UNIVERSITÄT SIEGEN

**Fakultät IV**
**Betriebssysteme und verteilte Systeme**
Prof. Dr. rer. nat. Roland Wismüller

# Aufgabenblatt 11
# Musterlösung

**Vorlesung Verteilte Systeme**
**Sommersemester 2020**

## Aufgabe 1: Replikation

**a)** Die drei wichtigsten Punkte, die für die Erzeugung von Replikaten sprechen sind Zuverlässigkeit, Datensicherheit und die Zugriffsgeschwindigkeit.

**b)** Arten von Replikaten:

- Permanent: dauerhaft vorliegende Replikate - Ausgangsmenge der Repliken, aus denen sich ein verteilter Datenspeicher zusammensetzt (Bsp. Mirroring von Webseiten, replizierte DNS-Server).
- Serverinitiiert: bei Bedarf durch den Server veranlasst - Kopien eines Datenspeichers, die existieren, um die Leistung zu steigern und welche durch die Initiative des Eigentümers des Dateispeichers erzeugt werden. (Bsp. Web Hosting Services, Content Delivery Networks)
- Clientinitiiert bzw. Client Caches: durch den Client erzeugt - lokale Speichermöglichkeit, die vom Client verwendet wird, um temporär eine Kopie der zuvor angeforderten Daten zwischenzuspeichern.

## Aufgabe 2: Aktive Replikation

Nein. Nehmen Sie als Beispiel Leseoperationen an, die für nicht veränderte Daten stattfinden, oder kommutative Schreiboperationen. Im Prinzip erlauben diese Situationen, dass auf unterschiedlichen Repliken unterschiedliche Reihenfolgen vorliegen. Es kann jedoch schwierig, wenn nicht unmöglich sein, zu erkennen, ob beispielweise zwei Schreiboperationen kommutativ sind.

## Aufgabe 3: Quorum-Protokolle

Read- und Write-Quorum müssen den beiden folgenden Bedingungen genügen (wobei $v$ die Gesamtzahl der verfügbaren Stimmen ist):

(1) $r + w > v$

(2) $w > v/2$

Mit Bedingung (1) wird erreicht, dass die Schnittmenge zwischen jedem Read- und Write-Quorum nicht leer ist, d.h. in einem partitionierten System kann ein Objekt nicht in einer Partition gelesen und in einer anderen geschrieben werden.

Bedingung (2) verhindert Schreib-Schreib-Konflikte.

Da hier jedes Replikat genau eine Stimme hat, gilt: $N = v$.

**a)** $N$ = 10, $r$ = 6, $w$ = 5

Hier wird Bedingung (2) verletzt; Schreib-Schreib-Konflikte werden nicht erkannt.

**b)** $N = 10, r = 3, w = 7$

Hier wird Bedingung (1) verletzt; Schreib-Lese-Konflikte werden nicht erkannt.

**c)** $N = 10, r = 1, w = N$

Dies ist ein korrektes Quorum-Protokoll. Durch $r = 1$ und $w = N$ reicht ein Replikat zum Lesen, wogegen alle Replikate zum Schreiben ben/ötigt werden. Unter der Annahme, dass Lesezugriffe (viel) häufiger vorkommen als Schreibzugriffe, ist dieses Protokoll laufzeitoptimal. Im Falle von Netzpartitionierungen sind jedoch keine Schreibzugriffe mehr möglich.

(Dieses Protokoll ist auch unter der Bezeichnung '*write locks all, read locks one*' bekannt. Es entspricht dem Vorgehen, das bei primärbasierten Protokollen zur sequentiellen Konsistenz gezeigt wurde.)

**d)** $N = 10, r = 5, w = 6$

Auch dies ist ein korrektes Quorum-Protokoll. Durch $r = 5$ und $w = 6$ werden Lesezugriffe zwar aufwendiger (im Vergleich zu Protokoll c), allerdings sind Schreibzugriffe in Netzpartitionen ab sechs Knoten möglich.

## Aufgabe 4: Quorum-basierte Replikation

Die folgenden Möglichkeiten (Schreibquorum $N_W$, Lesequorum $N_R$) sind erlaubt: (10,1), (9,2), (8,3), (7,4) und (6,5).

$N_W \leq 5$ ist nicht erlaubt, da sonst Schreib-/Schreibkonflikte auftreten könnten. Natürlich wären zu jedem $N_W$ auch höhere Werte für $N_R$ möglich, das ist aber nicht wirklich sinnvoll.

## Aufgabe 5: Programmierung: Konsistenzprotokolle

**a)** Die Lösung dieser Aufgabe finden Sie im Archiv `l11Files.zip`[1] auf der Vorlesungswebseite.

**b)** Die Schwierigkeit bei der Implementierung eines *local-write*-Protokolls liegt darin, herauszufinden, welcher Server gerade der aktuelle primäre Server ist, also an welcher Stelle sich die Primärkopie von $x$ gerade befindet. Dafür müssten noch weitere Mechanismen wie Positionsdienste implementiert werden.

## Aufgabe 6: Noch Fragen?

[1] `http://www.bs.informatik.uni-siegen.de/web/wismueller/vl/vs/l11Files.zip`