VIRACOCHA: Paralleles CFD-Postprocessing für die echtzeitfähige Visualisierung und Interaktion in VR-basierten Arbeitsumgebungen

Andreas Gerndt, Torsten Kuhlen, Christian Bischof
Virtual Reality Group
RWTH Aachen University
52074 Aachen
Tel.: +49 (0)241 80 24900
Fax: +49 (0)241 80 22134
E-Mail: {gerndt, kuhlen, bischof}@rz.rwth-aachen.de

Zusammenfassung: Durch den ständigen Performanzanstieg moderner Hochleistungsrechner und deren Verfügbarkeit für die Ingenieurswissenschaften wächst auch der Umfang an Ergebnisdaten im Bereich der Strömungssimulation. Diese Rohdaten sind ohne ein anschließendes Postprocessing weitestgehend unbrauchbar. Eine empfohlene Analysestrategie ist die interaktive Exploration innerhalb virtueller Umgebungen. Allerdings sind übliche Visualisierungssysteme kaum in der Lage, die anfallenden Datenmengen aufzubereiten und gleichzeitig eine echtzeitfähige Interaktion zu gewährleisten. Ein Erfolg versprechender Ansatz ist die Entkopplung der Merkmalsextraktion von der eigentlichen Visualisierung. In diesem Beitrag wird die Funktionalität des parallelen CFD-Postprozessors Viracocha beschrieben. Neben der Darstellung der verteilten Architektur steht zum einen das in Viracocha integrierte Datenmanagement im Vordergrund, wobei ein Schwerpunkt auf der Präsentation des Markov-Prefetchers liegt. Zum anderen wird das auf unterschiedlichen Multiresolution-Ansätzen basierende *Data Streaming* vorgestellt. Mit den entwickelten Ansätzen wird das Ziel verfolgt, auch große Simulationsdatensätze in virtuellen Umgebungen untersuchen zu können. Die Effizienz des Systems wird an einigen Beispielen exemplarisch präsentiert.

Stichworte: Interaktives CFD-Postprocessing, Parallelisierung, Markov-Prefetching, Data Streaming, Multiresolution

1 Einleitung

Noch vor wenigen Jahren benötigte die numerische Berechnung von Strömungsphänomenen einige Tage bis Wochen, wobei die zugrunde liegenden Strömungsfelder einfache Geometrien beschrieben und eher grob vernetzt waren. Erst mit der zunehmenden Verfügbarkeit leistungsfähiger Rechnersysteme wuchs die Bedeutung der numerischen Strömungssimulation (engl.: *Computational Fluid Dynamics*, CFD). Allerdings sind die erzeugten Simulationsdatensätze nicht direkt auswertbar. Vielmehr werden effiziente Verfahren für deren Analyse benötigt. So extrahiert erst das so genannte Postprocessing charakteristische Strukturen

und Merkmale, die durch geeignete visuelle Repräsentanten mithilfe der wissenschaftlichen Visualisierung einen Überblick über diese abstrakten Daten geben können.

Die weiterhin steigenden Rechenkapazitäten im Bereich des *High Performance Computing* (HPC) führen zu immer detaillierteren und exakteren Ergebnissen. Andries van Dam et al. weisen allerdings in [DamFo2000] auf das gewichtige Problem hin, dass die so gewonnenen Rohdaten wesentlich schneller wachsen als die Fähigkeit zur Weiterverarbeitung und Analyse. Langfristig lässt sich dieses Problem laut Autoren nur durch den Einsatz von Methoden der künstlichen Intelligenz lösen. Der Endanwender würde nur noch mit einer vorstrukturierten, handhabbaren Menge von aufbereiteten Daten konfrontiert. Für den kurz- und mittelfristigen Zeitraum schlagen die Autoren den Einsatz der *Immersive Virtual Reality* (IVR) vor, welche die interaktive Visualisierung mit dem Immersionseffekt virtueller Umgebungen verbindet.

Die explorative Analyse lebt ganz erheblich von der Interaktivität des verwendeten Systems. Für virtuelle Umgebungen lassen sich Interaktionskriterien definieren, die ein flüssiges Arbeiten sicherstellen sollen:

1. *Mindest-Bildwiederholungsrate:* Die Applikation darf eine minimale Bildwiederholungsrate nicht unterschreiten.

2. *Maximal-Systemreaktionszeit*: Zeit, die ein Anwender maximal toleriert, bis vom System ein Feedback auf eine Benutzereingabe erfolgen muss.

3. *Maximale Sekundärreaktionszeit*: Dieser Bewertungsmaßstab ist besonders wichtig für rechenintensive Anwendungen. Sie gibt an, bis wann Berechnungsergebnisse spätestens vorliegen müssen, damit der Anwender eine vergangene Berechnungsanforderung noch mit eintreffenden Daten in Beziehung setzen kann.

1.1 Ansätze verteilter Postprocessing-Systeme

HPC-Systeme und Visualisierungsrechner sind für ihre jeweiligen Einsatzbereiche optimiert. Während Hochleistungsrechner über einen großen Hauptspeicher, viele Prozessoren und eine schnelle Anbindung an Sekundärspeicher für die Deckung des Ressourcenbedarfs paralleler Berechnungsprogramme (z. B. Strömungssimulationsprogramme) verfügen, besitzen Visualisierungssysteme optimierte Graphik-Subsysteme, die vor allem für das aufwändige Rendern komplexer graphischer Szenen geeignet sind.

Es liegt nun nahe, dass der Hochleistungsrechner, der bereits die CFD-Simulationsdaten erzeugt hat, auch für das anschließende Postprocessing ausreichend Rechen- und Speicherkapazitäten zur Verfügung stellen kann. Die zentrale Idee ist daher alle Extraktionsberechnungen auf Parallelrechner auszulagern. Durch diese Entkopplung stehen auf dem Visualisierungsrechner alle Systemressourcen für die ausschließliche Darstellung der Extraktionsergebnisse zur Verfügung.

Um den Analysevorgang flexibel steuern zu können, werden die einzelnen Arbeitsschritte in einer Visualisierungspipeline zusammengefasst. Diese stellt auch die Grundlage für die parallele Verarbeitung von Extraktionsalgorithmen dar. Entsprechend der Parallelisierungsmöglichkeit können erhebliche Berechnungsbeschleunigungen (engl.: *Speed-Up*) erzielt werden. Die Grenzen zwischen dem Hochleistungs-*Backend* und dem Visualisierungs-*Frontend* sind dabei variabel definierbar.

Die Auslagerung rechenintensiver Prozesse auf Hochleistungsrechnern wird von einigen Postprocessing-Programmen verwendet. Bereits 1989 wurde MPGS (später kommerziell als EnSight vertrieben) von Grimslud und Lorig präsentiert [GriLo1989], das hierfür Cray-Supercomputer einsetzte. Eines der ersten verteilten, auf VR basierenden Systeme für die Strömungsvisualisierung wurde 1992 von Bryson et al. als *Distributed Virtual Windtunnel* [BryGe1992] vorgestellt. Das um ein vektorisiertes Postprocessing-Backend erweiterte VR-System lagert die Merkmalsextraktion ebenfalls auf einen Hochleistungsrechner aus. Ein weiteres, primär auf kollaboratives Arbeiten ausgerichtetes System ist COVISE [RanFra1998]. Um hierfür auch Mehrprozessorsysteme für eine Beschleunigung der Berechnungsschritte ausnutzen zu können, lässt sich wie bei vielen datenflussorientierten Programmen die zugrunde liegende Visualisierungspipeline in einzelne Berechnungsmodule aufteilen, die sodann verschiedenen Berechnungsknoten zugewiesen werden.

1.2 Aufbau des Beitrags

Im Folgenden soll das an der RWTH entwickelte, verteilte und VR-basierte Postprocessing-System *Viracocha* vorgestellt werden. Da große Datensätze eine besondere Herausforderung an das Datenmanagement stellen, wurde das *Viracocha Data Management System* (VDMS) als *Middleware* in Viracocha integriert. Die damit verbundenen Konzepte werden in Kapitel 3 beschrieben. Der Fokus liegt dabei auf wahrscheinlichkeitsbasierten Prefetching-Strategien. Die Beschreibung der Ansätze des *Data Streamings*, das vor allem zur Steigerung der Interaktivität durch Minimierung der Sekundärreaktionszeit während der explorativen Analyse in virtuellen Umgebungen dient, folgt sodann in Kapitel 4. Das anschließende Ergebniskapitel belegt die Effizienz der in Viracocha verankerten Verfahren. Eine Zusammenfassung beschließt diesen Beitrag.

2 Paralleles CFD-Postprocessing mit Viracocha

ViSTA FlowLib wurde als optionales Erweiterungsmodul für die an der RWTH Aachen entwickelte VR-Software ViSTA (*Virtual Reality for Scientific Technical Applications*) konzipiert. Es ist speziell auf die Bedürfnisse der wissenschaftlichen Visualisierung zugeschnitten. Da neben der Visualisierung instationärer Strömungsvorgänge die Behandlung sehr großer Datensätze eines der wesentlichen Ziele darstellt, wurde die speicherintensive Behandlung der Ausgangsdaten und das zeitaufwändige Berechnen von Extraktionsmerkmalen von der Visualisierung entkoppelt und auf einen zweiten Rechner ausgelagert. Während

ViSTA FlowLib nur noch für die echtzeitfähige Visualisierung zuständig ist, wird das eigentliche Postprocessing nun von einem Framework namens Viracocha [KuhGe2005] durchgeführt. Viracocha ist dabei nicht auf eine spezielle Hardware-Architektur festgelegt. So lassen sich einfach vernetzte Rechnerverbünde, optimierte Linux-Cluster bis hin zu beliebigen Supercomputersystemen einsetzen.

Viracocha verbindet den Parallelrechner (*WorkHost*) über einen Kommunikations- und einen Datenkanal mit dem Visualisierungsrechner (*VisHost*). Der Aufbau der Verbindungskanäle geschieht zur Laufzeit. Dadurch können *VisHost* und *WorkHost* unabhängig voneinander laufen. Der *WorkHost* fungiert als Dienstleistungsserver, der jederzeit Postprocessing-Aufträge über den Befehlskanal empfangen kann und die Ergebnisdaten über den Datenkanal zurücksendet. Die wesentlichen Objekte von Viracocha sind der *Scheduler* und eine feste Anzahl von *Workern* (vgl. Abbildung 1). Werden Berechnungsanfragen vom *VisHost* an den *WorkHost* abgesendet, werden diese vom *Scheduler* als Aufgabenverwaltungseinheit empfangen. Sind genügend *Worker* für die beauftragte Berechnung verfügbar, wird eine Arbeitsgruppe (*WorkGroup*) zusammengestellt, die sodann den Auftrag parallelisiert bearbeitet.

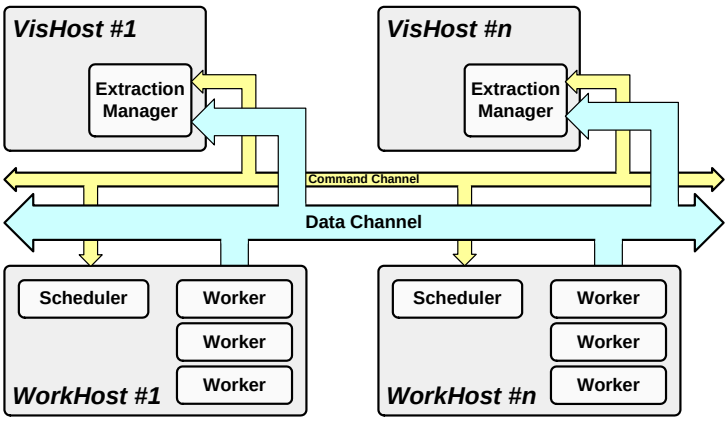

Abbildung 1: ViSTA FlowLib / Viracocha als verteiltes System

Viracocha gibt weder die zu verwendende Parallelisierungsschnittstelle noch die Kommunikationsprotokolle vor. Noch nicht einmal die tatsächlich zum Einsatz kommenden Berechnungsalgorithmen sind Viracocha bekannt. Somit lässt sich das Parallelisierungs-Framework z. B. auch für andere Bereiche als dem CFD-Postprocessing einsetzen. Diese Flexibilität wird durch die Definition mehrerer Abstraktionsschichten erreicht.

3 Das Viracocha Data Management System

Viracocha bietet mit den Ansätzen zur Parallelisierung bereits weit reichende Methoden für das CFD-Postprocessing an. Allerdings treten bereits mit einfachen Datensätzen Balancierungs-probleme auf, die vor allem durch das während der laufenden Berechnung erforderliche Nachladen von Daten herrührt. Daher wurde das *Viracocha Data Management System* (VDMS) entwickelt, das als *Middleware* fest in Viracocha integriert wurde. Dadurch steht es allen

Algorithmen zur Verfügung, wobei die Verwendung transparent stattfindet, d. h. dass der Algorithmus im Allg. von der Existenz des VDMS nichts weiß und somit auch sein Verhalten nicht kontrollieren muss. Folgende Hauptstrategien wurden hierfür realisiert:

Caching: Bereits geladene Daten werden, solange ausreichend Platz zur Verfügung steht, in applikationsnahen Speichern gehalten. Bei erneuter Anfrage des Datums wird es nicht vom entfernt liegenden Dateisystem geladen, sondern direkt aus dem Cache geholt.

Prefetching: Daten werden bereits vor der Anfrage in den Cache geladen. Wird das Datum dann tatsächlich angefordert, kann der Cache das Datum sofort liefern. Der Geschwindigkeitsvorteil wird dadurch erreicht, dass die Daten im Hintergrund parallel zur laufenden Berechnung geladen werden. Der Algorithmus muss daher im günstigsten Fall während einer Datenanfrage seine Berechnung nicht unterbrechen.

Ladestrategien: In Abhängigkeit von Datenstrukturen und verfügbaren Hardware-Ressourcen lassen sich optimierte Ladestrategien registrieren und ausführen. Liegen zum Beispiel parallele Dateisysteme vor, können entsprechende Ladestrategien definiert werden, die diese Technik ausnutzen.

Data Service: Dieser Ansatz verfolgt Lastbalancierung durch die optimale Zuordnung von Daten an berechnende *Worker*. Dabei kennt eine zentrale Instanz den Inhalt aller Caches, sodass einen nachfragenden *Worker* Daten für die Verarbeitung zugewiesen werden, die, wenn möglich, bereits in einem der Caches vorliegen.

In Viracocha kann vor allem durch das Caching benötigter Datensätze die Verarbeitungszeit deutlich verkürzt werden. Da es insbesondere bei großen Daten notwendig ist einzelne Dateien nur auf Anforderung zu laden, kommt dem Prefetching ebenfalls eine bedeutende Rolle zu. Die anderen beiden VDMS-Komponenten sind dagegen noch Gegenstand aktueller Arbeiten. So werden Ladestrategien erst dann interessant, wenn spezifische IO-Schnittstellen zur Verfügung stehen. Und der *Data Service*, der zwar bereits jetzt in der Lage ist parallele Berechnungen zu balancieren, kann wegen der notwendigen Zusatzkommunikation zwischen allen beteiligten Prozessen noch nicht zu einer deutlichen Steigerung des *Speed-ups* beitragen.

3.1 Markov-Prefetching

Die folgenden Betrachtungen beschränken sich auf einen Teilaspekt des VDMS, der sich mit Wahrscheinlichkeitsvorhersagen für das Prefetching von Datenblöcken beschäftigt. Der im VDMS integrierte Ansatz basiert auf Graphen, die angeben, mit welcher Wahrscheinlichkeit Blöcke als nächstes benötigt werden. Die Vorhersagen werden dabei auf Basis bisher abgefragter Werte getroffen. Dieser Ansatz wird auch als Markov-Prefetching bezeichnet [DouGru1999]. Hiefür definiert man einen Markov-*Predictor* i-ter Ordnung. Dieser durchsucht die bisherige Daten-Sequenz nach Fragmenten, die der Reihenfolge der i letzten Anfragen entspricht. Aufgrund der Daten, die den gefundenen Fragmenten folgen, wird eine Vorhersage für den

aktuellen Zeitpunkt getroffen. Die einfachste Variante der ersten Ordnung ermittelt das nächste Datum lediglich auf der Grundlage der letzten Anfrage. Abbildung 2 zeigt ein Beispiel.

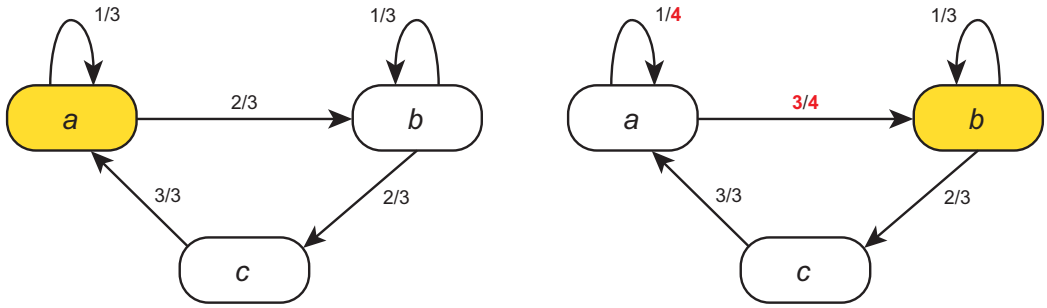

Abbildung 2: Wahrscheinlichkeitsgraph 1. Ordnung der Sequenz {*c*,*a*,*a*,*b*,*c*,*a*,*b*,*b*,*c*,*a*} (links), und nach der darauf folgenden Anfrage von *b* (rechts)

In Viracocha wird der Wahrscheinlichkeitsgraph zur Laufzeit konstruiert. Um aber bereits in der Anfangsphase, in welcher der Markov-Prefetcher noch keine oder ungenügende Informationen über Nachfolgerwahrscheinlichkeiten sammeln konnte, verwertbare Ergebnisse liefern zu können, wird in dieser Lernphase zusätzlich auf sequentielle Prefetcher zurückgegriffen. Das Markov-Prefetching wird dabei wie üblich durchgeführt. Kann der Markov-Prefetcher aufgrund fehlender Information keinen Nachfolger bestimmen, kommt automatisch z. B. die *One-Block-Lookahead*-Strategie (OBL) zum Einsatz. Bei der Analyse hat sich herausgestellt, dass diese Kombination sehr vielseitig und effizient sein kann.

3.2 Markov-Prefetching für das Particle Tracing

Der Haupteinsatzbereich des Markov-Prefetchers ist die Partikel-Verfolgung. Zum Startzeitpunkt der Partikel-Integration ist nicht bekannt, wie viele und in welcher Reihenfolge Datenblöcke geladen werden müssen. Daher erfolgt die Datenzuweisung dynamisch während der Berechnungslaufzeit, wobei der tatsächlich benötigte Block vom Algorithmus angefordert wird.

In der Lernphase, in der noch kein Wahrscheinlichkeitsgraph zur Verfügung steht, kommt neben der reinen Markov-Variante als alternative Strategie der OBL-Prefetcher zum Einsatz. Da ein Partikelverlauf nicht zwangsweise einer vorgegebenen Blockreihenfolge folgen muss, kann es dazu kommen, dass OBL vorwiegend ungeeignete Vorschläge für die Partikelintegration liefert. Dann bleibt das Prefetching während der Lernphase weiterhin wenig effektiv. Daher wurde in Viracocha die Möglichkeit eingebaut, die Lernphase durch einen initialen Wahrscheinlichkeitsgraphen zu ersetzen. Diese Initialisierung geht jedoch von einem Detailwissen über Einsatzdomäne und Datenstruktur aus, sodass die Algorithmusschicht von Viracocha den Markov-Prefetcher entsprechend mit Informationen versorgen muss. Folgende Ansätze sind hierfür implementiert worden:

Größe von Connection Windows: Multiblock-Datensätze weisen bereits Topologien auf, die auch als Graphen dargestellt werden können [GerSchi2005]. Ein nahe liegender Ansatz verwendet die Anzahl von Nachbarzellen der Verbindungsfenster zwischen zwei Blöcken

als Anhaltspunkt für Wahrscheinlichkeiten. Diese stellen die Kantengewichte im Graphen dar.

Stromlinien-Stichproben: Dieser Ansatz beruht ebenfalls auf der Multiblock-Topologie. Dabei werden die Wahrscheinlichkeiten, dass ein in einem Ausgangsdatenblock befindliches Partikel in dessen Nachbarblöcke gelangt, ermittelt. Hierfür werden in einem Präprozess eine Anzahl von Saatpunkten nach einer zuvor bestimmten Heuristik im zu untersuchenden Block verteilt und anschließend Stromlinien berechnet, bis diese den Block verlassen. Die IDs der betretenen Nachbarblöcke werden gezählt und für die Konstruktion des Graphen verwendet.

Zeitlevel-Gewichtung: Im instationären Fall, wenn beispielsweise Bahnlinien berechnet werden sollen, sind auch Blöcke des nächsten Zeitschrittes zu berücksichtigen. Der implementierte Ansatz verwendet trotzdem den Stromlinien-Initialisierungsgraphen, setzt dann jedoch eine zusätzliche Gewichtung für den nächsten Zeitschritt.

Zeitlevel-Sprung: Der Stromlinien-Initialisierungsgraph wird auch hier unverändert übernommen, jedoch wird die selektierte Block-ID dazu verwendet den Block vom bereits nächsten Zeitlevel zu laden. Abbildung 3 zeigt einen Ausschnitt aus dem Initialisierungsgraphen eines Multiblock-Motordatensatzes.

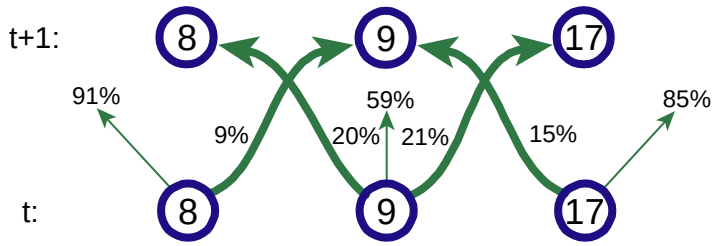

Abbildung 3: Der Markov-Prefetcher liefert beim Zeitlevel-Sprungverfahren immer einen Block vom nächsten Zeitlevel

4 Multiresolution und Streaming

Die Auslagerung des Postprocessings auf einen Hochleistungsrechner, Parallelisierungsstrategien und das Datenmanagement zielen hauptsächlich auf die Reduzierung der Gesamtlaufzeit. Diese Ansätze sind jedoch nicht in der Lage die Anforderungen des dritten Interaktionskriteriums der maximalen Sekundärreaktionszeit zu erfüllen. Selbst bei kleineren Problemgrößen stellen sich schnell Berechnungslaufzeiten ein, die sich über mehrere Sekunden bis hin zu einigen Minuten erstrecken können.

Es lässt sich jedoch beobachten, dass für eine interaktive Exploration durchaus schon vereinfachte approximative Ergebnisse ausreichen, um zu entscheiden entweder die Berechnung abzubrechen und Parameter für eine nächste Berechnungsiteration zu modifizieren, oder aber das

Endergebnis abzuwarten. Hinzu kommt, dass eine schnelle Präsentation von ersten Ergebnissen die subjektiv empfundene Wartezeit verkürzt. Auch wenn die visualisierten Teildaten noch keine aussagekräftigen Informationen enthalten sollten, wird die Verkürzung der Reaktionszeit als Steigerung von Interaktivität gewertet. Durch die Integration von Abbruch und Neustart des Kommandos mit modifizierten Parametern kommt man so einer explorativen Analyse schon sehr nahe. Die kontinuierliche Übertragung partieller Extraktionsergebnisse wird als *Data Streaming* bezeichnet. Entsprechende Ansätze lassen sich dabei in zwei grundsätzliche Gruppen unterteilen:

Teilergebnis-Streaming: Dies lässt sich dadurch erreichen, dass jeder Worker seine Datenblöcke bearbeitet und das jeweilige Gesamtergebnis zum *VisHost* sendet. Filigraner wird das Streaming, wenn bereits Teilergebnisse, die sich aus der Bearbeitung eines einzigen Datenblocks ergeben, gestreamt werden. Innerhalb der Blöcke lassen sich zusätzlich Zellen für das Streaming zusammenfassen. Beim Einsatz von Partikel-Verfolgung können dagegen einzelne Partikel-Positionen zu Streaming-Paketen gebündelt werden.

Ein anderer Ansatz, welcher Teilergebnisse für das Streaming produziert, verwendet den *Viewpoint* des Benutzers für eine *Front-to-Back*-Extraktion (*View Dependent*, VD). Hierfür muss allerdings der Ausgangsdatensatz in eine Baumstruktur überführt werden, die den Datensatz räumlich sortiert. In Viracocha wurde hierfür ein *Binary-Space-Partitioning*-Baum (BSP) verwendet.

Multiresolution-Streaming: Ein erster *Level-of-Detail*-Ansatz (LOD) verwendet einen durch Subsampling in mehrere Auflösungsstufen zerlegten Datensatz. Zuerst wird die niedrigste Stufe eingeladen, darauf Extraktionsberechnung durchgeführt und die Ergebnisse anschließend gestreamt. Danach werden die feiner aufgelösten Datensätze verwendet, bis der Originaldatensatz verarbeitet wurde.

Andererseits können auch die Extraktionsergebnisdaten als Multiresolution-Objekte aufbereitet und zum *VisHost* gestreamt werden. Jedoch sind diese *Top-Down*-Ansätze in der Regel nur für ein adaptives Rendering und nicht für eine schnelle Ergebnispräsentation geeignet. Daher sind progressive Extraktionsverfahren, die einen *Bottom-Up*-Ansatz verfolgen, gefordert. Ein Beispiel von progressivem Streaming ist in Viracocha mit auf *Subsampling* basierenden Schnittverfahren implementiert.

5 Ergebnisse

In den folgenden Testszenarios wurde zum einen ein Motordatensatz (*Engine*, 62 Zeitschritte, 23 Einzelblöcke, bis zu 201.434 Gitterpunkte / Zeitschritt) vom Aerodynamischen Institut der RWTH Aachen und zum anderen ein Turbinendatensatz (*Propfan*, 50 Zeitschritte, 144 Einzelblöcke, 2,533 Mio. Gitterpunkte / Zeitschritt) des DLR Köln verwendet. Als Parallelrechner kam eine SunFire-6800 mit 24 Prozessoren zum Einsatz.

5.1 Markov-Prefetching

Um eine Auskunft über das Verhalten des Markov-Prefetchers beim Extrahieren von Partikeln zu erhalten, wurden im Motordatensatz 64 Bahnlinien in neun aufeinander folgenden Zeitschritten berechnet. Hierfür kam ein adaptives Runge-Kutta-Verfahren 4. Ordnung zum Einsatz [GerSchi2005]. In der Initialisierungsphase wurde zusätzlich auf OBL zurückgegriffen. Für die Bewertung wurden zudem die Bahnlinien zum einen ohne aktivem VDMS (Messreihe „*Ohne*“) und zum anderen mit gecachten Datenblöcken (Messreihe „*Cached*“) berechnet. In Abbildung 4 sind die gemessenen Ergebnisse dargestellt.

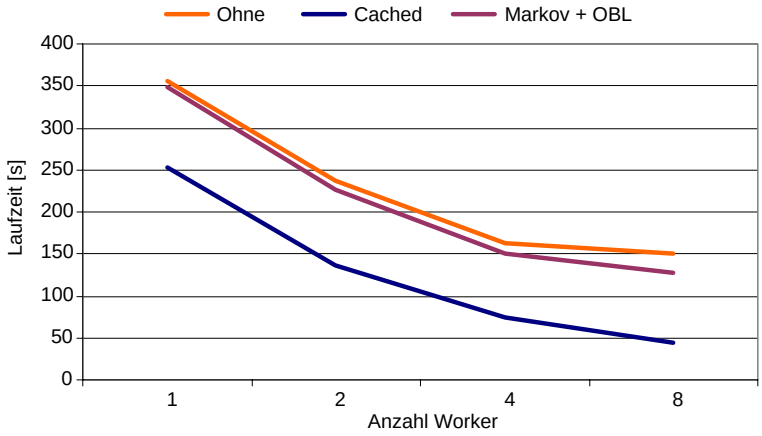

Abbildung 4: Berechnung von 64 Bahnlinien auf der SunFire 6800

Auffällig ist, dass trotz Verteilung der Berechnungslast die Ladelast fast konstant bleibt. Dadurch steigt dessen Anteil an der Gesamtlaufzeit von unter 30% auf über 70% an, sodass auch das Prefetching erheblich an Bedeutung gewinnt. Obwohl der Markov-Ansatz im seriellen Fall kaum Vorteile aufweist, steigt der Gewinn mit zunehmender Parallelisierung an. Die Gesamtlaufzeit wird im seriellen Fall um circa 2% reduziert und erreicht mit 8 *Workern* eine Einsparung von fast 15%. Die Ladelastreduktion verbessert sich von anfänglichen 6,8% auf 21,2%.

Alternative Markov-Ansätze, die Vorhersagen mithilfe von Initialisierungsgraphen treffen, haben sich in verschiedenen Testläufen noch nicht als ausgereift erwiesen. Dafür waren die erfassten Stichproben noch zu grob. Allerdings sind die hierdurch möglichen Erfolge auch wesentlich vom Datensatz abhängig. Während der Motordatensatz deutliche Turbulenzen aufweist und somit eine blockbasierte Strömungsrichtungstendenz häufig nicht eindeutig getroffen werden kann, sind Versuche mit anderen Datensätzen, die eine klare Strömungsrichtung aufweisen, viel versprechend.

5.2 Data Streaming

Für die Evaluierung von Streaming-Ansätzen wurden Isoflächen und Wirbelregionen nach der Lambda-2-Methode [JeoHu1995] extrahiert. Die Messreihen *Simple Iso* und *Simple Vortex* zeigen das Parallelisierungsverhalten ohne Streaming auf. Als Streaming-Varianten kamen mit *Viewer Iso* eine betrachteroptimierte Isoflächenextraktion und mit *Streamed Vortex* ein

blockweises Streaming von Wirbelregionen zum Einsatz. Die Streaming-Varianten arbeiteten im Gegensatz zu den Standardparallelisierungsansätzen mit gefüllten Caches, sodass die Auswirkungen der Lade-, Caching- und Prefetching-Strategien nicht Gegenstand der Untersuchung sind. Ergänzend wurden zwei weitere Messungen *Iso DataMan* und *Vortex DataMan* durchgeführt, die mit aktivem VDMS (vollständig gecachten Daten) aber ohne Streaming arbeiteten. Da somit fast ausschließlich die reine Berechnungszeit erfasst wurde, liegt hier eine untere Laufzeitschranke vor, die gut zur Bewertung der Streaming-Ansätze dienen kann.

Die Gesamtlaufzeiten zur Berechnung von Isoflächen sind in Abbildung 5 gegeben. Der Einfluss des Cachings von Daten ist beim Vergleich der Standardverfahren ohne und mit VDMS deutlich zu erkennen. Vergleicht man daneben den betrachterabhängigen Streaming-Ansatz *Viewer Iso* mit der Variante *Iso DataMan*, so stellt man eine um ein Vielfaches erhöhte Laufzeit fest. Dies ist der Mehraufwand, den man für das Streaming aufbringen muss. Der Anstieg lässt sich hauptsächlich durch die Online-Konstruktion des BSP-Baums für den VD-Ansatz begründen.

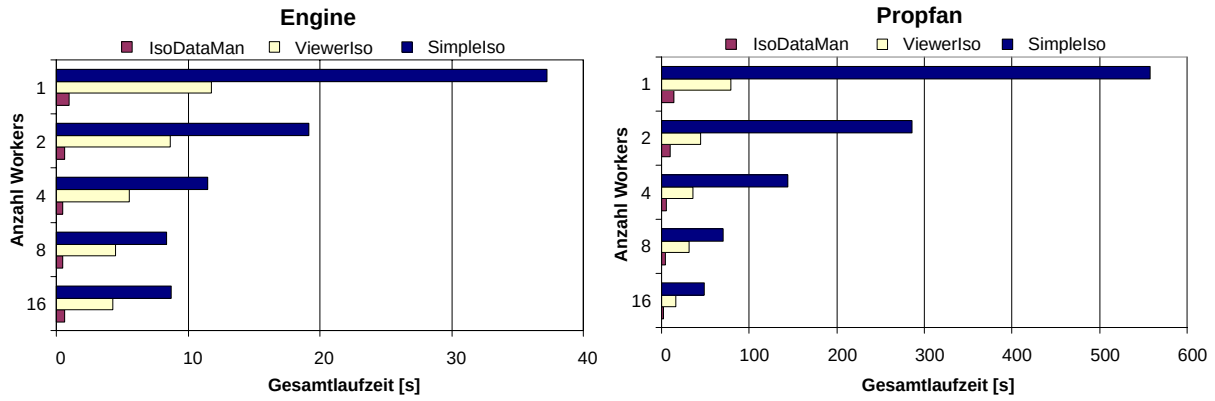

Abbildung 5: Gesamtberechnungszeiten für Isoflächen, Engine (links) und Propfan (rechts)

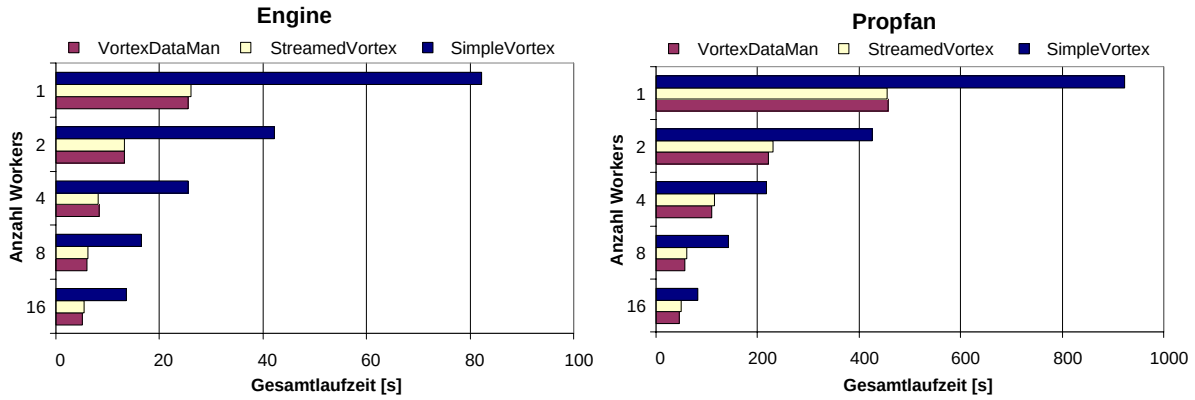

Abbildung 6: Gesamtberechnungszeiten für Wirbel, Engine (links) und Propfan (rechts)

In Abbildung 6 sind die Ergebnisse für die deutlich aufwändigeren Berechnungen von Wirbelregionen zusammengefasst. Die durch den Berechnungsalgorithmus entstehende höhere Last ist direkt ablesbar, wenn man das *Simple-Vortex*-Ergebnis mit der *Simple-Iso*-Laufzeitmessung in Abbildung 5 für den Fall, dass nur ein *Worker* arbeitet, vergleicht. Aber

auch die beiden anderen Verfahren brauchen nun deutlich länger. Betrachtet man *Vortex DataMan* wiederum als untere Schranke für die Gesamtlaufzeit, dann fällt auf, dass der *Streamed Vortex* fast gleich auf ist. Im Gegensatz zur vorher betrachteten Streaming-Version *Viewer Iso* werden hier keine zusätzlichen Baumstrukturen erzeugt. Der Vergleich gibt somit die reinen Kommunikationskosten wieder. Bedingt durch den hohen Berechnungsaufwand ist die zusätzliche Streaming-Kommunikation aber kaum noch wahrnehmbar.

Die Latenzzeiten, d. h. die Zeit bis zum Eintreffen der ersten Teildaten beim *VisHost*, wurden ausschließlich mit der Turbine durchgemessen. Die Latenzzeiten für den VDMS-verwendenden Standardextraktionsalgorithmus sind mit seiner Gesamtlaufzeit (vgl. Abbildung 5 rechts) identisch. Wie die Ergebnisse in Abbildung 7 zeigen, erfüllt der auf den VDMS setzende betrachteroptimierte Algorithmus die in ihn gesetzten Erwartungen. Bereits ab zwei *Workern* können innerhalb einer Sekunde die ersten *View-Dependent*-Isoflächenteile präsentiert werden. Werden die Berechnungen komplexer, dann steigt erwartungsgemäß die Latenzzeit. Dennoch liegen auch die Wartezeiten für den *Streamed Vortex* immer noch deutlich unter fünf Sekunden.

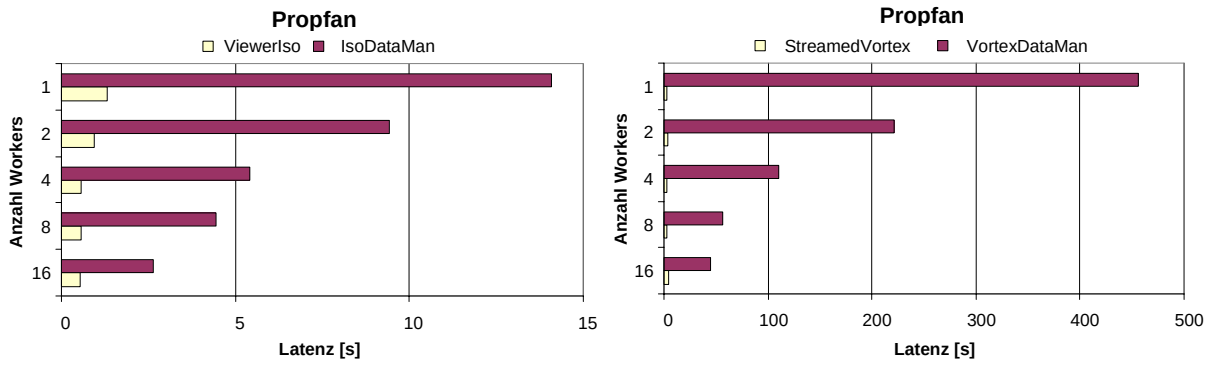

Abbildung 7: Latenzzeiten für die Isoflächen- (links) und Wirbel-Extraktion (rechts) unter Verwendung des Turbinendatensatzes

Zusammenfassend können beide hier untersuchten Streaming-Verfahren überzeugen. Der einfachere der beiden kommt gut mit der zusätzlichen Kommunikation zurecht. Aber auch der VD-Streaming-Algorithmus schneidet gut ab. Der Verzicht, den BSP-Baum als Offline-Datei einzulesen, ermöglichte die Untersuchung der vollständig anfallenden Streaming-Kosten. Angesichts dessen sind die Laufzeiten durchaus zufrieden stellend und die Latenzzeiten im gewünschten Bereich.

6 Zusammenfassung

Wesentliche Motivation des vorgestellten Postprocessing-Frameworks ist die echtzeitfähige Exploration, die durch die in der Einleitung definierten Interaktionskriterien bewertet wird. Mit der verteilten Architektur des entwickelten Postprocessing-Systems konnte das erste Kriterium bereits weitestgehend erfüllt werden. Im Zentrum der Bewertung stand jedoch vor allem das dritte Interaktionskriterium, das die Reaktionszeit des Postprocessing-Systems beurteilt. Um

diese auch als Latenzzeit definierte Größe zu reduzieren, wurde primär auf einen *Speed-up* durch Parallelisierung gesetzt. Der Datenmanager von Viracocha konnte durch Caching und durch eine verbesserte Balancierung, basierend auf Prefetching-Ansätzen, eine weitere Beschleunigung bewirken. Den bedeutendsten Einfluss auf die Reaktionszeit hat jedoch das integrierte Streaming.

7 Literatur

[BryGe1992] Bryson, S., Gerald-Yamasaki, M.: The Distributed Virtual Windtunnel, Proceedings, Conference on Supercomputing, Minneapolis, MN, S. 275 – 284, 1992.

[DamFo2000] van Dam, A., Forsberg, A., Laidlaw, D., LaViola, J., Simpson, R.: Immersive Virtual Reality for Scientific Visualization: A Progress Report, IEEE Computer Graphics and Application, 20 (6), S. 26 – 52, November / Dezember 2000.

[DouGru1999] Doug, J., Grunwald, D.: Prefetching Using Markov Predictors, IEEE Transaction on Computers, 48 (2), S. 121 – 133, 1999.

[GriLo1989] Grimsrud, A., Gray Lorig, G.: Implementing a Distributed Process between Workstation and Supercomputer, Proceedings, 1. International Conference on Applications of Supercomputers in Engineering, Southampton, S. 133 – 144, September 1989.

[GerSchi2005] Gerndt, A., Schirski, M., Kuhlen, T., Bischof, C.: Parallel Calculation of Accurate Path Lines in Virtual Environments through Exploitation of Multi-Block CFD Data Set Topology, Journal of Mathematical Modelling and Algorithms, 4 (1), S. 35 – 52, März 2005.

[JeoHu1995] Jeong, J., Hussain, F.: On Identification of a Vortex, Journal of Fluid Mechanics, Vol. 285, S. 69 – 94, 1995.

[KuhGe2005] Kuhlen, T., Gerndt, A., Assenmacher, I., Hentschel, B., Schirski, M., Wolter, M., Bischof, C.: Analysis of Flow Phenomena in Virtual Environments – Benefits, Challenges and Solutions, Proceedings, 11. International Conference on Human Computer Interaction, HCII 2005, Las Vegas, NV, Juli 2005.

[RanFra1998] Rantzau, D., Frank, K., Lang, U., Rainer, D., Wössner, U: COVISE in the CUBE: An Environment for Analyzing Large and Complex Simulation Data, Proceedings, 2. Workshop on Immersive Projection Technology, IPT98, Ames, IA, 1998.