Zentrum für sichere Informationstechnologie – Austria
Secure Information Technology Center – Austria

A-1030 Wien, Seidlgasse 22 / 9
Tel.: (+43 1) 503 19 63–0
Fax: (+43 1) 503 19 63–66

A-8010 Graz, Inffeldgasse 16a
Tel.: (+43 316) 873-5514
Fax: (+43 316) 873-5520

http://www.a-sit.at
E-Mail: office@a-sit.at

DVR: 1035461 ZVR: 948166612 UID: ATU60778947

SERVERLÖSUNGEN ZU CLOUD-BASED MOBILE AUGMENTATION

VERSION 1.1 - 12. JULI 2016

Andreas Reiter – andreas.reiter@a-sit.at

Zusammenfassung:
Trotz der Tatsache, dass mobile Endgeräte immer leistungsfähiger werden, sind sie nach wie vor in ihrer Funktionalität eingeschränkt. Eine erhöhte Leistungsfähigkeit geht einher mit erhöhtem Stromverbrauch und führt somit zu einer erheblichen Reduktion der Gerätelaufzeit. Aus diesem Grund wurden in den letzten Jahren Frameworks bekannt, die es ermöglichen, dynamisch Rechenleistung von anderen Recheneinheiten zu verwenden. Diese verfügbaren Frameworks nutzen aber bei Weitem nicht das Potential dieser Technologie aus. Bestehende Lösungen sind meist nur mit einer (Cloud-basierten) Recheneinheit fix verbunden und verwenden diese bei Bedarf.
In diesem Projekt soll gezeigt werden, dass es möglich ist, diese Zuordnung auch dynamisch zur Laufzeit durchzuführen, basierend auf gesammelten Performancewerten der Recheneinheiten. Außerdem wurden von bisherigen Systemen Sicherheits- bzw. Privatsphäre-Anforderungen nicht berücksichtigt. In diesem Projekt soll gezeigt werden, dass es möglich ist, auch Applikationen mit den beschriebenen Anforderungen mit dem Cloud-based Mobile Augmentation Ansatz zu betreiben.

Inhaltsverzeichnis

Inhaltsverzeichnis 1
1. Einleitung 2
2. Architektur 2
2.1. Netzwerk Infrastruktur 4
2.2. Datenbank der Recheneinheiten 4
2.3. Sicherheitsrelevante Bewertung von Recheneinheiten und auszulagernden Programmteilen 4
3. Implementierung 4
3.1. Netzwerk Infrastruktur 4
3.2. Datenbank der Recheneinheiten 5
3.3. Entscheidungskomponente und Bewertung der Vertrauenswürdigkeit 6
3.4. Auslagerungsengine 6
4. Evaluierung 6
5. Ergebnisse 8
6. Literaturverzeichnis 8

1. Einleitung

Mobile Endgeräte wie Smartphones und Tablets ermöglichen es Benutzerinnen und Benutzern zu jeder Zeit und an jedem Ort auf ihre Daten zuzugreifen. Dieser Trend eröffnet neue Möglichkeiten für Benutzerinnen und Benutzer, wirft aber auch neue Fragen auf.
Durch immer leistungsfähigere mobile Geräte werden einerseits neue Anwendungsgebiete erschlossen, andererseits steigt auch der Stromverbrauch der meistens Akkubetriebenen Smartphones und Tablets und beeinflusst die Laufzeit negativ. Um dieses Problem zu minimieren, wurde bereits in einem vorhergehenden Projekt ein *Cloud-based Mobile Augmentation (CMA)* Framework entwickelt, das ungenutzte Ressourcen von verschiedensten Geräten zum Vorteil des Smartphones bzw. des Tablets verwenden kann. Abbildung 1 zeigt eine Übersicht über das System wie im vorhergehenden Projekt entwickelt.

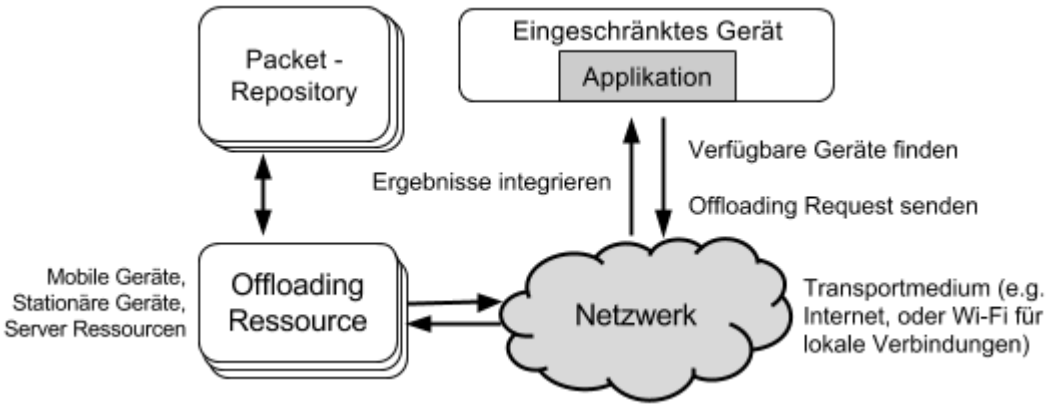

Abbildung 1 - Übersicht über das entwickelte CMA Framework

Diese Arbeit basiert auf dem Projekt *Plattformunabhängiges Cloud-based Mobile Augmentation System,* in dem Clientseitig eine solide Basis geschaffen wurde, um CMA über Plattformgrenzen hinweg zu ermöglichen.
Dieses Projekt behandelt zwei Teilaspekte von CMA-Systemen:

- Wie können Clients, die Rechenleistung anbieten, organisiert werden, sodass diese von anderen Clients effizient gefunden bzw. verwendet werden können?
- Wie kann zwischen Clients ein Vertrauensverhältnis aufgebaut werden, sodass auch sicherheitskritische Operationen ausgelagert werden können, ohne die Privatsphäre von Benutzern zu gefährden.

Die Ergebnisse wurde in einem Demonstrator realisiert der von der A-SIT Webseite[1] abgerufen werden kann.

2. Architektur

Bestehende Systeme und deren Eigenschaften wurden bereits ausgiebig in der Studio „Cloud-Based Mobile Augmentation Systems“ [1] behandelt. Prinzipiell können die darin untersuchten Systeme auf eine simple Architektur reduziert werden - wie in Abbildung 2 ersichtlich - und basieren prinzipiell auf folgenden Komponenten:

- Die *Applikation* beinhaltet Rechenintensive Aufgaben, die mittels Auslagerung beschleunigt und effizienter bzw. Akkuschonender ausgeführt werden können.
- Der *Applikationsprofiler* analysiert die Applikation, um Performancewerte von Programmteilen zu ermitteln, die ausgelagert werden können. Einige Frameworks stellen Echtzeitprofiler zur Laufzeit bereit, andere verwenden Offlineprofiler. Hierbei werden die Performancewerte in einem eigenen profiling Schritt ermittelt.
- Mit Hilfe der *Entscheidungskomponente* wird entschieden, ob es zu einem gewissen Zeitpunkt vorteilhaft ist, einen Programmteil auszulagern oder nicht. Dazu verwaltet die *Entscheidungskomponente* eine Liste von Performancewerten der verfügbaren Recheneinheiten.

[1] http://demo.a-sit.at/serverloesungen-zu-cloud-based-mobile-augmentation

- Die *Auslagerungsengine* kontrolliert den gesamten Prozess der Auslagerung eines Programmteils und übernimmt die Integration des zurückgelieferten Ergebnisses in das lokal laufende Programm.

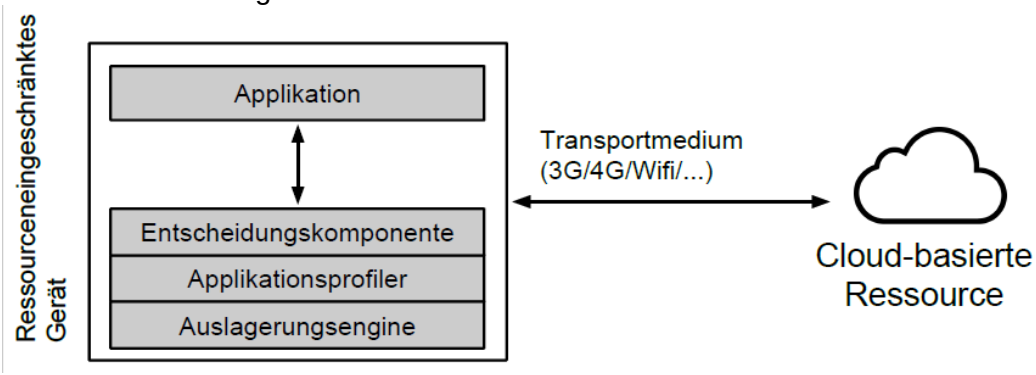

Abbildung 2 - Abstrahierte Architektur von bestehenden Systemen

Wie in Abbildung 2 dargestellt, werden die Recheneinheiten auf statische Art und Weise instanziert. Eine externe Recheneinheit ist dabei exklusiv einem Client zugeordnet. Dies erfordert auch von den Betreiberinnen bzw. Betreibern der Recheneinheiten zusätzliche Schritte, um eine direkte Verbindung zu ermöglichen:

- Firewall Einstellungen müssen angepasst werden, um Zugriff auf den entsprechenden Port zu ermöglichen.
- Router müssen konfiguriert werden, um Anfragen an den entsprechenden Rechner weiterzuleiten.
- Sollte das System ein automatisches Auffinden von Recheneinheiten unterstützen, muss die entsprechende Recheneinheit bei diesem zentralen System registriert werden.

Dabei handelt es sich um einen relativ beschränkten Ansatz. Deshalb wird die Architektur wie in Abbildung 3 dargestellt erweitert und verbessert.

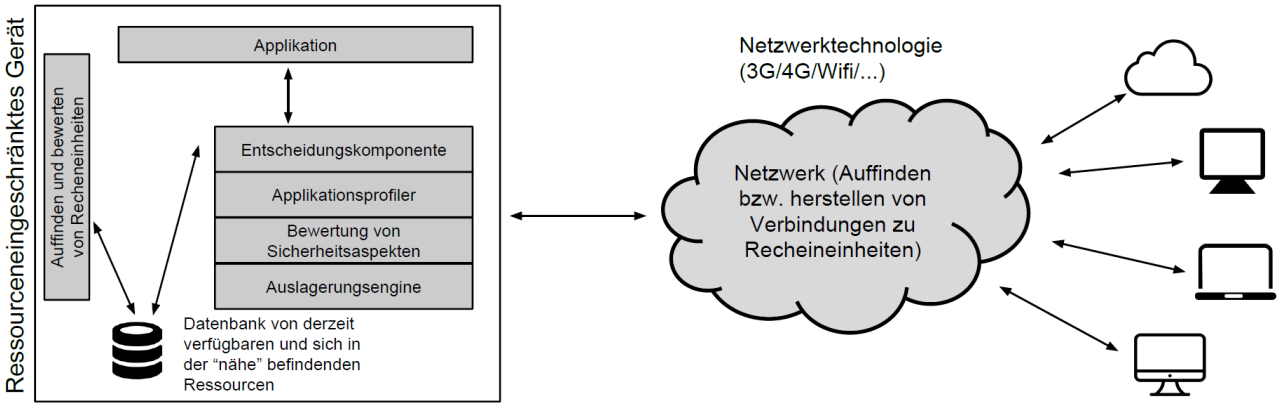

Abbildung 3 - Übersicht über die verbesserte Architektur

Statische Verbindungen zu externen Recheneinheiten werden durch eine dynamische Netzwerkinfrastruktur ersetzt mit dem Ziel, keinerlei Konfigurationsaufwand für Betreiber von Recheneinheiten zu verursachen.
Clientseitig wird die Architektur um folgende Komponenten erweitert:

- Eine Datenbank in der aktuell verfügbare Recheneinheiten aufgezeichnet und von der Auslagerungsengine schnell abgerufen werden können,
- eine Hintergrundkomponente, um neue Recheneinheiten zu finden und um deren Performancewerte zu ermitteln,
- eine Komponente zur Bewertung von sicherheitsrelevanten Eigenschaften von Programmteilen die ausgelagert werden sollten, bzw. zum Bewerten von externen Recheneinheiten.

2.1. Netzwerk Infrastruktur

Bei der hier eingeführten Netzwerk-Infrastruktur handelt es sich um ein selbstorganisierendes Netzwerk zum effizienten Auffinden von Recheneinheiten. Das Netzwerk ist auf zwei Ebenen organisiert:

- Die obere Ebene verwaltet alle vorhandenen Recheneinheiten in einem dezentralen Peer-to-Peer Netzwerk,
- die untere Ebene ermöglicht einen direkten Verbindungsaufbau zwischen Knoten.

Somit können die Vorteile aus beiden Welten genutzt werden: In Peer-to-Peer Netzwerken kann jeder Knoten mit jedem kommunizieren, auch wenn keine direkte Verbindung zwischen zwei Knoten besteht. In diesem Fall wird die Kommunikation über andere Knoten umgeleitet, bis schlussendlich der Zielknoten erreicht ist. Die Kommunikation über Peer-to-Peer Netzwerke ist generell langsamer als eine direkte Kommunikation, da im Normalfall mehrere Knoten zwischen Ursprungs- und Zielknoten liegen. Außerdem sind Peer-to-Peer Netzwerke dem ständigen Bei- und Austreten von Knoten ausgesetzt, vor allem wenn es sich um mobile Geräte handelt. Dies kann zu Verbindungsabbrüchen führen. Deshalb wird in diesem Fall das Peer-to-Peer Netzwerk nicht direkt zur Nutzdatenübertragung verwendet, sondern zum Auffinden von anderen Clients und unterstützend beim direkten Verbindungsaufbau zum Austausch von Verbindungsdaten.

2.2. Datenbank der Recheneinheiten

Die Architektur sieht eine Datenbank vor, in der sich die Verbindungsinformationen zu Recheneinheiten befinden, die für die aktuelle Applikation als gut befunden wurden. Die Kriterien dazu können beispielsweise sein:

- Latenz zur Recheneinheit,
- Bandbreite der Recheneinheit, oder
- Vertrauenswürdigkeit der Recheneinheit.

Der Hintergrundtask zum Auffinden und Bewerten von Ressourcen evaluiert diese Kennwerte auf periodischer Basis, und hält gegebenenfalls nach neuen Recheneinheiten Ausschau, falls Grenzwerte überschritten werden.

2.3. Sicherheitsrelevante Bewertung von Recheneinheiten und auszulagernden Programmteilen

In der Studie zu CMA Systemen [1] wurden sicherheitsrelevante Anforderungen für Mobile Cloud Computing Systeme definiert und bestehende Systeme analysiert. Keines der untersuchten Systeme berücksichtigt Anwendungen mit sicherheitstechnischen bzw. privacy Anforderungen zufriedenstellend.

Im Falle dieses Projekts wurde der Ansatz gewählt, dass die Entwicklering bzw. der Entwickler Codeteile mit ihren Sicherheitsanforderungen an die Recheneinheit kennzeichnet. Auf diese Weise kann das Auslagerungsframework feststellen ob eine Recheneinheit für die Ausführung eines bestimmten Codeblocks geeignet ist.

Die Bewertung von Recheneinheiten erfolgt Zertifikatsbasiert. Die Benutzerin bzw. der Benutzer kann bestimmten Zertifikaten ein höherer Sicherheitslevel zuordnen, und qualifiziert diese damit zum Ausführen von Programmteilen mit höheren Anforderungen.

3. Implementierung

Die Implementierung basiert auf dem in [2] entwickeltem Plattformunabhängigem CMA System. Das Framework basiert auf neuen Web-Technologien, und zielt auf Web Applikationen, aber auch auf *Cross-Platform* Applikationen, die in HTML5 entwickelt werden. Das Framework wurde also um die oben beschriebenen Komponenten erweitert, die in den folgenden Kapiteln erläutert werden.

3.1. Netzwerk Infrastruktur

Um ein Peer-to-Peer Netzwerk im Browser mittels Javascript zu realisieren, wird eine Technologie benötigt, die die Herstellung einer direkten Verbindung von Browser zu Browser ermöglicht. Die im Zuge dieses Projekts erweiterte Implementierung basiert auf WebRTC [3], um dies zu ermöglichen. WebRTC hat die Einschränkung, dass ein separater Kanal benötigt

wird, um initiale Verbindungsinformationen auszutauschen. Eine Übersicht über die WebRTC Technologie ist in Abbildung 4 ersichtlich.

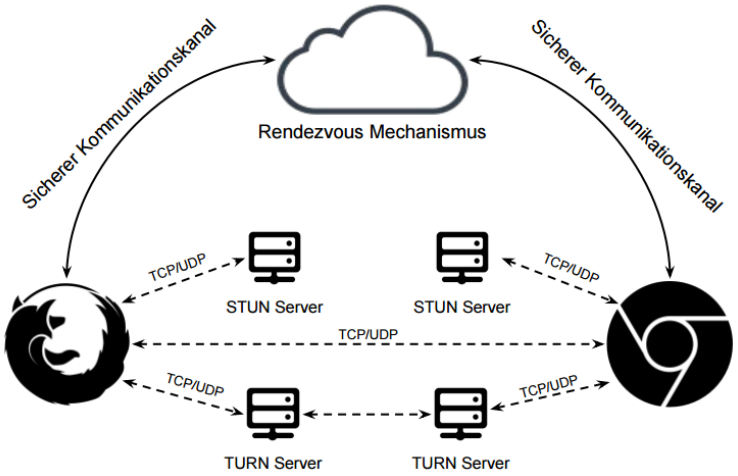

Abbildung 4 - WebRTC Übersicht

Verbindungsinformationen müssen über einen separaten sicheren Kommunikationskanal ausgetauscht werden, bevor eine direkte Verbindung hergestellt werden kann. Der Verbindungsaufbau wird dann unter Verwendung des *Interactive Connection Establishment* (ICE) Protokolls [4] durchgeführt. Das ICE Protokoll wird verwendet um eventuell bestehende *Network Address Translation* (NAT) Gateways zu erkennen, und zu umgehen. ICE verwendet dazu einerseits STUN Server [5] um die öffentliche IP Adresse von Rechnern zu erkennen, und andererseits, sollte trotzdem keine direkte Verbindung möglich sein, wird eine Verbindung über externe Relayserver unter Verwendung der TURN Technologie [6] hergestellt.
Basierend auf diesen Technologien wurde ein bestehendes Peer-to-Peer Framework erweitert. Es verwendet WebSockets zu zentralen Servern als Kanal zum Übertragen der initialen Verbindungsinformationen. Wurde eine Verbindung zum Peer-to-Peer Netzwerk hergestellt, wird die zentrale Serverkomponente nicht länger benötigt. Selbst zum Herstellen der direkten Client-Verbindungen, kann das Peer-to-Peer Netzwerk verwendet werden, um die initialen Verbindungsinformationen auszutauschen.

3.2. Datenbank der Recheneinheiten

Die Datenbank der aktuell verfügbaren Recheneinheiten wird fortlaufend überwacht um ausgefallene bzw. auffällige Recheneinheiten zu erkennen und entfernen zu können.
Die Anforderungen an Recheneinheiten können von Applikation zu Applikation unterschiedlich sein. Eigenschaften der Datenbank werden somit von der Applikation mitbestimmt wie beispielsweise: Anzahl der Recheneinheiten, die in der Datenbank eingetragen werden sollen, oder Schwellwerte für Latenz bzw. Bandbreite.
Das Auffinden und Bewerten von Recheneinheiten inkludiert wie in Abbildung 5 dargestellt folgende Abläufe:

1. Überprüfen ob alle Datenbankanforderungen erfüllt sind, wie die Anzahl der Recheneinheiten in der Datenbank, oder Schwellwerte für Latenz und Bandbreite. Sollten nicht alle Anforderungen erfüllt sein, wird mit Schritt 2 fortgesetzt, ansonsten der Vorgang unterbrochen.
2. Es wird nach neuen Recheneinheiten mit Hilfe des Peer-to-Peer Netzwerks gesucht.
3. Es wird versucht, eine direkte Verbindung zu den gefundenen Recheneinheiten herzustellen, sollte keine direkte Verbindung hergestellt werden können, so wird mit Schritt 2 fortgesetzt.
4. Es wird die Latenz der neu hergestellten direkten Verbindung ermittelt und die Recheneinheit in die eingefügt.
5. Die bestehende Verbindung wird zu einer TLS Verbindung aufgewertet, um der Recheneinheit ein entsprechendes Vertrauenslevel zuzuweisen.

Das Auffinden von neuen Recheneinheiten kann dahingehend optimiert werden, dass in Schritt 2 bereits eine Liste von Recheneinheiten abgerufen wird, die mit hoher Wahrscheinlichkeit eine niedrige Latenz zum Client aufweisen. Dies kann beispielsweise über *Geo-Location* basierte Ansätze aber auch über verbesserte Ansätze, wie von Agarwal und Lorch [7] vorgeschlagen, erfolgen.

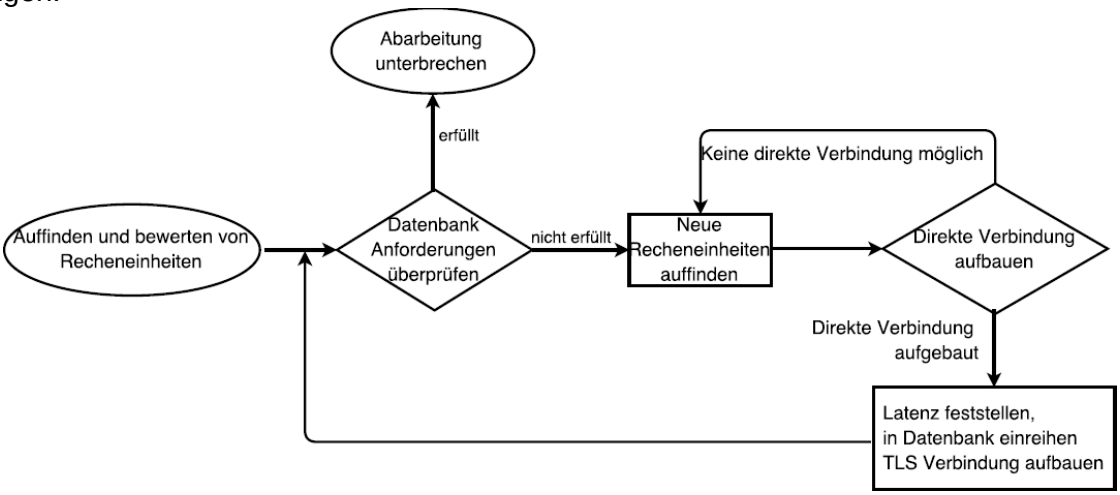

Abbildung 5 - Hintergrundtask zum Auffinden und Bewerten von Recheneinheiten

3.3. Entscheidungskomponente und Bewertung der Vertrauenswürdigkeit

Die Entscheidungskomponente gibt Auskunft darüber, ob es zum aktuellen Zeitpunkt Vorteile bringt, einen bestimmten Programmteil auszulagern und liefert gleichzeitig entsprechende Recheneinheiten, auf die ausgelagert werden könnte.
Die Entscheidungskomponente arbeitet eng mit den sicherheitsrelevanten Komponenten zusammen. Jedem auszulagerndem Programmteil sind Sicherheitsrelevante Parameter zugeordnet, die bestimmen, ob das Auslagerungsziel Vertrauenswürdig sein muss oder nicht.
Aktuell werden zwei Profile für die Entscheidungskomponente unterstützt:

- Das *Performanceprofil* optimiert die Rechenleistung für die Benutzerin bzw. den Benutzer. Applikationen die dieses Profil verwenden, werden also im besten Fall eine höhere Programmperformance erfahren, als würde die Applikation nur am mobilen Gerät ausgeführt werden.
- Das *Energiesparprofil* versucht trotz Auslagerung die Performance der Applikation weder zum positiven noch zum negativen zu beeinflussen, und erreicht damit den maximalen Energiespareffekt.

3.4. Auslagerungsengine

Die Auslagerungsengine bietet die technische Grundlage, um Programmteile auszulagern. Diese wurde bereits in [2] erläutert. Trotzdem wurden einige Verbesserungen an den Kommunikationseinrichtungen der Auslagerungsengine vorgenommen.

- Die Serialisierung von großen Objekten in eine JSON Repräsentation stellte sich als sehr Zeitaufwändig dar, und ist deshalb für diesen Use Case nicht geeignet. Es wird stattdessen *MessagePack*[2] verwendet.
- Außerdem werden übertragene Daten vor der Übertragung mittels *lz4*[3] komprimiert. Dabei handelt es sich um einen Komprimierungsalgorithmus der auf hohe Geschwindigkeiten ausgelegt ist und hat die Auswirkung, dass die übertragenen Daten erheblich verkleinert werden.

4. Evaluierung

Der Beitrag dieses Projekts ist es, *flexibles* und *sicheres* Teilen von Rechenleistung zu ermöglichen. Die Flexibilität wird dadurch gewährleistet, dass Benutzerinnen bzw. Benutzer

2 http://msgpack.org/

3 https://github.com/pierrec/node-lz4

nicht an einzelne Recheneinheiten bzw. virtuelle Maschinen gebunden sind, sondern dass diese von einer Vielzahl von Benutzerinnen und Benutzern verwendet werden können. Das eingeführte, dezentrale Model ist ausfallsicher und hat keinen *single-point-of-failure.* Die derzeitige Implementation kann nicht vollständig auf die zentrale Komponente verzichten, wegen Einschränkungen in der WebRTC Technologie. Trotzdem wird ein dezentraler Kommunikationskanal bereitgestellt über den direkte Verbindungen aufgebaut werden können. Der zentrale Punkt ist also nur für den initialen Verbindungsaufbau erforderlich.
Das entwickelte System wurde praktischen Tests unterzogen. Dazu wurde ein Testsetup verwendet wie in Abbildung 6 dargestellt.

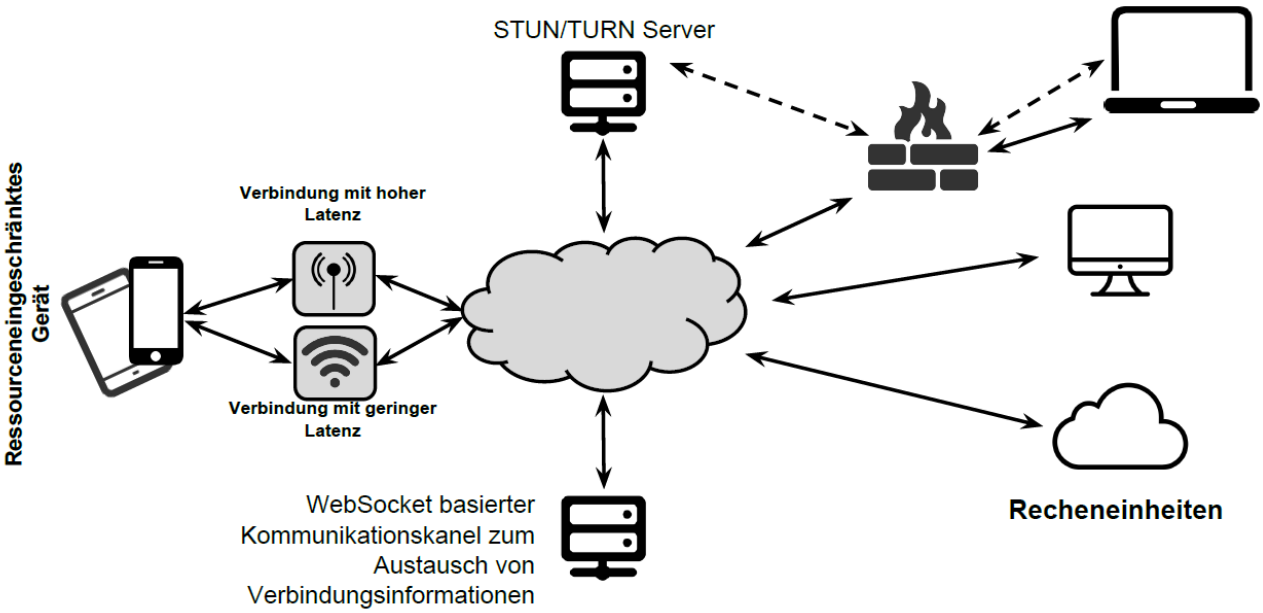

Abbildung 6 - Testsetup

Die Testumgebung besteht aus drei Recheneinheiten, wobei zwei davon direkt über das Internet erreichbar sind. Die dritte Recheneinheit kann nur über einen TURN Server verwendet werden. Ein WebSocket basierter Server wird zum Austausch der initialen Verbindungsinformationen verwendet. Die Testapplikationen wurden konfiguriert, sodass immer zwei Verbindungen zu Recheneinheiten bestehen müssen. Von der Auslagerungsengine bzw. der Entscheidungskomponente wird dann automatisch die Recheneinheit ausgewählt die die besseren Performance Werte aufweist.
Als mobile Endgeräte wurden Geräte mit Android 5, sowie mit Linux und Windows verwendet. iOS und Windows Phone weisen derzeit noch keine WebRTC Unterstützung auf.
Die Performance wurde anhand einer Raytracer[4] Testapplikation evaluiert. Die Eingabewerte der Applikation sind eine Beschreibung einer Szene, woraufhin die Szene als Bild berechnet wird. Die Ausgabe der Applikation ist eine Liste von Pixeln.
Die Szene wurde für die Performanceevaluierung mit 65535 Pixel generiert. Als Performancewert wird der Mittelwert aus 200 berechneten Bildern verwendet. Die Evaluierung für die ausgelagerte Ausführung wurde unter Verwendung der unmodifizierten Programmversion durchgeführt, aber auch unter optimierten, auf das Framework zugeschnittenen Varianten sowie einer parallelisierten Variante. Abbildung 7 zeigt die Ergebnisse der Evaluierung.
Ohne Optimierungen anzuwenden, war es bereits möglich, auf Android Geräten die Performance um 30% zu verbessern. Die optimierten Programmvarianten können für Android Geräte bereits eine Performanceverbesserung von 700% aufweisen. Auf anderen leistungsstärkeren Systemen fällt die Verbesserung geringer aber dennoch erheblich aus.
Weitere Performance und Energieverbrauchsevaluierungen wurden in *Flexible and Secure Resource Sharing for Mobile Augmentation Systems*[5] [8] veröffentlicht

[4] https://github.com/dartist/raytracer

[5] http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7474403

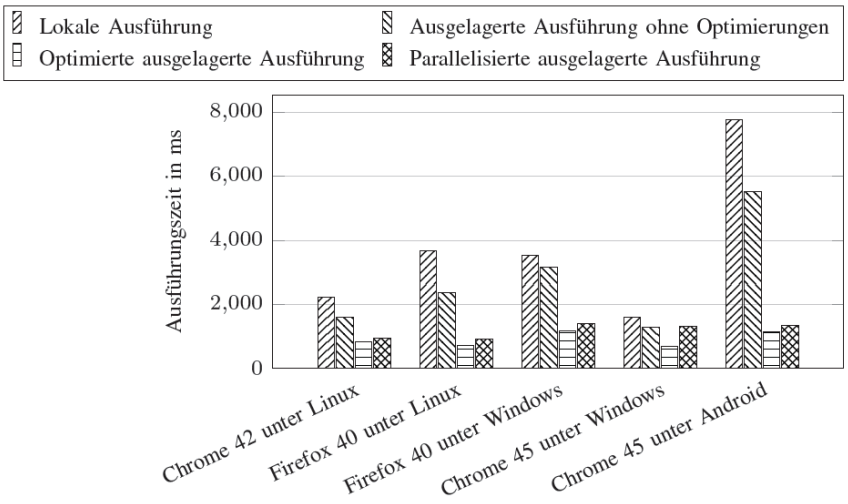

Abbildung 7 - Performanceevaluierung der Raytracer Applikation

5. Ergebnisse

In diesem Projekt wurde eine verbesserte Cloud-based Mobile Augmentation Architektur entwickelt die es ermöglicht, Recheneinheiten in *Hybrid Mobile Cloud Computing* Szenarien zwischen mehreren Benutzerinnen und Benutzern zu teilen, und so die fixe Zuordnung zu verhindern. Die Architektur und Proof-of-Concept Implementation ermöglicht den Einsatz dieses Systems auch für sicherheitskritischere Anwendungen, durch die Bewertung der Vertrauenswürdigkeit einer Recheneinheit basierend auf angebotenen Zertifikaten.
Die Ergebnisse sprechen für sich. Für ausgewählte Anwendungen kann eine Performancesteigerung von 30% im nicht optimierten Fall und weit darüber im optimierten Fall erreicht werden.

6. Literaturverzeichnis

[1] A. Reiter, „Cloud-based Mobile Augmentation Systems," 16. Februar 2015. [Online]. Available: http://demo.a-sit.at/cloud-based-mobile-augmentation-systeme/.

[2] A. Reiter, „Plattformunabhängiges Cloud-based Mobile Augmentation System," 22. Oktober 2015. [Online]. Available: http://demo.a-sit.at/platformunabhaengiges-cma-system/.

[3] A. Bergkvist, D. C. Burnett, C. Jennings, B. Narayanan und B. Aboba, „WebRTC 1.0: Real-time Communication Between Browsers," 13. Mai 2016. [Online]. Available: https://w3c.github.io/webrtc-pc/.

[4] J. Rosenberg, „Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols," 2010. [Online]. Available: https://tools.ietf.org/html/rfc5245.

[5] J. Rosenberg, R. Mahy, P. Matthews und D. Wing, „Session Traversal Utilities for NAT (STUN)," 2008. [Online]. Available: https://tools.ietf.org/html/rfc5389.

[6] R. Mahy, P. Matthews und J. Rosenberg, „Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)," 2010. [Online]. Available: https://tools.ietf.org/html/rfc5766.

[7] S. Agarwal und J. R. Lorch, „Matchmaking for online games and other latency-sensitive P2P systems," ACM SIGCOMM Computer Communication, vol. 39, no. 4, p. 315, 2009.

[8] A. Reiter und T. Zefferer, „Flexible and Secure Resource Sharing for Mobile Augmentation Systems," IEEE, 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud) , 2016.