
... kostet Zeit!



1. Aufgabenstellung
2. Tipps zur C++-Programmierung
3. Internet-Links, Quellenangaben

Internet-Links

- [1] Seite „Armadillo, C++ library for linear algebra & scientific computing“
<https://arma.sourceforge.net/>
(Abgerufen: 22.06.2023)
- [2] Seite „Options That Control Optimization“ In: Free Software Foundation, GNU Compilers Documentation
<https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html>
(Abgerufen: 22.06.2023)
- [3] Datei „Write-through with no-write-allocation.svg“ In: Wikimedia Commons
https://commons.wikimedia.org/w/index.php?title=File:Write-through_with_no-write-allocation.svg&oldid=603836429
(Abgerufen: 22.06.2023)
- [4] Datei „OpenMP“ In: Wikipedia – Die freie Enzyklopädie
<https://de.wikipedia.org/w/index.php?title=OpenMP&oldid=217641231>
(Abgerufen: 22.06.2023)

Internet-Links

- [5] Vipul Vaibhaw: „Matrix Multiplication: Optimizing the code from 6 hours to 1 sec“
<https://vaibhaw-vipul.medium.com/matrix-multiplication-optimizing-the-code-from-6-hours-to-1-sec-70889d33dcfa>
(Abgerufen: 22.06.2023)
- [6] Seite „Matrix Multiplication“ In: Algorithmica, an open-access web book dedicated to the art and science of computing
<https://en.algorithmica.org/hpc/algorithms/matmul/>
(Abgerufen: 22.06.2023)