Betriebssysteme, Übungsblatt 3

Aufgabe 1: Java Bibliothek

Die grundlegenden Operationen an Dateien wurden aufbauend auf der Interface „FileOperation“ erstellt. Relevant für die Sicherstellung der Konsistenz ist vor allem die schreibe-Funktion, die in der Datei „FileWriteOperation“ umgesetzt wurde. Die Konflikterkennung habe ich hier mit Hilfe von Hashing umgesetzt. Bevor eine Write-Operation durchgeführt wird, wird mit Hashing überprüft, ob die Metadaten sich seit dem initialisieren geändert haben, was auf eine externe Modifizierung der Datei hindeuten und somit einen Konflikt darstellt. Der Code für die Hash-Vergleich befindet sich in „FileMetaData.java“

Die eigentliche Konflikt-Behandlung für die einzelnen Operation erfolgt in „FileTransaction.java“. Hier werden ZFS-Snapshots verwendet, um beim Fehlschlag einer Transaktion (z.B. ein Konflikt) einen Rollback zu dem Snapshot auszuführen.

Für die Umsetzung der ZFS Befehle in Linux dient die „ZFSSnapshotManager.java“ Klasse. Hier werden die auszuführenden Terminal-Befehle durchgeführt.

„FileTransactionTest.java“ und „ParallelTransactionTest.java“ dienen zur Veranschaulichung und testen der Bibliothek.

Aufgabe 2: Prototyp einer Brainstorming-Anwendung

Das Brainstorming-Tool ist sehr simpel aufgebaut. Das Tool präsentiert dem Nutzer vier Möglichkeiten: Eine neue Idee erstellen, eine Idee lesen, ein Kommentar hinzufügen und das Tool schließen.

Das Erstellen einer neuen Idee sorgt für zwei weitere Aufforderungen. Der Name der Idee und einen Inhalt. Nach der Eingabe wird die Idee als eine Textdatei gespeichert, mit dem Namen der Idee als Name der Datei und der Inhalt als Inhalt der Datei.

Das Lesen einer Idee sorgt zunächst für eine Auflistung der existierenden Ideen. Hier kann anschließend der Name der zu lesende Idee eingeben werden. Nach erfolgreicher Eingabe wird der Inhalt der Idee ausgegeben.

Ein Kommentar hinzuzufügen erlaubt dem Nutzer einer Idee ein Kommentar hinzuzufügen. Das Kommentar wird dem Inhalt der Idee hinzugefügt und erhält einen Zeitstempel und eine Markierung um dieses als ein Kommentar zu identifizieren. Da die Write-Operation grundsätzlich alte Inhalte überschrieb, wird hier zunächst der alte Inhalt ausgelesen und hinzugefügt.

Aufgabe 3: Validierung

Das Validierungstool generiert eine Anzahl an zufälligen Kommentaren (Default: 100 pro existierender Idee) und verteil die dazugehörigen Transaktionen auf eine Anzahl and Threads (Default: 20), um eine parallele Ausführung der Transaktionen zu ermöglichen. Anschließend

werden die Transaktionen parallel durchgeführt. Ein zufälliger Delay zwischen 0 und X (Default: 5) Sekunden wird verwendet, um realistischere Prozessdauer zu simulieren.

Das Ergebnis der Simulation spricht für den optimistischen Ansatz, da die Anzahl and Konflikten sehr niedrig war. Allerdings ist hier zu bedenken, dass meine Simulation lediglich häufige Schreibbefehle untersucht und nicht konkurrierende Lese/Schreiboperation. Hierfür hatte ich leider keine Zeit mehr. Es scheint mir aber sehr wahrscheinlich das die Konflikte hier deutlich höher ausfallen würden und grundsätzlich die Anzahl an Konflikten maßgeblich größer wird, wenn die Zwischenzeit zwischen Lese- gefolgt von Schreibebefehlen eines Nutzers wächst. Sollte z.B. ein Nutzer eine Datei einlesen, diese modifizieren und nach mehreren Stunden speichern, ist die Wahrscheinlichkeit eines Konfliktes natürlich deutlich größer als bei direktem Schreiben.