Aufgabe 3

Florian Hofer, Werner Nutt
Thomas Tschager

Nut ~~MD5~~ Cracker

Überblick

In dieser Aufgabe sollen Sie ein verteiltes System aus mehreren RPIs programmieren, die zusammen MD5 Hashs knacken können.

- Das System soll *schnell* sein, indem es die nötige Arbeit möglichst gleichmäßig auf die beteiligten Rechner verteilen.
- Es soll außerdem *offen* sein, indem es neuen Rechnern erlaubt, sich dem System anzuschließen.
- Es soll *ausfallsicher (resilient)* sein, indem es auch bei Ausfall eines Rechners weiterarbeiten kann.

Wettbewerb

Am Ende des Semesters wird ein Wettbewerb stattfinden, bei dem alle Systeme in einem Turnier antreten und versuchen, als erste von einem Server gestellte Aufgaben zu lösen. Die dabei erworbenen Punkte fließen in die Bewertung Ihrer Lösung ein.
Der Wettbewerb wird aus zwei Runden bestehen. In der ersten Runde treten Teams von jeweils 4 RPIs an und es geht nur darum, möglichst schnell Lösungen zu den vom Server gestellten Aufgaben zu finden. In der zweiten Runde geht es neben Schnelligkeit auch um Offenheit und Ausfallsicherheit. Es treten Teams von zunächst 3 RPIs an. Dann wird ein vierter zugeschaltet, der vom 3-er Team aufgenommen werden muss. Danach werden abwechselnd einzelne Rechner vom Netz genommen und wieder zugeschaltet. Das Team muss trotzdem koordiniert und schnell weiterarbeiten und Aufgaben lösen.

Beitrag zur Endnote

Diese Aufgabe wird mit 20% auf die Endnote angerechnet. Davon entfallen 10% auf die Aspekte Schnelligkeit und Koordination und weitere 10% darauf, wie gut Sie Offenheit und Ausfallsicherheit realisieren.

Arbeitsweise und Zeitplan

Sie arbeiten in einem Team aus drei bis vier Studenten. Für die Entwicklung und Dokumentation haben Sie Zeit bis Mittwoch, den 23. Mai.
Sie entwickeln und testen Ihr System zunächst auf Ihren eigenen Rechnern, die Sie zusammenarbeiten lassen. In den zwei Labs am 10.5. und am 17.5. haben Sie Gelegenheit, das von Ihnen entwickelte System auf RPIs zu testen.
Bis Freitag, den 11.5., reichen Sie eine erste circa 2-seitige Dokumentation Ihres Projekts ein, in der Sie (i) die Probleme identifizieren, die das Projekt aufwirft, und (ii) ihre Lösungsansätze beschreiben. Für Ihre Dokumentation erhalten Sie Feedback bis zur Übung am 17.5. Ohne diese Dokumentation wird Ihre weitere Arbeit nicht gewertet.
Die Endversion Ihres Codes und Ihrer Dokumentation reichen Sie bis Mittwoch, 23.5., ein.
Im Lab am 31.5. findet der Wettbewerb statt.

Umgebung

MD5 (= Message Digest Algorithm 5) ist ein Algorithmus, der für einen gegebenen String (z.B. eine Datei, einen Teil einer Datei oder ein Passwort) einen 128-Bit-Hash-Wert berechnet. Ihre Aufgabe ist, einen verteilten MD5-Cracker zu schreiben und auf RPIs laufen zu lassen.
Für einen gegebenen MD5-Hash-Wert soll Ihr System einen (numerischen) String mit diesem Hash-Wert finden. Die Software sollte in der Lage sein, die Rechenlast auf mehrere vernetzte Computer zu verteilen.
Um die zu knackenden MD5-Hashes zu erhalten, muss Ihr System über die folgende RMI-Schnittstelle einen Server kontaktieren (in der Auswertung wird dies die Maschine des Übungsleiters sein):

```
package server;

import java.rmi.Remote;

public interface ServerCommInterface extends Remote {

        public void register(String teamName, String teamCode, ClientCommInterface cc)
                                                                        throws Exception;

        public void submitSolution(String teamNname, String teamCode, String sol)
                                                                        throws Exception;

        public void reregister(String teamName, String teamCode, ClientCommInterface cc)
                                                                        throws Exception;

        public String getTeamIP(String teamName, String teamCode) throws Exception;

}
```

- Mit der Methode *register* registriert sich Ihr System zum ersten Mal auf dem Server mit Ihrem Teamnamen, einem (geheimen) Teamcode und einer Objektreferenz auf das Objekt (vom Typ *ClientCommInterface*), das die Registrierung durchführt. Der Teamname wird während des Wettbewerbs vom Server gezeigt. Mit dem Teamcode können sich später Mitgliedsrechner des Teams beim Server identifizieren. Die Objektreferenz wird der Server verwenden, um Ihnen Aufgaben zu geben.

- Mit der Methode *submitSolution* können Sie Lösungen an den Server übermitteln. Dabei müssen Sie sich mit dem Code Ihres Teams ausweisen.

- Mit der Methode *reregister* kann sich ein anderer Rechner Ihres Teams als Kontaktpunkt beim Server anmelden. Nur eine der Maschinen Ihres Teams darf jeweils mit dem Server verbunden sein. Der Aufruf von *reregister* überschreibt die frühere Registrierung für Ihr Team.

- Mit der Methode *getTeamIP* kann ein Rechner Ihres Teams die IP-Adresse des Kontaktpunkts beim Server erfragen. Diese beiden Methoden können nützlich sein, wenn Ihr früherer Kontaktpunkt ausgefallen ist und ersetzt werden muss oder wenn ein neuer Rechner Mitglied Ihres Teams werden will.

Das Remote Interface *ClientCommInterface*, das Sie implementieren müssen, enthält eine Methode *publishProblem*, die der Server verwendet, um Ihnen einen MD5-Hash zum Knacken zu geben.

```
package client;

import java.rmi.Remote;

public interface ClientCommInterface extends Remote {

        void publishProblem(byte[] hash, int problemsize) throws Exception;

}
```

Um die Aufgabe einfach zu halten, verwenden wir als Passwort nur Ziffernfolgen, z.B. "53071" oder "1234567". Der Parameter *hash* gibt Ihnen die maximale ganze Zahl an, die das Passwort darstellen kann.
Implementieren Sie die Methode *publishProblem* so, dass sie von Ihrem Client mit zwei lokalen Variablen *hash* und *problemsize* aufgerufen wird. Der Aufruf wird als Seiteneffekt diesen beiden Variablen neue Werte zuweisen.

Aufgabenstellung

Ihre Aufgabe ist es, einen verteilten MD5-Cracker zu implementieren, der auf mehreren RPIs läuft. Nur eine Ihrer Maschinen kann zu jedem Zeitpunkt mit dem Server verbunden sein. Ihre Rechner müssen sich um die Verteilung der Aufgaben zwischen den Maschinen kümmern. Das System muss offen und ausfallsicher im oben beschriebenen Sinne sein. Sie können Java RMI, TCP oder UDP Sockets für die interne Kommunikation wählen.
Ihre Projektabgabe muss enthalten:

- den Quellcode der Software;

- eine Dokumentation mit einer Beschreibung a) der Strategie, mit der Sie das Problem teilen, b) der Mechanismen, mit denen Sie Offenheit und Ausfallsicherheit realisieren und b) der Kommunikation, die Sie zwischen Ihren Maschinen implementiert haben.

Evaluation

Dokumentation und Code sind am Mitwoch, dem 23.5., 23:55 Uhr, einzureichen. Die Lösungen werden am 31.5. in den Übungen in einem Wettbewerb bewertet.
In der ersten Runde, bekommt jedes Team vier Raspberry PIs. Im Wettbewerb gibt der Server eine Anzahl von Hashes für alle teilnehmenden Teams aus, und jedes Mal, bekommt das Team das den Hash zuerst knackt, einen Punkt. Für jede falsche Einreichung wird ein Punkt abgezogen. In der zweiten Runde Runde werden wie oben beschrieben abwechselnd Rechner dem Team zu- oder vom Team abgeschaltet.
Die Endnote berücksichtigt sowohl die Leistung im Wettbewerb als auch den Code und die Dokumentation.