Titel des Papers

Autor

May 18, 2024

Abstract

Die Art und Weise, wie auf Daten zugegriffen wird, kann einen erheblichen Einfluss auf die Performance eines Algorithmus haben. Insbesondere wenn große Datenmengen häufig zwischen langsamen Speichermedien und dem Prozessor ausgetauscht werden, spricht man häufig von einem Engpass des Speicherinterfaces. Durch effiziente Datenzugriffe und optimale Nutzung des Caches kann nicht nur die Performance signifikant gesteigert werden, sondern es wird auch sichergestellt, dass die Leistungsfähigkeit selbst bei der Verarbeitung großer Datenmengen stabil bleibt. In diesem Paper werden Methoden wie Loop Unrolling, Loop Fusion, Blocking analysiert. Um die Effektivität mancher Methoden zu demonstrieren, werden diese gebenchmarkt und mit den Standardmethoden verglichen.

1 Einleitung

Im letzten Jahrzehnt hat sich die Rechenleistung von Prozessoren erheblich gesteigert. In Abbildung 1 und 1.1 ist zu erkennen, dass seit der Einführung der 4th Generation im Jahr 2013 sich die Anzahl der Kerne bei Intel, die für den allgemeinen Verbrauchermarkt verfügbar sind, von 6 auf 24 Kernen erhöht hat. Es ist wichtig zu beachten, dass es zwar Prozessoren mit einer noch höheren Anzahl von Kernen gibt, diese jedoch in der Regel nicht für den Standard-Endverbraucher bestimmt sind. Ebenso zeigt sich in den Abbildungen 2 und 2.1 ein Anstieg der maximalen Taktfrequenz über die Jahre. Wenn man nun die theoretische maximale Rechenleistung, definiert als:
P_{max} = Anzahl der Kerne $\times$ Turbo Taktfrequenz $\times$ Flops pro Taktzyklus,
und der Entwicklung der Speicherbandbreite gegenüberstellt, wird in Abbildung 3 deutlich, dass die Zunahme der Speicherbandbreite nicht im gleichen Maße wie die Rechenleistung ansteigt. Im Detail hat sich die Bandbreite von 51.2 $\frac{\text{GByte}}{\text{s}}$ im Jahr 2013 auf 89.6 $\frac{\text{GByte}}{\text{s}}$ im Jahr 2024 erhöht. Parallel dazu ist die Performance im gleichen Zeitraum von 187.2 $\frac{\text{Flops}}{\text{s}}$ auf 1945.6 $\frac{\text{Flops}}{\text{s}}$ gestiegen. Diese Diskrepanz zwischen der gesteigerten Rechenleistung und der vergleichsweise langsamer wachsenden Speicherbandbreite verdeutlicht die Notwendigkeit einer Optimierung von Datenzugriffen. Ein effizienter Einsatz des Caches ist dabei unerlässlich, um die Leistung zu maximieren.

2 Berechnung der Performancegrenzen

In den nachfolgenden Unterabschnitten werden die Formeln vorgestellt, die zur Bewertung von loop-basiertem Code und zur Berechnung der Performancegrenzen herangezogen werden. Es ist jedoch wichtig zu berücksichtigen, dass diese Formeln lediglich eine Annäherung darstellen und nur unter bestimmten Bedingungen gültig sind. Beispielsweise wird vorausgesetzt, dass alle Ressourcen vollständig ausgeschöpft werden die die CPU zu bieten hat. Zudem basieren die Formeln auf Aspekten des idealen Cache-Modells, wie einem unendlich schnellen Cache und der Nichtexistenz von Latenzzeiten.

2.1 Maschinenbalance

Die Maschinenbalance B_m ist das Verhältnis aus der maximalen Bandbreite und der theoretischen Rechenleistung $B_m = \frac{B_{max}}{P_{max}}$. Sie beschreibt, wie viele Daten pro Flop übertragen werden können. Zur Veranschaulichung betrachten wir die B_m für den i7-9700K Prozessor, der 2018 erschienen ist und eine maximale Bandbreite von $41.6\,\frac{\text{GByte}}{\text{s}}$ sowie eine theoretische Rechenleistung von $8\,\text{Kerne} \times 4.9\,\text{GHz} \times 16\,\frac{\text{Flops}}{\text{Taktzyklus}} = 627{,}2\,\frac{\text{Flops}}{\text{s}}$ besitzt. Die Bandbreite muss noch in Fließkommazahlen umgerechnet werden, die pro Sekunde geladen werden können. Also $\frac{41.6\,\text{GB/s}}{8\,\text{Byte}} = 5.2\,\frac{\text{Fließkommazahlen}}{\text{s}}$. Daraus ergibt sich, dass $B_m = \frac{5.2\,\text{Fließkommazahlen/s}}{627.2\,\text{Flops/s}} \approx 0.008\,\frac{\text{Fließkommazahlen}}{\text{Flop}}$ geladen werden können. Mit anderen Worten, bis eine Fließkommazahl geladen ist, müssen 125 Flops auf einem Wert durchgeführt werden, bis der nächste Wert geladen ist. Die Maschinenbalance bei den neuesten Prozessoren wie dem i9-14900KS liegt bei ≈ 0.0058, und somit wären 172 Flops pro geladene Fließkommazahl erforderlich. Man könnte daher schlussfolgern, dass das Rechnen quasi kostenlos ist und das Laden der Werte den limitierende Faktor darstellt.

2.2 Codebalance

Die Codebalance $B_c = \frac{\text{Datenverkehr}}{\text{Flops}}$ ist das Verhältnis der zu ladenden und speichernden Fließkommazahlen und der Anzahl der Flops in einer loop Iteration. Hierbei zählt man aber nur die load und store Operationen, welche wirklich über den langsamen Datenpfad verläuft, daher wird l_i nicht mitgezählt, weil es sich im Register befindet. In Abbildung 4 ist ein solcher loop dargestellt, welcher 4 Flops ausführt und 3 Elemente aus den drei Arrays X, Y und Z lädt und eine speicher Operation in Z durchführt. Die Codebalance ist also $B_c = \frac{3+1}{4} = 1$.

2.3 Berechnung der zu erwartenden Performance

Um die maximal erreichbare Performance eines loop-basierten Codes zu berechnen, bestimmt man den Anteil der maximalen CPU-Performance, die tatsächlich erreicht werden kann. Dieser Anteil wird als ,,lightspeed" l bezeichnet und ist definiert als $l = \min(1, \frac{B_m}{B_c})$. Da diese Formel auf den ersten Blick nicht intuitiv ist, kann ein Beispiel zur Veranschaulichung dienen. Betrachten wir erneut den

Code aus Abbildung 4. Die Maschinenbalance B_m beträgt 0.5 $\frac{\text{Fließkommazahlen}}{\text{Flop}}$, während die Codebalance B_c 1 beträgt, da vier Fließkommazahlen geladen und gespeichert werden müssen und vier Flops ausgeführt werden. Da das Bereitstellen und Abspeichern von Fließkommazahlen nicht so schnell ist wie das Rechnen (was durch $B_m = 0.5$ ausgedrückt wird), kann man in der Zeit, in der man vier Flops ausführen würde, nur halb so viele, also $4 \times 0.5 = 2$ Fließkommazahlen laden und/oder abspeichern. Daher ist die maximale Performance, die erreicht werden kann, $l = \min(1, \frac{0.5}{1}) = 0.5$.

Oder anders betrachtet, es könnte einen Code geben, der doppelt so viele Flops ausführt wie er Fließkommazahlen lädt und speichert. Dies würde zu einer Codebalance von 0.5 führen. In einem solchen Szenario wäre die Anzahl der zu ladenden und speichernden Fließkommazahlen genau gleich der theoretisch möglichen Anzahl an Fließkommazahlen, die geladen und gespeichert werden könnten. Somit würde der maximal zu erreichende Anteil an P_{max} bei $l = \min(1, \frac{0.5}{0.5}) = 1$ liegen. Wenn dieser Wert nah an 1 liegt ist man nicht Speichergebunden. Die maximal erreichbare Performance ist somit $P = l \times P_{max}$ oder $P = \min(P_{max}, \frac{b_{max}}{B_c})$.

3 Loop Unrolling

4 Diskussion

Hier wird über die Bedeutung und Implikationen der Ergebnisse diskutiert.

5 Fazit

Hier wird das Fazit des Papers gezogen und ein Ausblick gegeben.

References

[1] Autor1, Titel des Referenzartikels, Journal, Jahr.

[2] Autor2, Titel des Referenzartikels, Journal, Jahr.