Virtueller Speicher

Florian Nehmer
Jakob Ledig

June 28, 2017

1 Code ergänzen

1.1 Ergänzen Sie den Code für die Methode getVirtualPageNum(int)

```
/**
 * @param virtAdr: eine virtuelle Adresse
 * @return Die entsprechende virtuelle Seitennummer
 */
private int getVirtualPageNum(int virtAdr) {
        return (virtAdr / PAGE_SIZE);
}
```

1.2 Ergänzen Sie den Code für die Methode getOffset(int)

```
/**
 * @param virtAdr
 *            : eine virtuelle Adresse
 * @return Den entsprechenden Offset zur Berechnung der realen Adresse
 */
private int getOffset(int virtAdr) {
        return (virtAdr % PAGE_SIZE);
}
```

1.3 Ergänzen Sie den Code für die Methode read(int, int)

```
/**
 * Datenwort von einer Adresse im virtuellen Speicher lesen
 *
 * @param pid
 *            Prozess-ID
 * @param virtAdr
 *            virtuelle Adresse
 * @return Datenwort auf logischer Adresse virtAdr oder -1 bei
 *         Zugriffsfehler
 */
public synchronized int read(int pid, int virtAdr) {
        Process process;
        int virtPageNum;
```

```
14          int offset;
15          PageTableEntry pte;
16
17          process = getProcess(pid);
18          virtPageNum = getVirtualPageNum(virtAdr);
19          offset = getOffset(virtPageNum);
20
21          // Uebergebene Adresse pruefen
22          if ((virtAdr < 0) || (virtAdr > VIRT_ADR_SPACE - WORD_SIZE)) {
23                  System.err.println("OS: read ERROR " + pid + ": Adresse " + virtAdr
24                                  + " liegt ausserhalb des virtuellen Adressraums 0 - " + VIRT_ADR_SPACE);
25                  return -1;
26          }
27          // Seitenadresse berechnen
28          virtPageNum = getVirtualPageNum(virtAdr);
29          offset = getOffset(virtAdr);
30          testOut("OS: read " + pid + " " + virtAdr + " " + " +++ Seitennr.: " + virtPageNum + " Offset: " + offset);
31
32          // Seite in Seitentabelle referenzieren
33          pte = process.pageTable.getPte(virtPageNum);
34          if (pte == null) {
35                  // Seite nicht vorhanden:
36                  testOut("OS: read " + pid + " +++ Seitennr.: " + virtPageNum + " in Seitentabelle nicht vorhanden");
37          } else {
38                  // Seite vorhanden: Seite valid (im RAM)?
39                  if (!pte.valid) {
40                          // Seite nicht valid (also auf Platte --> Seitenfehler):
41                          pte = handlePageFault(pte, pid);
42                  }
43          }
44
45          // Seitentabelle bzgl. Zugriffshistorie aktualisieren
46          pte.referenced = true;
47
48          // Statistische Zï¿½hlung
49          eventLog.incrementReadAccesses();
50
51          return pte.realPageFrameAdr;
52  }
```

1.4 Ergänzen Sie den Code für die Methode randomAlgorithm(PageTableEntry)

```
1   /**
2    * RANDOM-Algorithmus: ZufÃllige Auswahl
3    */
4   private PageTableEntry randomAlgorithm(PageTableEntry newPte) {
5           Random r = new Random();
6           int randomIndex = r.nextInt(pteRAMlist.size());
7           PageTableEntry output = pteRAMlist.get(randomIndex);
8           pteRAMlist.set(randomIndex, newPte);
9           return output;
10  }
```

2 Simulation

2.1 Daten

Table 1: My caption

MAX_RAM_PAGES_ PER_PROCESS	DEFAULT_ LOCALITY_FACTOR	RANDOM	CLOCK	FIFO
10	1	0,499	0,498	0,500
10	10	0,229	0,210	0,219
10	100	0,031	0,025	0,027
10	1000	0,00325	0,00254	0,00279
15	10	0,1119	0,1068	0,1128
20	10	0	0	0

2.2 Fragen

a) : Ist der Wert bei absolut zufälligen Zugriffsfolgen (Lokalitätsfaktor = 1) Ihrer Ansicht nach plausibel? Wenn ja, aufgrund welcher überlegung?
Tipp: Berücksichtigen Sie die Hauptspeicherzuteilung und die Programmgröße!

Bei 20 Seiten, wovon 10 im RAM sind ist die Chance bei komplett zufälligen Zugriffen 50% eine Seite im RAM zu treffen. Daher ist der Wert plausibel.

b) : In welcher Größenordnung liegt (bei diesem einfachen Simulationsmodell) der Leistungsunterschied zwischen CLOCK-, FIFO- und RANDOM-Algorithmus (in %)?

Der Größenunterschied ist sehr gering. Nur der Random ist immer ein wenig schlechter mit einem Unterschied von im Schnitt 13 %, wobei der Unterschied tendenziell höher ist, wenn der Lokalitätsfaktor höher ist.

c) : Welche Maßnahme zur Leistungssteigerung Ihres Computers können Sie ergreifen, wenn Sie große Programme mit schlechtem Lokalitätsverhalten ablaufen lassen wollen?

Man sollte in diesem Fall den RAM erweitern, da die Algorithmen bei ineffizienten Lokalitätsverhalten kaum einen Unterschied machen. Mit mehr RAM kann man einfach mehr in den Hauptspeicher laden und muss weniger auslagern.