Multithreading

Anselm de Jonge

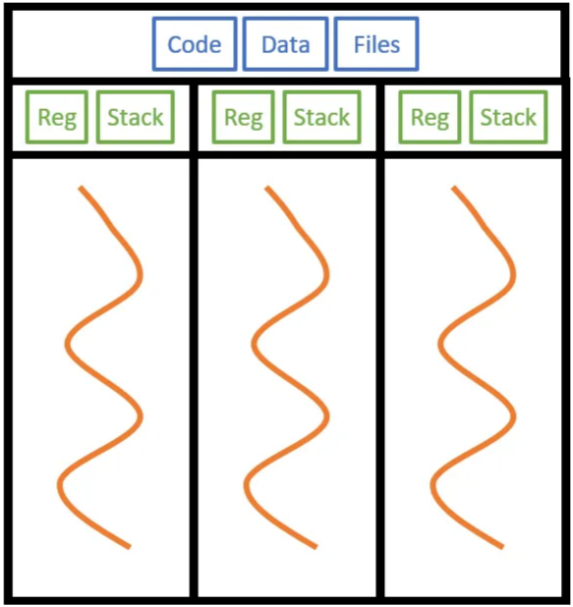
December 8, 2023

Was ist Multithreading?

Aufteilung meherer Programmaufträge auf parallel laufenden "Threads".

Beschleunigung des Programmablaufs.

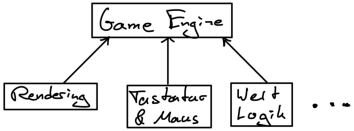
Erhöht jedoch komplexität des programms!



Anwendung

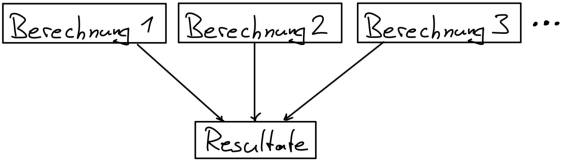
2 Arten der anwendung:

Aufteilung von verschiedenen Aufgaben
⇒ Multitasking.



Aufteilung von wiederholten größeren Aufgaben
⇒ Parallelisierung.

Parallelisierung gut zur Bearbeitung von großen Datenmengen!



Probleme

Race condition

- **2 Threads gleichzeitig am gleichen Speicher.**
- **Sog. Memorylocks können helfen.**

Deadlock

- **Z.B. 2 Threads brauchen 2 Memorylocks, beide bekommen aber nur einen.**
- **Deadlock - da beide Threads nicht fortsetzen können.**
- **Kann gezielt vermieden werden.**

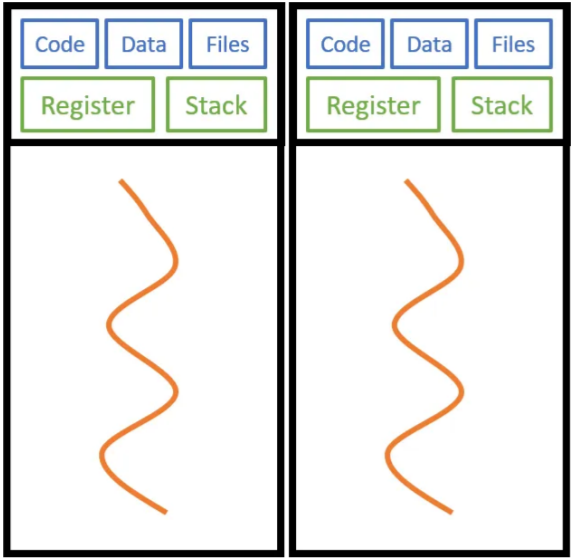
Multiprocessing

Ähnlich zum multithreading, jedoch alles seperat instanziert.

Kein Shared Memory etc.

Kein STD lib hierfür.

Implementierung und Auslastung härter.



Multithreading in C++

Threads in C++11 im Standardlibrary hinzugefügt.

Vorher <pthreads> Library.

```
std::thread thread_object (callable);
```

Callable objekte sind:

- **Funktions Pointer**
- **Lambda Ausdruck**
- **Funktions Objekt**

```
1 void funktion(param)
2 {
3   Code;
4 }
5 std::thread thread(funktion, params);
```

Multithreading in C++

Zusätzlich thread.join() um auf fertige Ausführung zu warten.

```
1  #include <thread>
2  using namespace std;
3
4  void f(param)
5  {
6    Code;
7  }
8
9  int main()
10 {
11     std::thread t(f, params); // Thread starten
12
13     t.join(); // Auf Thread warten
14
15     Weiterer Code;
16 }
```

Multithreading in Python

Sehr ähnlich zu C++.

```
1  import threading
2
3  def func(params):
4      Code
5
6
7  t = threading.Thread(target=func, args=(params))
8
9  t.start()
10
11 t.join()
12
13 Weiterer Code
```

Multithreading in Root

Multithreading zum teil automatisch implementiert.

`ROOT::EnableImplicitMT();`

```
// This activates implicit multi-threading
ROOT::EnableImplicitMT();

// The analysis below runs in parallel
ROOT::RDataFrame rdf("mytree", "myfile.root");
auto h = rdf.Filter("x > 0").Histo1D("x");
h->Draw();
```

```cpp
#include <iostream>
#include <vector>
#include <thread>
#include <chrono>

using namespace std;

void Sum(int n, int& sum) {
    sum = 0;
    for (int i = 1; i<=n; i++) {
        for (int j = 1; j<=i; j++) {
            sum += j;
        }
    }
}

int main() {
    int numThreads = 12; // Anzahl der Threads
    vector<thread> threads; // Vektor der spdter Threads enthdlt
    vector<int> results(numThreads, 0); // Vektor der Ergebnisse hdlt

    auto start = chrono::high_resolution_clock::now(); // Zeit Messung Start
    for (int i = 0; i < numThreads; ++i) {
        threads.emplace_back(thread(Sum,10000,ref(results[i]))); // Erzeugung der threads
    }

    for (auto& thread : threads) {
        thread.join(); // Warten auf fertige Ausführung
    }
    auto end = chrono::high_resolution_clock::now(); // Zeit Messung Ende
    auto duration = chrono::duration_cast<chrono::milliseconds>(end - start).count();
    cout << "Time taken: " << duration << " milliseconds" << endl;

    for (int i = 0; i < numThreads; i++) {
        cout << "Result " << i << ": " << results[i] << endl;
    }

    return 0;
}
```

Vergleich

Mit Multithreading

```
Time taken: 1114 milliseconds
Result 0: -787054544
Result 1: -787054544
Result 2: -787054544
Result 3: -787054544
Result 4: -787054544
Result 5: -787054544
Result 6: -787054544
Result 7: -787054544
Result 8: -787054544
Result 9: -787054544
Result 10: -787054544
Result 11: -787054544
```

Ohne Multithreading

```
Time taken: 1412 milliseconds
Result 0: -787054544
Result 1: -787054544
Result 2: -787054544
Result 3: -787054544
Result 4: -787054544
Result 5: -787054544
Result 6: -787054544
Result 7: -787054544
Result 8: -787054544
Result 9: -787054544
Result 10: -787054544
Result 11: -787054544
```

Quellen

https://towardsdatascience.com/multithreading-and-multiprocessing-in-10-minutes-20d9b3c6a867

https://www.geeksforgeeks.org/multithreading-in-cpp/

https://www.geeksforgeeks.org/multithreading-python-set-1/

https://root.cern/manual/multi_threading/

https://stackoverflow.com/