Universität Basel

Dr. Ralph Gasser
Dr. Marco Vogt

Informatiklabor — Herbst 2025

Übung 11 — Abgabe: 07.12.2025 (23:55 Uhr)

Tutoren: James Joggi, Roman Ostermiller, Eliot Portevin und Nicholas Salsi

Modalitäten: In dieser Übung benchmarken Sie eine zur Verfügung gestellte Applikation. Bitte geben Sie Ihre Messwerte, eine Dokumentation sowie das Benchmarking-Programm, welches Sie entwickelt haben, bis zum angegebenen Termin auf ADAM ab (siehe Abschnitt *Abgabe* weiter unten in diesem Dokument).

Motivation

Das Verständnis der Leistung einer Anwendung unter verschiedenen Bedingungen ist entscheidend für die Optimierung und Weiterentwicklung einer Applikation. Benchmarking, also die Messung der Leistung eines Programms, ist deshalb eine wesentliche Fähigkeit für jeden Softwareentwickler. Diese Übung vermittelt praktische Erfahrungen im Benchmarking einer Anwendung. Sie lernen, wie man die Laufzeit eines Programms für variierende Parameter misst.

Aufgabenbeschreibung

Ihre Aufgabe ist es, ein Programm zu schreiben, welches eine bereitgestellte Anwendung (`Simulation.jar`) benchmarkt. Diese Simulations-App ist so konzipiert, dass sie eine Anwendung nachahmt, deren Verhalten sich basierend auf bestimmten Eingaben ändert. Sie nimmt drei Argumente entgegen:

1. **Ihre Unibas E-Mail-Adresse**: Diese wird verwendet, damit Sie andere Daten messen können als Ihre Kommilitonen.
2. **Algorithmusnummer (1 oder 2)**: Stellt zwei verschiedene Algorithmen oder Modi dar, in denen die Anwendung betrieben werden kann.
3. **Ein Integer-Wert (0 bis 99)**: Stellt eine variable Eigenschaft der Anwendung dar, wie etwa die Anzahl der Kunden, die Datengrösse oder den Komplexitätsgrad.

Ihr Benchmarking-Programm soll messen, wie lange die Anwendung (`Simulation.jar`) für jede Kombination aus Algorithmusnummer und Integer-Wert (von 0 bis 99) läuft. Sie sollten:

- Den Algorithmus 1 und 2 mit allen Integer-Werten von 0 bis 99 ausführen.
- Die Ausführungszeit in Millisekunden für jeden Lauf messen.
- **Mindestens drei separate Durchläufe** für jeden Algorithmus durchführen, um Konsistenz und Zuverlässigkeit in den Daten zu gewährleisten.
- Die Ergebnisse in **einzelnen** CSV-Dateien mit zwei Spalten aufzeichnen: eine für den Integer-Parameter und die andere für die Laufzeit in Millisekunden. Bitte benennen Sie die Dateien nach dem Muster `algorithm[NUMBER]-[RUN].csv`, wobei `[NUMBER]` die Algorithmusnummer und `[RUN]` die Nummer der Wiederholung ist (z.B. algorithmus1-1.csv, algorithmus2-3.csv usw.).

Abgabe

Nach Abschluss der Übung sollten Sie Folgendes haben:

- Eine Benchmarking-Anwendung, welche die bereitgestellte Simulations-App wie beschrieben benchmarken kann.
- Eine Reihe von CSV-Dateien mit den Ergebnissen der Benchmarks.
- Eine Beschreibung, wie ihr Benchmarking-Programm ausgeführt werden muss.
- Eine Dokumentation, mit welcher Hardware, Software etc. ihre Messergebnisse erzielt wurden.

Bitte geben Sie diese Artefakte bis zur Abgabefrist auf ADAM ab.

Hinweise

- Sie können Ihr Benchmarking-Programm in einer Programmiersprache Ihrer Wahl schreiben, achten Sie aber darauf, eine vollständige Beschreibung zum Ausführen Ihres Programms beizulegen.
- Falls Sie noch wenig Programmiererfahrung haben, empfehlen wir Ihnen, die Anwendung in Java zu schreiben. Weiter unten finden Sie einige Code-Fragmente, welche Ihnen bei der Lösung helfen.
- Sie werden andere Messwerte erhalten als Ihre Kommilitonen, da ihre Mail-Adresse verwendet wird, um ein einzigartiges Verhalten zu simulieren.
- Die simulierte Ausführungszeit liegt zwischen 0 und zehn Sekunden. Sollten Sie nur

sehr kleine Ausführungszeiten im Bereich weniger Millisekunden messen, haben Sie vermutlich einen Fehler im Programmaufruf. Stellen Sie sicher, dass Sie auch Messwerte im Sekundenbereich messen.

- Um die Ausführung zu beschleunigen, können Sie Ihren Benchmark auch parallelisieren.

Die Simulation-App

Die Simulations-App ist ein Java-Programm, welches für diese Übung entwickelt wurde. Es generiert mithilfe der SHA-256-Hashfunktion aus der angegebenen E-Mail-Adresse sowie der spezifizierten Algorithmusnummer ein einzigartiges Polynom. Dieses Polynom wird so normiert, dass seine Funktionswerte im Bereich zwischen 0 und 7 liegen.

Der dritte Parameter der App repräsentiert den x-Wert. Der Polynomwert an dieser Stelle wird mit 1000 multipliziert, um die Simulationsdauer in Millisekunden zu erhalten. Um kleine Schwankungen zu simulieren, wird ein zufälliger Overhead zwischen 0 und 300 Millisekunden hinzugefügt. Die Verteilung dieser Zufallswerte wird durch eine quadratische Transformation angepasst, um die Häufigkeit kleinerer Werte zu erhöhen.

Da die Berechnungen der App einfach sind und die Laufzeit vorwiegend durch eine `sleep`-Funktion erzeugt wird, ist eine parallele Ausführung des Benchmarkings bis zu einem bestimmten Grad möglich. Sie können sich hierbei an der Anzahl der CPU-Kerne orientieren.

Der folgende Befehl zeigt, wie die App über die Kommandozeile aufgerufen werden kann:

```
java -jar Simulation.jar beispiel@unibas.ch 1 45
```

Dieser Befehl startet die Simulation mit der E-Mail-Adresse `beispiel@unibas.ch`, verwendet Algorithmus 1 und setzt den internen Wert auf 45.

Java-Codebeispiele

In diesem Abschnitt finden Sie einige Codebeispiele, welche Ihnen bei der Entwicklung Ihrer Benchmarking-Applikation helfen.

Überprüfung der Existenz der JAR-Datei

Als Erstes sollten Sie sicherstellen, dass die JAR-Datei existiert. Hier ist ein Beispielcode in Java, der diese Überprüfung durchführt:

```
import java.io.File;

File jarFile = new File("/pfad/zur/Simulation.jar");
if (!jarFile.exists()) {
    System.err.println("JAR wurde nicht gefunden: " + jarFile.getPath());
    return;
}
```

Dieser Code erstellt ein `File`-Objekt mit dem Pfad zur JAR-Datei und verwendet dann die Methode `exists()` des `File`-Objekts, um zu überprüfen, ob die Datei existiert. Wenn die Datei nicht existiert, wird eine Fehlermeldung auf der Konsole ausgegeben und das Programm beendet seine Ausführung.

Prozess-Initialisierung

Zum Starten eines Prozesses, der die Simulations-App ausführt, verwenden Sie die Klasse `ProcessBuilder`. Hier ist ein kurzes Beispiel, wie man einen Prozess für die Simulations-App initialisiert:

```
import java.io.IOException;

ProcessBuilder processBuilder = new ProcessBuilder(
    "java", "-jar", "/pfad/zur/Simulation.jar", "example@unibas.ch", "1", "50"
);
Process process = processBuilder.start();
process.waitFor();
```

In diesem Code:

- Der `ProcessBuilder` wird mit einem Befehl konfiguriert, der das Java-Laufzeitkommando (`java`), das `-jar` Flag, den Pfad zur JAR-Datei und die erforderlichen Argumente für die Simulation (E-Mail-Adresse, Algorithmusnummer und Integer-Wert) enthält.
- Mit `processBuilder.start()` wird der Prozess gestartet. Dieser Befehl gibt ein `Process`-Objekt zurück, das den laufenden Prozess repräsentiert.
- `process.waitFor()` hält den aktuellen Thread an und wartet, bis der externe Prozess beendet ist. Dies stellt sicher, dass die Ausführungszeit korrekt gemessen werden kann, bevor der nächste Prozess gestartet wird.

Messung der Ausführungszeit

Um die Ausführungszeit des Prozesses in Millisekunden zu messen:

```
long startTime = System.currentTimeMillis();
process.waitFor();
long endTime = System.currentTimeMillis();
long executionTime = endTime - startTime;
```

Schreiben einer CSV-Datei

Die gemessenen Laufzeiten sollen in eine CSV-Datei geschrieben werden. Hier ein Beispiel, wie man Daten in eine CSV-Datei schreibt:

```
import java.io.FileWriter;
import java.io.IOException;

try (FileWriter csvWriter = new FileWriter("ergebnisse.csv")) {
    csvWriter.append("Parameter,Laufzeit (ms)\n");
    csvWriter.append(x + "," + executionTime + "\n");
}
```

Parallelisierung des Benchmarkings

Für die Parallelisierung können Sie die `ExecutorService`-Klasse verwenden. Hier ein Beispiel, wie man mehrere Prozesse parallel startet:

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;

int threads = 4;
ExecutorService executor = Executors.newFixedThreadPool(threads);
for (int x = 0; x < 100; x++) {
    int finalX = x;
    executor.submit(() -> {
        // Prozess starten und Zeit messen
        // Ergebnisse in eine CSV-Datei schreiben
    });
}
executor.shutdown();
executor.awaitTermination(1, TimeUnit.HOURS);
```

Hinweis: Das Benchmarking zu parallelisieren, macht die Entwicklung deutlich anspruchsvoller und fügt einige neue Fehlerquellen hinzu. Es ist nicht notwendig zu parallelisieren, Sie können das Benchmarking auch Schritt für Schritt ausführen, es dauert nur etwas länger.