Inhaltsverzeichnis

1 Erzeugung von Parallelität

1.1 fork - join

fork erstellt eine ”Kopie”des aktuellen Prozesses, die Ausführung des ”Kindprozesses”beginnt an der Stelle, an der fork aufgerufen wurde.

1.2 cobegin

Alle Anweisungen werden parallel ausgeführt, es wird gewartet bis alle Arbeiten beendet sind.

1.3 forall

Kreuzung aus cobegin und for.

2 Parallelität am Beispiel Java

2.1 extends Thread

```
public class myThread extends Thread {
        public void run() {
                // do some work here
        }
}

new myThread().start();
```

2.2 implements Runnable

```
public class myRun implements Runnable {
        public void run() {
                // do some work here
        }
}
```

2.2.1 manuell ersteller Thread

```
new Thread(new myRun()).start();
```

2.2.2 ThreadPool

```
ExecutorService pool = Executors.newFixedThreadPool($ANZAHL_THREADS);
pool.execute(new myRun());
```

2.2.3 ForkJoinPool

```
ForkJoinPool pool = new ForkJoinPool();
myRun newRun = new myRun();
pool.invoke(newRun);
newRun.join();
pool.shutdown();
```

Innerhalb der run-Methode von myRun kann nun ein pool.invoke() oder pool.invokeAll() aufgerufen werden.

3 Arten - Klassifikation von Flynn

3.1 Single Instruction Single Data (SISD)

3.2 Single Instruction Multiple Data (SIMD)

3.3 Multiple Instruction Single Data (MISD)

3.4 Multiple Instruction Multiple Data (MIMD)

4 Speedup

4.1 Berechnung

$$S(n) = \frac{T_1(n)}{T_p(n)}$$

S	Speedup (relative Geschwindigkeitsteigerung; optimalerweise > 1)
T_1	Ausführungszeit im sequentiellen Fall
T_p	Ausführungszeit im parallelen Fall mit p Recheneinheiten

4.2 Idealer Speedup

Im Falle

$$S(n) = p$$

spricht man vom idealen Speedup.

4.3 Gesetz von Amdahl

Der Speedup fällt bei Steigender Thread-Anzahl ab, da sequentielle Teile existieren, die nicht parallelisierbar sind.
Es gilt:

$$T_p(n) = s(n) \cdot T_1(n) + \frac{p(n)}{p} \cdot T_1(n)$$

Und damit gilt:

$$S(n) = \frac{T_1(n)}{s(n) \cdot T_1(n) + \frac{p(n)}{p} \cdot T_1(n)} \leq \frac{T_1(n)}{s(n) \cdot T_1(n)} = \frac{1}{s(n)}$$

5 Parallele Effizienz

5.1 Berechnung

$$E_p(n) = \frac{S_p(n)}{p} = \frac{T_1(n)}{p \cdot T_p(n)}$$

E	Parallele Effizienz
S	Speedup
T_1	Ausführungszeit im sequentiellen Fall
T_p	Ausführungszeit im parallelen Fall mit p Recheneinheiten