Replikationsbasierte Skalierbare Parallelisierung Virtueller Umgebungen

Jens Müller-Iden

Institut für Informatik
Westfälische Wilhelms-Universität Münster
jmueller@uni-muenster.de

Abstract: Interaktive virtuelle Umgebungen sind in den letzten Jahren zu einer der wichtigsten Klassen über das Internet betriebener Anwendungen geworden, wie z. B. Telekonferenz- und Telepräsensanwendungen, virtuelle e-learning Klassenräume oder Online Computerspiele. Viele dieser Anwendungen erfordern eine gute Skalierbarkeit, um hohe Teilnehmerzahlen zu erreichen. Insbesondere *Massively Multiplayer Online Games (MMOG)* haben sich hierbei als Technologietreiber erwiesen und neue skalierbare Architekturen hervorgebracht. Der Stand der Forschung ermöglicht es allerdings bei weitem nicht, alle sinnvoll denkbaren Dimensionen interaktiver virtueller Umgebungen zu skalieren: Insbesondere die Dichte der dynamischen Objekte, so genannter *entities*, ist nicht skalierbar, was einer ganzen Reihe von wichtigen Anwendungen, u. a. schnellen Actionspielen, den Übergang zu einem massiv-mehrspielerfähigem Betrieb verwehrt. Dieser Artikel analysiert den Stand der Forschung und stellt einen neuartigen, replikationsbasierten Parallelisierungsansatz virtueller Umgebungen vor, der erstmalig die Skalierung der Entitätsdichte ermöglicht. Dabei wird vorgestellt, wie Konsistenz des replizierten Anwendungszustandes durch geeignete Synchronisierung sichergestellt wird und wie mittels einer Proxy-Server-Architektur die Antwortzeit für Nutzereingaben (Responsivität) optimiert werden konnte. Der Artikel stellt zudem praktische Evaluationsergebnisse mit Demonstrationsanwendungen vor, bei denen die replikationsbasierte Parallelisierung die erreichbare Entitätsdichte drastisch erhöht.

1 Einführung

In interaktiven virtuellen Umgebungen, so genannten *Distributed Virtual Environments (DVE)*, nehmen mehrere Benutzer über Client-Anwendungen an einem verteiltem System teil. Dieses System berechnet mit hoher Frequenz aus den Nutzereingaben neue Anwendungszustände, die in schneller Folge visualisiert den Eindruck einer flüssig und kontinuierlich fortschreitenden interaktiven virtuelle Welt vermitteln. Dabei kommt zumeist eine Architektur mit einem zentralem Server zum Einsatz, die zwar vergleichsweise einfach zu realisieren ist, jedoch nur ein limitierte Menge an Ressourcen bereitstellt. Zur Vergrößerung bestimmter Dimensionen verteilter Umgebungen, etwa der Größe der virtuellen Welt, der Anzahl oder der Dichte der Teilnehmer, ist eine skalierbare Architektur mit mehreren Servern, CPUs oder Cores zur Einhaltung der Echtzeitbedingungen dieser Anwendungen bei wachsendem Ressourcenbedarf nötig. Immer populärer werdende *Massively Multiplayer Online Games* [Woo] wie *World of Warcraft* oder *Everquest* nut-

zen Architekturen mit einer Vielzahl von Servern, die einzelne virtuelle Welten mittels *Zoning* [CXTL02] oder *Instancing* parallelisiert betreiben. Auch Lebenssimulationen wie *Second Life*, in denen Nutzer durch Avatare virtuell repräsentiert miteinander interagieren, nutzen diese bekannten Parallelisierungsansätze [RO03].

Bekannte Ansätze zur Parallelisierung virtueller Umgebungen ermöglichen nur die Skalierung bestimmter Aspekte. Insbesondere die Dichte der interaktiven Objekte (sog. Entitäten) als deren Anzahl bezogen auf eine vergleichsweise kleine virtuelle (Teil-)Welt fester Größe ist von bekannten Mechanismen nicht skalierbar. Aus diesem Grund sind wichtige Anwendungstypen mit kleiner virtueller Welt, wie z. B. populäre Spielgenres wie *First Person Shooter (FPS)* oder *Real-Time Strategy (RTS)* in der Teilnehmerzahl beschränkt. Um die Teilnehmerzahl in diesen Anwendungen von momentan wenigen Dutzend Nutzern auf mehrere hundert oder tausend Nutzer zu erhöhen, ist ein neuartiger Parallelisierungsansatz nötig, der die zur Verfügung stehenden Serverressourcen unabhängig von der Größe und Ausgestaltung der virtuellen Welt zu erhöhen vermag.

Dieser Artikel gibt einen Überblick über einen neuen replikationsbasierten Parallelisierungsansatz zur Skalierung der Dichte von interaktiven Entitäten in virtuellen Welten. Die Notwendigkeit hierzu und das grundlegende Berechnungsmodell virtueller Umgebungen wird in Abschnitt 2 motiviert. Den neuen Parallelisierungsansatz habe ich als wissenschaftlicher Mitarbeiter an der Universität Münster in der Arbeitsgruppe Parallele und Verteilte Systeme geleitet von Herrn Prof. Gorlatch entwickelt und in meiner Dissertation *Replication-based Scalable Parallelization of Virtual Environments* [MI07] beschrieben. Die Herausforderungen in der Entwicklung dieses Ansatzes lag zum Einen darin, überhaupt eine geeignete Parallelisierung zur serverseitigen Skalierung der Entitätsdichte zu entwickeln, da ein Übergang der bisherigen „embarrassingly parallel“-Ansätzen zu einer hochfrequent synchronisierten Parallelisierung nötig war. Zum Anderen erforderte der dann gewählte, in Abschnitt 3 beschriebene Ansatz die Implementierung einer effizienten Serverarchitektur und die Entwicklung hochperformanter Mechanismen zur Konsistenzhaltung, Responsivitätsoptimierung, Quantifizierung der Skalierbarkeit und Wahrung korrekter Ausführungsreihenfolgen von Nutzereingaben. Abschnitt 4 gibt einen Überblick über die zu Evaluationszwecken implementierten Demoanwendungen und stellt sowohl experimentelle als auch analytische Ergebnisse vor. Abschnitt 5 fasst den Artikel zusammen und reflektiert weitere Einsatzmöglichkeiten des Parallelisierungsansatzes.

2 Skalierung Interaktiver Umgebungen

Interaktive virtuelle Umgebungen erwecken durch eine hohe Berechnungsfrequenz (5-100 Hz je nach Anwendung) neuer Anwendungszustände auf Basis der Nutzereingaben den Eindruck eines kontinuierlichen Geschehens, vergleichbar mit einem interaktiv unmittelbar berechnetem Film. Beispiele sind kontinuierliche Anwendungen, die in einer virtuellen Welt stattfinden, also Telekonferenzanwendungen, E-Learning Klassenräume, Online Games oder kollaborative Konstruktionsumgebungen im CAE-Bereich. Abb. 1 zeigt Screenshots einiger solcher interaktiven Umgebungen.

(a) Second Life (b) World of Warcraft

Abbildung 1: Screenshots verschiedener interaktiver virtueller Umgebungen

Im Betrieb der virtuellen Umgebung werden Nutzereingaben durch die Clients an den Server übermittelt, dort ein neuer Anwendungszustand berechnet und dieser an die Clients zur Visualisierung zurückübertragen. Für Nutzer ergibt sich somit eine unmittelbar Rückkopplung (Feedback-Loop) von Eingaben und Auswirkungen: Drückt ein Nutzer eine Bewegungstaste, sollte sich der Avatar des Nutzers unmittelbar in Bewegung setzen und wieder stoppen, sobald die Taste losgelassen wird. Die Reaktionszeiten des Systems sollten maximal 500 ms (indirekt gesteuerte Strategiespiele [SGB+03]) bis 100 ms (direkt gesteuerte Anwendungen und Actionspiele [Arm01]) betragen. Im Ergebnis sind virtuelle Umgebungen als weiche Echtzeitsysteme [Kop97] anzusehen, bei denen verspätete Neuzustände (*lateness*) unmittelbar von Nutzern als unangenehme Unterbrechung des Anwendungsflusses (sog. *lag*) empfunden werden, der das „Eintauchen“ (*Immersion*) in das Geschehen unterbricht. Abb. 2 illustriert die kontinuierliche Aktualisierung der virtuellen Welt.

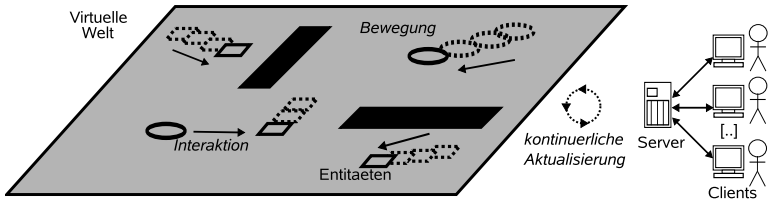


Abbildung 2: Kontinuierliche Verteilte Zustandsaktualisierung in Virtuellen Umgebungen

Grundsätzlich gilt, dass die benötigte Rechenzeit zur Durchführung einer Aktualisierung von der Anzahl der Nutzereingaben und Entitäten in der virtuellen Welt abhängt. Bei einer nicht skalierbaren Architektur, etwa einem einzelnen Server, besteht somit ein Limit der Anzahl der Teilnehmer und Entitäten, für die Zustandsaktualisierungen noch gemäß der vorgegebenen Aktualisierungsrate durchgeführt werden können. Es ist deshalb eine skalierbare Architektur und eine Parallelisierung der Zustandsaktualisierung erforderlich, um bestimmte, im nachfolgenden Abschnitt diskutierte Charakteristiken und *Dimensionen* der virtuellen Umgebung beliebig erhöhen zu können.

2.1 Dimensionen und Konzepte der Skalierung

Dieser Abschnitt stellt auszugsweise eine Analyse möglicherSkalierungsdimensionen und bekannter Parallelisierungsansätze aus meiner Dissertation vor. Die drei wichtigsten Dimensionen, insbesondere auch für Massively Multiplayer Online Games (MMOG), sind:

1.) Gesamtanzahl Nutzer einer Welt: In MMOG bezeichnet diese Dimension alle an einer gemeinsamen, meist riesigen virtuellen Welt (einem sog. *Shard* oder *Realm*) teilnehmenden Spieler. Sie kann durch bekannte Zoning- oder Instancing-Parallelisierung unter der Bedingung einer gleichmäßigen Verteilung der Nutzer gut skaliert werden. In kommerziellen Rollenspielen werden mehrere tausend Nutzer erreicht. Für Anwendungen mit kleiner virtueller Welt dagegen wie FPS- und RTS-Spielen besteht das übliche Maximum mangels einer geeigneten Parallelisierung bei etwa 100 Nutzern; diesen Anwendungen ist der Übergang zu einem massiv-mehrspielerfähigem Konzept bisher verwehrt.

2.) Größe der Virtuellen Welt: Skalierbarkeit in dieser Dimension ist üblicherweise von MMOG oder Virtual Life-Simulationen wie Second Life gefordert, in denen Nutzern eine riesige zusammenhängende virtuelle Welt bereisen können sollen. Zur Skalierung ist eine unlimitierte Erhöhung des serverseitigen Hauptspeichers, Festspeichers und der zur Verfügung stehenden Rechenleistung zur Verwaltung und „Belebung“ der virtuellen Welt mit simulierten NPC (Non Player Character) erforderlich.

3.) Dichte der dynamischen Entitäten: Die Dichte der Entitäten ergibt sich als deren Anzahl bezogen auf ein vergleichsweise kleines virtuelles Areal. Dieses Areal kann ein kleiner Teil einer großen virtuellen Welt sein oder eine komplette kleine virtuelle Welt. Diese Skalierbarkeit ist insbesondere in Anwendungen erforderlich, in denen Nutzer zur häufigen gegenseitigen Interaktion ermuntert werden, wie etwa in FPS und RTS Spielen mit- und gegeneinander zu kämpfen oder in Städten in Rollenspielen Handel zu treiben.

Die bekannte Zoning-Parallelisierung [CXTL02] ist geeignet, die ersten beiden Dimensionen *Gesamtnutzerzahl* und *Weltgröße* zu skalieren. Die dritte Dimension *Entitätsdichte* ist durch diesen Ansatz jedoch nicht skalierbar, wie im Folgenden diskutiert wird.

2.2 Stand der Forschung: Zoning-Parallelisierung

Bei der Zoning-Parallelisierung wird die Welt einer virtuellen Umgebung in disjunkte Teilwelten, den sog. *Zonen*, aufgeteilt. Die kontinuierliche Zustandsaktualisierung jeder Zone wird unabhängig durch einen einzelnen Server durchgeführt. Dieser Ansatz stellt eine unabhängige Kollektion von Einzelserver-Architekturen dar, deren Gesamtheit die übergeordnete, große virtuelle Welt repräsentiert. Abb. 3 illustriert ein Beispiel einer in vier Zonen unterteilten virtuellen Welt.

Kommunikation zwischen den Servern im Zoning-Ansatz findet nur zur Steuerung der Client-Übergabe statt, wenn ein Benutzer seinen Avatar von einer Zone in die andere bewegt. Der Zoning-Ansatz skaliert die Weltgröße perfekt: Die Welt kann beliebig durch Hinzufügen weiterer unabhängiger Zonen vergrößert werden. Die Gesamtnutzerzahl wird

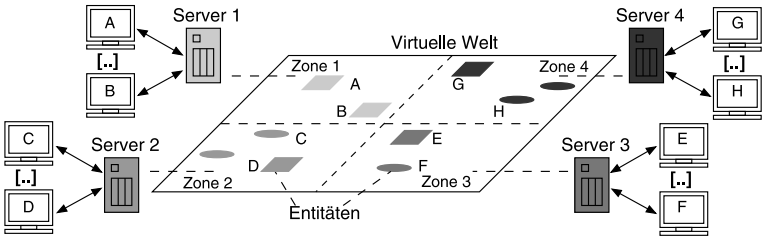


Abbildung 3: Zoning-Parallelisierung Virtueller Welten

ebenfalls gut skaliert, jedoch mit einer wichtigen Einschränkung: Die Avatare müssen gleichmäßig über die Zonen verteilt sein. Da deren Position aber durch die Benutzer selbst festgelegt wird, kann es durchaus sein, dass sich viele Avatare in einer Zone sammeln und dort die Entitätsdichte erhöhen bis zur Überlastung des einzeln zuständigen Servers erhöhen. Anwendungen müssen zur Überlastungsvermeidung Limitierungen einsetzen: So ist in Second Life z. B. ein festes Limit der Avatare pro Zone eingerichtet, nach dessen Erreichen keine weiteren Avatare zugelassen werden. Bei spezifischen points of interest oder virtuellen Veranstaltungen in einer bestimmten Zone ist das Limit schnell erreicht und nur ein Bruchteil der interessierten Benutzer können teilnehmen. Die Skalierung der Entitätsdichte, die durch Zoning nicht ermöglicht wird, ist ein wichtiges Erfordernis um Benutzer beliebig zusammenkommen und interagieren zu lassen.

2.3 Herausforderung: Skalierung der Entitätsdichte

Die Herausforderung in der Entwicklung eines Ansatzes zur Skalierung der Entitätsdichte lag zum Einen darin, überhaupt eine geeignete Parallelisierung zu entwickeln: Ein „embarrassingly parallel“-Ansatz mit unabhängiger Aufteilung der virtuellen Welt ist nicht geeignet, stattdessen ist eine feingranulare Parallelisierung der Zustandsaktualisierung erforderlich. Bei hoher Dichte ist die anwendungsspezifische Kopplung der Entitäten durch ihre Interaktionen untereinander jedoch derart stark, dass eine starke Synchronisierung innerhalb der parallelen Zustandsberechnung und somit ein Overhead unvermeidbar ist.

Die konkreten Herausforderungen an eine solche Parallelisierung sind somit:
Hohe Skalierbarkeit: Der Synchronisierungs-Overhead muss minimiert werden um einen guten speed-up zu erreichen.
Konsistenz: Der Anwendungszustand muss konsistent gemäß eines geeigneten Modells gehalten werden. Dazu muss ein konkretes Konsistenzmodell ausgewählt und angepasst, und die Synchronisierung entsprechend entwickelt werden.
Responsivität: Die Antwortzeit für Nutzereingaben muss so gering wie möglich sein, um direktes Feedback und hohe Nutzerimmersion zu erreichen.
Korrektheit: Die Ausführungsreihenfolge von Nutzereingaben sollte gleich sein ihrer realweltlichen Eingabereihenfolge, z. B. bei konkurrierenden (wenige ms Zeitabstand) Objektaufnahmeversuchen sollte der Nutzer das Objekt erhalten, der zuerst die Eingabe tätigte.

3 Neues Konzept: Replikationsbasierte Parallelisierung

Zur Skalierung der Entitätsdichte müssen die Berechnungen innerhalb einer einzelnen Zustandsaktualisierung parallelisiert werden. Sind beliebige Interaktionen zwischen den Entitäten möglich (wie i. A. der Fall), so benötigt jede an einer solchen Parallelisierung beteiligten Recheneinheit Zugriff auf den vollständigen Anwendungszustand. Aus diesem Grund ist der Zustand in der von mir entwickelten replikationsbasierten Parallelisierung auf jedem Server als Kopie oder *Replik* vorhanden, wie im Beispiel in Abb. 4 illustriert.

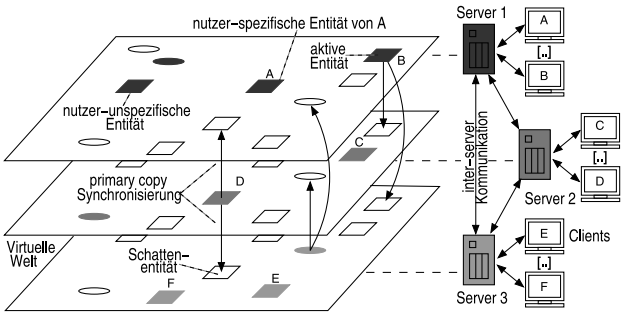


Abbildung 4: Replikations-basierte Parallelisierung Virtueller Welten

Durch die vollständige Replikation hat jeder Server Zugriff auf jede Entität, unabhängig von ihrer Position und der Ausgestaltung und Größe der virtuellen Welt. Zur Teilnahme kann sich ein Client somit zu einem einzelnen beliebigen Server verbinden.

Zur Erlangung von *Skalierbarkeit* und *Konsistenz* des Anwendungszustandes wurden Mechanismen für die (1) Berechnungsaufteilung der Entitäten und (2) die Synchronisierung der Repliken erarbeitet. Als Grundlage dienen dabei bekannte *eager* und *lazy* bzw. *update anywhere* und *primary copy*-Verfahren [WPS+00], deren spezifische Eignung zur Synchronisierung hochinteraktiver Anwendungen detailliert diskutiert werden. Im Ergebnis wird eine angepasste *lazy primary copy*-Synchronisierung verwendet, die *eventual consistency* garantiert. Die Hauptaspekte dieser Synchronisierung sind wie folgt:

- Jeder Server berechnet *nutzer-spezifischen Entitäten* (direkt von einem bestimmten Nutzer kontrolliert, z. B. der eigene Avatar) der direkt zu ihm verbundenen Clients.
- Die Berechnung *nutzer-unspezifischer Entitäten* (z. B. serverseitig kontrollierte Non Player Character, allgemeine Objekte) ist lastverteilend auf Server aufgeteilt.
- Von einem Server direkt berechnete Entitäten sind dort *aktive Repliken* (Primary Objects, Änderungen an alle Repliken kommunizieren) und auf allen anderen Servern *Schattenrepliken* (nur Lesezugriff)
- Der Nutzereingabetyp *Aktion* ändert nur den Zustand eigener nutzer-spezifischer Entitäten und wird direkt vom verbundenen Server ausgeführt, der die betroffenen Entitäten aktiv verwaltet und somit die notwendige Schreibmöglichkeit hat.
- Der Nutzereingabetyp *Interaktion* ändert möglicherweise den Zustand beliebiger Entitäten und wird somit an alle Server zur entfernten Ausführung weitergeleitet.

Dieses Synchronisierungsverfahren ermöglicht höchste Responsivität des Gesamtsystems. Insbesondere können Nutzereingaben vom Typ *Aktion*, die den Großteil der Eingaben darstellen (vor allem sind sämtliche Bewegungskommandos Aktionen) unmittelbar vom direkt verbundenen Server ausgeführt werden. Zur optimalen Ausnutzung dieser Eigenschaft habe ich eine nachfolgend vorgestellte Proxy-Server-Architektur zum Internet-Betrieb virtueller Umgebungen entwickelt. Die Skalierbarkeit der Berechnungsparallelisierung habe ich zudem analytisch untersucht und zu diesem Zweck das *Game Scalability Model* [MG05] entwickelt, welches eine feingranulare parametrisierte Quantifizierung der Berechnungen ermöglicht. Das GSM erlaubt einen analytischen Vergleich der Proxy-Server-Architektur mit herkömmlichen Client-Server- und Peer-to-Peer-Architekturen: Im Ergebnis kombiniert der neue Ansatz die Vorzüge dieser bekannten Architekturen und erweitert sie um die Möglichkeit der skalierbaren Berechnungsparallelisierung.

3.1 Verteilte Proxy-Server-Betriebsarchitektur

In der Proxy-Server-Architektur [MFGM04, MG04] zum Betrieb replikationsbasiert parallelisierter virtueller Umgebungen sind die Server einer einzelnen Partie weit im Internet verteilt, wie in Abb. 5 an einem Beispiel mit vier beteiligten Servern illustriert.

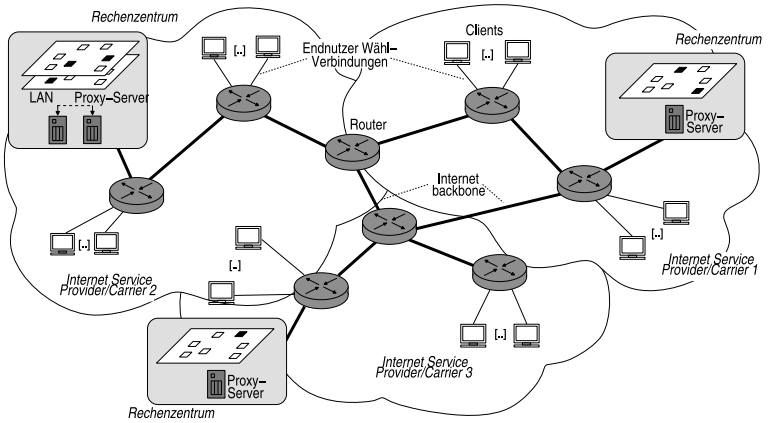


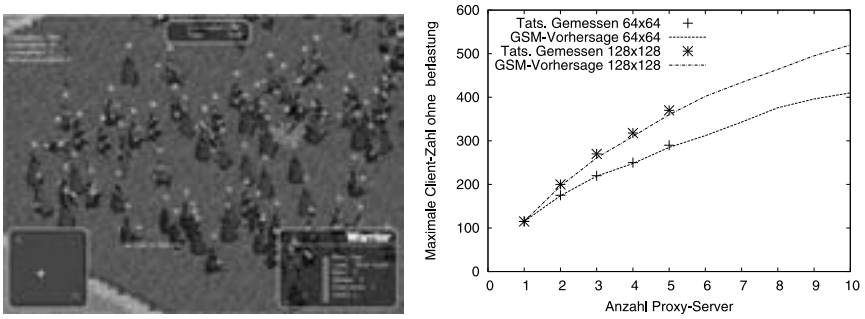
Abbildung 5: Verteilung der Replikationsserver als Proxies im Internet

In diesem verteilten Setup agieren die Server als anwendungsspezifische Proxies: Clients können den Server mit geringster Kommunikationslatenz und höchster Responsivität wählen. Diese Wahlfreiheit des Servers ist neuartig im Vergleich zur Zoning-Parallelisierung, in der zu einem fest bestimmten Zonenserver verbunden werden muss. Zudem werden mittels Zoning parallelisierte virtuelle Umgebungen üblicherweise in einem einzelnen lokalen Cluster und nicht weitverteilt über das Internet betrieben, so dass die Proxy-Server-Architektur einen deutlichen und neuartigen Zugewinn an Flexibilität des Betriebs und Responsivität der Verarbeitung von Nutzereingaben darstellt.

4 Demonstrationsanwendungen und Performance-Vorhersage

Zur praktischen Evaluierung des replikationsbasierten Parallelisierungsansatzes und der Proxy-Server-Architektur habe ich die C++-basierte Kommunikationsmiddleware *Game Proxy-Architecture (GPA)* entwickelt. GPA vereinfacht die Entwicklung verteilter virtueller Umgebungen und ermöglicht effiziente TCP- und UDP-basierte Client-Server- und Server-Server-Kommunikation (unicast und multicast) mit verschiedenen Auslieferungsgarantien über das Internet. GPA bildet die Basis für verschiedene in meiner Dissertation beschriebene Demonstrationsanwendungen [1], von denen im Folgenden das Spiel *Rokkatan* überblicksartig vorgestellt wird.

Das Spiel *Rokkatan* [MG06] wurde in Zusammenarbeit mit einer studentischen Projektgruppe [MPS05] auf Basis von GPA als Case Study eines massiv-mehrspielerf higen RTS-Spieles entwickelt. Abb. 6(a) zeigt einen Screenshot, der insbesondere die hohe mögliche Dichte der Avatare in großen, teambasierten Kämpfen um Flaggenpunkte illustriert.



(a) Spielsituation mit hoher Avatardichte (b) Vorhergesagte (GSM) und gemessene Spielerzahl

Abbildung 6: Rokkatan: Screenshot und experimentelle Ergebnisse

Das *Game Scalability Model* [MG05] ermöglicht die Vorhersage von benötigter Rechenzeit und Bandbreite einer konkreten Anwendung und somit die Vorhersage der maximalen Teilnehmerzahl (abhängig von einem zu definierenden „worst case“ der Entitätsdichte, der Weltgröße und der Zahl beteiligter Server). Abb. 6(b) zeigt für zwei Weltgrößen 64x64 und 128x128 Längeneinheiten, dass die GSM-Vorhersage für Rokkatan (Linien) die experimentell erreichten Teilnehmerzahlen (Messpunkte) sehr genau (max. 7 % Abweichung) vorhersagt. Insgesamt ist die erreichte Skalierbarkeit gut, mit 370 Clients auf der größeren Karte wurde ein speed-up von 3,2 mit fünf Servern erreicht (Setup schöpfte vorhandene 25 P IV 1,7 GHz Rechner des Testclusters zum Betrieb der Server und insbesondere der Clients voll aus), bezogen auf die maximal 115 möglichen Clients eines herkömmlichen Einzelserver-Setups. Das GSM sagt zudem die Erreichbarkeit von 500 Clients bei einer vertretbaren Zahl von 10 Servern voraus, wobei Rokkatan mit einer Aktualisierungsrate von 25 Hz vergleichbar fordernd wie kommerzielle Actionspiele (10 - 35 Hz) ist.

[1] Online API und Multimedia-Material unter http://pvs.uni-muenster.de/jmueller/thesis/index.html

5 Zusammenfassung und Verwandte Arbeiten

Skalierbarkeit der Entitätsdichte ist eine Grundvoraussetzung, um die Teilnehmerzahl in hochinteraktiven virtuellen Umgebungen zu erhöhen. Dieser Artikel stellt einen Überblick über die hierfür von mir entwickelte replikationsbasierte Parallelisierung dar und diskutiert neben der Proxy-Server-Betriebsarchitektur mit Rokkatan eine realistische Demonstrationsanwendung. In meiner Dissertation finden sich darüber hinaus weiterführende Arbeiten, die hier aus Platzgründen nur kurz erwähnt werden können: So wurde als weitere Demoanwendung das populäre Actionspiel *Quake 2* von einer Einzelserver-Version zu einer replikationsbasiert parallelisierten Version [MGSF07] portiert. Zudem habe ich *Korrektheit* als die Eigenschaft, die Reihenfolge von Benutzeraktionen in ihrer nebenläufigen Bearbeitung zu erhalten, analysiert und gezeigt, dass mit eigenen Erweiterungen bekannter Korrektheitsalgorithmen wie *local lag* oder *timewarp* [MVHE04] ein mit herkömmlichen Architekturen vergleichbares Maß an Korrektheit erreicht wird [MGG06].

Replikation wurde in früheren Arbeiten [Fun95] mit dem Ziel der Bandbreitenoptimierung eingesetzt. Der Einsatz zu Skalierungszwecken ist neu, das Forschungsinteresse wird jedoch zunehmend stärker. So repliziert das Colyseus-System [BPS06] Teile des Anwendungszustandes zur Skalierung der Gesamtnutzerzahl. Zur Skalierung der Entitätsdichte wurde eine alternative SMP-Parallelisierung von Quake 1 vorgestellt [AB04]. Insgesamt sind diese neuen Ansätze als Alternativen bzw. Ergänzung zu bestehenden Zoning- und Instancing-Parallelisierungen zu verstehen. Im europäischen *edutain@grid*-Projekt (IST 034601 Projekt „edutain@grid“, www.edutaingrid.eu) [FAA+07] wird aktuell die von mir entwickelte replikationsbasierte Parallelisierung zusammen mit anderen Ansätzen in eine generische Parallelisierungs- und Verteilungsmiddleware für E-Learning-, Trainings-, und Spielanwendungen integriert. Diese Middleware stellt in einem computational Grid dynamisch Ressourcen für diese interaktiven verteilte Anwendungen in Abhängigkeit von der aktuellen Nutzernachfrage bereit. Effektive Parallelisierungen, wie der von mir entwickelte Ansatz, bilden in solchen Systemen die wichtige Grundlage zum kosteneffizienten und skalierbaren Betrieb heutiger und zukünftiger interaktiver Anwendungen.

Literatur

[AB04] A. Abdelkhalek und A. Bilas. Parallelization and Performance of Interactive Multiplayer Game Servers. In *18th IPDPS*, Santa Fe, USA, April 2004. IEEE.

[Arm01] G. Armitage. Sensitivity of Quake 3 players to network latency, IMW2001.

[BPS06] A. Bharambe, J. Pang und S. Seshan. A Distributed Architecture for Multiplayer Games. In *PACM/ USENIX NSDI*, San Jose, USA, 2006.

[CXTL02] W. Cai, P. Xavier, S. J. Turner und B.-S. Lee. A Scalable Architecture for Supporting Interactive Games on the Internet. In *Proc. of 16th Workshop on Par. and Dist. Simulation*, Washington, D.C., May 2002. IEEE.

[FAA+07] T. Fahringer, C. Anthes, A. Arragon, A. Lipaj, J. Müller-Iden, C. J. Rawlings, R. Prodan und M. Surridge. The edutain@grid Project. In *GECON 2007*, 4685 *LNCS*, Rennes, France, August 2007. Springer.

[Fun95] T. A. Funkhouser. RING: A Client-Server System for Multi-User Virtual Environments. In *Symposium on Interactive 3D Graphics*, 1995.

[Kop97] H. Kopetz. *Real-Time Systems. Design Principles for Distributed Embedded Applications*. Kluwer, 1997.

[MFGM04] J. Müller, S. Fischer, S. Gorlatch und M. Mauve. A Proxy Server-Network for Real-time Computer Games. In *Euro-Par 2004*, 3149 *LNCS*, Pisa, Italy, Aug 2004. Springer.

[MG04] J. Müller und S. Gorlatch. A Scalable Architecture for Multiplayer Computer Games. In *INFORMATIK 2004*, 50 *LNI*, Ulm, Germany, August 2004. GI.

[MG05] J. Müller und S. Gorlatch. GSM: A Game Scalability Model for Multiplayer Real-time Games. In *IEEE Infocom 2005*, Miami, Florida, USA, März 2005. IEEE.

[MG06] J. Müller und S. Gorlatch. Rokkatan: scaling an RTS game design to the massively multiplayer realm. *ACM Computers in Entertainment*, 4(3):11, 2006.

[MGG06] J. Müller, A. Gössling und S. Gorlatch. On Correctness of Scalable Multi-server State Replication in Online Games. In *NetGames 2006*, Singapore, October 2006. ACM.

[MGSF07] J. Müller, S. Gorlatch, T. Schröter und S. Fischer. Scalability for Multiplayer Online Games via Proxy-Server Replication: A Case Study of Quake 2. In *16th IEEE HPDC*, Monterey, California, USA, June 2007. ACM.

[MI07] J. Müller-Iden. *Replication-based Scalable Parallelization of Virtual Environments*. Dissertation, Universität Münster, July 2007.

[MPS05] J. H. Metzen, A. Ploss und M. Schellmann. Rokkatan - ein massiv mehrspielerfähiges Echtzzeitspiel. In *Informatiktage 2005*, S-2 of *LNI*. GI, April 2005.

[MVHE04] M. Mauve, J. Vogel, V. Hilt und W. Effelsberg. Local-lag and Timewarp: Providing Consistency for Replicated Continuous Applications. *IEEE Transactions on Multimedia*, 6(1):47–57, February 2004.

[RO03] P. Rosedale und C. Ondrejka. Enabling Player-Created Online Worlds with Grid Computing and Streaming. In *Gamasutra*, September 2003.

[SGB+03] N. Sheldon, E. Girard, S. Borg, M. Claypool und E. Agu. The Effect of Latency on User Performance in Warcraft III. In *NetGames 2003*, Redwood City, USA, May 2003.

[Woo] B. S. Woodcock. MMORPG chart – An Analysis of MMOG Subscription Growth <http://www.mmogchart.com/>.

[WPS+00] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme und G. Alonso. Understanding Replication in Databases and Distributed Systems. In *Proc. of 20th ICDCS*, 2000.

Jens Müller-Iden, geb. Müller, wurde am 27. Februar 1979 in Berlin geboren. Nach Erlangen der Hochschulreife am Gymnasium Bad Nenndorf studierte er von 1999 bis 2003 Informatik an der Technischen Universität Braunschweig. Seit 2003 ist Jens Müller-Iden wissenschaftlicher Mitarbeiter bei Prof. Sergei Gorlatch an der Westfälischen Wilhelms-Universität Münster und promovierte 2007 mit Auszeichnung. Seine Forschungsinteressen umfassen Parallelisierung und Verteilung interaktiver Anwendungen, Skalierbarkeit und Korrektheit virtueller Umgebungen sowie Service-orientierte Architekturen und Grid-Computing. Jens Müller-Iden ist seit 2006 Leiter des Arbeitsbereiches „Real-time Services“ im europäischen Projekt *edutain@grid* und hat über 20 Arbeiten in Zeitschriften und Konferenzen im Bereich parallele und verteilte interaktive Systeme, Grid-Computing und virtuelle Umgebungen veröffentlicht.