Einfacher Masse-Feder-Dämpfer Performancebenchmark in Rust

R. Grünert
18. Januar 2024

CPU: i7 4790k

1 Umsetzung

In diesem Projekt wurde die Eulermethode zur numerischen Lösung der Differentialgleichung eines einfachen Mass-Feder-Dämpfer-Modells genutzt, um den Speedup bei der parallelen Verarbeitung zu untersuchen. Dabei werden mehrere Lösungen der Trajektorien für zufällige Dämpfungsfaktoren ermittelt und gemittelt. Die Lösungsaufgaben werden im Fall der Parallelverarbeitung gleichmäßig auf die Verarbeitungseinheiten aufgeteilt und anschließend ebenfalls gemittelt.

Rust stellt für den Multicore-Betrieb die crate `std::thread` zur Verfügung. Diese bietet die Funktion `spawn` zur Erstellung eines neuen Threads auf Basis der gegebenen Funktion. Man erhält einen `JoinHandle` für diesen Thread. Mittels `join`-Methode kann dann auf die Terminierung des Threads gewartet und das Ergebnis „eingesammelt“ werden. Es folgt ein Beispiel:

```
let handle: std::thread::JoinHandle<f32> = std::thread::spawn(|| {
  1.0 + 1.0
});

result = handle.join()?; // 2.0
```

Spawnt man mehrere Threads, werden diese bereits automatisch auf die CPU-Verarbeitungseinheiten aufgeteilt. Um Kernaffinitäten zu spezifizieren, bietet sich die `core_affinity` crate an, welche sich einfach in die Threadfunktion integriert.

2 Ausführung

Die sequentielle und parallele Lösung wurden jeweils mit dem Optiomierungslevel `release` kompiliert. Beide erhielten dann eine Gesamtzahl von 2000000 Lösungen.

Zur Ausführung des parallelen Programmes kann der Befehl `cargo run --release -- 20000000 run-stepped` verwendet werden. Dabei sind folgene Aufrufargumente notwendig:

```
Usage: MassSpringDamper_Euler_Benchmark <NUM_RUNS> <COMMAND>

Commands:
  run-cores    Run on specific cores
  run-all      Run on all available cores
  run-stepped  Run on all cores, increasing the core count each time
  help         Print this message or the help of the given subcommand(s)

Arguments:
  <NUM_RUNS>  Total number of computations

Options:
  -h, --help     Print help
  -V, --version  Print version
```

3 Ergebnisse

Das Script `plot.R` erstellt einen Graphen der zuletzt berechneten Trajeketorie des Systems , ein Beispiel ist in Abb. 1 zu sehen. Mit dem Script `plot_amdahl.R` kann der speedup aus der Datei `speeds.csv` grafisch dargestellt werden. Dabei werden die Dateien `plot_amdahl_duration.pdf` und `plot_amdahl_speedup.pdf` erstellt. Abb. 2 zeigt das Ergebnis. Der Speedup läuft bis zu 4 Kernen fast perfekt linear, ab 5 Kernen gibt es allerdings einen negativen Sprung, welcher Möglicherweise auf Hyperthreading zurückzuführen ist.

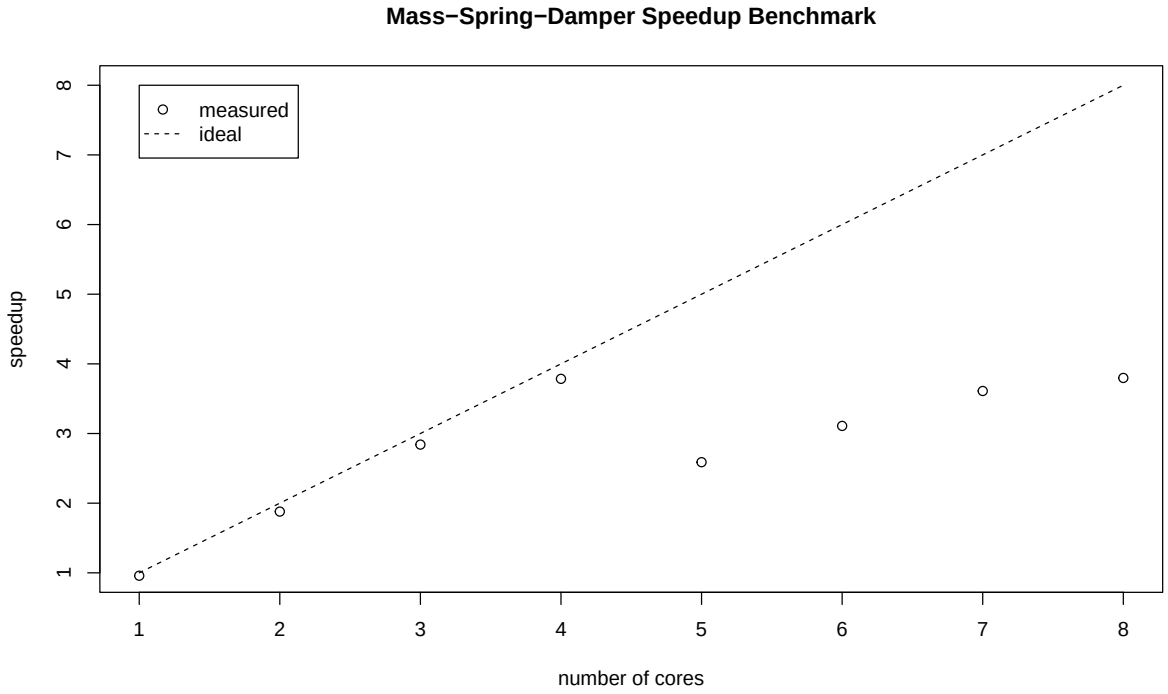

Abbildung 1: Benchmarkergebnis zum Speedup.

4 Mögliche Verbesserungen

- Die Funktion `MSDThreads::spawn_threads` sollte eigentlich eine Closure als Aufrufparameter erhalten, welche die Basis-`MSDParameter` bindet und die Funktion somit unabhängig von der eigentlichen MFD-Problemstellung macht
- Das Teilen der Basisparameter des MFD-Systems könnte einfacher über `Arc`-Typen geschehen, statt eine statische Lebensdauer zu fordern
- Warnungen beseitigen
- Test auf anderer CPU, möglicherweise mit mehr Kernen

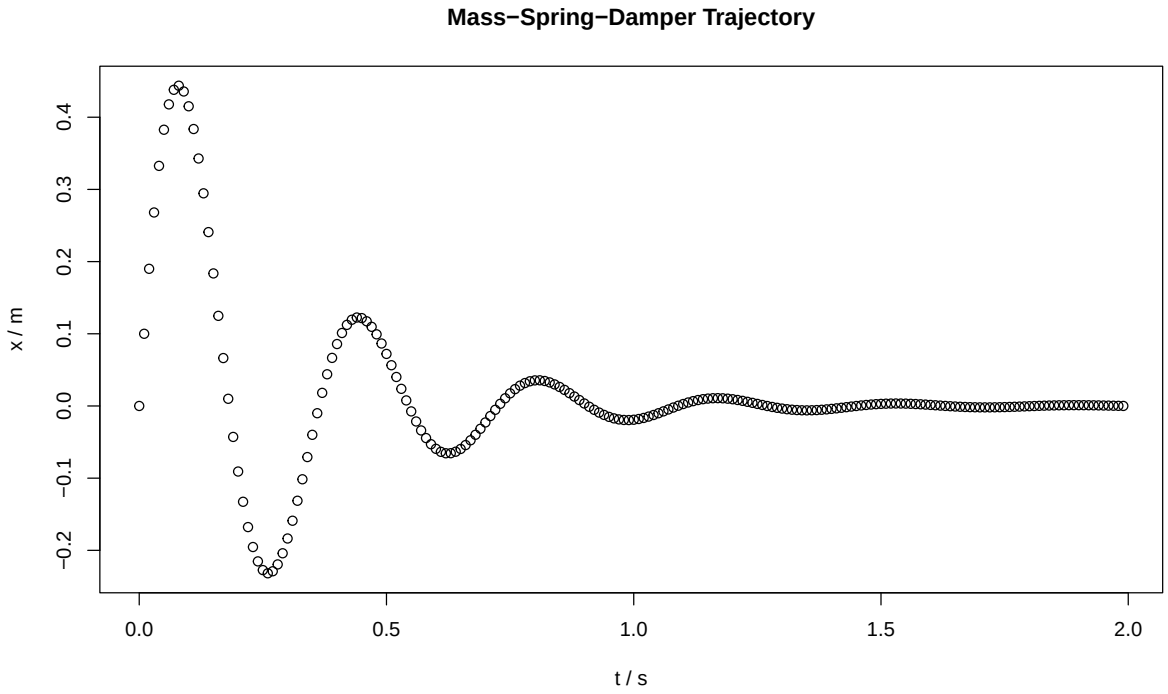

Abbildung 2: Beispieltrajektorie.