universität ulm

Verteilte Systeme

Institut für Verteilte Systeme | Wintersemester 2025/2026
Alexander Heß, Prof. Dr.-Ing. Franz J. Hauck

Übungsblatt 3: Java RMI

Abgabetermin: 20.11.2025, 16:00Uhr

Aufgabe 1: Remote Key-Value Store (4)

In dieser Aufgabe sollen Sie mit Hilfe von Java RMI einen entfernte Key-Value Store realisieren. Erstellen Sie hierfür zunächst ein Java Interface `RemoteKVStore`, welches vom Interface `java.rmi.Remote` erbt und zusätzlich die folgenden Methoden definiert:

- `readRemote(String key)`

  Gibt den Wert zu einem übergebenen Schlüssel zurück. Falls kein Wert für den entsprechenden Schlüssel vorhanden ist, soll eine `RemoteException` geworfen werden.

- `writeRemote(String key, String value)`

  Speichert das übergebene Schlüssel-Wert Paar im KVStore ab. Falls bereits ein Wert für diesen Schlüssel vorhanden ist, soll dieser überschrieben werden.

- `removeRemote(String key)`

  Löscht das Schlüssel-Wert Paar basierend auf dem übergebenen Schlüssel aus dem KVStore. Falls kein Wert für den entsprechenden Schlüssel vorhanden ist, soll eine `RemoteException` geworfen werden.

Implementieren Sie dieses Interface nun in einer Klasse `RMIKVStore` und erstellen Sie zusätzlich einen Konstruktor, welcher eine `RMI Registry` erzeugt und den Server-Stub dort exportiert. Um den Key-Value Store zu realisieren, können Sie auf die Klasse `java.util.HashMap<K,V>` zurückgreifen.

Demonstrieren Sie die Funktionalität ihrer Implementierung in einer Klasse `RMIClient` , indem Sie die implementierten Methoden sinnvoll entfernt aufrufen.

Aufgabe 2: Client-side Cache (6)

In dieser Aufgabe sollen Sie ihre Implementierung aus Aufgabe 1 so erweitern, dass jeder Client einen lokalen Cache besitzt in dem diejenigen Schlüssel-Wert Paare gespeichert werden auf die der Client zugegriffen hat. Hierzu wird auch ein Mechanismus benötigt um die Werte im lokalen Cache bei Modifikationen durch andere Clients invalidieren zu können. Dies soll in dieser Aufgabe mit Hilfe von Subscriptions gelöst werden, wobei der Server entfernt auf den Cache des Clients zugreifen kann. Gehen Sie dafür folgendermaßen vor:

1. Erstellen Sie ein Interface `SubscribeKVStore`, welches von ihrem `RemoteKVStore` Interface erbt und zusätzlich die folgenden Methoden definiert:
   - `subscribe(String key, Subscriber sub)`
     Erstellt eine Subscription für den übergebenen Schlüssel. Falls der Schlüssel nicht vorhanden ist, soll eine `RemoteException` geworfen werden.
   - `unsubsribe(String key, Subscriber sub)`
     Entfernt eine Subscription für den übergebenen Schlüssel. Falls der Schlüssel nicht vorhanden ist, soll eine `RemoteException` geworfen werden.
2. Implementieren Sie das Interface `SubKVStore` nun in einer Klasse `SubRMIKVStore`, welche neben dem KVStore nun auch eine `HashMap` für die Subscriptions verwaltet. Bei einem Schreib- oder Löschvorgang sollen nun alle Clients die eine Subscription für den betroffenen Schlüssel hinterlassen haben ebenfalls aktualisiert werden.
3. Erstellen Sie nun ein Interface `Subscriber`, welches vom Interface `java.rmi.Remote` erbt und die folgenden beiden Methoden definiert:
   - `updateEntry(String key, String value)`
     Aktualisiert einen Eintrag im lokalen Cache des Subscribers. Falls kein Eintrag für den übergebenen Schlüssel vorhanden ist, soll eine `RemoteException` geworfen werden.
   - `removeEntry(String key)`
     Aktualisiert einen Eintrag im lokalen Cache des Subscribers. Falls kein Eintrag für den übergebenen Schlüssel vorhanden ist, soll eine `RemoteException` geworfen werden.
4. Erstellen Sie eine Klasse `CachedRMIClient`, welche das Interface `Subscriber` implementiert, und zusätzlich die folgenden Eigenschaften besitzen soll.
   - Einen Konstruktor welcher einen Server Stub für einen `SubRMIKVStore` erzeugt.
   - Eine Methode `write(String Key, String value)`, welche das übergebene Schlüssel-Wert Paar sowohl im lokalen Cache, als auch im entfernten KVStore speichern soll.
   - Eine Methode `remove(String key)`, welche den Wert sowohl aus dem lokalen Cache, als auch aus dem entfernten KVStore löschen soll.
   - Eine Methode `read(String key)`, welche versucht den Wert zum übergebenen Schlüssel aus dem lokalen Cache zu lesen und diesen zurückgibt. Falls kein Wert gefunden wird, soll versucht werden vom entfernten Key-Value Store zu lesen. Falls auch dort kein Wert gefunden wird, soll `null` zurückgegeben werden.
5. Demonstrieren Sie nun die Funktionalität ihrer Implementierung in einer `Test` Klasse, indem Sie eine `SubRMIKVStore` Instanz sowie mindestens zwei `CachedRMIClient` Instanzen erzeugen, und die implementierten Methoden sinnvoll aufrufen.