

Paralleles Ising Modell

Thomas Karl

17. April 2019

1 Theorie

Im Ising-Modell wird angenommen, dass die Spins, welche das magnetische Moment der Atome oder Ionen bestimmen, nur zwei diskrete Zustände annehmen können (Spinwert ± 1). Die Richtung im Raum bleibt aber offen.

Wir minimieren mittels Metropolis-Algorithmus den Hamiltonian:

$$H = -\frac{1}{2} \sum_{ij} J_{ij} s_i^z s_j^z - B \sum_{i=1}^V s_i^z \quad \text{mit } s_i^z = \pm 1 \quad (1)$$

In diesem Fall ist die Kopplungskonstante $J_{ij} = J = 1$ und das Magnetfeld ist $B = 0$.

Der Anfangszustand ist zufällig gewählt: In zwei Dimensionen mit periodischen Randbedingungen sind insgesamt $V = N_x \cdot N_y$ (Volumen) Spins entweder 1 oder -1 .

Beim Metropolis-Algorithmus werden zufällig Spins geändert um neue Konfigurationen zu erzeugen. Eine neue Konfiguration wird mit der Wahrscheinlichkeit

$$p = \min \left(1, \exp \left(-\frac{\Delta E}{k_b \cdot T} \right) \right) \quad (2)$$

angenommen. ΔE ist dabei die Differenz der Energie zum vorherigen Zustand, k_b die Boltzmann-Konstante und T die Temperatur, die auf $T = \frac{1}{10k_b}$ gesetzt wird. Bei dieser Temperatur sollten sich alle Spins gleich ausrichten und die Simulation liefert pro Volumen die Gesamtenergie -2 .

2 Algorithmus

Der Algorithmus funktioniert nur, falls es sich bei der Anzahl der Kerne um eine Quadratzahl handelt.

Die Prozesse werden als Quadrat angeordnet (Abb. 1) und erhalten ihren Teil des Gitters. Zusätzlich erhalten sie den Rand (rot). Dann führt jeder Prozess seinen Update Schritt durch.

Danach folgt der Randaustausch. Jeder Prozess ist Teil von zwei Ringen und sendet seinen äußeren Rand (rot) an den inneren Rand (blau) des entsprechenden Nachbarprozesses. Dieser empfängt ihn und speichert ihn ab. Zum Beispiel sendet *Rank 0* seinen rechten Rand an den linken von *Rank 1* und seinen unteren Rand an den oberen von *Rank 2*. Um ein *Deadlock* zu vermeiden, beginnen zuerst die diagonalen Prozesse mit Senden, die anderen empfangen. Dann wird getauscht.

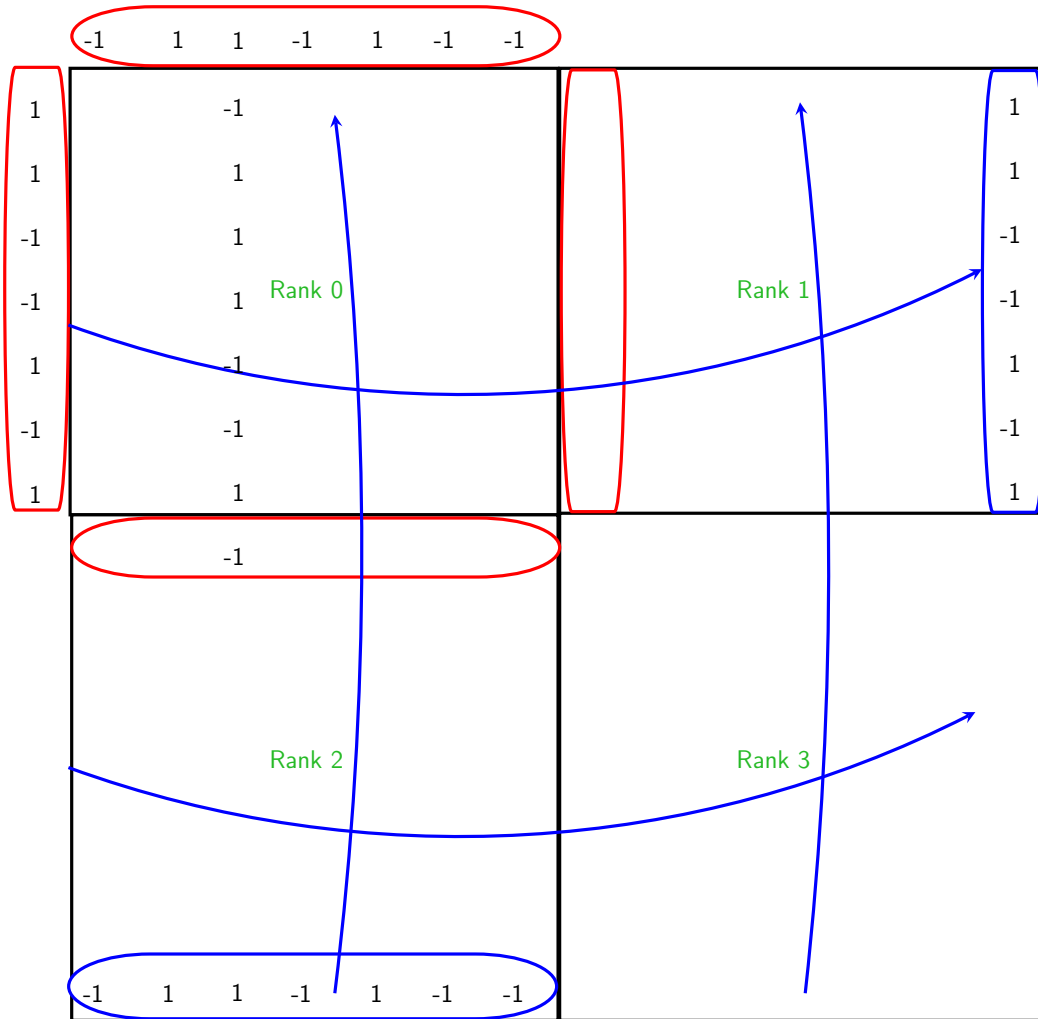


Abbildung 1: Schematische Darstellung des Prozessorlayouts.

3 Evaluation

Es wurden jeweils 10 Simulationen mit 10000 Schritten durchgeführt und die Laufzeit gemittelt. Diese wurden doppelt-logarithmisch für verschiedene Kernzahlen gegen die Länge des Gitters $s = N_x = N_y = \sqrt{V}$ aufgetragen (Abb. 2).

Da mit s das Volumen quadratisch steigt, also auch die Zahl der zu bearbeitenden Spins, hat der Algorithmus das Laufzeitverhalten $\mathcal{O}(s^2)$.

Die Fitkonstante im Exponenten liefert nicht wie erwartet 2. Das liegt daran, dass beim

Fitten davon ausgegangen wurde, dass Anteile, die linear mit s wachsen, vernachlässigbar seien. Gerade für kleine Gittergrößen trifft dies aber nicht zu.

Setzt man den Exponenten auf einen festen Wert und fittet nur den Vorfaktor, so ergeben sich Speedups von 3,5 und 4,84. Die Abweichung zum maximalen Speedup 6 im zweiten Fall ergibt sich dadurch, dass die Simulation mit einer Quadratzahl von Prozessen (also 9) gestartet werden muss. Bei der Hälfte der Zeit bleiben also die Hälfte der Kerne unbeschäftigt ($6 - 1,5 = 4,5$).

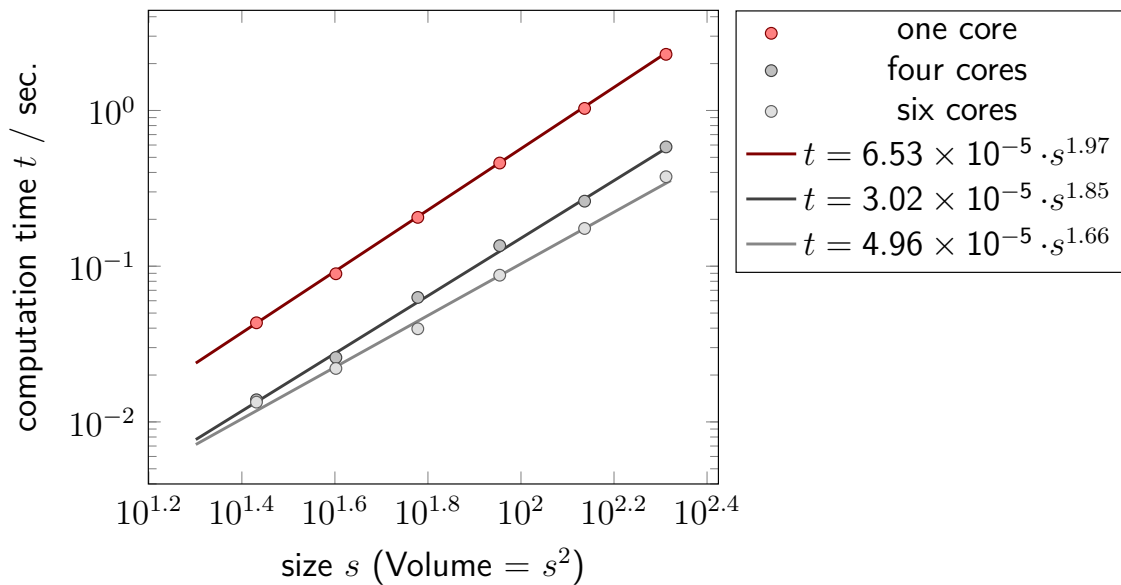


Abbildung 2: Verbrauchte Rechenzeit gegen die Länge des Gitters für alle drei Kernzahlen und Fitfunktionen auf doppelt-logarithmischer Skala.