Lösungsansätze zur Visualisierung im High Performance Computing und Networking Kontext

Stephan Olbrich[‡] Tino Weinkauf[†] André Merzky[†] Harald Knipp[†]
Hans-Christian Hege[†] Helmut Pralle[‡]

[†]Konrad-Zuse-Zentrum (ZIB), Berlin
{weinkauf,merzky,knipp,hege}@zib.de

[‡]Regionales Rechenzentrum für Niedersachsen (RRZN)
Lehrgebiet Rechnernetze und Verteilte Systeme (RVS), Universität Hannover
{olbrich,pralle}@rrzn.uni-hannover.de

Abstract: In diesem Paper stellen wir ein Echtzeit-Szenario zur Überwachung und Steuerung von entfernt laufenden Großsimulationen vor, das auf einer Extraktion von Geometrien aus den Simulationsdaten, deren Übertragung über Hochgeschwindigkeitsnetze zu einem Client und der daraufhin erfolgenden Simulationssteuerung beruht. Wir stellen die Probleme eines solchen Szenarios dar und präsentieren Lösungen für diese Probleme. Basis dieses Papers ist das BMBF-geförderte DFN-Projekt „Anwendungen der Teleimmersion in Weitverkehrsnetzen“ - einem Gigabit-Testbed mit Partnern in Berlin (ZIB) und Hannover (RRZN/RVS, Institut für Meteorologie und Klimatologie).

1 Szenario und Problematik

Im Zusammenhang mit Anwendungen des Hochleistungrechnens wird bei der grafischen Aufbereitung von Ergebnissen zu Explorations- oder Präsentationszwecken (Wissenschaftliche Visualisierung) ein Dilemma beobachtet. Die Ursache liegt in der unausgewogenen Leistungsfähigkeit der verschiedenen Systemkomponenten, die üblicherweise für die Simulationsrechung und zur Visualisierung verwendet werden.

Einerseits nimmt die Leistungsfähigkeit der Rechner weiterhin exponentiell zu („Moore’s Law“), und insbesondere durch Parallelisierung wird die Berechnung extrem großer Simulationsszenarien in kurzer Zeit möglich. Andererseits können die Ergebnisse derartiger Anwendungen häufig nicht mehr zeitnah interpretiert werden, weil der Zeitaufwand zur grafischen Aufbereitung oftmals um Größenordnungen größer ist als der zur Durchführung der Simulationsrechnung. Die vorhandenen Unzulänglichkeiten betreffen die konzeptionellen Ansätze zur Partionierung der Simulations- und Visualisierungspipeline auf verschiedene Rechner unterschiedlicher Charakteristika, den Datentransport, die Daten- bzw. Feature-Extraktion, das Mapping bzw. die Erzeugung von geometrischen 3D-Objekten,

die Renderingsysteme sowie die Präsentations- und Interaktionstechnik.

Der klassische Ansatz, Ergebnisdaten zu speichern und die grafische Aufbereitung (Postprocessing) anschließend auf speziellen Visualisierungsrechnern durchzuführen, verbietet sich für bestimmte Grand-Challenge-Anwendungen (z. B. instationäre Strömungssimulation mit typ. 10^8 - 10^9 Gitterpunkten, 10^4 Zeitschritten) aus mehreren Gründen:

- wegen des großen Volumens der Ergebnisdaten,
- wegen der hohen Kommunikationsanforderungen sowie
- wegen der hohen Rechenaufwände für das „Visualization Mapping“.

Da die Komplexität dieser Prozesse häufig vergleichbar mit derjenigen der eigentlichen Simulationsrechnung ist, müssten dafür idealerweise auch ähnlich leistungsfähige Rechner verwendet werden. Allerdings skalieren die Leistungsparameter der verfügbaren Hochleistungs-Visualisierungsrechner nicht im gleichen Maße wie die der Simulationsrechner (High-Performance Computing).

2 Lösungsansätze

Das Gesamtproblem lässt sich in mehrere Komponenten partitionieren, die auf verschiedenen, über leistungsfähige Datennetze gekoppelten Rechnern mit jeweils speziellen Charakteristika ablaufen können. Die innovativen Ansätze, die zurzeit vom RRZN/RVS und ZIB im Rahmen eines BMBF/DFN-Projekts zur Lösung typischer Probleme im Kontext „High-Performance Computing and Networking“ entwickelt werden, basieren auf einem verteilten Systemkonzept.

Dabei kommen folgende fortgeschrittene Basistechnologien zum Einsatz:

- Parallele Datenextraktion zur effizienten Generierung geometrischer 3D-Szenen aus den Daten von hochkomplexen Simulationen,
- 3D-Streaming-System zur interaktiven 3D-Visualisierung in räumlich verteilten Szenarien:
  - Synchronisierte Ausspielung von 3D-Szenensequenzen,
  - Interaktiv räumlich und in der Zeitachse navigierbare 3D-Animation,
  - Online-Visualisierung - „on-the-fly“ während der Simulationsrechnung,
  - Kooperative Betrachtung - synchronisierte Sicht, Videoconferencing,
  - Interaktive Steuerung von Simulationsparametern.
- Gigabit-Ethernet im lokalen Netz und im Weitverkehrsnetz durch Nutzung einer transparenten WDM-Strecke (Wave Division Multiplex) im Gigabit-Wissenschaftsnetz G-WiN.

Auf die ersten beiden Aspekte wird in den folgenden Abschnitten näher eingegangen.

3 Parallele Datenextraktion – Generierung und Transport von 3D-Szenen

Bei Simulationen auf Supercomputern fallen i.a. Datenmengen von derart großem Umfang an, dass diese ebenfalls nur durch Großrechner zu verarbeiten sind. Somit ist es notwendig, den ersten Teil der Visualisierung, die Extraktion von Geometrien aus den Simulationsdaten, neben der Simulation selbst ebenfalls auf dem gleichen Supercomputer durchzuführen. Dies bringt jedoch folgende Probleme mit sich:

1. Die Visualisierungsalgorithmen müssen die Simulationsdaten lesen und interpretieren können. Von einer Anpassung der Simulation an die Visualisierung kann nicht ausgegangen werden.
2. Die Simulationsdaten dürfen aus Speicherplatz- und Zeitgründen nicht kopiert werden.
3. Die Visualisierungsalgorithmen müssen parallelisiert werden.
4. Die extrahierten Geometrien müssen zur eigentlichen Darstellung über ein Netzwerk gesendet werden.

Die Probleme 1-3 werden durch die am ZIB entwickelte *Parallel Data Extraction and Visualization Library (PaDEV)* gelöst. Über einen speziellen C++–Template-Mechanismus können verschiedenste Datenlayouts den Visualisierungsalgorithmen zur Verfügung gestellt werden, welche über ein einheitliches Interface auf die Daten zugreifen können. Abbildung 1 zeigt die Trennung zwischen Daten-, Kommunikations-, und Visualisierungsebene.
Zur Lösung des Problems 4 bedient sich die PaDEV der in den folgenden Abschnitten beschriebenen libDVRP.

3.1 Effiziente Generierung von 3D-Szenen

Werden die Teilergebnisse einer parallelen Simulationsrechnung entsprechend der zu Grunde liegenden Gebietszerlegung direkt auf dem jeweiligen Prozessor in geometrische 3D-Objekte (z. B. Slicers, Isosurfaces, Streamlines) aufbereitet, so werden zwei Ziele erreicht:

- Datenkompression
  Gegenüber den Rohdaten ist die Datenmenge der 3D-Szenen im Allgemeinen deutlich geringer.
- Reduktion des Zeitaufwands
  Die Parallelisierung ermöglicht - im Vergleich zum traditionellen Postprocessing auf einem separaten Grafikrechner - eine ausgewogenere Visualisierungspipeline.

Nach einigen wenig erfolgreichen Experimenten mit der Visualisierungsbibliothek VTK [SML97] wurde zu diesem Zweck eine MPI-basierte [GLS99], von Fortran aufrufbare,

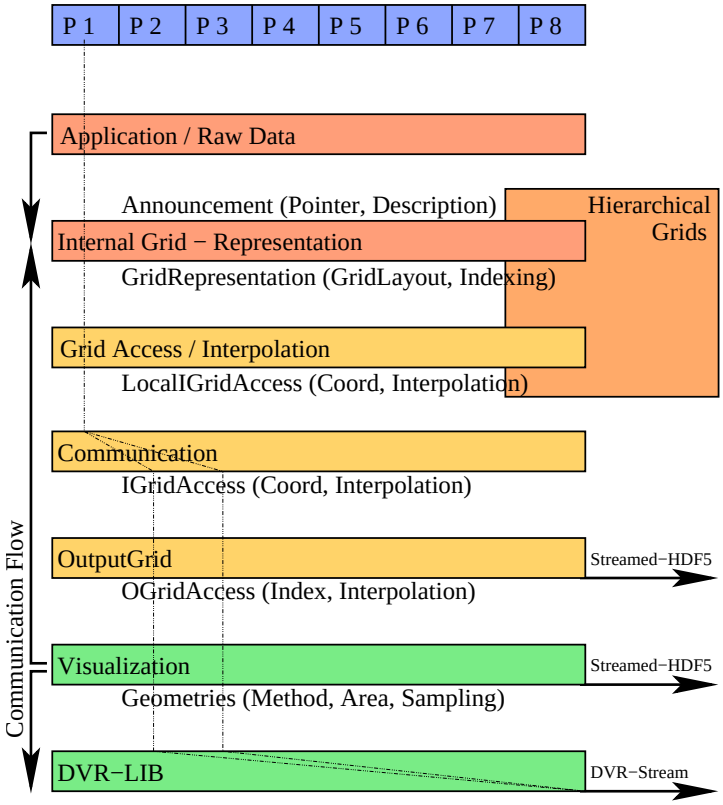

Abbildung 1: Aufbau der „Parallel Data Extraction and Visualization Library“ (PaDEV)

eigene Visualisierungsbibliothek in C entwickelt [OPR01]. Diese „libDVRP“ stellt einen einfachen Satz parallelisierter Methoden zur Visualisierung von Volumendaten auf rektilinearen Gittern (Coloured Orthogonal Slicer, Isosurface[1]) sowie Funktionen zur direkten Erzeugung von 3D-Objekten (Polygone, Linien, Punkte) bereit.

Durch die zur Zeit erfolgende Kopplung der PaDEV an die libDVRP wird das Gesamtsystem um die Möglichkeit zur Verarbeitung beliebiger Gitter- und Datentypen erweitert, da die PaDEV aufgrund ihrer modularen Gestaltung leicht angepasst werden kann.

Die libDVRP, welche nur für bestimmte Anwendungsfälle (z. B. rektilineare Gitter) optimiert ist, wurde bereits erfolgreich in mehrere bestehende, parallele Simulationsprogramme integriert. Ein Anwendungsbeispiel aus der Meteorologie wird in Abbildung 2 gezeigt. Durch die Aufheizung der Luft von der Oberfläche her bildet sich eine konvektive Grenzschicht. Die Strömung an dieser Schicht wird hier durch passiv transportierte Partikel und deren Trajektorien visualisiert. Zudem wird die Luftfeuchtigkeit durch Isoflächen (Wol-

[1]Basierend auf einer seriellen Implementierung des Marching-Cube-Algorithmus [LC87]: Polyreduce, by Jesper James; http://hendrix.ei.dtu.dk/software/software.html

ken) dargestellt.

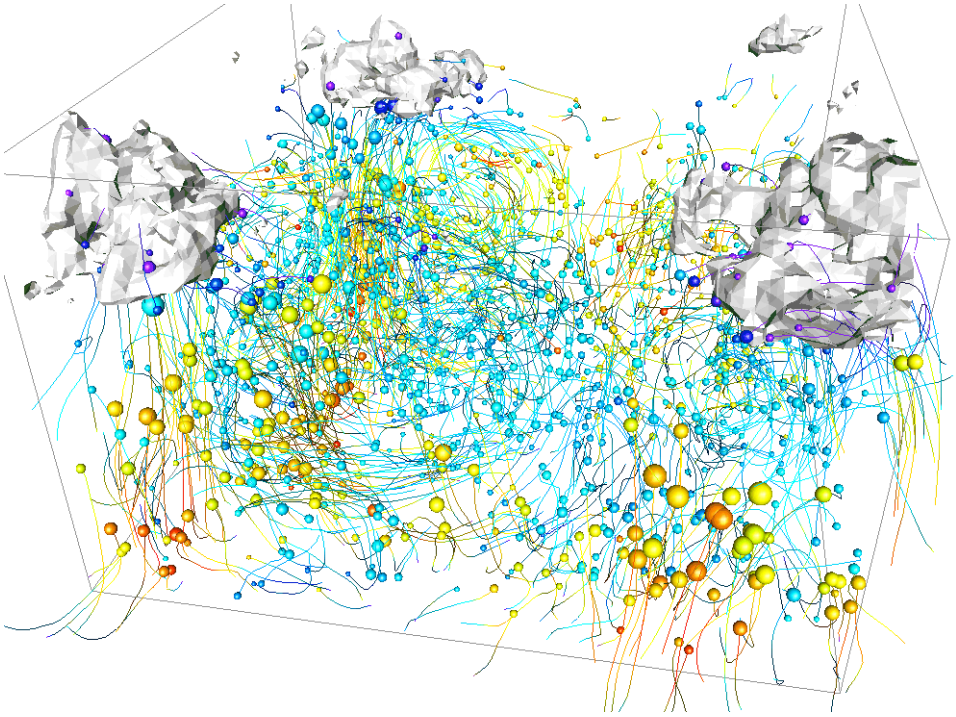

Abbildung 2: Konvektive Grenzschicht. In Zusammenarbeit mit dem Institut für Meteorologie und Klimatologie der Universität Hannover.

3.2 Kopplung mit dem 3D-Streaming-System

libDVRP beinhaltet die Option, 3D-Szenen(-Sequenzen) entweder auf Dateien zu speichern (lokal oder auf FTP-Server) oder einen kontinuierlichen DVRP-Datenstrom zu erzeugen. Die letztgenannte Möglichkeit wurde als Client-Funktionalität in Bezug auf den in 4 beschriebenen 3D-Streamingserver implementiert. Auf diesem werden die 3D-Szenen gespeichert, um sie on-the-fly (live streaming) oder zeitversetzt (on-demand streaming) auf räumlich verteilten Viewer-Clients präsentieren zu können.

Darüber hinaus stellt libDVRP auch Funktionen für „Computational Steering“ bereit. Diese dienen dazu, aktuell angekoppelte Viewer-Clients über die steuerbaren Parameter zu informieren sowie auf dem DVRP-Rückkanal Parameteränderungen zu empfangen und in der parallelen Simulationsrechnung auf die beteiligten Prozessoren zu verteilen.

Die ursprüngliche libDVRP-Implementierung blockierte während des DVRP-Datentransports sämtliche Prozessoren außer dem einen, auf dem die Kommunikation ablief (siehe auch

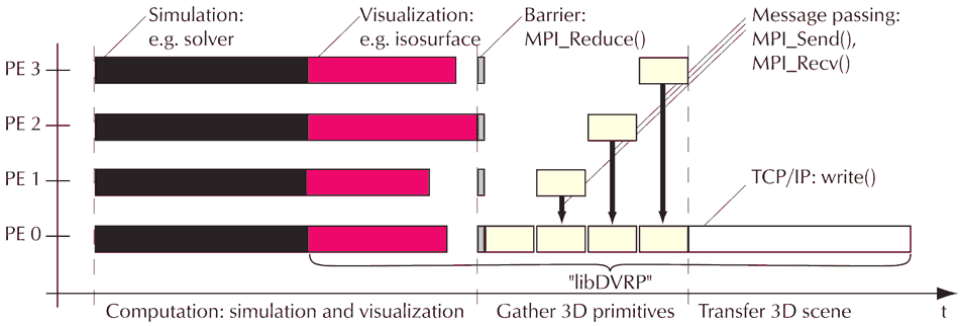

Abbildung 3: Prinzipieller Ablauf der eng gekoppelten Simulation und Visualisierung mit libDVRP. Hier erfolgt der Datentransport zum Streamingserver auf PE 0, dabei blockieren die PEs 1-3.

Abbildung 3). Da diese Vorgehensweise für eine große Prozessoranzahl zu einem deutlichen Engpass führen kann, wurde eine Option zur asynchronen Ausgabe eingeführt. Dabei wird der Datentransport mittels eines dedizierten Prozessors, der vorher mit der MPI-Funktion „Split Communicator“ abgekoppelt wird, jeweils zeitlich überlappt mit der Ausführung des nächsten Rechen- und Datenextraktionsschritts durchgeführt.

3.3 Effizienz-Analyse – Datentransfer in libDVRP: blockierend vs. asynchron (Pipelining-Option)

Verwendete Symbole

T_S **:** Zeitaufwand zur Ausführung eines sequentiell implementierten Simulations- und Visualisierungsschritts.

T_0 **:** Zeitaufwand zum Transport der Ergebnisse eines Simulations- und Visualisierungsschritts zum Streamingserver (hängt von Datenvolumen und Datenrate ab).

k **:** Konstante zur Charakterisierung der Anwendung.

$$k = \frac{T_S}{T_0} \tag{1}$$

N **:** Prozessor-Anzahl.

$T_P(N)$ **:** Zykluszeit der parallelisierten Simulations-, Visualisierungs- und Transfer-Schritte.

$S_P(N)$ **:** Speedup bei der Parallelisierung.

$$S_P(N) = \frac{T_S + T_0}{T_P(N)} \tag{2}$$

$E_P(N)$ **:** Effizienz der Parallelisierung.

$$E_P(N) = \frac{S_P(N)}{N} \tag{3}$$

Vereinfachende Annahme

Simulation und Visualisierung skalieren linear für beliebige N. D. h. ohne den Transport von Ergebnissen zum Streamingserver $T_0 = 0$ gelte:

$$T_P(N) = \frac{T_S}{N} \quad \Rightarrow \quad S_P(N) = N \quad \Rightarrow \quad E_P(N) = 1 \tag{4}$$

Zwei Parallelisierungsstrategien bezüglich des Datentransfers

1. *Blockierender Datentransport*
   Der Transfer wird auf einem Prozessor durchgeführt, dabei blockieren alle anderen Prozessoren. Es gilt:

$$\text{Zykluszeit:} \quad T_P(N) = \quad \frac{T_S}{N} + T_0 \tag{5}$$

$$\text{Speedup:} \quad S_P(N) = \quad \frac{T_S + T_0}{\frac{T_S}{N} + T_0} \quad \Rightarrow \quad S_P(N) = \frac{k+1}{\frac{k}{N} + 1} \tag{6}$$

$$\text{Effizienz:} \quad E_P(N) = \quad \frac{T_S + T_0}{T_S + N \cdot T_0} \quad \Rightarrow \quad E_P(N) = \frac{k+1}{k+N} \tag{7}$$

2. *Asynchroner Datentransport (Pipelining)*
   Der Transfer wird auf einem zusätzlichen Prozessor durchgeführt, und zwar zeitlich überlappt mit dem jeweils nachfolgenden Simulations- und Visualisierungsschritt. Es gilt:

$$\text{Zykluszeit:} \quad T_P(N) = \quad \max\left(T_0, \frac{T_S}{N-1}\right) \tag{8}$$

   Hier sind zwei Fälle zu unterscheiden:

   (a)

$$T_0 \leq \frac{T_S}{N-1} \text{ bzw. } k \geq N-1. \text{ In diesem Fall gilt: } T_P(N) = \frac{T_S}{N-1} \tag{9}$$

$$\text{Speedup:} \quad S_P(N) = \frac{(T_S + T_0) \cdot (N-1)}{T_S} \tag{10}$$

$$\Rightarrow \quad S_P(N) = \frac{(k+1) \cdot (N-1)}{k} \tag{11}$$

$$\text{Effizienz:} \quad E_P(N) = \frac{(T_S + T_0) \cdot (N-1)}{N \cdot T_S} \tag{12}$$

$$\Rightarrow \quad E_P(N) = \left(1 + \frac{1}{k}\right) \cdot \left(1 - \frac{1}{N}\right) \tag{13}$$

(b)

$$T_0 > \frac{T_S}{N-1} \text{ bzw. } k < N-1. \text{ In diesem Fall gilt: } T_P(N) = T_0 \tag{14}$$

$$\text{Speedup:} \quad S_P(N) = \frac{T_S + T_0}{T_0} \tag{15}$$

$$\Rightarrow \quad S_P(N) = k+1 \tag{16}$$

$$\text{Effizienz:} \quad E_P(N) = \frac{T_S + T_0}{N \cdot T_0} \tag{17}$$

$$\Rightarrow \quad E_P(N) = \frac{k+1}{N} \tag{18}$$

Verlauf von Speedup und Effizienz in einer Beispielanwendung

In Abbildung 4 sind Speedup und Effizienz in Abhängigkeit von der Prozessoranzahl für die beiden betrachteten Transfer-Alternativen (ohne / mit Pipelining) mit $k = 32$ grafisch dargestellt. Die Illustration beruht auf den Zusammenhängen gemäß Gleichungen (6), (7), (11), (13), (16) und (18).

Im Vergleich zwischen den jeweiligen beiden Kurven in Abbildung 4 ist festzustellen, dass ein Vorteil der Pipelining-Strategie erst oberhalb einer bestimmten Prozessoranzahl existiert. Durch Gleichsetzung der Gleichungen (2) und (6) bzw. (3) und (7) ergibt sich als Bedingung für diesen „Break-Even-Point“:

$$N \cdot (N-1) = k \tag{19}$$

Für den in Abbildung 4 dargestellten Fall mit $k = 32$ gilt:

$$N \geq 7 \tag{20}$$

Vorteile der Pipelining-Strategie

1. *Bessere Skalierbarkeit der Parallelisierung.*
   Zum Beispiel sinkt die Effizienz – verglichen mit der blockierenden Version – erst bei ca. doppelt so hoher Prozessoranzahl unter 50%.

- ohne Pipelining: $E_P(N) \geq 0.5 \Rightarrow k \leq k+2$ (aus (7))
- mit Pipelining: $E_P(N) \geq 0.5 \Rightarrow k \leq 2k+2$ (aus (17) und (18))

2. *Geringere Burstiness, d. h. Glättung des Datentransports.*
   Mit $k \leq N-2$ führt das Pipelining-Verfahren zu nahezu kontinuierlicher Datenübertragung.

3. *Geringere Anforderung an die Datenrate für den jeweiligen Datentransport.*
   Zur Erzielung einer Effizienz von mindestens 50% – verglichen mit der blockierenden Version – bei gleich großer Prozessoranzahl eine nur ca. halb so hohe Datenrate erforderlich.

   - ohne Pipelining: $E_P(N) \geq 0.5 \Rightarrow k \geq N-2$
   - mit Pipelining: $E_P(N) \geq 0.5 \Rightarrow k \geq \frac{1}{2}(N-2)$

   Da T_S in beiden Fällen identisch ist, kann k (gemäß Gleichung (1)) bei gleichbleibendem jeweiligen Datenvolumen als nahezu proportional zur Datenrate angenommen werden.

Praktische Anwendung der Pipelining-Option (DVRP_Split)

In Abbildung 5 ist der tatsächliche Zeitablauf in einer praktischen Simulations- und Visualisierungsanwendung veranschaulicht. Die zeitliche Überlappung des Datentransport und nahezu vernachlässigbare Behinderung der Simulationsrechnung ist deutlich erkennbar.

4 3D-Streaming-System

Um die Ergebnisse einer laufenden Simulationsrechnung „on the fly“ betrachten und interpretieren zu können, ist es erforderlich, Ergebnisextrakte (z. B. 3D-Szenen) als kontinuierlichen Datenfluss zu einer 3D-Viewer-Komponente zu übertragen. Neben einer Erzielung kurzer Latenzzeiten wird durch das dabei realisierte Pipelining auch eine Parallelisierung der Supercomputer- (Simulation und Datenextraktion) und Visualisierungsprozesse (3D-Rendering) erreicht.

Dafür können Streamingverfahren, die ursprünglich für Audio- bzw. Video-Anwendungen entwickelt wurden, angewandt werden. Auf diesem Prinzip beruht das am RRZN/RVS entwickelte System zur Abspielung von räumlich und in der Zeitachse interaktiv navigierbaren „3D-Filmen“. Dabei werden die Datenflüsse für Steuerungskommandos (RTSP - Real-Time Streaming Protocol, RFC 2326, [SRL98]) und synchronisierbare Mediendatenströme (hier: binäre Repräsentation von 3D-Szenen - „DVRP“) über separate TCP/IP-Verbindungen realisiert.

In Analogie zum Video-Streaming im WWW diente das 3D-Streaming-Abrufsystem zunächst zur Ausspielung vorbereiteter Szenensequenzen („Play“). Später wurden noch die Betriebsmodi „Record“ und „Live Streaming“ hinzugefügt. Dadurch wird eine Online-Generierung und -Speicherung virtueller 3D-Szenen unterstützt, und es werden sowohl

synchrone als auch asynchrone Visualisierungsszenarien ermöglicht. Außerdem wurde ein Rückkanal für „Computational Steering“ integriert (Abbildung 6). [OP01] [OP98] [OP99]

5 Zusammenfassung

Auf der Basis der vorgestellten Ansätze zur Lösung der zunehmend im Kontext „High-Performance Computing and Networking“ zu beobachtenden Probleme bei der Visualisierung in aufwändigen Simulationsszenarien wird zurzeit ein Werkzeugkasten zusammengestellt, der leistungsfähige Unterprogrammbibliotheken, Server-Software und 3D-Viewer-Programme beinhaltet.

Entsprechend der Gebietszerlegung, wie sie üblicherweise in komplexen Simulationsrechnungen für massiv-parallele Supercomputer erforderlich ist, erfolgt eine Datenextraktion in geometrische 3D-Objekte bereits am Ort der Quelle, d. h. auf dem jeweiligen Prozessor. Damit wird der Prozess der Generierung von 3D-Szenen, ebenso wie die eigentliche Simulationsrechnung, parallelisiert. Außerdem wird das Datenvolumen, im Vergleich zu den Rohdaten, deutlich reduziert.

Darüber hinaus legt der iterative Charakter einer Simulation zeitlich veränderlicher Phänomene, in denen häufig Tausende von Zeitschritten zu berechnen sind, die Anwendung von Streamingverfahren nahe. Auf dieser Grundlage werden, im Gegensatz zu der bislang üblichen Handhabung relativ großer Ergebnisdateien, auch interaktive Anwendungen in dynamischen, hochkomplexen Simulations- und Visualisierungsszenarien ermöglicht.

Leistungsfähige Weitverkehrsnetze - wie die im Gigabit-Wissenschaftsnetz heute verfügbaren transparenten WDM-Kanäle, auf die z. B. Gigabit-Ethernet unmittelbar aufgesetzt werden kann - erlauben darüber hinaus räumlich verteilte, tele-immersive „Virtual Lab“-Konzepte. Damit können z. B. Ergebnisse zentralisiert betriebener Höchstleistungsrechner auf weit entfernten Wissenschaftler-Arbeitsplätzen bereits während der Rechnung als 3D-Animation räumlich betrachtet werden, Simulationen können interaktiv gesteuert werden, und auch eine gemeinsame Nutzung geografisch verteilter Forschergruppen wird durch eine 3D-Viewer-Software unterstützt, in der Videoconferencing und 3D-Telepointer integriert sind.

Literaturverzeichnis

[GLS99] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI*. MIT Press, 1999.

[LC87] W. E. Lorensen and H. E. Cline. Marching Cubes: a high resolution 3D surface reconstruction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[OP98] S. Olbrich and H. Pralle. High-Performance Online Presentation of Complex 3D Scenes. In H. R. van As, editor, *IFIP TC-6 Eighth International Conference on High Performance Networking – HPN ’98*, Vienna, Austria, 1998. Kluwer Academic Publishers.

[OP99] S. Olbrich and H. Pralle. Virtual Reality Movies – Real-Time Streaming of 3D Objects. In *Computer Networks - The Challenge of Gigabit Networking*, volume 31, Lund, Schweden,

November 1999. Elsevier. selected papers from the TERENA-NORDUnet Networking Conference.

[OP01] S. Olbrich and H. Pralle. A Tele-Immersive, Virtual Laboratory Approach based on Real-Time Streaming of 3D Scene Sequences. In *Proceedings of ACM Multimedia Conference*, 2001.

[OPR01] S. Olbrich, H. Pralle, and S. Raasch. Using Streaming and Parallelization Techniques for 3D Visualization in a High-Performance Computing and Networking Environment. In *HPCN 2001 – International Conference on High Performance Computing and Networking*, volume 2110 of *Lecture Notes in Computer Science*, Amsterdam, 2001. Springer. June 25-27.

[SML97] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit*. Prentice Hall, 2 edition, 1997. URL: http://www.kitware.com/.

[SRL98] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP), April 1998. RFC 2326.

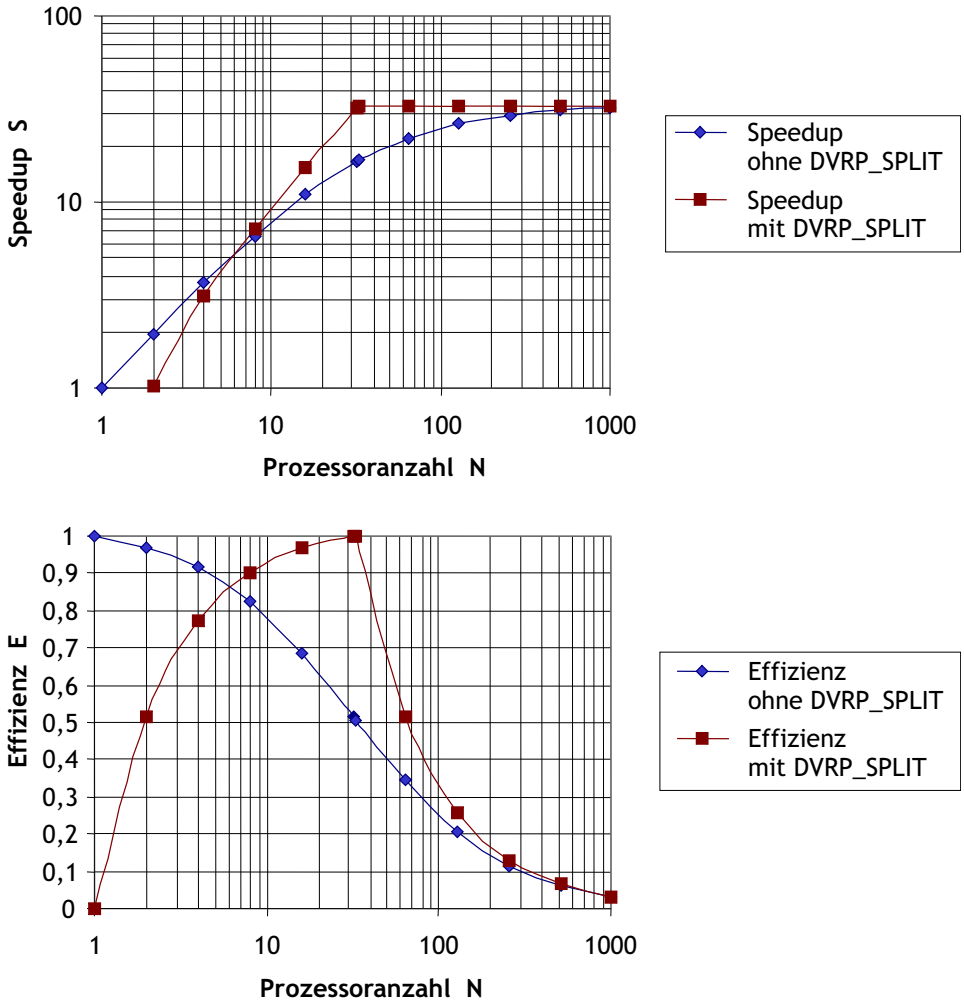

Abbildung 4: Speedup und Effizienz in Abhängigkeit von der Prozessoranzahl für die beiden betrachteten Transfer-Alternativen (ohne / mit Pipelining) in einer Beispielanwendung mit $k = 32$.

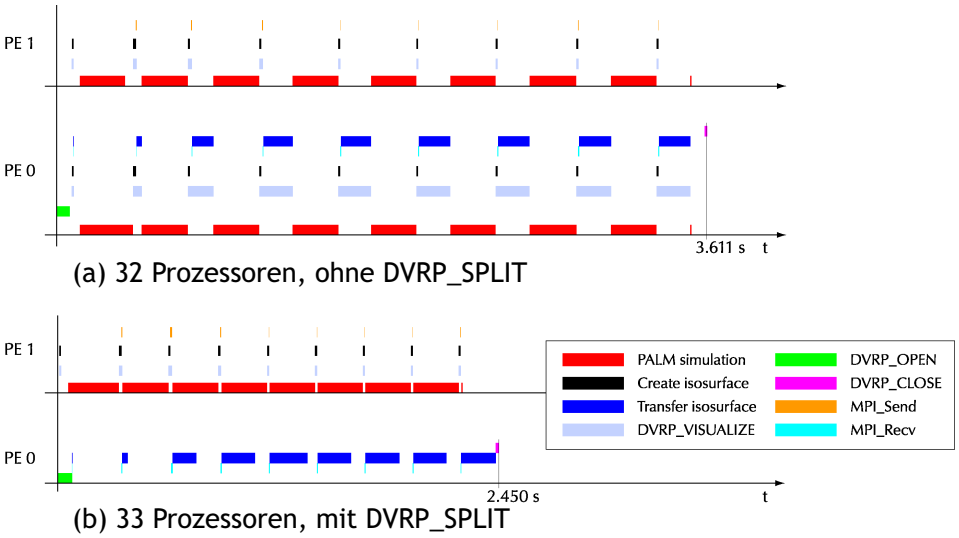

Abbildung 5: Time limes, showing atmospheric simulation (159x31x32 grid points), isosurface visualization and transfer (from Cray T3E to SGI Origin 200, connected via 100 Mbps Ethernet) steps, in a (small) case study on a Cray T3E. Only the first 2 PEs are shown, the other PEs behave similar to PE 1. Improvement by using DVRP_SPLIT: 32%. 3D data volume (9 DVR files): 3.363.592 bytes.

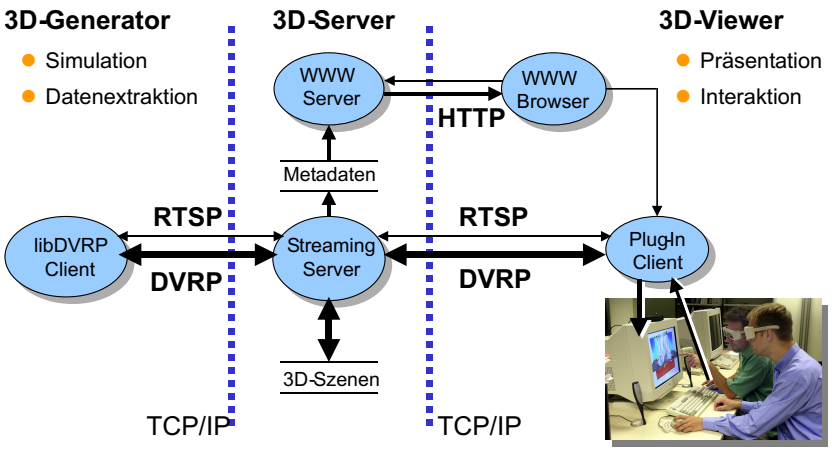

Abbildung 6: 3D-Streaming-Server: Vermittler und Pufferspeicher zwischen 3D-Quelle (Simulationsrechnung) und 3D-Viewer (VR-System).