HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

Blatt 1

Betriebssysteme

Luis Staudt

a) Seitenfehler bei zweidimensionalem Feld

Ausschnitt A (spaltenweise Zugriff): 2048 + 1 = 2049 Ausschnitt B verursacht 33 Seitenfehler (1 für das Programm und 32 für die zeilenweise geladenen Datenseiten), da die Speicherzugriffe der natürlichen row-major Anordnung folgen und jede Seite nur einmal geladen wird.

Ausschnitt B (zeilenweise Zugriff): 32 + 1 = 33 Ausschnitt A verursacht hingegen 2.049 Seitenfehler (1 für das Programm und 2.048 für die Datenzugriffe), weil der spaltenweise Zugriff auf zeilenweise gespeicherte Daten bei nur 3 verfügbaren Datenrahmen zu ständigem Ein- und Auslagern aller 32 Datenseiten für jede der 64 Spalten führt.

b) Seitentabelle mit virtuellen und physischen Adressen

Bei einer virtuellen Adresse von 48 Bit und einer Seitengröße von 8 KiB ($= 2^{13}$ Byte) werden 13 Bit für den Offset innerhalb einer Seite verwendet, somit bleiben 35 Bit $(48 - 13)$ für die Seitennummer. Die Seitentabelle enthält daher $2^{35} = 34.359.738.368$ Einträge, wobei die Information über die 32-Bit physische Adresse für diese Berechnung nicht benötigt wird.

c) Invertierte Seitentabellen und Hashtabelle

Die invertierte Seitentabelle enthält einen Eintrag für jeden der $2^{15}(32.768)$ physischen Seitenrahmen $(256MiB/8KiB)$. Damit durchschnittlich weniger als ein Eintrag pro Hashtabellen-Slot existiert, muss die Hashtabelle mindestens $2^{16}(65.536)$ Einträge umfassen, da dies die nächstgrößere Zweierpotenz ist, die größer als die Anzahl der physischen Rahmen ist.

d) Scheduling-Strategien und durchschnittliche Prozessdurchlaufzeit

1. Round Robin (Multiprogramming mit fairer CPU-Aufteilung)

Bei dieser Strategie wird jedem Job reihum eine kleine Zeitscheibe zugeteilt; wenn wir 1-Minuten-Zeitscheiben in der Reihenfolge A-B-C-D-E annehmen, endet Job C nach 10 Minuten, Job D nach 18 Minuten, Job C nach 24 Minuten, Job D nach 28 Minuten und Job E nach 30 Minuten. Die durchschnittliche Durchlaufzeit beträgt (10 + 18 + 24 + 28 + 30)/5 = 22 Minuten.

2. Prioritätsscheduling

Bei Ausführung nach absteigender Priorität (B-E-A-C-D) ergibt sich: Job B nach 6 Minuten fertig, Job E nach 14 Minuten (6+8), Job A nach 24 Minuten (14+10), Job C nach 26 Minuten (24+2) und Job D nach 30 Minuten (26+4). Die durchschnittliche Durchlaufzeit beträgt (6 + 14 + 24 + 26 + 30)/5 = 20 Minuten.

3. First Come First Serve

Bei der Ankunftsreihenfolge A-B-C-D-E ergibt sich: Job A nach 10 Minuten fertig, Job B nach 16 Minuten (10 + 6), Job C nach 18 Minuten (16 + 2), Job D nach 22 Minuten (18 + 4) und Job E nach 30 Minuten (22 + 8). Die durchschnittliche Durchlaufzeit beträgt (10 + 16 + 18 + 22 + 30)/5 = 19,2 Minuten.

4. Shortest Job First

Bei Ausführung nach kürzester Laufzeit (C-D-B-E-A) ergibt sich: Job C nach 2 Minuten fertig, Job D nach 6 Minuten (2 + 4), Job B nach 12 Minuten (6 + 6), Job E nach 20 Minuten (12 + 8) und Job A nach 30 Minuten (20 + 10). Die durchschnittliche Durchlaufzeit beträgt (2 + 6 + 12 + 20 + 30)/5 = 14 Minuten.

e) CPU-Effizienz bei Round-Robin-Scheduling

1. Warum gilt für Q = ∞ und Q > T dieselbe Effizienz?

Wenn Q > T oder Q = ∞, wird der Prozess immer nach Zeit T blockieren, bevor sein Quantum abläuft. In beiden Fällen läuft der Prozess für die Zeit T und blockiert dann, was zu identischem Verhalten führt.

2. CPU-Effizienz für Q = ∞ bzw. Q > T

In diesem Fall läuft ein Prozess bis er nach Zeit T blockiert. Dann erfolgt ein Prozesswechsel, der Zeit S dauert. Dieser Zyklus wiederholt sich.

In jedem Zyklus beträgt die Rechenzeit T und die Gesamtzeit T + S. Daher ist die CPU-Effizienz:

$$\eta = \frac{T}{T+S}$$

3. CPU-Effizienz für S < Q < T (T sehr viel größer als Q)

In diesem Fall läuft ein Prozess für sein gesamtes Quantum Q, dann erfolgt ein Prozesswechsel, der Zeit S dauert. Dieser Zyklus wiederholt sich, bis der Prozess insgesamt Zeit T gelaufen ist, woraufhin er blockiert.

Die Anzahl der Ausführungen eines Prozesses vor dem Blockieren ist ungefähr T/Q (da T $\gg$ Q). In jedem Zyklus beträgt die Rechenzeit Q und die Gesamtzeit Q + S. Für T/Q solche Zyklen beträgt die Gesamtrechenzeit T und die Gesamtzeit (T/Q) · (Q + S) = T + (T·S)/Q.

Daher ist die CPU-Effizienz:

$$\eta = \frac{T}{T + \frac{T \cdot S}{Q}} = \frac{1}{1 + \frac{S}{Q}}$$

4. CPU-Effizienz für Q = S

Mit der Formel aus III erhalten wir:

$$\eta = \frac{1}{1 + \frac{S}{Q}} = \frac{1}{1 + \frac{S}{S}} = \frac{1}{2}$$

Die Effizienz beträgt 50%.

5. CPU-Effizienz für Q ≈ 0

Wenn Q sehr klein ist, dominiert der Overhead des Prozesswechsels. Mit der Formel aus III:

$$\eta = \frac{1}{1 + \frac{S}{Q}} \approx 0 \text{ für } Q \approx 0$$

Die Effizienz nähert sich 0%. Da das System hauptsächlich mit Prozesswechseln beschäftigt ist, wird die CPU nicht genutzt.

f) Einfluss von Seitenfehlern auf die Speicherzugriffszeit

Zuweisung zahl 0

Java nutzt referenzen auf Objekte, somit verweisen die gleichen Zahlen immer auf die gleiche stelle im Speicher, somit wird für das gesamte Array nur ein einziger Speicherplatz reserviert.

Prozess mit ausreichend großem Heap

Nein, deshalb mehrere Prozesse mit einem Heap von 8 GB.

Spalte Seitenfehler/s im Resourcemonitor

Nach bereits 4 Prozessen waren bereits >300 Fehler/s pro Prozess.

Zeiten pro inneren Schleifendurchlauf

Es waren keine Unterschiede zu erkennen, ich vermute aber das mit Seitenfehlern die Zeiten steigen sollten.

Swapfile Verwendung

Vor dem Start 0 GB, danach ca zwischen 0 und 2 GB. Kurzzeitig auf 4 GB, aber dann wieder runter.

CPU Auslastung

Die CPU war nie bei 100% Auslastung, aber die Auslastung war immer zwischen 50 und 80%. Die Auslastung von nur 50 bis 80% lässt sich vermutlich da durch Erklären, dass die Prozesse nicht CPU intensiv sind und nicht alle Kerne ausgelastet werden.