Marcell Kehmstedt
m.kehmstedt@tu-bs.de

Parallele Effizienz

Die parallele Effizienz ist neben dem Speedup eine weitere typische Kenngröße für parallele Berechnungen auf P Prozessoren. Sie ist definiert als:

$$\text{Effizienz(P)} := \frac{\text{Speedup(P)}}{P}$$

Lokale Innere Produkte

Definition

In der Linearen Algebra bezeichnet man mit *Inneres Produkt* oder *Skalarprodukt* auf einem Vektorraum V eine positiv definite symmetrische Bilinearform $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$ für die folgende Bedingungen gelten: $\forall x, y, z \in V, \forall \lambda \in \mathbb{R}$

a) Positive Definitheit: $\langle x, x \rangle \geq 0$, $\langle x, x \rangle = 0$ genau dann, wenn $x = 0$

b) Symmetrie: $\langle x, y \rangle = \langle y, x \rangle$

c) Bilinearität

- $\langle \alpha x + \beta y, z \rangle = \alpha \langle x, z \rangle + \beta \langle y, z \rangle$
- $\langle x, \alpha y + \beta z \rangle = \alpha \langle x, y \rangle + \beta \langle x, z \rangle$

Es gibt mehrere Abbildungen, die diese Bedingungen erfüllen und demnach gleichwertig als *Innere Produkte* bezeichnet werden. Unter dem sog. *Standardskalarprodukt* im $\mathbb{R}^n$ versteht man in der Lineraren Algebra dagegen folgende konkrete Abbildung:

$$(\forall x, y \in \mathbb{R}^n) : \langle x, y \rangle := \sum_{i=1}^{n} x_i y_i = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$$

Unter Anwendung des Assoziativgesetzes der Addition lässt sich die Summe auch schreiben als:

$$\sum_{i=1}^{n} x_i y_i = \sum_{i=1}^{\lfloor \frac{n}{P} \rfloor} x_i y_i + \sum_{i=\lfloor \frac{n}{P} \rfloor + 1}^{2\lfloor \frac{n}{P} \rfloor} x_i y_i + \cdots + \sum_{i=(P-1)\lfloor \frac{n}{P} \rfloor + 1}^{n} x_i y_i$$

Die P Teilsummen sind ihrerseits Skalarprodukte auf Ausschnitten der Vektoren x und y und lassen sich unabhängig voneinander berechnen. Man bezeichnet diese als *Lokale Innere Produkte*.

Rekursive Verdopplung

Die Idee, die Bildung der Skalarprodukte auf lokale Skalarprodukte runterzubrechen und diese anschließend zusammenzuführen, lässt sich fortführen und so können die lokalen Skalarprodukte ihrerseits auf noch kleinere lokale Skalarprodukte zurückgeführt werden. Haben wir also 8 Prozessoren, die von 0 bis 7 durchnummeriert seien, so bilden im ersten Schritt alle 8 Prozessoren zunächst lokale Skalarprodukte $s(0), s(1), \ldots, s(7)$ auf ihren Vektorstücken. In einem zweiten Schritt berechnen die Prozessoren $0, 2, 4, 6$ die Teilsummen $s(0:1), s(2:3), s(4:5), s(6:7)$. Im dritten Schritt berechnen die Prozessoren $0, 4$ die Teilsummen $s(0:3), s(4:7)$, welche im letzten Schritt von Prozessor 0 zum Endergebnis zusammengeführt werden.

Marcell Kehmstedt
m.kehmstedt@tu-bs.de

Butterfly-Schema

Ein negativer Aspekt des Verfahrens der rekursiven Verdopplung ist, dass von Schritt zu Schritt weniger Prozessoren beschäftigt werden. Dieses kann dadurch verbessert werden, indemin $\log_2 p$ Schritten jeweils $p/2$ Paare anstelle einzelner Teilsummen gebildet werden. Dazu tauschen je zwei Prozesse ihre lokale Teilsumme miteinander aus und bilden entsprechend neue Teilsummen.

Dazu tauscht Prozessor $0 = 000_b$ zunächst seine Teilsumme $s(0)$ mit Prozessor $1 = 001_b$ gegen die Teilsumme $s(1)$. Daraufhin berechnen **beide** die Teilsumme $s(0:1)$. Anschließend tauscht Prozessor $0 = 000_b$ die Teilsumme $s(0:1)$ mit Prozessor $2 = 010_b$ gegen die Teilsumme $s(2:3)$ und Prozessor $1 = 001_b$ tauscht die Teilsumme $s(0:1)$ mit Prozessor $3 = 011_b$ gegen die Teilsumme $s(2:3)$. Und so weiter. Es ergibt sich eine Butterfly-Kommunikationsstruktur, wie sie auch typisch für die FFT (Fast Fourier Transformation) ist.

Gram-Schmidt Orthonormalisierungsverfahren

Das Orthogonalisierungsverfahren nach Gram-Schmidt berechnet zu einem System $\{w_i\}_{i\in\mathbb{N}_n} \subseteq \mathbb{R}^m$ von n linear unabhängigen Vektoren ein Orthogonalsystem $\{v_i\}_{i\in\mathbb{N}_n}$ von n paarweise orthogonalen Vektoren, das denselben Vektorraum erzeugt. Es gilt:

$$(\forall i,j \in \mathbb{N}_n, i \neq j) : \langle v_i, v_j\rangle = 0$$
$$span(\{w_i\}_{i=1,\dots,n}) = span(\{v_i\}_{i=1,\dots,n})$$

Der Algorithmus lautet wie folgt:

$$\begin{aligned}
v_1 &= w_1 \\
v_2 &= w_2 - \tfrac{\langle v_1,w_2\rangle}{\langle v_1,v_1\rangle} v_1 \\
v_3 &= w_3 - \tfrac{\langle v_1,w_3\rangle}{\langle v_1,v_1\rangle} v_1 - \tfrac{\langle v_2,w_3\rangle}{\langle v_2,v_2\rangle} v_2 \\
&\vdots \\
v_n &= w_n - \sum_{i=1}^{n-1} \tfrac{\langle v_i,w_n\rangle}{\langle v_i,v_i\rangle} v_i
\end{aligned}$$

Orthonormalisierung

Die Orthonormailiserung ist eine Spezialisierung der Orthogonalisierung in der Form, als dass für die Vektoren v_i zusätzlich zur Orthogonaliät $(\forall i,j \in \mathbb{N}_n, i \neq j) : \langle v_i, v_j\rangle = 0$ eine Normierung der v_i gefordert. Es soll also gelten:

$$(\forall i \in \mathbb{N}_n) : \|v_i\| = \langle v_i, v_i\rangle = 1$$

Im Algorithmus kann dazu nach jeder Berechnung eines der v_i auch sogleich seine Normierung erfolgen. Dazu wird der Vektor v_i mit dem Kehrwert seiner eigenen Länge multipliziert.

$$v_i := \|v_i\|^{-1} v_i = \frac{1}{\|v_i\|} v_i = \sqrt{\langle v_i, v_i\rangle} v_i$$

Da die v_i nach jedem Schritt auch sogleich normiert werden, kann die Bildung der Skalarprodukte $\langle v_i, v_i\rangle$ in jedem nachfolgenden Schritt entfallen.

Eine Alternative zur Normierung nach jedem Schritt ist es zunächst ein Orthogonalsystem $\{v_i\}$ zu bestimmen und dessen Vektoren anschließend zu normieren.

Marcell Kehmstedt
m.kehmstedt@tu-bs.de

Aufgabe 1 Parallelisierung Innerer Produkte

a) Implementieren Sie die Berechnung des Standardskalarproduktes gemäß der mathematischen Definition zunächst als serielles Programm und messen Sie dessen Peakperformance (in GFlops). Wählen Sie dazu eine geeignete Problemgröße n.
Tipp: Führen Sie Messungen für unterschiedliche Problemgrößen durch bis diese in einen Sättigungsbereich kommen.

b) Parallelisieren Sie nun die Berechnung indem Sie auf P Prozessoren jeweils Lokale Innere Produkte entsprechend der Definition berechnen und diese anschließend aufsummieren.

c) Parallelisieren Sie nun nach dem Verfahren der Rekursiven Verdopplung.

d) (Bonus) Parallelisieren Sie zu guter Letzt nach dem Butterfly-Schema. Sie dürfen sich zur Vereinfachung auf Zweierpotenzen für die Anzahl der verwendeten Prozesse beschränken.

Das Ergebnis soll bei allen drei parallelen Implementierungen am Ende der Berechnung auf allen Prozessen verfügbar sein. Testen Sie Ihre Implementierungen für $n = 10^3$ und $n = 10^7$ sowie $P = 2, 4, 6, \ldots, 32$. Messen Sie die jeweils benötigte Zeit und bestimmen Sie jeweils den Speedup und die parallele Effizienz (in %) gegenüber der seriellen Berechnung. Stellen Sie Ihre Ergebnisse für den Speedup anschaulich in je einem gemeinsamen Plot dar (einen für jedes n). Interpretieren Sie Ihre Ergebnisse.

(10+5 Punkte)

Aufgabe 2 Gram-Schmidt Orthonormalisierungsverfahren

a) Implementieren Sie das Orthonormalisierungsverfahren von Gram-Schmidt zunächst in einer seriellen Variante und messen Sie dessen mittlere Laufzeit für $n = 10^3$ und $m = 10^4$.

b) Überlegen Sie sich eine geeignete Parallelisierung und implementieren Sie diese. Messen Sie wieder die mittlere Laufzeit für $n = 10^3$ und $m = 10^4$ jeweils für $P = 2, 4, 6, \ldots, 32$ Prozessoren. Bestimmen Sie jeweils den Speedup und die Effizienz und plotten sie den Speedup als Funktion über der Anzahl der verwendeten Prozessoren.

Überprüfen Sie ferner die korrekte Berechnung Ihrer Algorithmen, indem sie die Längen und die Orthogonalität der Ergebnisvektoren überprüfen.

(15 Punkte)