# Entwicklung eines neuen Verfahrens zur Serverzuweisung von Avataren in verteilten virtuellen Systemen

Sven Allers

HAW Hamburg,
sven.allers@haw-hamburg.de

**Zusammenfassung.** Verteilte virtuelle Systeme dienen dem interagieren mehrerer Parteien in einer 3D-Umgebung über ein Netzwerk. Sobald gewisse Dimensionen erreicht werden ist es unerlässlich, dass ein Lastverteilungsverfahren genutzt wird. Das Heat Map Verfahren ist ein globales Verfahren, welches die virtuelle Umgebung in ein Raster aufteilt, um eine optimale Zuordnung der Avatare zu Servern, zu ermitteln. Die folgende Arbeit beschäftigt sich mit den Arbeiten die im Rahmen dieses Verfahrens bereits getätigt wurden und gibt einen Überblick über mögliche zukünftige Entwicklungen.

**Schlüsselwörter:** Lastverteilung, verteilte virtuelle Systeme

## 1 Einführung

Verteilte virtuelle Systeme (DVE) ermöglichen die Interaktion mehrerer Teilnehmer in virtuellen 3D-Welten. Ihre Einsatzgebiete sind weitreichend. So werden sie im militärischen Bereich, für Simulationen sowie im Entertainmentbereich eingesetzt [33]. Die Wohl bekannteste Form von DVEs sind Online-Spiele. In diesem Bereich kann die Anzahl der Teilnehmer auf über 100.000 steigen [17]. Um eine solch hohe Anzahl behandeln zu können, muss eine geeignete Lastverteilungsstrategie vorliegen. Eine häufig eingesetzte Technik um große Teilnehmerzahlen effizient zu behandeln ist das Sharding. Bei dieser Technik wird die Spielwelt mehrfach repliziert. Dies ermöglicht es, die Spielermengen innerhalb der Spielinstanzen zu begrenzen und bei hohem Andrang neue Instanzen zu starten. Ein bekanntes Beispiel hierfür ist World of Warcraft [19]. Solche Shards beschränken jedoch die Interaktionsmöglichkeiten zwischen den einzelnen Teilnehmern, da Teilnehmer auf den verschiedenen Instanzen nicht miteinander interagieren können. Es gibt jedoch auch Spiele, die diese Beschränkung nicht haben möchten. In Eve-Online spielen alle Spieler in einem gemeinsamen Universum. In einem solchen Fall muss es Möglichkeiten geben die virtuelle Welt über mehrere Maschinen zu verteilen. Eine häufig eingesetzte Technik zur Verteilung der Last ist das Partitionieren der Umgebung. Hierbei wird die virtuelle Umgebung in mehrere Regionen unterteilt, wobei jede Region von einem Server verwaltet wird [28]. Dies kann auch in Kombination mit Sharding geschehen. In einem solchen Fall wird jeder Shard zusätzlich in mehrere Regionen aufgeteilt und von einem Servercluster verwaltet. So geschieht dies z.B. in World of Warcraft [1, 19]. Die Regionen, die ein Server verwaltet, müssen

jedoch nicht fix sein. In dynamischen Partitionierungsverfahren kann die Aufteilung der virtuellen Welt der Lastsituation angepasst werden [10,16]. Häufig wird die virtuelle Umgebung dazu in kleinere Zellen eingeteilt, die bei Überlastsituationen an andere Server abgetreten werden können [9,35]. Allerdings kann es trotz solcher Verfahren zu Überlastsituationen kommen, wenn die Last durch Avatare in einzelnen Regionen in einem kurzen Zeitraum rapide ansteigt. Die Gründe hierfür können von neuen Erweiterungen [34], bis hin zu einer unglücklichen Verschiebung der Last aufgrund von unvollständigen Informationen reichen [18]. Dies liegt am sogenannten Flocking. Das Bedeutet, dass die Avatare nicht gleichmäßig in der virtuellen Umgebung verteilt sind, sondern es Bereiche gibt, die von Avataren deutlich stärker besucht werden als andere [8]. Solch ein Verhalten konnte auch in Studien nachgewiesen werden [27]. Einige Verfahren verzichten auf die Partitionierung der virtuellen Umgebung und verteilen die Avatare unabhängig von ihrer lokalen Position [12,20]. Bei allen Verfahren ist es jedoch schwierig eine optimale Lösung zu finden. Denn beim Client Assignment handelt es sich um ein NP-vollständiges Problem [8]. Wird es als Gruppierungsproblem aufgefasst ist es sogar nachweislich NP-schwer [13]. Ein weiterer Aspekt der betrachtet werden kann um Überlastsituationen zu verhindern, ist die Skalierbarkeit der Systeme. Peer-to-Peer Systeme bieten hier eine Möglichkeit effizient zu skalieren [25]. Diese haben jedoch das Problem, dass eine Synchronisierung der Daten aufwendiger ist. Auch das Schützen sicherheitskritischer Aspekte ist deutlich aufwändiger [7,25]. Andere Ansätze erhöhen die Skalierbarkeit, indem auch externe Resourcen von verschiedenen Dienstleistern gebucht werden können, wenn die Rechenleistung des eigenen Rechenzentrums nicht mehr ausreicht [7,24]. Das in dieser Arbeit vorgestellt Verfahren verfolgt das Ziel, die Auslastung der Server zu optimieren indem, durch räumliches Gruppieren der Avatare, der Kommunikations- und Synchronisationsaufwand verringert wird. Zeitgleich wird die Verteilung nicht durch eine Partitionierung beschränkt, um flexibler auf spontane Lastsituationen reagieren zu können.

## 2 Stand der Forschung

Eine der am häufigsten eingesetzten Strategien ist die Partitionierung [10,16,28]. In einigen Verfahren werden Avatare jedoch auch unabhängig von ihrer räumlichen Position auf die Server verteilt. Es gibt verschiedene Motivationen solch einen Ansatz zu benutzen. In manchen Fällen werden viele Lastverteilungsverfahren als zu komplex empfunden, sodass einfache Verfahren wie Round Robin eingesetzt werden [20]. In anderen Fällen wird erhofft eine flexiblere Anpassbarkeit an die Lastsituation zu erreichen, da es weniger Abhängigkeiten zwischen Avatar und zugewiesenem Server gibt [4]. Die Verfahren können global sowie lokal verwaltet werden. Bei globalen Verfahren stehen die Zustandsinformationen sämtlicher Server zur Verfügung. In der Regel gibt es bei solchen Verfahren eine zentrale Instanz, die die Informationen sammelt, damit sie zur Berechnung zur Verfügung stehen. Ein Beispiel hierfür ist der Load Collector, der die Lastinformationen sämtlicher Server erhält und bei drohender Überlast kontaktiert wird, um eine Umverteilung zu initialisieren [18]. Bei lokalen Verfahren kann jeder Server selbstständig eine Umverteilung der Last initialisieren, wenn eine Überlastsituation droht. Hierzu stehen ihnen in der Regel die Lastinformationen von

benachbarten Servern sowie ihre eigenen zur Verfügung [16]. Ein oft genutztes Konzept zur Verwaltung von Avataren ist die Area of Interest (AoI). Hierbei handelt es sich um den Bereich, der für einen Avatar potentiell interessant ist, weil er ihn sehen oder Objekte in dem Bereich manipulieren kann [21].

Im kommerziellen Kontext ist eine statische Partitionierung der Spielwelt eine der verbreitetsten Methoden [11, 28]. Es kommen allerdings auch dynamische Verfahren zum Einsatz. So wird in Eve-Online mithilfe eines binären Baumes versucht die Systeme der Spielwelt optimal aufzuteilen [34]. Ein weiteres Verfahren zur dynamischen Partitionierung sind sogenannte Mikrozellen. Hierbei wird die virtuelle Welt in viele kleine Zellen unterteilt. Diese können bei Überlastsituationen an andere Server abgetreten werden, die weniger stark belastet sind [9]. Dieses Verfahren gibt jedoch keine Garantien darüber, ob die verwalteten Regionen der Server zusammenhängend bleiben. Um jedoch den Kommunikationsaufwand und den Migrationsaufwand zwischen den Servern niedrig zu halten, ist es sinnvoll möglichst zusammenhängende Regionen von Servern verwalten zu lassen. Deshalb werden bei vielen zellenbasierten Ansätzen Zellen nur an benachbarte Server abgetreten. Dazu werden nur Zellen ausgewählt, die sich an der Grenze zu dem ausgewählten benachbarten Server befinden [10, 26]. Beim Locality Aware Partitioning wird auch primär versucht die Last an benachbarte Server abzutreten. Sollten diese jedoch zu stark ausgelastet sein, können sie eine Annahme der Last verweigern. In einem solchen Fall können Zellen auch an nicht benachbarte Server abgetreten werden. Dies erhöht jedoch die Interserverkommunikation. Ist diese zu hoch, so werden Regionen aggregiert, um sie zu verringern [8]. Ein anderer Ansatz ist die Lastverteilung mittels Hitzediffusionsgleichung. Diese wird genutzt, um zu berechnen, wie viel Last von den einzelnen Servern abgetreten werden muss, damit sich die Belastung des Systems ausgleicht [10]. Es gibt jedoch auch die Ansicht, dass solche Verfahren zu aufwendig sind. Deshalb gibt es den Vorschlag bei Überlast fest definierte Bereiche an benachbarte Server abzutreten [16].

Das objektorientierte Management ist ein Beispiel für eine Avatarverwaltung ohne Partitionierung. Sollte ein Server überlastet sein, so werden die Avatare nach ihrem Präsenzfaktor sortiert und auf die am geringsten ausgelasteten Server umsortiert. Der Präsenzfaktor beschreibt hierbei, in wie vielen AoIs von anderen Avataren sich ein Avatar befindet [23]. Einige gehen noch weiter und wollen gar keinen spezialisierten Verwaltungsalgorithmus einsetzen. Stattdessen wird vorgeschlagen einen NAT einzusetzen und die Avatare mit Standardverfahren, wie Round Robin zu verteilen [20]. In einem weiteren Verfahren wird für jeden Avatar ein eigener Prozess gestartet. Dies soll die virtuelle Umgebung skalierbarer machen [12].

## 3 Forschungslücke

Die meisten Verfahren zur Verteilung von Avataren beschäftigen sich mit der Partitionierung der Spielwelt [10, 19, 32]. Trotz starker Dynamisierung kann es aufgrund von übermäßigem Flocking immer noch zu Überlastsituationen kommen [34]. Zudem haben Partitionierungsverfahren den Nachteil, dass ein erhöhter Kommunikationsaufwand

betrieben werden muss, wenn sich Avatare in Grenzbereiche befinden. So müssen alle Objekte bezogen werden, die sich innerhalb des AoI eines Avatars befinden, wenn dieser sich an der Grenze zu einer anderen Region befindet. Alle Änderungen die durch den Avatar auf diesen Objekten geschehen, müssen auch an den verwaltenden Server übermittelt werden [16]. Sollte ein Avatar zudem in kurzer Zeit regelmäßig die Grenze einer Partition überschreiten, so muss dieser auch jedes mal migriert werden. Dieser Umstand beschränkt die Möglichkeiten des Spieldesigns, als dass dies Berücksichtigt werden muss und keine Hot Spots an Grenzregionen existieren dürfen. Deshalb wurde mit dem Heat Map Verfahren ein Algorithmus entwickelt, der ohne Partitionierung auskommt. Im Rahmen des Forschungsprojektes Timadorus lag zudem bereits ein Persistenzsystem vor, welches die Möglichkeit bietet dieselbe Region durch mehrere Server verwalten zu lassen [5]. Durch ausnutzen dieser Eigenschaft sollte eine flexiblere Verteilung der Last erreicht werden, da bei extremen Lastsituationen nicht zwingend auf eine Partitionierung geachtet werden muss. Dennoch profitiert das System stark von einer Gruppierung der Avatare [5, 6]. Damit ist gemeint, dass Avatare die räumlich nah beieinander liegen auch demselben Server zugewiesen werden. Dies liegt zum einen daran, dass das Risiko verringert wird, dass zwei Avatare zeitgleich versuchen dasselbe Objekt zu manipulieren, zum anderen müssen die Daten zwischen weniger Systemen synchronisiert werden. Ziel war es daher ein Verteilungverfahren zur Verfügung zu stellen, dass ohne Partitionierung arbeitet und dennoch in der Lage ist die Avatare zu gruppieren.

Werden bisherige Verfahren zur Verteilung betrachtet, die ohne Partitionierung arbeiten, so gibt es in der Regel kein konkretes Ziel zur genauen Verteilung der Avatare. Beim objektorientierten Management wird die Verteilung optimiert, indem die Avatare mit dem höchsten Präsenzfaktor auf die schwach ausgelasteten Server verschoben werden [23]. Eine genaue Zuordnung von Avataren aufgrund eines bestimmten Attributs ist nicht vorgesehen. Auch bei einer Zuteilung über einen NAT [20] oder wenn für jeden Avatar ein eigener Prozess gestartet wird [12], wird keine Annahme über die genaue Verteilung gemacht. Der entwickelte Algorithmus verfolgt jedoch das Ziel ein den Kommunikations- und Synchronisationsaufwand mittels der Informationen über die räumliche Position der Avatare zu minimieren und diese entsprechend zu verteilen. Ein solches Verfahren stand, nach aktuellem Kenntnisstand, jedoch noch nicht zur Verfügung.

# 4 Konzept

Das Heat Map Verfahren wurde zur Validierung in das Timadorus Projekt integriert. Dort konnte bereits nachgewiesen werden, dass der Algorithmus auch praktisch in der Lage ist Avatare zu gruppieren und dabei effizient arbeitet [4]. Auch zukünftige Entwicklungen werden zur Validierung in dieses System integriert. Im Folgenden soll dieses deshalb näher betrachtet werden. Zudem wird das Heat Map Verfahren noch einmal vorgestellt.

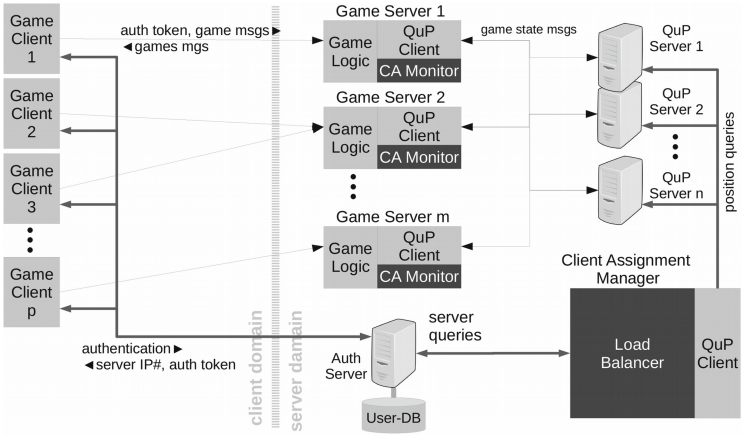


Abb. 1: Möglicher Aufbau für Timadorus [4]

### 4.1 System zur Validierung des Verfahrens

Alle weiteren Entwicklungen werden im Kontext des Timadorus Projektes validiert. Eine Referenzarchitektur liegt für das Projekt bereits vor (Abb. 1). Der Client Assignment Manager ist hierbei das System in dem das Verfahren zur Verteilung der Avatare umgesetzt wurde. Es besteht aus zwei Komponenten, dem CA Monitor sowie dem Load Balancer. Der CA Monitor hat die Aufgabe den Zustand des Spielservers zu überwachen und diesen an den Load Balancer zu übermitteln. Entsprechend wird er in den Spielserver integriert. Der Load Balancer sammelt die Informationen der Spielserver und nutzt diese um für die Avatare einen geeigneten Server zu ermitteln. Nachdem ein Avatar einem Spielserver zugewiesen wurde, erfolgt in periodischen Abständen eine Neuberechnung. Sollte bei dieser Berechnung ein anderer Server ermittelt werden als der Avatar gerade zugewiesen ist, so wird dieser auf den neuen Server migriert [4]. Eine Besonderheit an beiden Komponenten ist, dass es möglich ist, die Art der Verwaltung der Avatare zu konfigurieren. Eine Konfigurationsmöglichkeit ist z.B. das Heat Map Verfahren. Dies ermöglicht ein schnelles Austauschen der Verfahren, wenn z.B. mehrere miteinander verglichen werden sollen [2]. Im Hintergrund nutzt der Client Assignment Manager das QuP-System [6] um die genaue Position einzelner Avatare zu erhalten. Hierbei handelt es sich um das Persistenzsystem von Timadorus, welches den Zustand der simulierten Welt verwaltet.

### 4.2 Das Heat Map Verfahren

Das Heat Map Verfahren beschreibt den Algorithmus mit dem die Avatare über die Spielserver verteilt werden. Ziel ist es, die Avatare nach ihrer räumlichen Position zu gruppieren. Dabei soll jedoch auch die Auslastung der Spielserver berücksichtigt werden. Sollte ein Server eine zu starke Auslastung aufweisen, so soll es deshalb die Möglichkeit geben, dass die Gruppierung ignoriert wird und ein alternativer Server

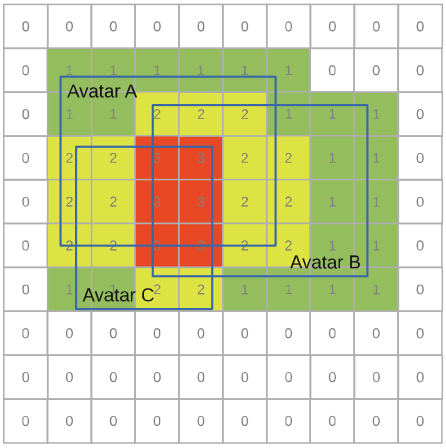


Abb. 2: Heat Map für einen Server mit AoIs der Avatare [4]

ausgewählt wird. Dies spiegelt sich auch in der allgemeinen Kostenfunkton wider:

$$cost(a, s) = w0 * -g(a, s) + w1 * cpu - w2 * memory \tag{1}$$

Die Funktion beschreibt die Kosten, wenn ein Avatar $a$ dem Server $s$ zugewiesen wird. $g$ beschreibt den Gruppierungsfaktor und somit die Qualität der Gruppierung, wenn ein Avatar dem Server zugeordnet wird. $cpu$ beschreibt die Auslastung des Prozessors und $memory$ entspricht dem noch zur Verfügung stehenden Speicher. Mithilfe von $w0$, $w1$ und $w2$ können diese Faktoren gewichtet werden [4].

### Die Heat Map

Die Heat Map ist die Datenstruktur in der die Verteilung der Avatare abgelegt wird. Hierzu wird die Spielwelt in ein dreidimensionales Raster eingeteilt. Die Präzision $P$ definiert wie viele Zellen pro Achse existieren. Die Heat Map zeigt für jeden Server an wie viele AoIs von Avataren sich in einer Zelle der Spielwelt befinden. Je mehr AoIs von verschiedenen Avataren sich in einem Zelle befinden desto Wärmer ist diese. In Abb. 2 ist ein Beispiel für einen Server zu sehen (zur vereinfachten Darstellung zweidimensional). Der Wert einer Zelle zeigt dabei an, wie viele AoIs sich in ihr befinden [4].

### Gruppierungsberechnung

Zur Berechnung der Qualität der Gruppierung wird betrachtet, welche Zellen sich mit dem AoI eines Avatars überschneiden. Anschließend werden die Werte für jede Zelle mit der es eine Überschneidung gibt aufsummiert. Der ermittelte Wert entspricht dem Gruppierungsfaktor. Sollte es keine Überschneidung mit einem Feld geben, in dem sich ein AoI eines Avatars auf dem Server befindet, so wird die Manhattan Distanz zur nächsten Zelle berechnet auf der dies zutrifft. Hierbei entspricht $Distanz = -g$ [4].

Sollte einem Server kein Avatar zugewiesen sein, so wird zunächst berechnet, wie viele Felder jeder Server bei einer idealen Verteilung belegen sollte:

$$IdealFieldCount = g0\frac{P^3}{ServerCount} + g1\frac{UsedFields}{ServerCount} \tag{2}$$

Diese Funktion ist in zwei Ideale unterteilt. Zum einen, dass die komplette Spielwelt gleichmäßig verteilt ist und zum anderen, dass die tatsächlich genutzten Zellen gleichmäßig verteilt sind. Gewichten lassen sich diese beiden mittels $g0$ und $g1$. Dieses wird in Relation zu der durchschnittlichen Anzahl an belegten Felder gesetzt:

$$Distance = \frac{\sqrt[3]{IdealFieldCount} - \sqrt[3]{AvgFieldCount}}{2} \tag{3}$$

Aus dieser Berechnung kommt eine Distanz heraus, ab der auf einen nicht belegten Server gewechselt werden sollte. Auch hier gilt $Distanz = -g$ [4].

Für alle genanten Verfahren gilt, dass das Ergebnis anschließend mit $(\frac{1}{P})^3$ multipliziert wird, damit der berechnete Wert möglichst unabhängig von der Präzision ist [4].

## 5 Erste Ergebnisse

Zu Beginn der Arbeiten lag bereits ein erstes System vor. Dies hatte die initiale Zuweisung mithilfe des Heat Map Verfahrens implementiert. Dazu war auch der CA Monitor sowie der Load Balancer umgesetzt, sodass ein Informationsaustausch bereits stattfand [2]. In den folgenden Arbeiten sollte das System erweitert bzw. verfeinert werden. Im Folgenden sollen diese Entwicklungen näher betrachtet werden.

### 5.1 Reassignment

Im ersten Schritt sollten die Voraussetzungen geschaffen werden, um Avatare im laufenden Betrieb migrieren zu können. Dazu wurde eine Möglichkeit zur Neuberechnung der Serverzuweisung von Avataren und ein kompletter Prozess geschaffen, der den Migrationsprozess abdeckt. Zur Neuberechnung wird für jeden Avatar in der Umgebung ein Prozess gestartet, der periodisch einen Server für den Avater berechnet. Wird ein anderer Server ermittelt als der Avatar gerade zugewiesen ist, so wird der Migrationsprozess gestartet. Dazu werden zunächst sämtliche Daten des Avatars vom neu zugewiesenen Server geladen, damit der Wechsel möglichst flüssig stattfindet. Nachdem dies stattgefunden hat wird der aktuell zugewiesene Spielserver informiert, dass der Avatar wechseln muss. Dieser übermittelt nun den Auth-Token an den neuen Spielserver und teilt dem Client mit, dass dieser einem neuen Spielserver zugeordnet ist. Anschließend verbindet sich der Client mit dem neuen Spielserver (Abb. 3) [4].

Bei diesem Ansatz handelte es sich zunächst um einen Proof of Concept, der zeigen sollte, dass es mit dem System auch möglich ist Avatare zu migrieren und trotzdem

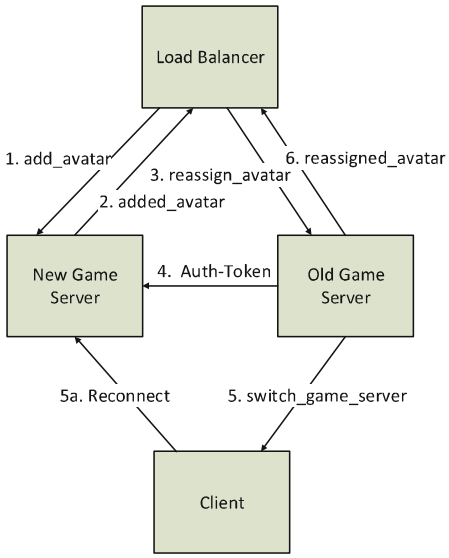


Abb. 3: Reassignment Prozess [4]

die Gruppierungseigenschaften zu erhalten. Entsprechend wurden stark vereinfachte Tests entwickelt. In diesen wurde für jeden Avatar ein eigener Prozess gestartet, in dem dieser in Richtung eines zuvor definierten Zieles bewegt wird. Damit konnte erfolgreich gezeigt werden, dass die Gruppierung auch über die Zeit erhalten bleibt.

### 5.2 Verfeinerung des Systems zur Neuberechnung

Nachdem eine erste Version zur Migration von Avataren zur Verfügung stand, wurde die Umsetzung verfeinert. Ziel war es eine flexiblere Umgebung zu haben, in der verschiedene Verfahren zur Neuberechnung verglichen werden können. Dazu wurden im CA Monitor sowie im Load Balancer jeweils neue Komponenten geschaffen, die das Verfahren abstrahieren. Die Komponente im Load Balancer ist über einen Konfigurationsparameter in der Lage verschiedene Verfahren zu starten. Eine dieser Verfahren ist z.B. die periodische Neuberechnung wie sie bereits vorliegt. Die Komponente innerhalb des CA Monitors hat die Aufgabe Informationen für das Neuberechnungsverfahren zu sammeln. Entsprechend wurde auch das Protokoll zur Zustandsübermittlung an den Load Balancer angepasst.

### 5.3 Entwickeln einer Testumgebung

Ein weiteres Ziel war es eine Testumgebung zu schaffen, die es ermöglicht eine realistischeres Verhalten der Avatare zu simulieren. Dazu wurde zunächst die Simulation der Avatare stärker abstrahiert. Entstanden ist ein allgemeiner Player-Simulator, der in den Spielserver-Mock eingebunden wird (Abb. 4). Zum Start eines Player-Simulator-Prozesses wird das gewählte Bewegungsmuster übergeben und der Prozess startet

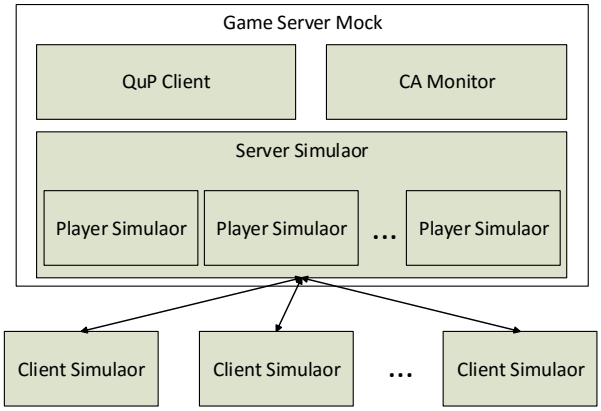


Abb. 4: Spielserver-Mock [3]

mit der entsprechenden Implementierung. Zur Simulation eines realistischeren Bewegungsmusters wurde außerdem ein weiteres Muster implementiert. Hierbei handelt es sich um eine Mischung aus dem Random Way Point Muster (RWP) [15] und Hot Points All (HPA) [29]. Wie in RWP wird jedem Avatare eine zufällige Menge an Zielen zugewiesen, welcher dieser ansteuern kann. Zusätzlich gibt es eine Menge von Punkten, die jedem Avatar zugewiesen werden. Diese sollen Hot Points simulieren, die von allen Avataren angesteuert werden. Jeder Avatar ermittelt zunächst aus der zugewiesenen Menge ein zufälliges Ziel und bewegt sich auf dieses zu. Zusätzlich wird eine zufällige Zeit definiert, für die sich ein Avatar in diese Richtung bewegt. Ist die Zeit abgelaufen oder erreicht der Avatar das Ziel vor Ablauf der Zeit, so wird ein neues zufälliges Ziel aus der Menge der zugewiesenen Ziele ausgewählt. [3]

Zudem wurde eine Möglichkeit geschaffen, um Avatare verzögert starten zu können. Dazu wird angegeben wie viele zu Beginn der Simulation bereits initialisiert sein sollen und in welchen Abständen die restlichen Avatare hinzugefügt werden sollen. Damit soll ein weiteres hinzustoßen neuer Spieler simuliert werden. Damit die Elemente der Spielwelt erzeugt werden können gibt es einen World Creator. Auch in diesem Fall kann die genutzte Implementierung mithilfe der Konfiguration gesteuert werden. So könnte z.B. ein alternatives Verteilungsmuster oder ein Verfahren, dass die Elemente aus einer Datei liest implementiert werden. Aktuell können die Avatare nach drei Mustern über die Spielwelt verteilt werden: Gleichmäßig, in Richtung eines Gebietes gedrängt und in mehreren Cluster [3]. Hierbei handelt es sich um Standardverteilungen, wie sie auch in einer Reihe von anderen Tests eingesetzt werden [21–23, 29, 30].

# 6 Zukünftige Arbeiten und Risiken

Eine Reihe von Erweiterungen konnte bereits erfolgreich Umgesetzt werden. Es gibt jedoch noch zahlreiche Punkte an denen in Zukunft weiter gearbeitet werden kann. Während der nächste Schritt, nämlich eine Verbesserung der Neuberechnung, fest eingeplant ist, gibt es für darauf folgende Arbeiten noch eine Reihe von Möglichkeiten. Im Folgenden wird deshalb aufgeführt, welche Arbeiten noch anstehen bzw. möglich sind.

## 6.1 Verbessertes Verfahren zur Neuberechnung

Als nächster Schritt soll die Neuberechnung zur Migration der Avatare verbessert werden. Ziel ist es eine intelligentere Auswahl an potentiell zu migrierenden Avataren zu haben und die Häufigkeit der Migration pro Avatar zu minimieren. Außerdem soll die aktuelle Eigenschaft, dass eine Gruppierung der Avatare weiterhin besteht, erhalten bleiben.

Aktuell findet die Neuberechnung für jeden Avatar periodisch statt. Dies sorgt für eine hohe Anzahl an unnötigen Berechnungen, da auch Avatare berechnet werden bei denen ein Serverwechsel voraussichtlich wenig Sinn macht. Zudem wird nicht darauf geachtet, die Anzahl an Migrationsprozesse zu minimieren. So kann es bei vereinzelten Avataren alle 2,5 Minuten zu einer Migration kommen [3]. Damit die Berechnung für eine kleinere Anzahl an Avataren stattfindet, soll eine Neuberechnung primär für diejenigen stattfinden, die sich am Rand eines Spielservers befinden. Dieses Konzept ist schon aus vorherigen Arbeiten bekannt [21, 22]. In beiden Arbeiten werden die sich am Rand befindlichen Avatare über einen Graph ermittelt. Da jedoch für jede Suche der komplette Graph erneut aufgebaut werden muss, dauert die Suche bei großen Mengen relativ lange. Deshalb wurde das Layer und Communication Refinement Partitioning um ein Parallel Partitioning erweitert. Dennoch werden für eine Neuberechnung bei 2500 Avataren 1,5 Sekunden benötigt [21]. Der Ameisenalgorithmus benötigt bei 2500 Avataren sogar zwischen ca. 5,5 und 23,2 Sekunden [22].

Auch wenn es im Heat Map Verfahren keine klassischen Ränder gibt, weil dieselbe Region von mehreren Servern verwaltet werden kann, so macht es Sinn Avatare zur Neuberechnung zu bevorzugen, die viele andere Avatare in ihrer näheren Umgebung haben, die von anderen Servern verwaltet werden. Dafür gibt es bereits zwei Ansätze. Bei einem würde analysiert werden, wie viele der Avatare, die sich innerhalb des AoI eines Avatars befinden, von einem anderen Server verwaltet werden. Je größer diese Zahl ist, desto größer ist die Wahrscheinlichkeit, dass der Avatar den Server wechseln muss. Um dies zu ermitteln, würde allerdings eine starke Abhängigkeit zum QuP bestehen, da auf Funktionalitäten aus diesem zurückgegriffen werden müsste. Eine davon ist jedoch noch nicht implementiert, sodass ein sofortiges Entwickeln nicht möglich wäre. Zudem müsste geschaut werden wie Avatare behandelt werden, die keinen anderen Avatar innerhalb ihrer AoIs haben. Beim zweiten Ansatz würde geschaut werden, welche Zellen der Heat Map von mehreren Servern verwaltet werden. Avatare dessen AoIs sich innerhalb dieser Zellen befinden sind gute Kandidaten für eine Neuberechnung.

Zudem könnte nach Zellen gesucht werden, die sich isoliert von anderen Zellen eines Servers befinden und von Zellen anderer Server umgeben sind. Dadurch würde eine stärkere Unabhängigkeit vom QuP entstehen.

Zum Testen soll das Verfahren mit anderen Verfahren, wie einer einfachen Partitionierung verglichen werden. Geprüft werden soll dabei zunächst, ob die Gruppierung der Avatare erhalten bleibt. Außerdem soll verglichen werden, inwiefern sich das neue Verfahren auf das Migrationsverhalten der Avatare auswirkt. Zudem soll überprüft werden, ob sich das Verfahren positiv auf die Leistungsfähigkeit der Spielserver auswirkt.

## 6.2 Autoscaling

Sobald die Neuberechnung verbessert wurde gibt es eine Reihe von Folgemöglichkeiten, an denen weitergearbeitet werden kann. Ein Aspekt der behandelt werden könnte, ist das Autoscaling der virtuellen Umgebung. Hierbei geht es darum neue Serverinstanzen zu starten, wenn die Last des Gesamtsystems sehr hoch ist und Serverinstanzen zu entfernen, wenn die Belastung gering ist.

Es gibt bereits Arbeiten, die sich mit der Thematik Skalierung von DVE-Systemen auseinandersetzen. In der Regel geht es dort auch um das Auslagern von Resourcen in die Cloud [7, 24]. Ein Faktor der beim Auslagern von Resourcen auch eine Rolle spielt sind die Kosten, die damit verbunden sein können. Deshalb wird in [7] ein Kostenfaktor integriert, durch den Kosten und Nutzen abgewogen werden können. Ein weiterer Aspekt kann eine möglichst starke Auslastung einzelner Resourcen sein, um Energie und damit auch Kosten, aufgrund der weniger benötigten Resourcen, zu sparen [14, 19].

Am Beispiel von RuneEscape konnte bereits gezeigt werden, dass sich neuer Content auf die Lastbedingungen auswirken können. Dies kann zu weniger sowie zu mehr Teilnehmern und somit auch Last führen [24]. Um darauf angemessen reagieren zu können sind Skalierungsverfahren nötig. Damit dies zuverlässig geschehen kann werden Vorhersagen benötigt [24]. Dazu muss das Spielerverhalten analysiert werden. Es gibt zwar schon eine Reihe von Untersuchungen [19, 27, 30], es müssen jedoch auch entsprechende Daten in der laufenden virtuellen Umgebung gesammelt werden, um Vorhersagen über zukünftige Lastsituationen treffen zu können. Eine interessante Fragestellung, die zudem betrachtet werden kann, ist wie beim Up- und Downscaling eine gute Grupierung erhalten bleiben kann. Gerade beim Downscaling sollen die Avatare, die sich auf einem Server befinden, auf andere Server umverteilt werden. Hat der Algorithmus gut gearbeitet, so befinden sich auf dem Server vor allem Avatare, die sich räumlich nah beieinander befinden. Werden diese einfach auf andere Server verteilt, so wird die Gruppierung zerrissen. Werden alle Avatare nur auf einen Server getan, so kann dieser jedoch auch stark überlastet werden.

Um Vorhersagen über die Belastung treffen zu können, müssen Data Mining Methoden verwendet werden. Hierbei handelt es sich um ein weiteres Gebiet, das erlernt werden muss. Dies birgt das Risiko, dass der Zeitaufwand groß werden kann.

### 6.3 Verbesserung der Heat Map

Ein weitere Möglichkeit, die betrachtet werden kann, ist eine weitere Optimierung der Heat Map. Hierbei gibt es zwei Aspekte die verbessert werden können. Zum einen eine effizientere Suche innerhalb der Heat Map, wenn es für einen Avatar keine Überschneidungen mit anderen Servern gibt und zum anderen, dass sich das Raster bei Bedarf automatisiert verfeinert. Ersteres behandelt die Problemstellung die entsteht, wenn das Raster sehr groß ist, ein Avatar jedoch mit keiner Zelle eines Servers Überschneidungen hat. In einem solchen Fall wird die Manhattan Distanz zur nächsten Zelle eines Servers berechnet. Dies erfolgt aktuell über eine Breitensuche, bei der immer weiter eine Zelle nach außen gegangen wird. Dies wird jedoch bei sehr großen Rastern äußerst ineffizient. Aktuell gibt es nur eine einfache Optimierung. Bei dieser werden nach jedem Schritt die hypothetischen Kosten berechnet. Überschreiten diese die aktuell niedrigsten Kosten eines anderen Servers, so wird die Suche abgebrochen. Beim zweiten Aspekt geht es darum, dass das Raster der Heat Map automatisiert verfeinert wird, wenn es innerhalb einer Zelle stärkere Lastsituaionen gibt. Gedacht ist dieses Schema für grobe Raster. Angenommen es gibt z.B. ein Universum mit Planeten, so kann es sinnvoll sein, zwischen den Planeten ein eher grobes Raster zu haben, weil sich dort wenige Avatare befinden, während es im Bereich der Planeten sinnvoller sein könnte, ein engeres Raster zu nutzen, weil sich dort mehr Avatare befinden. Mit einem dynamischen Raster könnte auch auf Gegebenheiten auf dem Planeten reagiert werden. Bei einem dynamischen Raster müsste wie in Abschnitt 6.2, das Verhalten der Avatare ausgewertet werden um eine zuverlässige Rasterung zu erreichen.

Sollte ein dynamisches Raster angepeilt werden, so entstehen ähnliche Risiken wie in Abschnitt 6.2, da auch hier Data Mining Methoden eingesetzt werden müssten. Sollte dies der Fall sein, so müsste zunächst eine einfachere Variante betrachtet werden.

## 7 Fazit und Ausblick

Diese Arbeit hat das Heat Map Verfahren vorgestellt und aufgezeigt, in welchen Bereichen bereits an diesem weitergearbeitet wurde. So wurde dargestellt, dass ein dynamischer Migrationsprozess und eine Testumgebung samt Avatarsimulation entstanden sind. Zudem wurde auf zukünftige Entwicklungsmöglichkeiten und deren Risiken eingegangen.

Auch außerhalb der aufgezeigten Weiterentwicklungen gibt es Möglichkeiten, an denen in Zukunft noch gearbeitet werden kann. So sind der Load Balancer und die Heat Map aktuell Bottlenecks sowie Single Points of Failure, da sie nicht verteilbar sind. Auch das dynamische starten einzelner Dungeons auf einer dynamisch ermittelten Maschine, wäre eine weitere Entwicklungsmöglichkeit, an der gearbeitet werden kann. In einigen Arbeiten wird auch der Kommunikationsaufwand zwischen Clients und Servern betrachtet und versucht diese zu optimieren [31,32]. Dies ist auch ein interessanter Aspekt für Timadorus. Insbesondere wenn von einer global verteilten virtuellen Umgebung ausgegangen wird.

## Literatur


1. Ali, A. A. F. ; Ismail, A. A. S. A. ; Bade, A. : An Overview of Networking Infrastructure for Massively Multiplayer Online Games. In: *Comp.Utm.My* (2009), Nr. October 2008, 619–628. http://comp.utm.my/pars/files/2013/04/An-Overview-of-Networking-Infrastructure-for-Massively-Multiplayer-Online-Games.pdf
2. Allers, S. : *Lastverteilung in einer clusterbasierten verteilten virtuellen Umgebung*, HAW Hamburg, Bachelor of Science, 2015
3. Allers, S. : Testumgebung für die Avatarbasierte Lastverteilung in verteilten virtuellen Systemen. (2017), S. 1–18
4. Behnke, L. ; Allers, S. ; Wang, Q. ; Grecos, C. ; Luck, K. von: Avatar Density Based Client Assignment. In: *Entertainment Computing â€“ ICEC 2016* Bd. 1, 2016. – ISBN 9783319461007, S. 137–148
5. Behnke, L. ; Grecos, C. ; Luck, K. von: QuP : Graceful Degradation in State Propagation for DVEs. (2014)
6. Behnke, L. ; Wang, Q. ; Grecos, C. ; Luck, K. von: Budget based dynamic state update aggregation. In: *Proceedings of the 7th ACM International Workshop on Massively Multiuser Virtual Environments - MMVE '15* (2015), 25–26. http://dx.doi.org/10.1145/2723695.2723696. – DOI 10.1145/2723695.2723696. ISBN 9781450333542
7. Carlini, E. ; Ricci, L. ; Coppola, M. : Flexible load distribution for hybrid distributed virtual environments. (2013). http://dx.doi.org/10.1016/j.future.2012.09.004. – DOI 10.1016/j.future.2012.09.004
8. Chen, J. ; Wu, B. ; Delap, M. ; Knutsson, B. ; Lu, H. ; Amza, C. : Locality aware dynamic load management for massively multiplayer games. In: *Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming - PPoPP '05* (2005), 289. http://dx.doi.org/10.1145/1065944.1065982. – DOI 10.1145/1065944.1065982. ISBN 1595930809
9. De Vleeschauwer, B. ; Van Den Bossche, B. ; Verdickt, T. ; De Turck, F. ; Dhoedt, B. ; Demeester, P. : Dynamic microcell assignment for massively multiplayer online gaming. In: *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games - NetGames '05* (2005), 1–7. http://dx.doi.org/10.1145/1103599.1103611. – DOI 10.1145/1103599.1103611. ISBN 1595931562
10. Deng, Y. ; Lau, R. W. H.: Heat diffusion based dynamic load balancing for distributed virtual environments. In: *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology* 1 (2010), Nr. 212, 203–210. http://dx.doi.org/http://doi.acm.org/10.1145/1889863.1889910. – DOI http://doi.acm.org/10.1145/1889863.1889910. ISBN 978–1–4503–0441–2
11. Deng, Y. ; Lau, R. W. H.: Dynamic load balancing in distributed virtual environments using heat diffusion. In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 10 (2014), feb, Nr. 2, 1–19. http://dx.doi.org/10.1145/2499906. – DOI 10.1145/2499906. – ISSN 15516857
12. Elfizar ; Baba, M. S. ; Herawan, T. : Object-Based Viewpoint For Large-Scale Distributed Virtual Environment. 28 (2015), Nr. 4, S. 301–317
13. Falkenauer, E. : A New Representation and Operators for Genetic Algorithms Applied to Grouping Problems. In: *Evolutionary Computation* 2 (1994), jun, Nr. 2, 123–144. http://dx.doi.org/10.1162/evco.1994.2.2.123. – DOI 10.1162/evco.1994.2.2.123. – ISSN 1063–6560
14. Hindman, B. ; Konwinski, A. ; Zaharia, M. ; Ghodsi, A. ; Joseph, A. D. ; Katz, R. H. ; Shenker, S. ; Stoica, I. : Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center. In: *NSDI* 11 (2011)

15. Johnson, D. B. ; Maltz, D. A.: Dynamic Source Routing in Ad Hoc Wireless Networks. In: *Mobile Computing* 353 (1996), 153–181. http://dx.doi.org/10.1007/b102605. – DOI 10.1007/b102605. ISBN 978–0–7923–9697–0
16. Kazem, I. ; Ahmed, D. T. ; Shirmohammadi, S. : A visibility-driven approach to managing interest in distributed simulations with dynamic load balancing. In: *Proceedings - IEEE International Symposium on Distributed Simulation and Real-Time Applications, DS-RT* (2007), S. 31–38. http://dx.doi.org/10.1109/DS-RT.2007.11. – DOI 10.1109/DS–RT.2007.11. – ISBN 0769530117
17. Kushner, D. : Engineering everquest. In: *IEEE SPECTRUM* 42 (2005), Nr. 7, S. 28
18. Lau, R. W.: Hybrid load balancing for online games. In: *Proceedings of the international conference on Multimedia - MM '10* (2010), 1231. http://dx.doi.org/10.1145/1873951.1874194. – DOI 10.1145/1873951.1874194. ISBN 9781605589336
19. Lee, Y. T. ; Chen, K. T.: Is server consolidation beneficial to MMORPG? A case study of world of warcraft. In: *Proceedings - 2010 IEEE 3rd International Conference on Cloud Computing, CLOUD 2010* (2010), S. 435–442. http://dx.doi.org/10.1109/CLOUD.2010.57. – DOI 10.1109/CLOUD.2010.57. ISBN 9780769541303
20. Lu, F. ; Parkin, S. ; Morgan, G. : Load balancing for massively multiplayer online games. In: *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games - NetGames '06* (2006), 1. http://dx.doi.org/10.1145/1230040.1230064. – DOI 10.1145/1230040.1230064. ISBN 1595935894
21. Lui, J. C. S. ; Chan, M. F.: An efficient partitioning algorithm for distributed virtual environment systems. In: *IEEE Transactions on Parallel and Distributed Systems* 13 (2002), Nr. 3, 193–211. http://dx.doi.org/10.1109/71.993202. – DOI 10.1109/71.993202. – ISSN 10459219
22. Morillo, P. ; Fernández, M. ; Orduña, J. : An ACS-based partitioning method for distributed virtual environment systems. In: *Proceedings International Parallel and Distributed Processing Symposium* 00 (2003), Nr. C. http://dx.doi.org/10.1109/IPDPS.2003.1213283. – DOI 10.1109/IPDPS.2003.1213283. – ISBN 0–7695–1926–1
23. Morillo, P. ; Orduña, J. M. ; Fernández, M. ; Duato, J. : An Adaptive Load Balancing Technique For Distributed Virtual Environment Systems. In: *Proceedings of the 15th IASTED international PDCS-03* (2003), S. 256–261
24. Nae, V. ; Iosup, A. ; Podlipnig, S. ; Prodan, R. ; Epema, D. ; Fahringer, T. : Efficient management of data center resources for Massively Multiplayer Online Games. In: *International Conference for High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008.* (2008), Nr. November, S. 1–12. http://dx.doi.org/10.1109/SC.2008.5213940. – DOI 10.1109/SC.2008.5213940. ISBN 978–1–4244–2834–2
25. Neumann, C. ; Prigent, N. ; Varvello, M. ; Suh, K. : Challenges in peer-to-peer gaming. In: *ACM SIGCOMM Computer Communication Review* 37 (2007), Nr. 1, S. 79. http://dx.doi.org/10.1145/1198255.1198269. – DOI 10.1145/1198255.1198269. – ISBN 0146–4833
26. Ng, B. ; Si, A. ; Lau, R. W. ; Li, F. W.: A multi-server architecture for distributed virtual walkthrough. In: *Proceedings of the ACM symposium on Virtual reality software and technology - VRST '02* (2002), 163. http://dx.doi.org/10.1145/585767.585768. – DOI 10.1145/585767.585768. ISBN 1581135300
27. Pittman, D. ; GauthierDickey, C. : A measurement study of virtual populations in massively multiplayer online games. In: *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games - NetGames '07* (2007), 25–30. http://dx.doi.org/10.1145/1326257.1326262. – DOI 10.1145/1326257.1326262. ISBN 9780980446005
28. Rosedale, P. ; Ondrejka, C. : *Enabling Player-Created Online Worlds with Grid Computing and Streaming*. http://www.gamasutra.com/resource{_}guide/20030916/rosedale{_}pfv.htm. Version: 2003, Abruf: 2015-08-10

29. Rueda, S. ; Morillo, P. ; Orduña, J. M. ; Valencia, U. D.: A Peer-To-Peer Platform for Simulating Distributed Virtual Environments. (2007). ISBN 9781424418909
30. Suznjevic, M. ; Matijasevic, M. : Player behavior and traffic characterization for MMORPGs : a survey. (2013), S. 199–220. http://dx.doi.org/10.1007/s00530-012-0270-4. – DOI 10.1007/s00530–012–0270–4
31. Ta, D. N. B. ; Zhou, S. : A network-centric approach to enhancing the interactivity of large-scale distributed virtual environments. In: *Computer Communications* 29 (2006), Nr. 17, S. 3553–3566. http://dx.doi.org/10.1016/j.comcom.2006.05.015. – DOI 10.1016/j.comcom.2006.05.015. – ISSN 01403664
32. Ta, D. N. B. ; Zhou, S. ; Cai, W. ; Tang, X. ; Ayani, R. : Efficient zone mapping algorithms for distributed virtual environments. In: *Proceedings - Workshop on Principles of Advanced and Distributed Simulation, PADS* (2009), S. 137–144. http://dx.doi.org/10.1109/PADS.2009.10. – DOI 10.1109/PADS.2009.10. ISBN 9780769537139
33. Waters, R. C. ; Barrus, J. W.: The rise of shared virtual environments. In: *Spectrum, IEEE* 34 (1997), Nr. 3, S. 20–25. http://dx.doi.org/10.1109/6.576004. – DOI 10.1109/6.576004
34. X, C. P.: *Building a Balanced Universe - EVE Community*. http://community.eveonline.com/news/dev-blogs/building-a-balanced-universe/. Version: 2013, Abruf: 2017-01-15
35. Zhang, W. ; Zhou, H. : A Dynamic Mapping Method to Keep Region Connectedness in DVE Systems. (2016), S. 0–3