

Prof. Dr. Guido Moerkotte

Email: moer@pi3.informatik.uni-mannheim.de

Pit Fender

B6, 29, Raum C0.05

68131 Mannheim

Telefon: (0621) 181-2517

Email: pfender@pi3.informatik.uni-mannheim.de

Anfrageoptimierung
Herbst-/Wintersemester 07

8. Übungsblatt
14. November 2007

Aufgabe 1

Die 2-Phasen Heuristik verwendet zunächst Iterative Improvement (II) und dann Simulated Annealing (SA).

Aufgabe 1 a)

Wie verhält sich der Algorithmus, wenn die beiden Algorithmen vertauscht werden?

Lösung

SA liefert normalerweise ein lokales Optimum, d.h. ein anschließendes II ist sinnlos

Aufgabe 1 b)

Ist es sinnvoll, Tabu Search anstelle von SA zu verwenden?

Lösung

Tabu Search könnte anstelle von SA verwendet werden, da es ebenfalls ein lokales Optimum verlassen kann. Allerdings ist hier der Nutzen von der 1. Phase nicht so klar, da keine "Temperatur" eingestellt werden kann.

Aufgabe 2

Formulieren Sie die Auswahl des Joinverfahrens als genetischen Algorithmus. D.h. gegeben ein Jointree muss pro Join das Verfahren (Hash-Join, Sort-Merge etc.) festgelegt werden. Wie sehen Repräsentation, Mutation und Crossover aus?

Lösung

- Repräsentation: Vector mit Joinverfahren (ein Eintrag pro Join)
- Mutation: Zufälligen Eintrag ändern
- Crossover: Vektoren zufällig mischen (d.h. $c[i]=\text{rnd}(2)?a[i]:b[i]$)

Aufgabe 3

Aufgabe 3 a)

Im folgenden Listing wird die Implementierung des in der Vorlesung vorgestellten Skippy-Benchmarks gezeigt. Erklären Sie die Funktionsweise des Benchmarks.

```
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <fcntl.h>
#include <unistd.h>

#define SINGLE_SECTOR 512

int
main() {
    int lNoMeasure = 1000;
    char lBuffer[SINGLE_SECTOR];
    int i;
    int lTime;

    struct timeval lTimeValStart;
    struct timeval lTimeValEnd;
    struct timezone lTimeZone;
    for(i = 0; i < SINGLE_SECTOR; ++i) {
        lBuffer[i] = 'x';
    }

    int lFd = open("/dev/sda3", O_RDWR | O_SYNC);
    if(lFd < 0) {
        fprintf(stderr, "Error occurred while opening raw device\n");
        exit(-1);
    }
    for(i = 0; i < lNoMeasure; ++i) {
        /* time the following sequence, and output <i, time> */
        gettimeofday(&lTimeValStart, &lTimeZone);
        lseek(lFd, i * SINGLE_SECTOR, SEEK_CUR);
        write(lFd, lBuffer, SINGLE_SECTOR);
        gettimeofday(&lTimeValEnd, &lTimeZone);

        lTime = (lTimeValEnd.tv_sec - lTimeValStart.tv_sec) * 1000000;
        lTime += (lTimeValEnd.tv_usec - lTimeValStart.tv_usec);

        printf("%d %d\n", i, lTime);
    }
}
```

```

    }
    close(lFd);
}

```

Lösung

Vereinfacht ergibt sich folgende Code-Struktur:

```

open(raw disk device)
for(i ... Number of Measurements)
    read start time
    lseek(raw device, i * Size of a Sector);
    write(1 Sector)
    read end time
close(raw device)

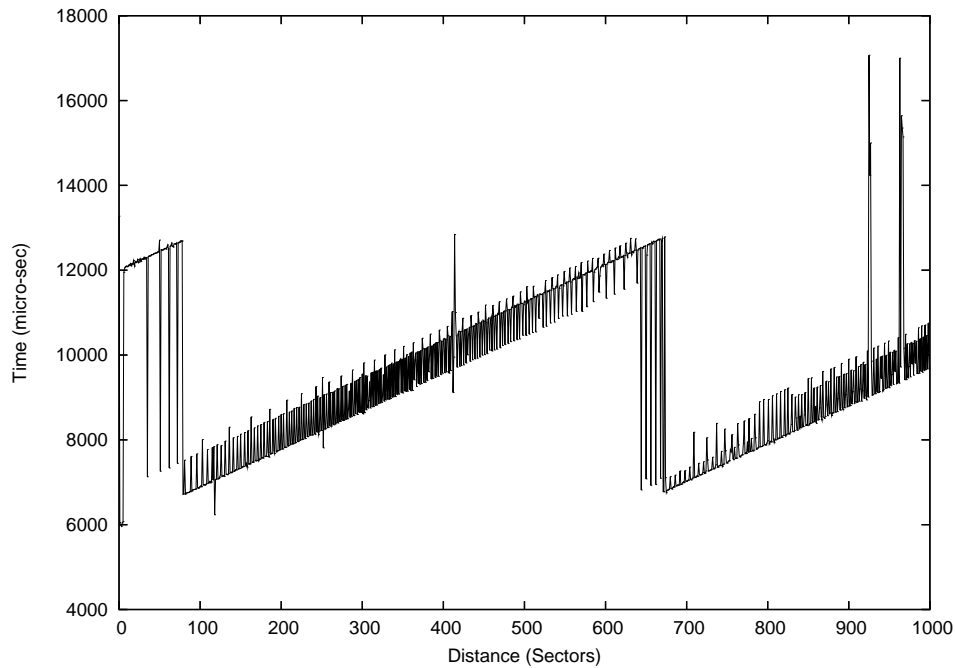
```

Der Benchmark schreibt jeweils einen Sektor auf Platte und springt dann $i, i \in [1, MAX]$ Sektoren weiter. Eine Seek- und Schreibeoperationen gliedert sich in folgende Punkte:

1. w start,
2. w at disk,
3. w at surface,
4. w under head und
5. w end.

Die *MTM* (Minimum Time to Media) umfasst die Punkte 1) bis 3). Das Rotational Delay ergibt sich dann aus den Punkten 3) bis 4). Um die Ergebnisse interpretieren zu können ist ein weiterer Wert *STM* von Bedeutung. *STM* steht für die Anzahl der Sektoren einer gegebenen Zone, die während der *MTM* am Plattenarm vorüberlaufen. Zum Startzeitpunkt befindet sich der Lesekopf über der gewünschten Position, an der bei einem $i = 0$ und damit einer Schrittweite von 0 Sektoren ein Sektor beschrieben werden soll. Durch die Verzögerung der *MTM* hat sich zum Zeitpunkt 3) die Platte bereits weitergedreht, so dass bis zur nächsten Umdrehung gewartet werden muss. Im Diagramm ergibt sich daraus für $x \approx 0$ die Full Rotation Time; abzulesen auf der Abszisse. Mit zunehmender Schrittweite erhöht sich auch die Wartezeit. Kurz vor dem negativen Sprung entspricht die Schrittweite nahezu dem Wert *STM*. Die Wartezeit ergibt sich nun aus der Summe von Full Rotation Time und *MTM*. Überschreitet die Schrittweite die Anzahl der Sektoren *STM*, dann sind weniger Sektoren an am Lesekopf vorbeigelaufen wie zu überspringen sind. Das Rotational Delay ist 0. Auf der Y-Achse ist die *MTM* ablesbar. Da ein Head-Switch im gleichen Zylinder weniger Zeit benötigt, als ein Cylinder Switch, entsprechen die kleineren Ausschläge der Head Switch Time und die Größeren der Cylinder Switch Time.

Aufgabe 3 b)



Interpretieren Sie die mit dem Benchmark ermittelten Beispielmessungen! Welche Parameter lassen sich ablesen?

Lösung

Ergebnis der Benchmarkmessung:
Man kann folgende Daten ablesen:

- Rotational Latency: 12ms
in Höhe des y-Wertes eines Spungs
- Head Switch: 0,7ms
kleinere Ausschlag zwischen zwei benachbarten Punkten
- Cylinder Switch: 1,2ms
grösserer Ausschlag zwischen zwei benachbarten Punkten
- MTM+Transfer Time: 6,5ms
niedrigster y-Wert

Folgende Daten stehen im Datenblatt (IBMIC35L018UWD210-0):

- Rotational Latency: 12ms (Time for ResolutionL 6ms + Average Latency: 3ms)
- Head Switch: 0,72ms
- Cylinder Switch: 0,88ms
- 16 Zonen
- 3 Köpfe, 2 Platten