

Paralleles Raytracen auf einem heterogenen PVM-Cluster

Christian Plattner, Christian Gloor*

20. März 2001

Zusammenfassung

Die Vorlesung "Computer System Benchmarking and Performance Analysis" an der ETH Zürich, gelesen von Prof. Thomas Stricker, beschränkt sich nicht auf das Vermitteln von theoretischen Inhalten, sondern fördert auch die Einsicht der Studenten in das praktische Benchmarking, indem die Studenten anhand eines kleinen Projektes die in der Vorlesung vorgestellten Techniken anwenden sollen.

Wir haben diese Vorlesung im Wintersemester 2000/2001 besucht und uns im Rahmen eines solchen Mini-Projektes entschlossen, Performancemessungen des Programmes *Povray* unter dessen Ausführung auf einem *Linux-Cluster* zu betrachten. *Povray* ist ein sogenannter Raytracer, welcher aus einer textuellen Beschreibung von Objekten, Lichtquellen und Betrachtungsposition fotorealistische Bilder produziert. Da dies ein rechenintensiver Vorgang ist, der auf einem einzelnen, separierten Computer je nach Komplexität des zu berechnenden Bildes sehr lange dauern kann (bis zu mehreren Stunden oder gar Tagen), liegt die Idee nahe, diese Arbeit auf mehrere Computer zu verteilen.

1 Einleitung

Christian Gloor arbeitet für die Computerschule *DIGICOMP AG*. Die Computerschule *DIGICOMP* besitzt in Zürich 27 Schulungsräume, in welchen jeder Arbeitsplatz mit einem eigenen Personal Computer ausgestattet ist. Im Schnitt besitzt jeder dieser Räume 12 Arbeitsstationen.

Die Schulungsräume werden von Montag bis Freitag ganztags für Seminare genutzt, an Samstagen nur teilweise und an Sonn- und Feiertagen gar nicht. Da es sich bei diesen Rechnern um eine geballte Ladung Rechenpower handelt und diese Rechenleistung während der Nacht und an Wochenenden quasi brachliegt, kam Christian Gloor auf die Idee, diese Rechner in den Randstunden mittels des Betriebssystems Linux zu einem Cluster zusammenzufügen um so ihre Rechenleistung möglichst gebündelt für eigene Experimente nutzen zu können.

Der Grundgedanke des verteilten *Povray* Rechnens stammt aus einem Anwendungsbeispiel im Buch "Linux Clusters" [SPE]. Die ersten Experimente mit dem Cluster führte Christian Gloor im Dezember 2000 statt, Mitte Januar 2001 fand sich dann auch Christian Plattner dazu. Am Cluster wurde vor allem an den Wochenenden gearbeitet, da dann einerseits ein ungestörtes Arbeiten möglich war und andererseits die Schulungsräume vollständig für den Clusterbetrieb zur Verfügung standen.

2 Physikalischer Aufbau des Netzwerks/Clusters

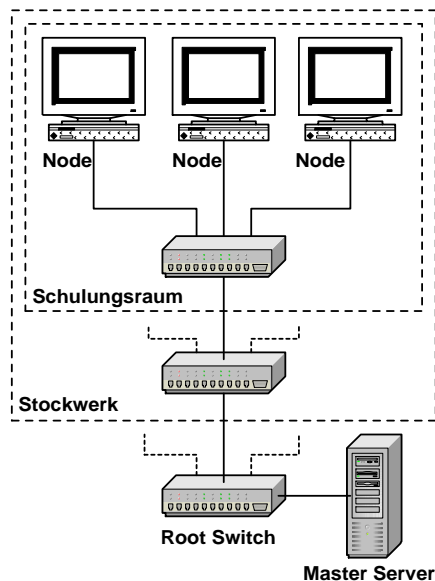
In unserem Cluster sind 3 Komponenten zu unterscheiden:

- Nodes (auch Clients genannt), die Rechner in den Schulungsräumen
- Master Server, Rechner welcher u.a. die Aufträge an die Nodes verteilt

*cplattne@iic.ethz.ch, cgloor@iic.ethz.ch

- Netzwerk, verbindet den Master Server mit den Nodes

Die Schulungsräume der Firma DIGICOMP befinden sich in verschiedenen Stockwerken des Gebäudes. Jeder Raum verfügt über einen eigenen Switch (jeder PC ist also direkt an einen Switch angeschlossen) und auf jedem Stockwerk befindet sich wiederum ein Switch, welcher die verschiedenen Räume eines Stockwerks bedient. Diese Art der Vernetzung wird auch auf Gebäudeebene fortgesetzt, es existiert folglich ein Root-Switch, welcher die verschiedenen Stockwerke miteinander verbindet. Sämtliche Netzwerkverbindungen sind in 100Mbit full-duplex ausgeführt.



Der Masterserver im Cluster, also der Rechner, welcher die Aufträge an die Nodes verteilt, ist direkt an den Root Switch angeschlossen. Da dieser nicht an der Berechnung des Bildes teilnimmt (in der Theorie, siehe Bemerkung weiter unten bei der zweiten Messung) und nur für die Verteilung der Aufgabe zuständig ist, reicht ein Pentium-I, welcher mit 166 MHz getaktet und mit 192 RAM ausgestattet ist, vollkommen aus.

Bei den verwendeten PCs in den Schulungsräumen handelt es sich um Personal Computer der Marke *Compaq*, alle sind mit mindestens 128MB RAM ausgestattet. Sie besitzen einen Pentium-II oder Pentium-III Prozessor mit Taktfrequenzen zwischen 350 MHz und 866 MHz. Jeder PC verfügt ausserdem über ein CDROM-Laufwerk. Pro Schulungsraum sind alle Arbeitsplätze mit identischen Computern ausgestattet.

3 Die Cluster-Software

Da die Node-Rechner normalerweise als Schulungs-PCs verwendet werden, war es uns nicht möglich (bzw. nicht erlaubt), Software auf der Harddisk dieser Systeme zu installieren. Die Verwendung des BOOTP-Protokolls war leider aus technischen Gründen nicht möglich. Man entschied sich deshalb für eine Lösung basierend auf CDROMs; glücklicherweise unterstützen die verwendeten Compaq PCs das Booten von CDROM problemlos.

Die bootfähigkeit der verwendeten CDROM (im folgenden einfach CD genannt) wurde durch die Verwendung des sogenannten "El-Torito" Formats erreicht. CDs, welche in diesem Format beschrieben werden, enthalten folgende zwei Komponenten: Ein Bootfloppy Image und ein normales ISO-9660 Filesystem.

In einer ersten Version dieser CD enthielt das ISO-9660 Filesystem eine abgemagerte (bzw. auf unter 650MB reduzierte) und leicht modifizierte RedHat Linux 7.0 Installation [Redhat], die

Bootfloppy enthielt einen Linuxkernel (Version 2.2.16) und eine initiale Ramdisk (initrd) mit einem von uns erstellten Startskript (sogenanntes linuxrc).

Mit Hilfe dieser CD konnte zwar schon experimentiert werden, sie hatte aber zwei grundlegende Probleme:

1. Da das Root-Filesystem von CD gemountet wurde, musste die CD waehrend des Clusterbetriebes im Node Rechner verbleiben: Folglich hätte man also fuer einen grossen Cluster sehr viele CDs benoetigt.
2. Alle Instruktionen für den Node Rechner waren auf der CD abgelegt - Bei jeder Änderung der Clusteraufgabe hätte also eine neue CD (bzw. ein ganzes Set von CDs) gebrannt werden müssen.

Da dies kein tragbarer Zustand war, wurde viel Aufwand in die Herstellung einer "flexiblen" CD gesteckt.

Die neuste Version der CD (mittlerweile die 42. Version, alle fehlgeschlagenen Exemplare mitgezählt) verfügt - unter anderem - über folgende Features:

- Das CD ISO-9660 Filesystem wird nicht als Root-Filesystem gemountet, sondern in eine Ramdisk kopiert (welche das Root-Filesystem darstellt) - Das ISO-9660 Filesystem ist mittlerweile gerade noch 52 MB gross.
- Nach dem Kopieren des CD-Filesystems wird die Schublade des CD-Drives automatisch geöffnet - der Operator des Clusters kann die CD entnehmen und in eine andere (potentielle) Node Maschine einlegen. So kann mit Hilfe von nur etwa 15 CD-Kopien ein grosser Cluster manuell in sehr kurzer Zeit hochgefahren werden.
- Nach dem Hochfahren ladet der Node ein Shellskript vom Masterserver herunter (mittles des Programmes *wget*). Dieses steuert die weiteren Aktionen des Nodes. Man erreicht so eine grosse Flexibilität - die CD bleibt hochgradig wiederverwendbar.
- Auf dem CD-Filesystem befinden sich nur noch die nötigsten Binaries. Das /usr Dateisystem wird vom Masterserver mittels NFS gemountet (als Aktion innerhalb des vom Masterserver geladenen Shellskriptes).
- Durch einen beliebigen Tastendruck wird der Node neu gestartet (der Bildschirm enthält nur die Meldung "Press any key"). Dies hat den Vorteil, dass der Cluster nie manuell heruntergefahren werden muss: Falls sich ein Kursteilnehmer am Montag Morgen an einen PC setzt, löst er (unwissentlich) einen Reboot des Rechners aus.

Als Kernel für die Nodes verwenden wir die Version 2.4.1 [Kernel].

Der Masterserver ist eine Redhat Linux 7.0 Installation mit Kernel 2.2.16. Bei dem von ihm mittels NFS exportierten /usr Filesystem handelt es sich um sein eigenes /usr Directory. Um eine Uebersicht über alle im Cluster vorhandenen (aktiven) Nodes zu erhalten, wird eine Postgres Datenbank verwendet [PGSQL]. Die Nodes werden anhand von ihrer IP-Adresse, welche sie mittels DHCP bezogen haben, identifiziert. Der DHCP Server gehört zur vorhandenen Infrastruktur. Die Nodes tragen sich in dieser Datenbank periodisch ein - momentan alle 5 Minuten, um die Netzbelastung gering zu halten. So kann leicht festgestellt werden, welche Nodes noch aktiv oder neu hinzugekommen sind. In dieser Datenbank kann auch für jeden Node festgelegt werden, ob er einen neuen Auftrag hat. Der Node prüft deshalb beim periodischen Update ein Flag in der Datenbank und lädt sich gegebenenfalls ein neues Shellskript mit auszuführenden Kommandos herunter.

4 Povray im Cluster

Die Povray Raytracing Software wurde für den Betrieb auf Einzelplatzmaschinen entwickelt. Dank eines Patches [PPV] kann man die Software unter Zuhilfenahme von PVM [PVM] auch auf einem Cluster laufen lassen. Dabei wird das zu berechnende Bild in Teilbereiche eingeteilt, welche von verschiedenen Nodes unabhängig voneinander berechnet werden können. Der Masterrechner teilt dazu den Nodes die Teilaufgaben zu und setzt am Ende des Berechnungsvorganges die Resultate aller beteiligten Nodes zu einem Gesamtbild zusammen.

Im Clusterbetrieb interessieren vor allem zwei Parameter der PVM/Povray Konfiguration:

- Anzahl der Node Rechner, welche an einer Bildberechnung beteiligt sind
- Art der Aufteilung des zu berechnenden Bildes in Fragmente, bzw. Jobs für die beteiligten Nodes

Man muss hier anmerken, dass die Anzahl der Fragmente keinesfalls der Anzahl der beteiligten Nodes entspricht - Im generellen Fall ist die Anzahl der Fragmente viel grösser als die der an der Berechnung beteiligten Nodes. Ein Node verarbeitet folgedessen während der Berechnung eines Bildes im allgemeinen mehrere Fragmente.

Es ist grundsätzlich damit zu rechnen, dass die Netzbelastung und der administrative Aufwand seitens des Masterservers mit der Anzahl der Clients und mit kleineren Fragmenten stark ansteigt. Phänomene die auftreten könnten sind Paketverluste bzw. Paketstau in den Switches hervorgerufen durch folgende zwei Ursachen:

- Eine grosse Anzahl von Rechnern welche gleichzeitig Daten an den Masterserver zurückschicken
- Eine generelle höhere Kommunikationsrate durch die Verwendung von kleineren Fragmenten

Ob diese Phänomene tatsächlich auftreten würden, war Gegenstand unserer Messungen, welche im folgenden beschrieben werden.

5 Messungen

Die Messungen am Cluster wurden alle an einem Wochenende durchgeführt, so konnte sichergestellt werden, dass keine anderen Benutzer das Netzwerk belasten würden.

Für alle Messreihen wurde dasselbe zu berechnende Bild verwendet (siehe Anhang).

5.1 1. Messreihe

Die der Messreihe zugrundeliegende Fragestellung lautet: Wie gut ist das Raytracen skalierbar, d.h. wie weit lässt sich durch Erhöhen der Anzahl der Rechner das Ausführungstempo steigern?

Bei dieser Messung wurden nur die Pentium-III 866 Mhz Rechner verwendet. Das Bild wurde immer in 16x16 Pixel grosse Fragmente unterteilt. Es wurden pro Anzahl Rechner 3 Messungen durchgeführt. In die nachfolgende Tabelle wurde jeweils die *niedrigste* Laufzeit übernommen - die Werte stellen also eine *obere Schranke* fuer die Zeit dar, welche der Cluster im Idealfall für die Berechnung in einer bestimmten Konfiguration braucht. Wir geben zu, dass diese Werte nicht die durchschnittliche Leistung des Clusters wiedergeben, allerdings bekommt man so eine gute Abschätzung wie gut die maximale Performance des Clusters in einer bestimmten Konfiguration ist.

Ein grosses Problem mit PVM sind "Ausreisser" die auf ein zu spätes Abliefern eines der letzten Pakete (Teil) zurückzuführen sind. Aus uns unbekanntem Gründen kommt dies beim Einsatz von PVM ab und zu vor. Verzögert sich ein beliebiges Fragment in der Mitte der Berechnung, wirkt sich das lediglich zu $\frac{1}{n}$ (mit der Anzahl Maschinen n) aus, weil die anderen Maschinen noch am

rechnen sind. Nicht so am Schluss, wenn alle anderen bereits warten. Der entstehende Delay kann sich im Sekundenbereich auswirken!

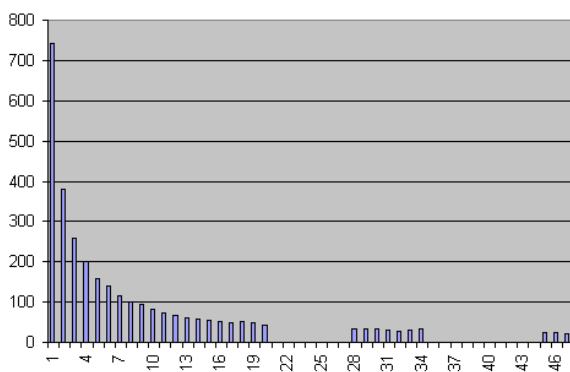
Ohne diese "Ausreisser" sind die Messergebnisse einer Konfiguration mit einer bestimmten Anzahl Nodes sehr konstant (auf die Sekunde genau). Leider konnten die Werte nur auf die Sekunde genau erfasst werden.

Beim betrachten der Messwerte in der Tabelle fällt auf, dass nicht alle Nodeanzahlen von 1 bis 47 durchgeprüft wurden. Dies hat seinen Grund in der PVM Software: Es ist zum Teil nicht möglich, gewisse Nodesets zu verwenden, da PVM eigenmächtig einzelne Nodes ignoriert und nur eine Teilmenge der von uns angegebenen Nodeliste zur Anvisierung von Teils verwendet. Aus Zeitgründen konnte auf dieses Phänomen nicht weiter eingegangen werden.

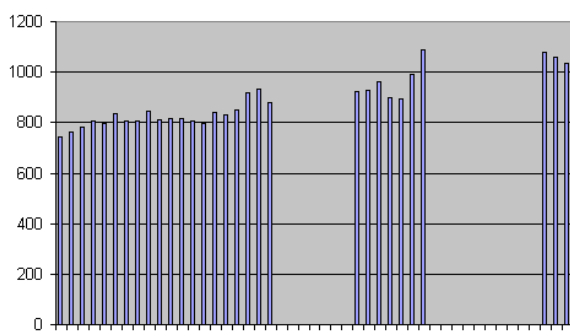
Für die Ermittlung der Werte wurden nur die Pentium-III Rechner mit 866MHz verwendet.

Anzahl Nodes	1	2	3	4	5	6	7	8	9	10
Zeit in s	741	381	260	202	159	139	115	101	94	81
Anzahl · Zeit	741	762	780	808	795	834	805	808	846	810
Anzahl Nodes	11	12	13	14	15	16	17	18	19	20
Zeit in s	74	68	62	57	56	52	50	51	49	44
Anzahl · Zeit	814	816	806	798	840	832	850	918	931	880
Anzahl Nodes	28	29	30	31	32	33	34	45	46	47
Zeit in s	33	32	32	29	28	30	32	24	23	22
Anzahl · Zeit	924	928	960	899	896	990	1088	1080	1058	1034

Es folgen die grafischen Darstellungen der Messdaten:



X-Achse: Anzahl Nodes, Y-Achse: Zeit in [s]



X-Achse: Anzahl Nodes, Y-Achse: Gesamtzeit

5.2 Interpretation der 1. Messreihe

Der erhoffte Zusammenbruch der Performance ab einem gewissen Parallelitätsgrad ist leider nicht eingetroffen, der Cluster skaliert (wenn auch nicht mehr allzu überragend mit steigender Node Anzahl) in dem von uns gemessenen Bereich. Der von uns erhoffte sichtbare Knick tritt leider

nicht auf. Wir gehen jedoch davon aus, dass in einer zukünftigen noch zu tätigenen Messung dieser Knick ab einer gewissen Anzahl Clients auftreten wird und somit auch nachgewiesen werden kann. Aus Zeitgründen musste auf eine weitergehende Messung verzichtet werden.

Der berechnete "Anzahl · Zeit" Wert ist ziemlich aufschlussreich, gibt er doch ein gewisses Kostenmass für die Berechnung an (Akkumulierte Rechenzeit: So lange waren die beteiligten Nodes blockiert für die Berechnung des Bildes). Solange man weniger als 16 Nodes benützt, pendelt der Wert im Bereich von 800 Sekunden, ab ca. 16 Nodes jedoch steigt er an. Falls man für jede Sekunde Rechenzeit im Cluster zahlen müsste, wäre man wohl unter diesen Umständen kaum bereit eine beliebige Parallelisierung zu bezahlen, da die Kosten mit ansteigender Anzahl Clients schnell steigen und sich der dazugewonnene Performancegewinn in Grenzen hält.

5.3 2. Messreihe

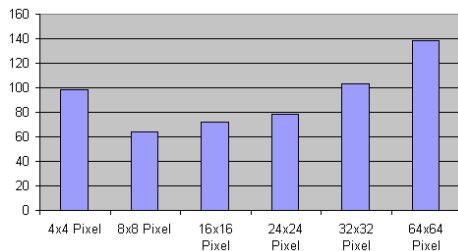
Die der Messreihe zugrundeliegende Fragestellung lautet: Bei konstanter Clusterkonfiguration (gleiche Nodes), wie verhält sich die Performance in Abhängigkeit von der gewählten Fragmentgrösse?

Leider war es aus zeitlichen Gründen nicht mehr möglich, die 4x4 Messung auf der zweiten Clusterkonfiguration durchzuführen.

21 PCs, 10 mit je 866 MHz und 11 mit je 350 MHz

Unterteilung	∅	1	2	3	4	5
4 · 4 Pixel	98.6s	101s	95s	97s	102s	98s
8 · 8 Pixel	64s	66s	67s	67s	66s	58s
16 · 16 Pixel	71.8s	75s	81s	62s	87s	54s
24 · 24 Pixel	78.6s	83s	53s	103s	79s	75s
32 · 32 Pixel	103.2s	82s	117s	118s	117s	82s
64 · 64 Pixel	138.2s	133s	173s	120s	146s	119s

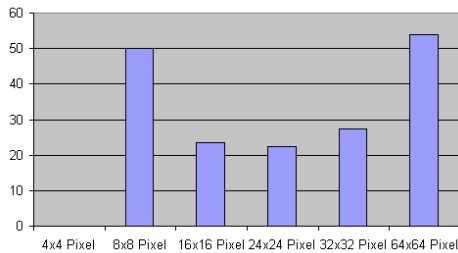
Grafische Repräsentation:



46 PCs, 27 mit je 866 MHz und 19 mit je 733MHz

Unterteilung	∅	1	2	3	4	5
4 · 4 Pixel	—	—	—	—	—	—
8 · 8 Pixel	50s	77s	31s	66s	45s	31s
16 · 16 Pixel	23.6s	23s	24s	22s	25s	24s
24 · 24 Pixel	22.4s	37s	22s	24s	25s	24s
32 · 32 Pixel	27.4s	23s	30s	26s	32s	26s
64 · 64 Pixel	53.8s	51s	52s	64s	49s	53s

Grafische Repräsentation:



5.4 Interpretation der 2. Messreihe

Obwohl in der zweiten Messreihe die 4x4 Messung nicht durchgeführt wurde, ist trotzdem sichtbar, dass bei verschiedenen Clusterzusammensetzungen die optimale Fragmentgrösse variiert. Interessant ist auch, dass die gemessenen Zeiten bei den Fragmentgrössen welche in der Nähe des Optimums liegen, jeweils am konstantesten sind.

Eigentlich wäre es zu erwarten gewesen, dass die Rechenzeit etwa umgekehrt proportional zur Grösse der Fragmente ist, da mit abnehmender Fragmentgrösse der Kommunikationsaufwand steigt. Allerdings dauern Berechnungen ab einer bestimmten Fragmentgrösse wieder zunehmend länger:

Der Masterserver verteilt jedem Client ein Fragment zur Berechnung, ist dieser damit fertig, erhält er ein neues, dies wiederholt sich solange bis alle abgearbeitet sind. So scheidet gegen den Schluss des Berechnungsvorganges ein Client nach dem anderen aus, bis der letzte sein Fragment abgeliefert hat. Aufgrund der heterogenität des Clusters ist es dem Masterserver nicht möglich, dafür zu sorgen, dass alle Clients exakt zur selben Zeit fertig werden. So warten am Schluss alle Rechner auf ein paar wenige (meistens nur einen), die noch am Rechnen sind.

Es wurde in der PVM-Implementation versucht, diesen Effekt auszugleichen, indem man zu jedem Rechner seine Leistung angeben kann, damit der Scheduler diese berücksichtigen kann. Allerdings ist diese Implementation fehlerhaft und wird vom PVM-Team anscheinend nicht mehr weiter verfolgt.

Eine andere Möglichkeit wäre die dynamische Anpassung der Fragmentgrösse. Zu Beginn der Berechnung könnten grosse Fragmente verteilt werden, gegen den Schluss kleinere.

Hier ein Beispiel einer Verteilung:

```
PVM Task Distribution Statistics:
  host name [ done ] [ late ]      host name [ done ] [ late ]
  172.16.6.121 [ 5.00% ] [ 0.00% ]  172.16.3.246 [ 5.00% ] [ 2.83% ]
  172.16.3.179 [ 6.67% ] [ 2.75% ]  172.16.2.64 [ 7.00% ] [ 2.58% ]
  172.16.2.60 [ 5.33% ] [ 2.75% ]  172.16.3.183 [ 5.00% ] [ 0.00% ]
  172.16.2.56 [ 5.33% ] [ 2.83% ]  172.16.2.57 [ 6.33% ] [ 2.83% ]
  172.16.2.59 [ 6.00% ] [ 2.83% ]  172.16.2.219 [ 6.00% ] [ 2.83% ]
  172.16.7.96 [ 5.00% ] [ 0.00% ]  172.16.2.52 [ 7.33% ] [ 2.67% ]
  172.16.4.104 [ 4.33% ] [ 0.00% ]  172.16.7.98 [ 5.33% ] [ 2.83% ]
  172.16.4.103 [ 6.00% ] [ 2.50% ]  172.16.5.101 [ 7.00% ] [ 2.67% ]
  172.16.2.129 [ 6.67% ] [ 2.92% ]  172.16.0.45 [ 0.67% ] [ 0.08% ]
```

Es ist zu sehen, dass die Rechner gleichmässig belastet werden. Der letzte, 172.16.0.45, ist der Masterserver, welcher mit einer wesentlich schlechteren CPU ausgestattet ist. Es ist aufgrund eines Fehlers in der PVM-Implementierung nicht möglich gewesen, diesen aus der Konfiguration zu entfernen. Auch dies ist etwas, dass in zukünftigen Messungen unbedingt verbessert werden muss.

6 Zusammenfassung

Das Ziel dieser Arbeit das Gewinnen von Daten über Povray im Cluster Umfeld und deren Auswertung und Interpretierung. Es muss jedoch gesagt werden, dass der grösste Teil der benötigten Zeit nicht für die Messungen und deren Interpretation investiert wurde, sondern für das Bereitstellen der ganzen Client-Software Infrastruktur und Messanordnung (insgesamt mehrere Wochenende).

Zwar existiert einiges an Literatur über den Aufbau von Clustern, jedoch sind diese Dokumente meist nur teilweise von Nutzen, gibt es doch nicht "den Standard-Cluster".

Es musste sehr viel Zeit in die Herstellung der Boot-CD investiert werden. Dies war ein mühsamer Vorgang, welcher nicht im geringsten etwas mit der folgenden Messung zu tun hatte. Mühsam war dieser Vorgang vorallem, weil die Herstellung einer solchen Boot-CD nicht etwas ist, was die grosse Masse der Linux-User interessiert und deshalb das Angebot an Dokumentation nicht sehr vielfältig ist. Ein grosser Effort musste in die Erkundung des Boot-Vorganges gesteckt werden, da die Benutzung der Linux Initial Ramdisk nicht ganz ohne Fallstricke ist. Da die CD derart gestaltet wurde, dass der Cluster mit beliebigen Aufgaben betreut werden kann, wird sie also auch in Zukunft von Nutzen sein und ist nicht einfach ein Abfallprodukt dieses Projektes.

Die durch unsere Messungen erhaltenen Resultate sind nicht gerade atemberaubend, haben uns aber ein besseres "Feeling" für die Konfiguration des Clusters gegeben. Es hat sich klar gezeigt, dass blindes Parallelisieren und das heruntersetzen der Fragmentgrösse auf sehr tiefe Werte nicht unbedingt den ultimativen Performancegewinn darstellen.

7 Ausblick

Leider fehlte uns eine genaue Zeitplanung - wir müssen zugeben, dass durch unser Vorgehen der eigentliche Messvorgang (und die anschliessende Interpretation) unter grossem Zeitdruck stattgefunden hat und deshalb nicht in dem Ausmass ausgeführt werden konnte, wie wir uns dies gewünscht haetten. Zukünftigen Messvorgängen am Cluster muss unbedingt eine bessere Zeitplanung vorausgehen.

Der Lerneffekt bezüglich des Clusters und seines Handlings allerdings darf als maximiert betrachtet werden. Da wir uns sehr für Cluster interessieren, werden wir auch in Zukunft von den hier erlernten Techniken Gebrauch machen können.

In zukünftigen Versuchen mit dem Cluster würden uns vorallem folgende Konstellationen interessieren:

- Performance Messungen mit noch mehr PCs. Fragestellung: Ab wann skaliert das System (in Bezug auf die Performance) überhaupt nicht mehr?
- Einfluss der Zusammensetzung des Clusters auf die Performance. Dabei ginge es darum zu erfahren, ob sich der Einsatz von langsamen Rechnern in bestimmten Situationen nicht negativ auswirken würde bzw. ob man sie nicht lieber weglassen sollte.
- Aufbau eines extrem verteilten Clusters mit Hilfe aller PCs der Firma DIGICOMP, nicht nur jene am Standort Zürich. Die Fragestellung hierbei ist, wie sich das WAN (Wide Area Network) auf die Gesamtperformance auswirkt.

8 Anhang

8.1 Berechnetes Bild

8.1.1 Source

Um die Nodes des Clusters möglichst stark zu belasten, wurde ein Bild mit hoher Komplexität gewählt.

```
#include "colors.inc"
#include "metals.inc"
#include "textures.inc"
#include "finish.inc"

camera {location <1.5,1.6, -1.2> look_at <0.5, 0.3, 0.4> }

union{
    #include "kugel_crash1.inc"
    pigment { P_Chrome3 }
```

```

        finish { F_MetalE }
    }

media { intervals 10 scattering { 1, rgb<0.0.7,0> } samples 1, 10 density { turbulence 0.8} }

plane { <0, 1, 0>, 0.46
    texture { DMFWood6 scale 2 }
    normal { wrinkles }
    finish { phong 0.8 phong_size 2000 }
}

plane { <0, -1, 0>, -5
    pigment { color Gray }
}

light_source { <-1, 0.7, -1> color White media_interaction on}
light_source { <2, 0.7, 1> color White media_interaction on}
light_source { <0.7, 0.7, -1> color Red media_interaction on}
light_source { <-1.5, 0.5, -1.5> color White media_interaction on }

```

Das durch #include "kugel_crash1.inc" eingebundene File wird mit folgendem Perl-Script erstellt:

```

#!/usr/bin/perl

$it = 4;

$i = 0;
$x1 = 0;
$y1 = 0;
$z1 = 0;
$x2 = 1;
$y2 = 1;
$z2 = 1;

$n = 20;
$d = 1.05/$n;
$d2 = $d/2;

for ($i=0;$i<=$n;$i++) {
    for ($j=0;$j<=$n;$j++) {
        for ($k=0;$k<=$n;$k++) {
            $lx1 = (($x2-$x1)/$n)*$i;
            $ly1 = (($y2-$y1)/$n)*$j;
            $lz1 = (($z2-$z1)/$n)*$k;
            printbox($lx1, $ly1, $lz1);
        }
    }
}

sub printbox {
    my ($x1, $y1, $z1) = @_;
    if (($y1<0.5) && (rand(1)>0.0)) {
        $fallx = (5*rand(1)-2.5);
        $fally = (5*rand(1)-2.5);

        ##$fall = $y1 if ($fall>$y1);
        $falling = "translate <-$fallx,0,-$fally>";
        $rx = rand(($fallx+$fally)*50);
        $ry = rand(($fallx+$fally)*50);
        $rz = rand(($fallx+$fally)*50);
        $visible = '0'; $visible = '1' if (rand(3)>($fallx*$fallx+$fally*$fally));
        $y1=0.5;
    } else {
        $falling="";
        $rx = rand(0);
        $ry = rand(0);
        $rz = rand(0);
        $visible = '1';
    };
    print "box {
    <0,0,0>,
    <$d,$d,$d>
    rotate <$rx,$ry,$rz> translate <$x1, $y1, $z1>
    $falling }\n"
    if (((($x1-0.5)**2+($y1-0.5)**2+($z1-0.5)**2)<(0.5**2)) && ($visible eq '1') && (rand(1.4)>1)) ;
    print "sphere {
    <0,0,0>, $d2
    rotate <$rx,$ry,$rz> translate <$x1, $y1, $z1>
    $falling }\n"
}

```

```

    if (((($x1-0.5)**2+($y1-0.5)**2+($z1-0.5)**2)<(0.5**2)) && ($visible eq '1') && (rand(3)>1)) ;
}

```

Schlussendlich wurde die Berechnung auf dem Cluster folgendermassen gestartet:

```

#/bin/bash

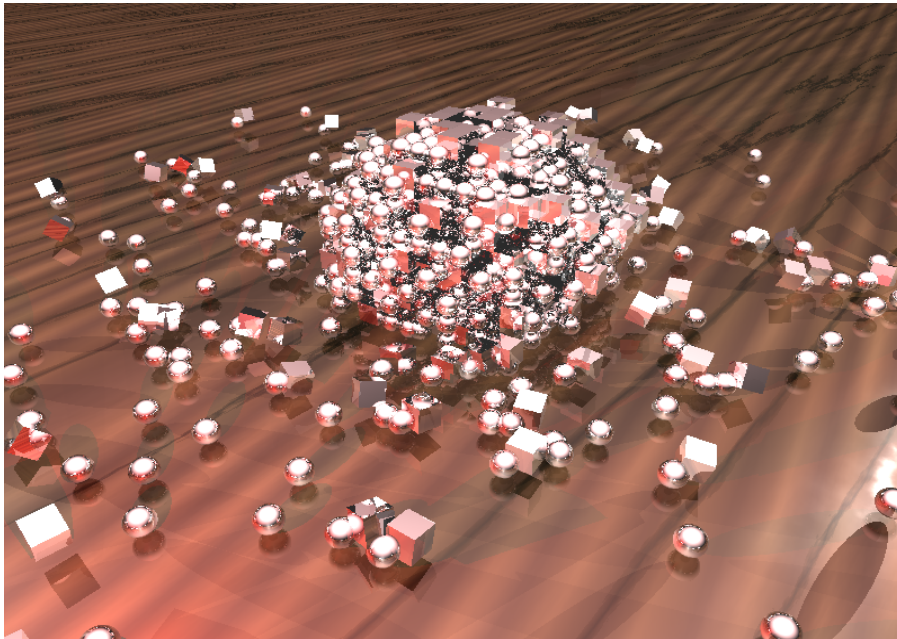
povfile="kugel_crash1.pov"
outfile="kugel_crash_$1"
otherfiles="kugel_crash.pl kugel_crash1.inc"

if /usr/bin/pvmpov +v1 -d +ft -x +a0.300 +r3 -q9 -w640 -h480 -mv2.0 +b \
-ND/home/bilder/kugel/ -NS/usr/bin/pvmpov -I/home/bilder/kugel/$povfile \
$2 +FP +O $outfile.ppm +L/home/povray/include | tee pov_output.txt

then
    tar czf $outfile.tar.gz $povfile $otherfiles make.sh
else
    cat pov_output.txt
    echo -e "\nErrors occurred, processing aborted"
fi

```

8.1.2 Bild



8.2 PC-Ausstattung

Typ 1, Verfügbarkeit ca. 120 Stück:

```

model name      : Pentium II (Deschutes)
cpu MHz         : 348.207
cache size     : 512 KB
bogomips       : 694.68

```

Typ 2, Verfügbarkeit ca. 50 Stück:

```

model name      : Pentium III (Coppermine)
cpu MHz         : 863.727
cache size     : 256 KB
bogomips       : 1723.59

```

Vom Typ 2 existieren ein paar wenige Exemplare mit einer leicht geringeren Leistung von 733 Mhz.

Literatur

- [SPE] David HM Spector: Building Linux Clusters, O'Reilly, July 2000
- [Redhat] RedHat, <http://www.redhat.com>
- [Kernel] Linux Kernel, <http://www.kernel.org>
- [RJAI] Raj Jain: The Art of Computer Systems Performance Analysis, John Wiley and Sons, 1991
- [POV] Persistence of Vision Raytracer, <http://www.povray.org/>
- [PPV] PVM patch for POV-Ray, <http://www.luga.de/~flierl/pvmpov/>
- [PVM] PVM: Parallel Virtual Machine, http://www.epm.ornl.gov/pvm/pvm_home.html
- [PGSQL] PostgreSQL, <http://www.postgresql.org>