Master-Thesis zum Erlangen des akademischen Grades

Master of Science (M.Sc.)

im Studiengang Informatik im Fachbereich Elektrotechnik und Informatik

Bewertung verschiedener verteilter Dateisysteme auf Linux-basierten Cluster-Rechnern

vorgelegt von

Name: Janke Vorname: Felix

Geb. am: 14.10.1987 in: Greifswald

Erstgutachter: Prof. Dr. rer. nat. Michael Koch
Fachbereich Elektrotechnik und Informatik
Fachhochschule Stralsund

Zweitgutachter: Prof. Dr.-Ing. Andreas Noack
Fachbereich Elektrotechnik und Informatik
Fachhochschule Stralsund

Greifswald, den 16. September 2013

Vorwort

Das Rechenzentrum der Ernst-Moritz-Arndt-Universität plant die Anschaffung eines neuen Compute-Clusters, welcher mehrere Server mit hunderten Kernen aufweisen wird. Dieser dient zur Unterstützung von Doktoranden und Studenten, die in ihrer Tätigkeit komplexe Berechnungen durchführen müssen.
Insbesondere Mitarbeiter, die in physikalischen, chemischen, mathematischen oder biochemischen Bereichen tätig sind, benötigen leistungsstarke Hardware, damit mehr Rechenoperationen ausgeführt werden können. Dies ist möglich, indem Aufgaben schneller ausgeführt werden und das es während der Arbeitszeit zu keiner Zeit zu Speicherlimitierungen kommen kann. Aufgrund dessen ist es für das Rechenzentrum zur Ausarbeitung eines optimalen Nutzungskonzeptes wichtig, dass für die gegebenen Anforderungen das möglichst optimale Dateiystem herausgefunden wird.
Besonders dieses ist in solchen komplexen Strukturen ein elementar wichtiges Produkt, da es oftmals als Flaschenhals ausgemacht werden kann. Das Lesen und Schreiben zwischengespeicherter Werte ist oftmals langsamer, als die Berechnungen selbst. Zudem ist das Zusammenfassen mehrerer Festplatten zu einem komplexen Datenspeicher notwendig, um eine Limitierung der Festplattenkapazitäten nicht zu schnell zu erreichen.
Die folgende Arbeit gibt einen konkreten Überblick darüber, was ein Dateisystem ist, wie diese im allgemeinen aufgebaut sind und zusätzlich wird der Unterschied zwischen regulären und verteilten Dateisystemen aufgezeigt. Durch die nähere Betrachtung dieser, sowie durch verschiedene Testverfahren, welche für einzelne Systeme durchgeführt werden, wird eine Entscheidung bezüglich eines Systems getroffen.
Zur Verfügung gestellt wird dafür ein kleinerer Cluster, welcher aus drei einzelnen Servern besteht. Auf diesen können verschiedene Dateisysteme installiert und unterschiedliche Tests durchgeführt werden. Besondere Punkte, die für das Universitätsrechenzentrum für die Entscheidungsfindung notwendig sind, sind die Performance, Stabilität und die Administrationsmöglichkeiten. Das gewählte Produkt sollte open-source verfügbar sein, das bedeutet, das System muss kommerziell zur Verfügung stehen, ohne zusätzliche Kosten zu verursachen. Insbesondere bei solchen Systemen ist es nicht möglich, eine pauschalisierte Aussage über die Stabilität zu treffen, da sich viele noch im Entwicklungsstadium befinden. Als Entscheidungshilfe dient der Blick in andere Rechenzentren, die solche Dateisysteme effektiv einsetzen.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Die Ernst-Moritz-Arndt-Universität Greifswald zählt zu den ältesten Universitäten im deutschsprachigen Raum. Bereits im Jahr 1456 wurde diese auf Initiative von Bürgern der Hansestadt Greifswald gegründet. Aktuell sind über 11.700 Studenten an der Universität eingeschrieben.
Das Universitätsrechenzentrum verwaltet dabei alle technischen Aufgaben, die zur Unterstützung der Kernprozesse durch Computer anfallen. Dazu zählt der Betrieb des gesamten Datennetzes mit der Zusammenarbeit von Routern, Switches und Firewalls, sowie die Verwaltung der DNS- und DHCP-Services.

Das Universitätsrechenzentrum verwaltet zusätzlich die gesamte Telefonie- und E-Mail-Kommunikationsstruktur der Universität. Zentrales Thema ist allerdings das Nutzermanagement und die Verwaltung von Compute-Servern für komplexe Berechnungen in mathematischen, medizinischen, chemischen, biochemischen und physikalischen Bereichen. Für solche Anwendungen sind Hochleistungsserver nötig, um die Bearbeitungszeit möglichst gering zu halten. Durch optimierten, parallelisierten Quellcode ist ein Einsatz von Cluster-Servern sinnvoll, da mehrere Aufgaben gleichzeitig ausgeführt werden können. Außerdem kann eine Lösung der Limitierung von Festplattenkapazitäten durch das Zusammenfassen zu großen Dateisystemen gefunden werden.

Das Rechenzentrum der Ernst-Moritz-Arndt-Universität Greifswald plant dazu die Anschaffung eines neuen Compute-Clusters, welcher mehrere hundert Kerne beinhalten wird. Für ein optimales Nutzungskonzept ist ein direkter Vergleich von verfügbaren Dateisystemen für solche Aufgabengebiete im Hinblick auf Performance, Stabilität, Skalierbarkeit, Ausfallsicherheit und Verwaltbarkeit nötig. Die folgende Arbeit dient dazu, eine Entscheidung hinsichtlich des Dateisystems zu treffen. Dafür werden diese im einzelnen in ihrer Arbeitsweise miteinander verglichen.

Aus den Erkenntnissen werden bestimmte Systeme ausgewählt, die zur Durchführung

von verschiedenen Testverfahren auf einem kleineren Compute-Cluster installiert werden. Dieser Vergleich soll darlegen, welches System für die vorherrschenden Anwendungsbereiche am besten geeignet ist. Das daraus resultierende Fazit beeinflusst die Entscheidung, welches Dateisystem auf dem Hochleistungscluster eingesetzt werden soll.

Die enge Mitarbeit mit Nutzern des Compute-Clusters hilft dabei, ein Dateisystem herauszufinden, welches den Ansprüchen optimal gerecht werden kann. Dafür wird der Netzverkehr des Universitätsrechenzentrums auf dem aktuellen Cluster untersucht, um zu erkennen, welche Dateigrößen hauptsächlich genutzt werden. Dies ist darum von großer Bedeutung, weil die unterschiedlichen Arten von Dateisystemen unterschiedlich auf verschiedene Dateigrößen reagieren und entsprechend skalieren.

1.2 Bedeutung der Arbeit

Durch die hohen Kosten für einen neuen Compute-Cluster, die auf etwa eine Million Euro beziffert werden, ist es für das Universitätsrechenzentrum äußerst wichtig, dass dieser auch die möglichst maximale Rechenleistung zur Verfügung stellen kann. Dafür sind viele Kriterien von entscheidender Bedeutung. Diese beginnen bei der Auswahl der korrekten Hardware, die für das Einsatzgebiet nötig ist, über die Anbindung der einzelnen Server durch verschiedene Netzwerktopologien, wie zum Beispiel InfiniBand, Myrinet oder 10GbE (10-Gigabit-Ethernet), bis zu den verschiedenen Arten von Kühlsystemen.

Bei großen Compute-Clustern auftreten kann, ist, dass oftmals die Bearbeitungszeit eines Problems häufig schneller durchgeführt werden kann, als das die Werte auf Datenträgern abgelegt werden können. Dieser Flaschenhals soll bei dem neuen Cluster möglichst vermieden werden.

Das Thema GreenIT spielt auch hierbei eine zentrale Rolle. Dazu zählt das optimale Verhältnis zwischen Betriebskosten und vorhandener Leistung, sowie die Nutzung der Abwärme des Serversystems zur Erwärmung der Warmwasservervorgung des Rechenzentrums. Auch wird darauf geachtet, dass die eingesetzte Hardware möglichst umweltschonend hergestellt wurde und nach der Nutzung entsprechend recycled werden kann. Dafür ist eine enge Zusammenarbeit mit den Herstellern notwendig.

Ohne ein performantes Dateisystem ist es nicht möglich, die theoretisch zur Verfügung stehenden Leistung möglichst effizient zu nutzen. Daher ist ein solcher Vergleich von verschiedenen Distributed-File-Systems von großer Bedeutung, da dieser aufzeigt, welches System in welchen Einsatzgebieten die beste Performance bietet.
Hierbei darf man allerdings den Punkt der Verwaltbarkeit nicht außer Acht lassen. Das Universitätsrechenzentrum ist bestrebt eine zuverlässige, sichere und einfach zu konfigurierende Implementation zu finden, welche ohne großen Aufwand und zusätzlichen speziellen Kenntnissen zu administrieren ist.

1.3 Aufbau der Arbeit

Die Arbeit ist aufgeteilt in verschiedene Kapitel, die den Aufbau von Compute-Clustern erklären, die regulären und verteilten Dateisysteme beschreibt und miteinander vergleicht, sowie die durchgeführten Testverfahren beschreiben und auswerten. Anschließend erfolgt ein Fazit, welches Dateisystem für welche Einsatzgebiete besser geeignet sind und welches das Universitätsrechenzentrum für den neuen Hochleistungscluster einsetzen könnte.

Inhaltsverzeichnis Im Inhaltsverzeichnis wird ein kurzer und prägnanter Überblick über das Thema gegeben und welche Punkte bei der Untersuchung bearbeitet werden.

Compute-Cluster Im zweiten Kapitel der Ausarbeitung wird der Aufbau und die Funktionsweise von Compute-Clustern beschrieben. Dieses Wissen wird insbesondere für das bessere Verstehen der Arbeits- und Wirkungsweise von Filesystems und verteilten Dateisystemen benötigt.

Dateisysteme In diesem Kapitel werden reguläre Dateisysteme vorgestellt, die sich auf den meisten Linux-basierten Computern befinden. Insbesondere das BTRFS-Filesystem ist hierbei von großer Bedeutung, da es zum einen als Dateisystem der Zukunft gehandelt, aber bereits jetzt schon bei einigen verteilten Dateisystemen als Unterlage empfohlen wird.

Distributed Filesystems Insbesondere für den Betrieb in großen Serverfarmen ist der Einsatz von verteilten Dateisystemen eine sehr gute Alternative, um den Flaschenhals der begrenzten Lese- und Schreibgeschwindigkeiten von einzelnen Festplatten und Festplattenverbunde zu minimieren, indem das System automatisch eine Datei auf mehrere Server und dessen eigenen Speicher auslagert. Dabei existieren verschiedene Ansätze, die in unterschiedlichen Anwendungen zum Einsatz kommen, diese werden in diesem Kapitel beschrieben.

Tests Im Kapitel Tests werden einzelne Testszenarien vorgestellt, die bei dieser Untersuchung zum Einsatz gekommen sind. Dabei untersuchen diese unterschiedliche Eigenschaften des Filesystems, von der Lese- und Schreibgeschwindigkeit, bis hin zu einem Lasttest des Systems.

Testauswertung Die vorgestellten Testszenarien werden nicht nur theoretisch ausgearbeitet, sie werden auch auf einem vom Universitätsrechenzentrum zur Verfügung gestellten Serververbund nachgestellt und auf Performance, Skalierbarkeit, Stabilität und Verwaltbarkeit miteinander verglichen. Dabei sollte sich herauskristallisieren, welche Lösung sich für welche Einsatzgebiete am besten eignet.

Fazit Die durchgeführten Tests, sowie dessen Auswertungen hinsichtlich verschiedener Aspekte führt dazu, dass das Universitätsrechenzentrum eine konkrete Übersicht erhält, welche Distributed-Filesystems-Arten sich aktuell auf dem Markt befinden, wie diese aufgebaut sind und welches für das Aufgabengebiet des neuen Compute-Clusters am besten geeignet ist. Im Fazit wird daher eine Empfehlung gegeben, welches System auf dem neuen Cluster eingesetzt werden sollte.

Kapitel 2

Compute-Cluster

Ein Compute-Cluster ist in der Informationstechnologie ein Verbund aus Servern, die miteinander meist durch spezielle Hardware vernetzt sind und untereinander kommunizieren können, dabei erscheint dieser Zusammenschluss als eine zentrale Einheit nach außen.
Der Begriff **Cluster** steht dabei für „Rechner-Schwarm“ oder „Rechner-Haufen“. Cluster und Constellations sind Gruppen von Rechnern in einem „High-Performance-Netzwerk“, welches besonders für **parallele Berechnungen** optimiert wurde.
Dabei existieren insbesondere im grundlegenden Aufbau, in der Netztopologie und in der Art des Speicherns der verwendeten Daten große Unterschiede, die dazu führen, dass der eine oder andere Punkt zu Begrenzungen der Performance führt und die Leistung des Clusters beeinträchtigt.

Abbildung 2.1: Juqueen Cluster in Jülich [3]

Compute-Cluster müssen spezielle Eigenschaften erfüllen, die für jeden Anwendungsfall unterschiedlich sein können, daher wurde mit der Zeit eine Typisierung entwickelt, die für jeden Einsatzzweck unterschiedlich aufgebaut ist. Grundlegende Eigenschaften bleiben allerdings bestehen, wenn diese auch für jede Art verschiedene Gewichtungen beinhalten.

Grundlage für den Aufbau und dem Betrieb eines Compute Clusters ist das 1967 entwickelte **„Amdahls-Gesetz“**, was besagt, dass „ [...] sich der nicht parallelisierbare Anteil eines Programms auf die Gesamtrechenzeit auswirkt [...]“ [2]. Das heißt, dass ein sequentieller Anteil eines Programms immer die Gesamtzeit des Rechnens so beeinflusst, dass diese immer höher ist, als bei rein parallelen Programmen.

Dieses Gesetz wird heutzutage als Basis für Multiprozessor- und Cluster-Computer angesehen.

$$\mathrm{T(p)} = T(1) * \frac{1-\alpha}{p} + T(1) * \alpha$$

$$\mathrm{S(p)} = \frac{p}{(1-\alpha) + p * \alpha}$$

Quelle: [28]

In der angegeben Formel ist $T(1)$, die Zeit, die ein Prozessor für ein sequentielles Programm benötigt, α der sequentielle Anteil des Programms, sowie p die Anzahl der verfügbaren Prozessoren. Somit ergibt sich $T(p)$ als Zeit, die ein Programm benötigt, wenn diesem p Prozessoren zur Verfügung stehen. Der mögliche Speedup $S(p)$ zeigt dabei auf, wie effizient die Parallelisierbarkeit arbeiten kann.
Es ist zu erkennen, dass die Geschwindigkeit immer vom sequentiellen Anteil eines Programmes abhängig ist.

Ein Compute-Cluster ist ein Zusammenschluss mehrerer einzelner Server, die durch einen Hauptknoten miteinander verbunden sind und somit die Möglichkeit besteht, einzelne Teilaufgaben eines Programms auf verschiedene Teilknoten auszugliedern, um einen parallelen Ablauf zu erhalten.

In der heutigen Zeit werden Compute-Cluster für die Bereitstellung von Daten in der **Cloud** verwendet, da diese mit sehr vielen Anfragen umgehen können und diese auf verschiedene Hosts aufteilbar ist. Außerdem ist die Nutzung von Compute-Clustern in „High-Performance-Netzwerken“ elementar, da durch Parallelisierung die Performance durch mehrere vorhandene Serverknoten stark ansteigen kann.

2.1 Aufbau eines Compute-Clusters

Um den Aufbau eines Computer-Clusters zu beschreiben ist die Unterteilung in verschiedene Arten von großer Bedeutung, da sich diese in ihren Grundeigenschaften unterscheiden können. Es besteht die Möglichkeit, einen Hochverfügbarkeitscluster aufzubauen, Load-Balancing-Cluster und High-Performance-Computing-Cluster haben eine andere Arbeitsweise. Allerdings können heutzutage Mischformen auftreten, sodass die Übergänge zwischen den einzelnen Arten fließend ist. So ist es möglich, einen Hochverfügbarkeitscluster aufzubauen, der aber trotzdem eine sehr hohe Performance aufweisen kann.

2.1.1 Hochverfügbarkeitscluster

Unter einem Hochverfügbarkeitscluster versteht man einen Serververbund, der eine sehr hohe **Ausfallsicherheit** bietet. Dies wird dadurch erreicht, dass zum einen besondere Hardware genutzt wird, die nicht nur auf Performance optimiert wurde, sondern auch auf Langlebigkeit. Des Weiteren gewährleistet man diesen Aspekt durch **redundante Hardware**, die entweder im gleichen Serverrack eingebaut wird und bei einem Ausfall eines Systems aktiviert werden und die Arbeit übernimmt oder es wird oftmals der gleiche Aufbau eines Hochverfügbarkeitsclusters in getrennten Räumen oder Häusern untergebracht. Dies hat den Vorteil, dass bei einem Ausfall des gesamten Servers die Daten automatisch auf den Sicherheitsserver übertragen werden und somit weitergearbeitet werden kann und anschließend der Server-Administrator die Fehlerquelle ausmachen und beheben kann, ohne das es zu Einschränkungen in der Betriebszeit kommt.

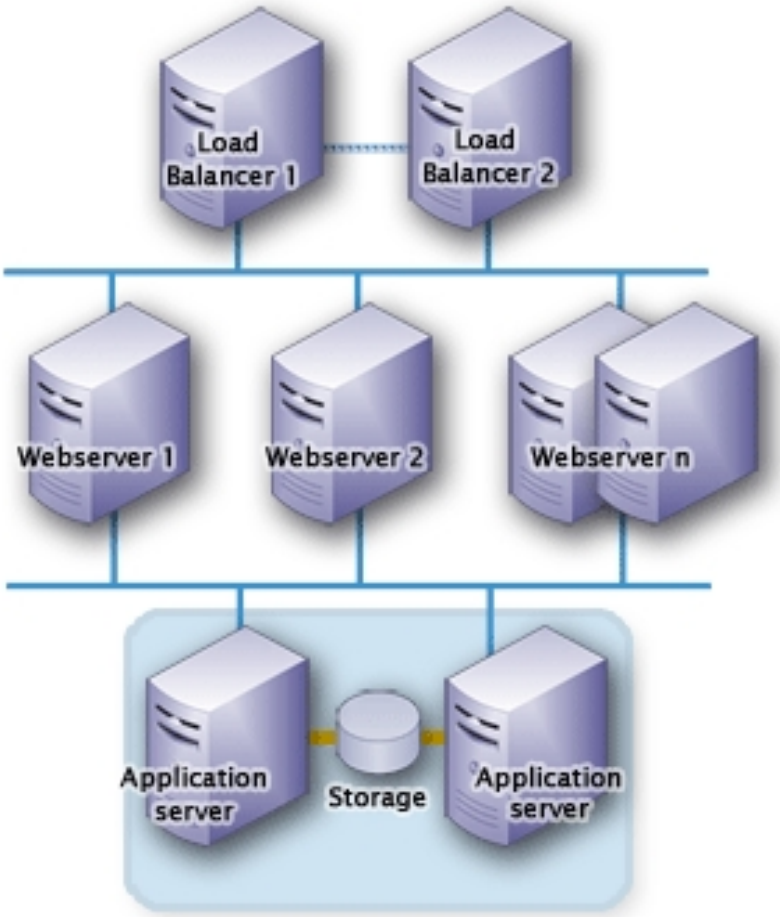

Abbildung 2.2: Hochverfügbarkeit durch Redundante Hardware [7]

Hochverfügbarkeitscluster haben durch den Aufbau einen Nachteil: Es muss Hardware angeschafft werden, die lediglich im Notfall aktiviert wird und keine zusätzliche Rechenleistung zur Verfügung stellt.

Jedoch bei besonders wichtigen Bereichen, die auf eine Hochverfügbarkeit angewiesen sind, ist dieser Umstand unumgänglich, da es sonst zu hohen Ausfallkosten für den Betreiber kommen kann.

Eingesetzt werden solche Arten von Compute-Cluster in Bereichen bei Datenbanksystemen oder Cloud-Computing-Systeme, die eine möglichst hohe Verfügbarkeit aufweisen müssen, sodass die gespeicherten Daten, bzw. die Rechenzeit jederzeit zur Verfügung gestellt werden kann.

2.1.2 Load-Balancing-Cluster

Load-Balancing-Cluster dienen der **Lastverteilung** innerhalb eines komplexen Netzwerkes. Dies ist in Umgebungen nötig, wo viele Recheneinheiten zur Verfügung stehen und die benötigte Rechenleistung sehr hoch ist.

Die Verteilung der einzelnen Lasten auf die einzelnen Server erhöht die Effizienz der gesamten Hardware, weil dadurch gewährleistet werden kann, wenn bestimmte Aufgaben nicht gut parallelisierbar sind, dass diese verteilt abgearbeitet werden können.

Ein Beispiel dafür wäre ein Verbund aus vielen Servern, die als Suchmaschine im Internet eingesetzt werden. Durch die vielen verschiedenen Anfragen, die nichts miteinander gemein haben, muss eine Lastverteilung stattfinden, da sonst einzelne Server bei besonders vielen Anfragen schnell an die Leistungsgrenze gebracht werden können.

Das Load-Balancing vermindert die Rechenzeit und erhöht die Performance des gesamten Systems. Bei besonders vielen gleichzeitigen Anfragen und Aufgaben ist eine Lastverteilung für eine möglichst performante Leistung unumgänglich.

In der heutigen Zeit können solche Load-Balancing-Cluster auch die theoretische Geschwindigkeit jedes einzelnen angebundenen Servers abspeichern und können somit die Last besser verteilen, da komplexe Aufgaben auf schnelleren Prozessoren ausgeführt werden, kleinere Anwendungen auf, wenn vorhanden, langsamere.

Abbildung 2.3: Beispiel von Load-Balancing durch Cluster [8]

Eingesetzt werden solche Cluster im Bereich Web- und Application-Services, da man dort mit einer hohen Anzahl von Anfragen rechnen muss, die alle in möglichst geringer Zeit bearbeitet werden sollen. Daher ist eine Lastverteilung auf mehrere Server nötig, da die Performance eines einzelnen oftmals nicht ausreicht und dieser zusammenbrechen könnte.

2.1.3 High-Performance-Computing-Cluster

Diese Art von Compute-Clustern ist auf eine möglichst hohe **Performance** getrimmt. Genutzt werden diese hauptsächlich bei der Berechnung von wissenschaftlichen Problemen, sowie beim Rendern von 3D-Anwendungen und -animationen.

Die Berechnungen werden auf mehrere Hosts aufgeteilt, um einen parallelen Ablauf zu erhalten. Dabei unterscheidet man, ob diese Verteilung global für alle Hosts gleichzeitig durchgeführt wird oder ob jeder sein eigenes spezielles Paket erhält, mit dem dieser rechnen soll. Dabei werden alle verfügbaren Recheneinheiten für die Berechnungen genutzt, damit das Problem in möglichst kurzer Zeit gelöst werden kann.

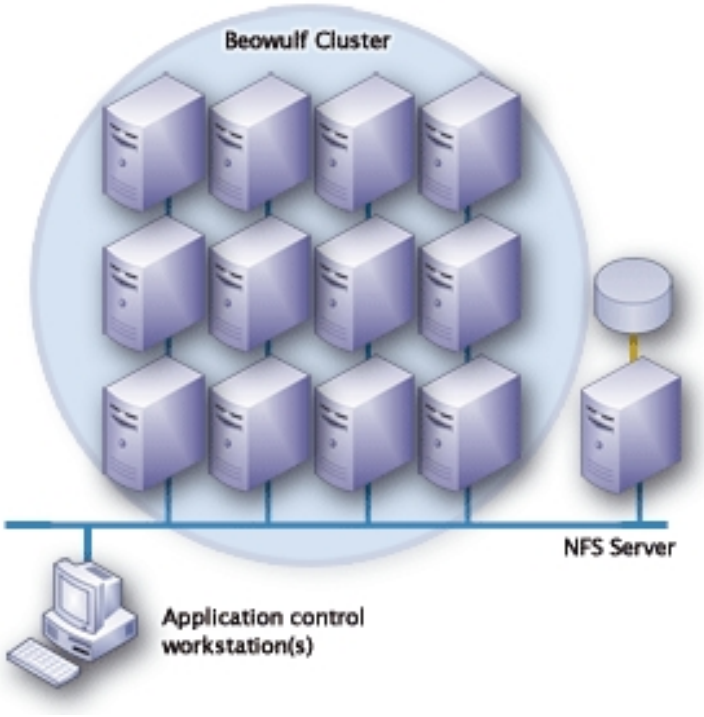

Abbildung 2.4: Beispiel von High-Performance-Cluster [9]

Eingesetzt werden solche Cluster-Systeme hauptsächlich bei komplexen Berechnungen in naturwissenschaftlichen Bereichen, weil diese die höchstmögliche Rechenkapazität benötigen und weniger als Storage dienen. Auch viele Suchmaschinen nutzen solche High-Performance-Cluster in deren Rechenzentren, um viele Anfragen in möglichst geringer Zeit ausführen zu können.

Gerade in solchen Anwendungsgebieten entstehen auch Mischformen, besonders zwischen Load-Balancing-Cluster und High-Performance-Cluster.
Eine Kombination mit einem Hochverfügbarkeitscluster ist hierbei möglich, widerspricht allerdings dem Konzept des High-Performance-Computing-Clusters. Daher entstehen immer mehr Mischformen, sodass eine genaue Abgrenzung immer schwieriger ist, die Grundgedanken bleiben aber immer erhalten.

2.2 Eigenschaften

Ein Compute-Cluster hat je nach Anwendungsfall verschiedene wichtige Eigenschaften, die erfüllt sein müssen, damit man davon ausgehen kann, dass das gesamte System korrekt funktioniert. Allerdings müssen einige grundlegende Voraussetzungen immer gegeben sein.

2.2.1 Fail-Over

Fail-Over bedeutet in der IT-Industrie das automatische **Umschalten der Arbeitsprozesse** eines Servers auf einen anderen im Falle eines Fehlers oder Defekts oder des gesamten Abschaltens eines Clusters und der Übernahme der Aktivitäten eines Backup-Clusters. Dies soll den Zweck haben, dass die Ausfallzeiten innerhalb der Arbeitszeiten minimiert werden und ein Administrator die Möglichkeit besitzt, diese Fehler zu beheben.

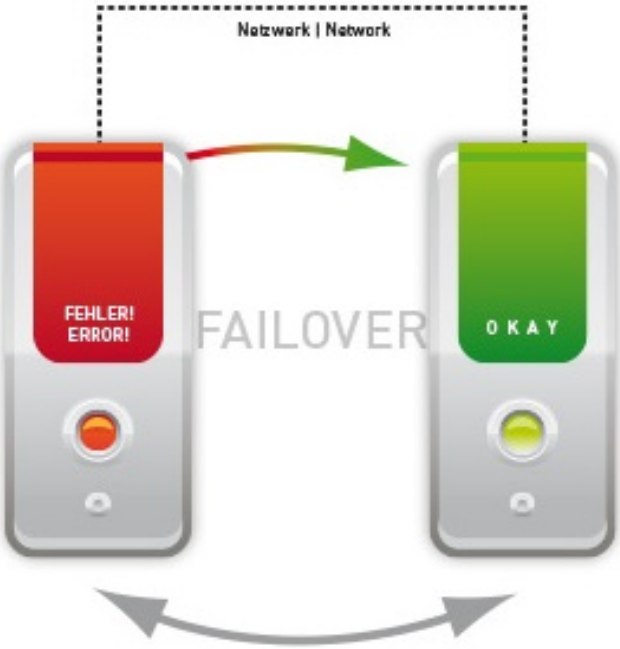

Abbildung 2.5: Fail-Over-Cluster [11]

Ein solcher Fail-Over-Cluster besteht aus mindestens einem Haupt- und einem Backup-Cluster, der im Falle eines Fehlers ohne administratives Eingreifen die Aufgaben übernimmt. Dazu muss die Hardware physikalisch miteinander verbunden sein und es muss die Möglichkeit bestehen, dass die beiden Recheneinheiten auf einen **gemeinsamen Speicherbereich** zugreifen können, damit eine Übergabe der laufenden Prozesse möglich ist. Für solche Übertragungen werden von verschiedenen Herstellern unterschiedliche Lösungen angeboten, die dieses übernimmt.

Diese Eigenschaft ist besonders bei Hochverfügbarkeitsclustern wichtig, da die Ausfallzeiten eines gesamten Clusters minimiert werden. Dieser ist zwar heruntergefahren, aber die Benutzer können durch das automatische Einschalten des Backup-Servers weiter arbeiten. Meist geschieht dies in sehr kurzer Zeit, sodass es zu keinem Arbeitsausfall kommt. Hinzu kommt, dass dies die Verfügbarkeit bzw. Hochverfügbarkeit des Systems erheblich verbessert.

2.2.2 Hochverfügbarkeit

Die Verfügbarkeit eines Systems ist in der heutigen Zeit wichtiger denn je geworden. Ein möglicher Serverausfall kann großen Unternehmen sehr viel Geld kosten und führt häufig dazu, dass Mitarbeiter völlig arbeitsunfähig sind. Daher existieren verschiedene Kenngrößen, die bei der Berechnung der Verfügbarkeit von Servern, bzw. Clustern genutzt werden. Wichtig ist diese Eigenschaft besonders bei, wie der Name vermuten lässt, Hochverfügbarkeitsclustern.

„Ein System gilt als hochverfügbar, wenn eine Anwendung auch im Fehlerfall weiterhin verfügbar ist und ohne unmittelbaren menschlichen Eingriff weiter genutzt werden kann.“ [5]

Die Kenngröße „Verfügbarkeit“ lässt sich berechnen durch:

$$\text{Verfügbarkeit} = \frac{Uptime}{Downtime + Uptime}$$

Aus dieser Formel ist zu erkennen, dass die Verfügbarkeit besser wird, je kleiner die Downtime ist, das ist die Zeit, wo das System nicht zur Verfügung steht. Bei dieser Formel müssen auch geplante Wartungsarbeiten und ungeplante Abstürze mit einberechnet werden, die durch den Ausfall von einzelnen Komponenten zu Stande kommen kann oder durch das Ersetzen ganzer Hardwareeinheiten.

Um Fehler im System und die daraus resultierende Nichtverfügbarkeit zu minimieren, muss es die Möglichkeit besitzen, diese so früh wie möglich erkennen zu können. Das kann durch so genanntes „Proaktives Monitoring“ [5] geschehen, welches meist durch die Hardware-Hersteller bereits integriert ist. Ein Beispiel bei Festplatten sind die S.M.A.R.T.-Werte, die bereits vor einem eigentlichen Defekt den Administrator warnen können, das es zu Problemen mit der Hardware kommen wird.

Neben der ständigen Überwachung der Hardware ist es aber nicht auszuschließen, dass diese auch abrupt einen Defekt aufweist, zum Beispiel durch eine Überspannung im System. Solche sogenannte **„Single Point of Failure“** (kurz: SPOF) können durch intelligente Hard- und Software im Server selbst und auch in der Netzwerkarchitektur bereits mit eingeplant werden. Insbesondere bei Netzwerken ist dies sehr wichtig, denn diese verbinden alle Server eines Rechenzentrums miteinander. Fällt ein Server aus, der

möglicherweise durch einen Backup-Server gesichert ist, kann dieser SPOF bereits eliminiert werden, allerdings ist es häufig so, dass beide Server über den gleichen Netzwerkswitch an das gesamte Netzwerk angebunden sind. Fällt dieser Switch aus, ist keine Kommunikation mehr möglich und somit kommt es zu einem Systemausfall. Daher muss man auch bei der Entwicklung der Netztopologien solche Eventualitäten zu beachten, um ein möglichst hochverfügbares System zu erhalten.

Durch den Einsatz eines Fail-Over-Clusters wird die Downtime stark verringert und dementsprechend die Verfügbarkeit verbessert, denn der Administrator hat nun die Möglichkeit die defekte Hardware auszutauschen und den primären Server wieder zu starten. Dies kann einige Zeit in Anspruch nehmen, insbesondere wenn für die defekte Hardware kein Ersatz direkt vorhanden ist und bestellt werden muss.

Um ein solches Umschalten ohne Datenverlust zu gewährleisten, müssen die Cluster gleichzeitig auf einen gemeinsamen Speicher zugreifen können. Dies wird in der heutigen Zeit über ein zentrales Storage gelöst, wo alle wichtigen Komponenten, auf die ein Cluster Zugriff haben muss, zentral abgelegt werden. Bei einem Ausfall des Hauptclusters kann nun der Backup-Server automatisch die Arbeit fortsetzen, die unterbrochen wurde. Die Downtime ist daher sehr gering, da diese nur die Startzeit des Backup-Servers beträgt.

Besonders bei großen Serverfarmen und in Rechenzentren wird die Verfügbarkeit in verschiedene Klassen unterteilt. Diese unterteilen sich zwischen maximal 3,7 Tagen und 3 Sekunden Downtime im Jahr.

Stufen der Verfügbarkeit			
Verfügbarkeitsklasse	**Bezeichnung**	**Verfügbarkeit in Prozent**	**Downtime pro Jahr**
2	stabil	99	3,7 Tage
3	verfügbar	99,9	8,8 Stunden
4	hochverfügbar	99,99	52,2 Minuten
5	fehlerunempfindlich	99,999	5,3 Minuten
6	fehlertolerant	99,9999	32 Sekunden
7	fehlerresistent	99,99999	3 Sekunden

Abbildung 2.6: Verfügbarkeitsklassen [6]

Die Übersicht in Abbildung 2.6 zeigt deutlich, dass die Einteilung sich prozentual lediglich in den Nachkommastellen ändert, die Downtime aber rapide sinkt.
Wenn ein Unternehmen auf eine absolute Hochverfügbarkeit angewiesen ist, zum Beispiel im Falle eines Call-Centers, die 24 Stunden täglich und 365 Tage im Jahr erreichbar sein müssen, muss eine absolut geringe Downtime durch hochverfügbare Server- und Netzwerk-Hardware zur Verfügung stehen.

Da in der Downtime auch geplante Wartungsarbeiten mit einberechnet werden, können andere Unternehmen, die geregelte Arbeitszeiten aufweisen, auf eine geringere Verfügbarkeit zurückgreifen, da die Wartungsarbeiten außerhalb der Öffnungszeiten durchgeführt werden können. Bei dieser Überlegung ist es aber nicht absolut gewährleistet, dass es während der Arbeitszeit durch ungeplante Ausfallzeiten zu Einschränkungen kommen kann.

2.2.3 Skalierbarkeit

„Skalierbarkeit ist die Fähigkeit, Ressourcen so zu erweitern, dass (im Idealfall) die Dienstkapazitäten linear zunehmen. Das wesentliche Merkmal einer skalierbaren Anwendung ist, dass bei steigender Auslastung lediglich zusätzliche Ressourcen und keine umfangreichen Veränderungen der Anwendung selbst erforderlich sind.“ [12]

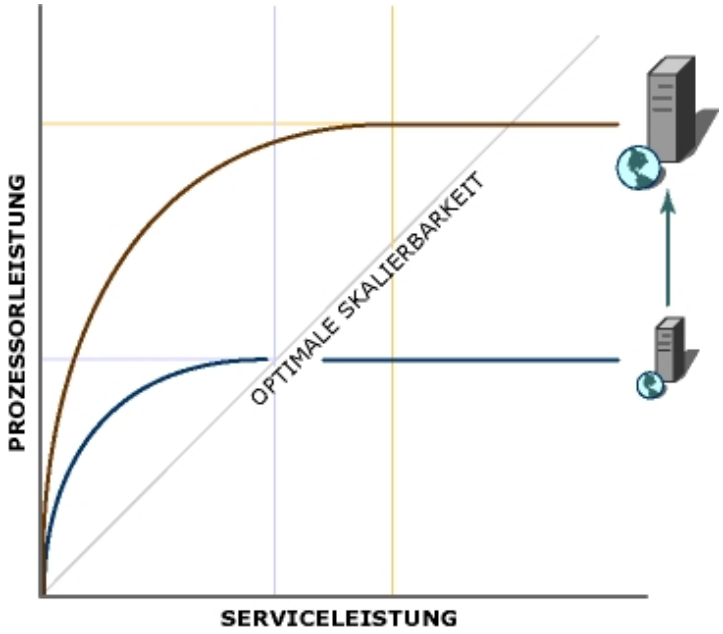

Abbildung 2.7: optimale Skalierbarkeit [13]

Im Optimalfall soll die Skalierbarkeit dazu führen, dass durch das Hinzufügen von Hardware in einen Verbund eine lineare Verbesserung der Performance eintreten soll. Dies ist hauptsächlich für High-Performance-Cluster von großer Bedeutung, da diese jede Recheneinheit dazu benötigen, die Arbeitsgeschwindigkeit zu erhöhen.

Um eine solche möglichst **lineare Skalierung** zu erreichen muss der ausgeführte Quellcode auch auf die Anzahl der verfügbaren Recheneinheiten ausführbar sein. Man erhält keinen zusätzlichen Geschwindigkeitsvorteil, wenn man ein Programm auf einem großen Cluster ausführt, welches mit maximal 4 CPU-Kernen umgehen kann.
Die Parallelisierbarkeit muss softwareseitig gegeben sein. Ebenso hat das Filesystem des Clusters einen entscheidenen Einfluss auf die Skalierbarkeit. Wenn die Begrenzung des Filesystems erreicht ist und keine weiteren Server mehr angesprochen werden können, dann endet auch die Skalierbarkeit. Solche Punkte müssen im Vorfeld einer Anschaffung unbedingt beachtet werden.

2.2.4 Verwaltbarkeit

Das Thema Verwaltbarkeit ist von sehr großer Bedeutung im Bereich Cluster-Computing, da ein Administrator gleichzeitig die Verwaltung über verschiedene einzelne Server, die in einem Verbund von außerhalb erreichbar sind, übernehmen muss, aber auch die vorhandene Netzwerktopologie innerhalb des Clusters und die Anbindung ins übrige Netzwerk überwachen muss.

Zusätzlich kann ein externes Storage zur Verfügung stehen, insbesondere bei Hochverfügbarkeitsclustern. Dies sind viele Punkte, die gleichzeitig beachtet werden müssen, daher ist eine gute Verwaltbarkeit von großer Bedeutung.

Hersteller von großen Compute-Clustern und Serverfarmen besitzen eine eigene Verwaltungssoftware für ihre eigenen Geräte, allerdings sind diese grundlegend unterschiedlich, sodass jeder Administrator für jeden Hersteller eine eigene Schulung benötigt.

Durch die komplexen Verwaltungsmöglichkeiten, die insbesondere sehr große Firmen anbieten, hat man allerdings die Möglichkeit, schnell einen Überblick über alle möglichen Ressourcen zu bekommen. Solche Management-Tools überwachen die gesamte Hardware innerhalb des Clusters und meldet sofort, wenn es zu Fehlern kommt.

2.3 Cluster-Filesystems

Compute-Cluster sind ein Verbund aus mehreren einzelnen Servern, die von außerhalb als ein einziger Server angesprochen werden können. Dabei ist die Unterscheidung wichtig, ob es sich hierbei um einen Cluster mit externem Storage handelt, oder ob jeder einzelne Server einen bestimmten Teil an Festplattenkapazitäten zur gemeinsamen Benutzung anbietet.

Bei einem externen Storage ist die wichtige Frage, ob es sich innerhalb des abgeschlossenen Systems um ein Filesystem handelt, welches mit möglichst großen Datenmengen, zum Beispiel die Anzahl von Dateien oder die Dateigröße, performant umgehen kann oder ob sich dieses mehr auf Sicherheit stützt.

Die bestehende Hardware-Konfiguration spielt dabei eine sehr große Rolle. Zum Beispiel können durch einen Hardware-RAID-Controller („Redundant Array of Independent Disks“ [15]) bestimmte Konfigurationen aufgebaut werden, die die einzelnen Festplatten als bestimmte Einheit zusammenfasst, um so die Performance und/oder die Sicherheit zu erhöhen.

Es besteht sowohl die Lösung, dass jeder Server innerhalb eines Clusters seine eigene Festplattenkapazität bereitstellt, als auch Storage-Lösungen, dass dort verschiedene Dateisteme für jedes Betriebssystem installiert werden können und mit Hilfe von Netzprotokollen darauf zugegriffen werden kann. Dies vereinfacht die Installation der Dateisysteme, denn häufig wird für UNIX-Umgebungen **XFS** oder **EXT2/3/4** genutzt, für Microsoft Windows Serverumgebungen **NTFS**-Dateisysteme.

Es existieren weitere, weniger bekannte, wie zum Beispiel BTRFS, die einen interessanten Ansatz mit sich bringen, allerdings sich noch im Entwicklungsstadium befinden und daher insbesondere für hochverfügbare Systeme noch relativ uninteressant sind, da diese sichere und stabile Formate einsetzen müssen, da ein möglicher Datenverlust enorme Kosten entwickeln kann.

Eine weitere Möglichkeit ist das Zusammenfassen einzelner Festplatten innerhalb der verschiedenen Server durch den Einsatz von verteilten Dateisystemen, **Distributed Filesystems**. Diese haben die Aufgabe, dass durch einen Master-Server die Daten automatisch auf verschiedene Festplatten verteilt werden.

Diese beinhalten häufig dann keine vollständigen Dateien mehr, sondern die Datei wird in einzelne Objekte zerlegt und diese werden verteilt gespeichert. Das hat den Vorteil, dass die Daten von unterschiedlichen Servern zur gleichen Zeit angefragt werden können und so theoretisch die Performance ansteigen sollte.

Dafür gibt es in der heutigen Zeit verschiedene Ansätze, die unterschiedliche Herangehensweisen an dieses Problem aufweisen. Jedes Dateisystem unterscheidet sich hinsichtlich der Performance, Sicherheit und Skalierbarkeit gegenüber der Konkurrenz.

Kapitel 3

Dateisysteme

Klassische Dateisysteme beschreiben die allgemeine Verwaltung der Abspeicherung von Daten auf physikalischen Datenträgern und dienen als direkte **Schnittstelle** zwischen dem Betriebssystem und den einzelnen Partitionen auf Festplatten und anderen Speichermedien.
Dafür definiert jedes Dateisystem einen festgelegten Raum, wo die Daten eindeutig beschrieben werden. Dem Nutzer wird anschließend die Datei und das Verzeichnis zur Verfügung gestellt, man kann unter normalen Umständen die logische Speicherung auf dem Datenträger nicht erkennen.

Die grundlegenden Aufgaben eines Filesystems sind das Anlegen von Verzeichnissen und Dateien, die Speicherung von Datumsinformationen, die Rechteverwaltung, sowie die Verwaltung von langen Dateinamen und deren besonderer Aufbau. Abhängig vom Filesystem können im Dateinamen Sonderzeichen und Umlaute vorhanden sein.

Um einen optischen Performancegewinn aufzuzeigen und die Fehlertoleranz zu minimieren, existieren mittlerweile viele Dateisysteme, die das „Journaling“ unterstützen. „Journaling“ bedeutet, dass Änderungen an Daten und Verzeichnissen nicht direkt auf dem Datenträger geschrieben werden, sondern diese in einer externen Tabelle zwischengespeichert werden. Das führt zu einer hohen Geschwindigkeit beim Schreiben, da dies erst dann durchgeführt wird, wenn die Auslastung des Datenträgers nicht so hoch ist. Zum anderen können kurzfristige Änderungen durchgeführt werden, ohne die eigentliche Datei bearbeiten zu müssen. Ein vollständiges Rückholen einer Datei ist bis zu dem Zeitpunkt, bis die Änderungen geschrieben werden, auch möglich. Eingesetzt wird dieses Verfahren bereits im NTFS-Filesystem für Windows, Apple verwendet dieses im HFS Plus Dateisystem für Mac OS und Linux bietet diese Möglichkeit für EXT3/4 an, sowie für XFS, BTRFS und ReiserFS.

3.1 XFS

XFS wurde im Oktober 1993 von der Firma SGI (Silicon Graphics, Inc.) der Öffentlichkeit vorgestellt. Als Weiterentwicklung des nur intern genutzten Dateisystems EFS gilt es heutzutage als eines der häufigsten verwendeten Dateisysteme, insbesondere für Systeme mit einer großen Anzahl von CPU's, CPU-Kernen und Festplatten. Bereits im Linux Kernel 2.4 wurde XFS offiziell eingeführt. Der Hersteller verspricht sehr hohe Transaktionsgeschwindigkeiten, das bedeutet eine sehr hohe Input-Output-Rate, sehr schnelle Wiederherstellungsmechanismen, effizientes Allokieren von Speicher, eine perfekte Anbindung der Netzwerkhardware, sowie massive Skalierbarkeit.

3.1.1 Aufbau von XFS

XFS ist ein reines **64-bit** Dateisystem, welches in der Lage ist, mit $2^{63} = 9$ Exabyte Daten umzugehen. Insbesondere für Storage-Hoster ist dies ein Wert, welcher auch in der heutigen Zeit zu betrachten ist. Dazu ist XFS in der Lage, beim Verbund aus mehreren Systemen skalierbar zu arbeiten.

Das Dateisystem beinhaltet einige wesentliche Eigenschaften, die die Unterschiede zu anderen Systemen verdeutlichen. Die Eigenschaft **„GRIO“** („Guaranteed IO Bandwith“) ist dabei herauszustellen. Diese garantierte Mindestgeschwindigkeit ist besonders bei Streaming-Servern wichtig, da diese einen konstanten Datenstrom anbieten müssen. Eine weitere Eigenschaft nennt man **„Disk Quota“**. Dieses Feature besagt, dass für spezielle Benutzer des Systems Begrenzungen des Speicherplatzes eingeführt werden können.

Das Filesystem bietet spezielle Technologien zur effizienten Speicherbehandlung an, welches die Performance beim Reservieren und Freigeben von Speicher wesentlich erhöht. Möglich macht dies eine besondere Unterteilung der Zuweisung von Daten. Das Dateisystem ist dabei unterteilt in gleich große **„Allocation Groups“**, welche zwischen 16MB und 1TB groß sein können. Da diese Gruppen sehr groß werden können, ist besonders die Performance bei großen Dateien sehr gut, da diese keinen großen Overhead erzeugen, da nur sehr wenige einzelne „Allocation Groups“ ausgewertet werden müssen.

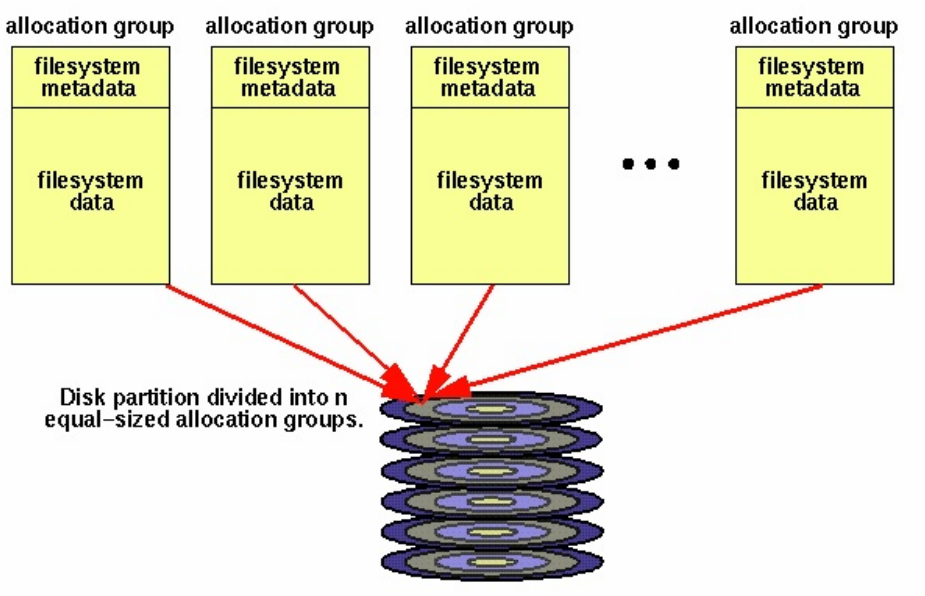

Abbildung 3.1: Aufbau des XFS-Filesystems [19]

Aufbau der Metadaten Jede Gruppe ist unterteilt in einen Metadata- und Filesystemabschnitt. Der Metadatenabschnitt ist in unterschiedliche Blöcke eingeteilt, die zum einen zum optimalen Ausnutzen des vorhandenen Speichers genutzt werden, zum anderen auch den **B+-Tree** beinhaltet, sowie die **Inode-Liste**. Ein B+-Tree ist eine Indexstruktur, welche die eigentlichen Datenelemente leidiglich in den Endknoten (Blattknoten) speichert und die inneren Knoten nur die Schlüssel beinhaltet. Das hat den Vorteil, dass ein schneller und paralleler Zugriff auf den Baum möglich gemacht wird.

Während andere Filesystems die Inodes, die die Metadaten der gespeicherten Dateien enthält, fest verankert, bietet XFS die Möglichkeit, diese dynamisch zur Laufzeit zu erzeugen und anzulegen. Das nennt man **„dynamische Inode Allkoation“**. Auch ist eine Begrenzung nicht gegeben, es ist also möglich, eine sehr hohe Anzahl von Dateien in einer einzigen „Allocation Group“ abzulegen.

3.1.2 Fehlerbehandlung

Insbesondere nach unerwarteten aufgetretenen Fehlern ist es elementar wichtig, dass aktuelle Dateisysteme in der Lage sind, diese Fehler zu korrigieren, ohne das die Anzahl von Dateien, die verwaltet werden, dabei entscheidend sind. Dieses Feature ist elementarer Bestandteil von XFS, welches unabhängig der Größe und der Auslastung des Filesystems eine vollständige Recovery-Funktion direkt mit sich bringt.

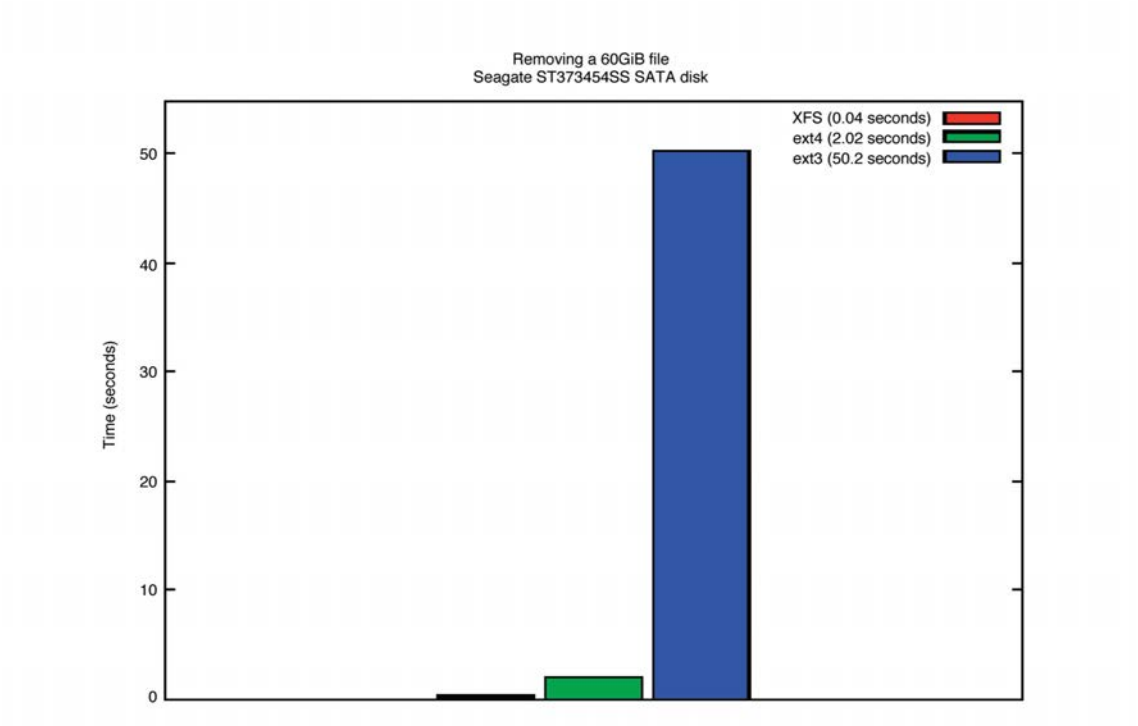

Abbildung 3.2: Performance von XFS beim Löschen von vielen Dateien [18]

3.1.3 Aktuelle Entwicklungen

XFS wird heutzutage von der Firma SGI gepflegt, was die regelmäßigen Updates auf Linux-Systemen zeigen. Auch darum ist dieses Filesystem eine interessante Alternative, sowohl für Rechnersysteme mit einer sehr hohen Anzahl von CPUs und sehr viel Speicher, als auch für den privaten Anwender, da die Möglichkeit zur Rekonstruktion bei fehlgeschlagenden Transaktionen immer eine sehr gute Möglichkeit der Datenwiederherstellung bietet. Insbesondere seit dem Jahr 2010, wo die grundlegende Struktur an die heutigen Bedürfnisse angepasst wurden, ist XFS wieder ein Filesystem, welches in den Punkten Input-Output-Performance, Sicherheit und Stabilität im Vergleich mit dem Standard-Filesystem ext4 viel Zuspruch findet, insbesondere wenn es um die Behandlung von sehr großen Dateien geht.

3.2 ext4

Das „Extended File System“ ext wurde ursprünglich in seiner ersten Version 1992 von Remy Card veröffentlicht. Während dieses Format bereits zur damaligen Zeit eine Datengröße von 2GB verwalten konnte, wurde 1993 der Nachfolger ext2 veröffentlicht, welcher zunächst 2TB, ab dem Linux 2.6 Kernel 32TB verwalten konnte. 2001 wurde das Journaling zu dem Filesystem hinzugefügt und ext3 genannt.
Journaling bedeutet, dass Änderungen am Filesystem vor dem endgültigen Schreiben auf den Datenträger zunächst in einem festgelegten Speicherbereich geschrieben wird, um zu erreichen, dass ungewollte Transaktionen wieder rückgängig gemacht werden können. Das hat zunächst einen großen Vorteil, allerdings leidet die Gesamtperformance des Systems darunter, da die Limitierung des festgelegten Journaling-Bereiches ausschlaggebend für die Geschwindigkeit ist.

Im Dezember 2008 wurde die aktuelle Version des „Extended File Systems“ ext4 vorgestellt, welche, wie im Vergleich von ext2 zu ext3, keine Erweiterung, sondern eine völlige Neuentwicklung darstellt. Insbesondere in den Punkten Performance, Skalierbarkeit und Sicherheit wurden viele Verbesserungen angebracht, dass das System auf den aktuellen Stand der Zeit bringt und damit auch in der heutigen Zeit zum aktuell am häufigsten verwendeten Filesystem auf Linux-basierten Computern ist.

3.2.1 Vorteile von ext4

Zu den Vorteilen von ext4 gehört, dass es eine vollständige Funktionskompatibilität zu ext3 beinhaltet, dass bedeutet zum einen, dass Systeme, welche noch auf dem alten Standard formatiert sind, diese in ext4 umgewandelt werden können, ohne dass Datenverlust zu beklagen ist. Insbesondere bei sehr sensiblen Daten ist ein solcher Umstieg meist nicht ohne eine gründliche Datensicherung im Vornherein möglich, ext4 bietet diese Funktion bereits an, ohne am Betriebssystem selbst Änderungen vornehmen zu müssen.

Die Größe des Filesystems und die Größe der zu speichernden Daten hat sich ebenfalls deutlich erhöht. Während ext3 nur mit maximal 16TB Speicherplatz und maximal 2TB große Dateien umgehen konnte, ist es jetzt möglich **1EB** (1 Etabyte = 1024PB, 1 Petabyte = 1024TB) zu adressieren, wobei einzelne Dateien bis zu **16TB** groß sein

können. Möglich macht dies, dass ext4 **48bit** große Blöcke adressiert, die Gegenwart zeigt, dass eine 64bit Umsetzung folgen muss, um mit anderen Filesystemen insbesondere in großen Rechnerverbunden, die sehr große Daten speichern, mithalten zu können. Positiv ist auch, dass die Grenze der möglichen Unterordner aufgehoben wurde, diese lag bei ext3 noch bei 32000.

3.2.2 Aufbau ext4

Das Dateisystem ext4 ist nicht wie XFS in der Lage 64-bit-Blöcke zu adressieren, es verwaltet 48bit große Adressbereiche. Dies stellt in der Realität kaum ein Problem dar, da die maximale Dateisystemgröße 1 Exabyte beträgt, ohne Einschränkungen in der Anzahl der Unterordner zu besitzen. Die Dateilänge ist allerdings auf 255 Zeichen begrenzt.

Um eine hohe Arbeitsgeschwindigkeit zu erreichen wurde das Adressieren und Verwalten von Speicherblöcken im Vergleich zu den Vorgängern grundlegend verändert. Während ältere Dateisysteme für jeden festgelegten Block Daten in einem Schema abspeichern („indirect block mapping scheme“ [20]) existieren in ext4 Ausdrücke, die in einem Befehl einen gesamten Speicherbereich für eine Datei darstellen. Die Größe dieser **„Extents“** ist dabei variabel, bei sehr großen Dateien ist es möglich, dass diese in mehreren „Extents“ gespeichert werden.

Ein weiterer Unterschied zu ext3 ist das **„Multiblock Allocation“**- Feature. Ext3 fragte für jeden Block, der geschrieben werden sollte den „Block Allocator“ an. Dies konnte auch schon bei kleinen Dateien zu großen Problemen führen, bei einer 100MB großen Datei und einer Blockgröße von 4KB wurde die Anfrage 25600 mal durchgeführt.

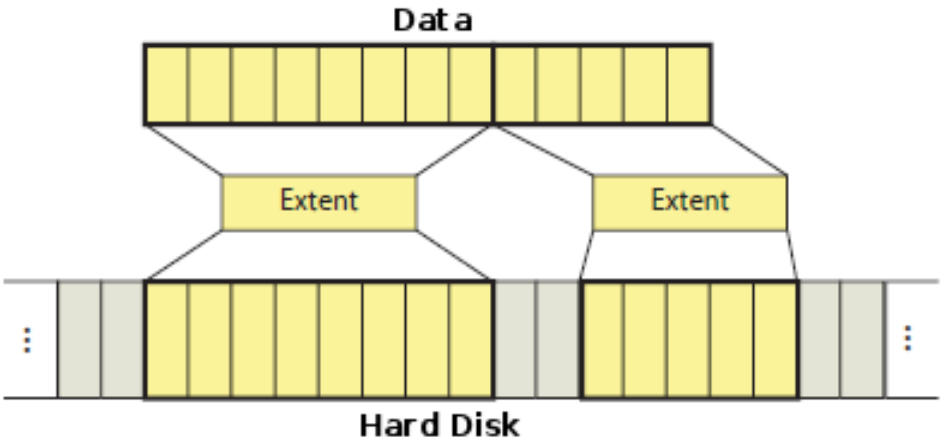

Abbildung 3.3: Extents im ext4 Dateisystem [21]

Ext4 reduziert die Anfragen auf eine einzige, indem sofort der gesamte Bereich abgefragt wird. Dies hat zusätzlich den Vorteil, dass das Filesystem die Daten besser verteilen kann, da bereits beim ersten Kontakt die gesamte Größe der zu speichernden Datei übertragen wird. Dies führt zu einer besseren Auslastung und einer höheren Performance und Effizienz des Filesystems.

Für eine direkte Zuordnungsmöglichkeit von Metadaten zu den gespeicherten Daten in den Blöcken arbeitet ext4 genau wie XFS mit sogenannten **Inode-Blöcken**.

3.2.3 Verbreitung von ext4

Sowohl XFS als auch ext4 haben eine große Entwicklungszeit überstanden und sind heutzutage die am häufigsten eingesetzten Dateisysteme. Bei Linux-Distributionen für den Endanwender, wie zum Beispiel Ubuntu, Debian, openSuSe oder auch CentOS ist dies das Standarddateisystem, bei der automatischen Installation. Dies zeigt, dass das System ausgereift und eine Performance aufzeigt, welche für den Endanwender keine Änderungen nötig macht. Auch die Stabilität ist gegeben.

Eine vollständig andere Herangehensweise zeigt BTRFS, welches sich zwar noch in Entwicklung befindet, aber bereits jetzt stabil und in vielen Fällen schneller als alle Kontrahenten ist.

3.3 BTRFS

Das neuartige Filesystem BTRFS wurde als „Next Generation Linux“ [22] vorgestellt, welches insbesondere die Probleme und Einschränkungen der standardisierten Systeme ausmerzen soll. Im Jahr 2007 stellte Oracle das Produkt offiziell vor, im Linux-Kernel 2.6.29 wurde es offiziell aufgenommen und steht für jeden Nutzer zur Verfügung.
Bis zum aktuellen Stand der Zeit ist BTRFS allerdings noch nicht als stabil gekennzeichnet worden, das bedeutet, dass es immernoch zu Fehlern beim längerfristigen Einsatz kommen kann und nicht als „sicheres“ Filesystem anzuerkennen ist. Möglicher Datenverlust bei unvorhergesehenen Fehlern ist daher leider nicht ausgeschlossen.

3.3.1 Aufbau BTRFS

Im Vergleich zu XFS und ext3/4 basiert BTRFS auf eine andere Art der Formatierung eines Datenträgers. Wie ZFS für Sun-Systeme basiert BTRFS auf das **Copy-On-Write** System. Copy-On-Write bedeutet, dass geänderte Daten in neue Adressbereiche geschrieben werden. Nach dem erfolgreichen Schreiben erfolgt anschließend die Aktualisierung des B-Trees, die Metadaten zeigen solange auf die alten Speicherbereiche. Durch diese Eigenschaft sind **Snapshots** möglich, die einen Teil des Baumes direkt in andere Bereiche auslagern kann und auch zu anderen Dateisystemen zusätzlich eingehangen werden können.

BTRFS bietet besondere Eigenschaften, die in anderen Systemen nicht zur Verfügung stehen. Die **Verschlüsselung auf Dateisystemebene** erhöht die Sicherheit und dient der besseren Zugriffskontrolle der gespeicherten Daten, die **Komprimierung** unterstützt das Speichermanagement.
Auch das Journaling wurde grundlegend verändert, BTRFS bietet eine vollständig neuartige Verwaltung der Daten- bzw. Metadaten.

BTRFS ist ein Dateisystem, welches mit **64bit** großen Adressräumen arbeitet. Das bedeutet, eine einzelne Datei kann bis zu **16EiB** groß werden. Die Größe des Filesystems ist unbegrenzt.

3.3.2 Metadatenbehandlung in BTRFS

BTRFS steht für B-Tree-Filesystem, das aufzeigt, dass die Metadaten und Datenblöcke in Baumstrukturen aufgezeichnet werden, die jeweils aufeinander zeigen. Ähnlich wie bei ext4 existieren auch hier Extents, die für große Dateien einen Speicherbereich festlegen können, ohne einen großen Overhead zu erzeugen. Die dynamischen Inode-Tabellen unterstützden dies. Durch das dynamische Abspeichern und der Möglichkeit von Copy-On-Write können einzelne Teilbäume einfach kopiert werden, somit können Snapshots erstellt werden, diese dynamisch zur Laufzeit eingebunden oder entfernt werden. Diese Teilbäume nennt man Subvolumes.

Die Unterstützung von **Software-RAID** (Redundant Array of Independent Disks [15]) können mehrere Festplatten gleichzeitig angesprochen werden. BTRFS unterstützt dabei die Formate RAID-0 („Striping“), RAID-1 („Mirroring“) und RAID-10 („Striping + Mirroring“). RAID-5 und RAID-6 sind in Planung der Entwickler. Somit können mehrere Subvolumes auf einem Datenträger angelegt und ineinander geschachtelt werden.

3.3.3 Sicherheit der Daten in BTRFS

Das Dateisystem nutzt **Checksummen**, die sicherstellen, dass die Metadaten und Datenblöcke korrekt ins Filesystem übertragen wurden. Bei der Nutzung von redundanten Speichermöglichkeiten können eventuelle Fehler automatisch korrigiert werden, indem der korrekte Wert vom anderen Medium übertragen wird.
Diese Möglichkeit wird häufig für das Anlegen von inkrementellen Backups genutzt, die bei einem Ausfall die Daten schnell wiederherstellen können.
Um solche Fehler ausfindig zu machen, existieren verschiedene Ansätze. Zum einen werden die Daten automatisch abgeglichen, zum anderen ist eine **„offline filesystem check“**-Möglichkeit eingebaut, welche Fehler finden und korrigieren kann.

Durch das Anlegen und Abspeichern dynamischer Baumeinträge ist ein Verändern der Größe des Dateisystems im laufenden Betrieb möglich, die Defragmentierung kann auch live erfolgen.

3.3.4 Rechtevergaben in BTRFS

Ähnlich wie ext4 bietet BTRFS Zugriffsbeschränkte Rechtevergaben an. Dies geschrieht über **„Access Control Lists“** (ACLs) nach dem POSIX-Standard („Portable Operating System Interface“). Die Möglichkeit der Verschlüsselung unterstützt die Rechtevergabe zusätzlich, da ausgewählt werden kann, welcher Benutzer in der Lage ist, die Daten korrekt zu entschlüsseln.
Diese Möglichkeit, bereits im Dateisystem die Rechtevergaben durchführen zu können, erhöht zudem die Sicherheit des Systems.

3.3.5 Ausblick

Durch die Möglichkeit parallelen Zugriff auf den Verzeichnisbaum zu gewähren und der verteilten Speicherung der Daten ist eine hohe Performance zu erwarten. BTRFS bietet zudem eine direkte Umwandlung von bestehenden ext3/4 Filesystemen an, in der Praxis funktioniert diese hervorragend im laufenden Betrieb.

Zum aktuellen Zeitpunkt befindet sich BTRFS bereits im offiziellen Linux-Kernel, ist aber noch als experimentell gekennzeichnet. Die Entwicklung ist nicht abgeschlossen, es werden in regelmäßigen Abständen Updates zur Verfügung gestellt.

Von vielen Expertens soll BTRFS das Dateisystem der Zukunft werden, weil die grundlegenden Ansätze bei der Entwicklung des Systems noch nicht vollständig implementiert wurden. Die Geschwindigkeit beim Lesen und Schreiben können bereits in dem frühen Entwicklungsstadium mit denen von XFS und ext4 mithalten, es ist sogar vom Hersteller versprochen, dass dieses noch schneller werden kann.
Auch die Möglichkeit, direkte Snapshots von einem Verzeichnisbaum anzulegen und somit inkrementelle Sicherungen anzulegen, ist heutzutage einzigartig und ist noch in keinem anderen Dateisystem mit eingearbeitet worden.

3.4 Zusammenfassung

In der heutigen Zeit existieren sehr viele verschiedene Dateisysteme, die auf der ganzen Welt für verschiedene Einsatzzwecke genutzt werden. Die vorgestellten Systeme ext4 und XFS sind die am häufigsten verwendeten auf Linux-basierten-Betriebssystemen, weil sich diese bereits seit fast 20 Jahren in der Entwicklung befinden und die grundlegende Performance im Umgang mit unterschiedlichen Dateigrößen und -fragmentierungen anbieten.

XFS ist ein Dateisystem, welches häufig auf NAS-Servern („Network Attached Storage“) zum Einsatz kommt, weil dieses besonders bei großen Dateien sehr performant ist. Auch durch die Eigenschaft „GRIO“ („Guaranteed IO Bandwith“) wird XFS häufig für Streaming-Server verwendet, da eine grundlegende Geschwindigkeit trotz hoher Anzahl von Transaktionen garantiert werden kann.

Das Dateisystem **ext4** ist in der heutigen Zeit das Standardsystem bei Neuinstallationen von Linux-Distributionen. Die Verbreitung kommt durch den performanten Umgang mit kleinen Dateien zustande, die häufig bei Betriebssystemen selbst zum Vorschein kommen. Diese Eigenschaft führt zu einem optischen Performancegewinn beim Arbeiten mit einer Distribution.
Die Geschwindigkeit liegt durchschnittlich mit XFS auf einem Niveau.

BTRFS gilt in der Linux-Welt als mögliches Dateisystem der Zukunft. Die Entwicklung wird in den letzten Jahren auch durch diesen Hype zusätzlich vorangetrieben, da durch den neuartigen Aufbau des Systems sich auch die Entwicklung von verteilten Dateisystemen beeinflussen lässt.
Allerdings ist es zum aktuellen Zeitpunkt nicht als stabil im Linux-Kernel gekennzeichnet worden, sodass von einem produktiven Arbeiten mit diesem System abzuraten ist.

	ext4	XFS	BTRFS
Adressgröße	48bit	64bit	64bit
max. Größe	1EiB	9EiB	16EB
max. Dateigröße	16TB	16TB	16EB
Extents	ja	ja	ja
Journaling	ja	ja	ja
Aufbau Metadaten	Tabelle	B+Tree + Inode	B+Tree
Komprimierung	nein	nein	ja
Snapshots	nein	nein	inkrementell
RAID-Konfigurationen	alle	alle	RAID0,1,10
Verbreitung	sehr hoch	hoch	gering

Abbildung 3.4: Zusammenfassung Eigenschaften regulärer Dateisysteme

Die vorliegende Tabelle in Abbildung 3.4 zeigt einen generellen Überblick der vorgestellten regulären Dateisysteme. Alle diese können mit sehr großen Datenmengen umgehen, sie unterscheiden sich hauptsächlich im Aufbau der Metadaten, sowie den einzelnen Eigenschaften.

Während BTRFS eine Komprimierung und Verschlüsselung auf Dateisystemebene anbietet, fehlen diese Punkte vollständig bei ext4 und XFS. Mit diesen können hingegen Software-RAID-Systeme mit allen verfügbaren Modi eingerichtet werden, BTRFS unterstützt zum aktuellen Zeitpunkt lediglich die Modi RAID-0, RAID-1 und RAID-10.

In den folgenden Tests dient ein reguläres Dateisystem jeweils als Untergrund. Dabei wurden alle Durchläufe mit dem ext4-Dateisystem durchgeführt, weil dieses bei kleineren Dateien performanter arbeitet als XFS. Für das Universitätsrechenzentrum sind Dateigrößen bis maximal 2GByte von Interesse, häufig befinden sich die Größen allerdings im Bereich von 10-100MByte.

BTRFS wurde im Zusammenhang der Testumgebung installiert, um die Arbeitsweise von Snapshots zu erkennen, allerdings wird von einem produktiven Einsatz des Systems abgeraten. Dieser Umstand und die möglichst hohe Realitätstreue der Testverfahren ließen die Entscheidung verdeutlichen, dass BTRFS nicht zur Auswahl steht, im Hinblick auf die Zukunft allerdings im Auge behalten werden sollte.

Kapitel 4

Distributed Filesystems

Ein Dateisystem dient der automatischen Speicherzuweisung zum Ablegen von Dateien auf einem Computer. Dateien bestehen dabei dabei aus dem eigentlichen Inhalt und den Metadaten. Die Metadaten speichern alle wichtigen Informationen ab, wie zum Beispiel die Größe der Datei, die Zugriffsrechte und das Datum der letzten Änderung.

Reguläre Dateisysteme dienen somit als **Schnittstelle** zum physischen, mit dem Computer verbundenen, Speichermedium. Ein verteiltes Dateisystem kommt hauptsächlich in Clusterverbunden zur Anwendung, dort sind die physikalischen Speichermöglichkeiten verteilt in einzelnen Servern zu finden.

Ein verteiltes System fasst die verschiedenen Speichermedien zentral zusammen und zeigt dieses als einen großen Datenträger an. Die zentrale Verwaltung ist dadurch auch möglich. Dort können unterschiedliche Partitionen und Filesysteme installiert werden. Diese dienen in diesem Fall der Zugriffskontrolle für einzelne Nutzer. Nötig dafür ist, dass es eine zentrale Kontrolle geben muss, der die Zuweisung vornimmt, welche Daten auf welchem Medium gespeichert werden.

Anforderungen an verteilte Dateisysteme Die **Zugriffstransparenz** muss gewährleistet werden, dass bedeutet, dass ein Nutzer des Systems keinen Unterschied bemerken darf, ob er eine Operation auf eine lokale Datei ausführt oder auf eine, die in einem verteilten Dateisystem abgelegt ist. In diesem Punkt spielt die **Ortstransparenz** mit ein, diese regelt, dass es keine Rolle spielt, auf welchem physikalischen Datenträger die Datei abgelegt wird.

Besonders wichtig bei solchen Systemen ist die Möglichkeit, **parallel auf eine Datei zugreifen** zu können, ohne das die Zugriffe einander beeinflussen. Bei Schreibzugriffen ist dies problematisch, dabei muss die Datei entsprechend für andere Nutzer gesperrt werden, wenn diese bereits als „in Bearbeitung“ gekennzeichnet ist. Das System muss dafür sorgen, dass bei Fehlern von Clients und des Servers keine Inkonsistenzen auftreten. Das kann auch durch eine sehr hohe Anzahl von Anfragen geschehen, diese dürfen sich nicht auf die Antwortzeiten auswirken.

Ein weiterer wichtiger Punkt ist die Hardware- und Betriebssystem-Transparenz - **Heterogenität**. Wenn in einem Serververbund unterschiedliche Hard- und Software zur Anwendung kommt ist es wichtig, dass das Dateisystem für den Anwender diese umwandelt, sodass dieser keinen Unterschied bemerkt.
Viele Distributed-File-Systems implementieren eine **Replikationsmöglichkeit** direkt. Das bedeutet, dass Daten mehrfach abgespeichert werden auf unterschiedlichen physikalischen Datenträgern, um eine möglichst hohe Ausfallsicherheit zu gewährleisten. Wenn Replikationen eingesetzt werden, darf der Nutzer aber nur eine Datei angezeigt bekommen, da die anderen nur zur Sicherung verwendet werden. Dabei ist es auch von Bedeutung, dass der korrekte Speicherungsort für den Client nicht sichtbar ist. Besonders wenn durch verschiedene Verfahren der Speicherort verändert wurde, darf dieses nicht direkt zu erkennen sein.

Ein wichtiger Punkt, der bei der Betrachtung von verteilten Dateisystemen zu nennen ist, ist das **Caching**. Caching bedeutet, dass ein gewisser Speicherplatz auf dem Client zur Verfügung gestellt wird, der die Daten zur schnelleren Bearbeitung zwischenspeichert. Das erhöht die Geschwindigkeit und sorgt für Entlastung der Netzwerkschnittstellen.
Caching hat ein sehr großes Problem, welches gelöst werden muss. Die Daten müssen immer aktuell gehalten werden. Das bedeutet, wenn die originale Datei im verteilten Dateisystem geändert wird, muss dieses entsprechend die Änderungen an jeden Client, der die Daten zwischenspeichert, weitergeben. Das gleiche gilt selbstverständlich für die andere Richtung. Dieser Konflikt ist nur dann zu lösen, wenn der Metadatenserver alle Informationen über die abgelegten Kopien beinhaltet und bei entsprechenden Änderungen diese auch an die Kopien weiterleitet. Wenn dies nicht geschieht, kann es zu Inkonsistenzen kommen.

4.1 Unterschied zu regulären Filesystems

Die Hauptaufgabe aller Dateisysteme besteht darin, Dateien und Informationen abzuspeichern und diese zur Verfügung zu stellen. Dabei darf dem normalen Nutzer nicht bekannt gemacht werden, wo die Dateien abgelegt werden und in welchem Format. Alle Systeme müssen bestimmte **Zugriffsrechte** anbieten, weil, insbesondere bei parallelen Speicherzugriffen, Inkonsistenzen auftreten können.
Jedes Dateisystem muss selbstständig die Speicherung der Daten durchführen, diese zur Verfügung stellen (entsprechende Rechte vorausgesetzt) und den verfügbaren Speicherplatz möglichst optimal ausnutzen.

Der Unterschied zwischen regulären und verteilten Dateisystemen besteht in der Art der Speicherung. Während reguläre Systeme immer einen physikalischen Speicher direkt ansprechen müssen, können bei Rechnerverbunden mehrere Speicher, die sich verteilt in den unterschiedlichen Knoten befinden, angesprochen und zusammenfasst werden, sowie in das Filesystem mit eingebunden werden.
Diese Art setzt eine schnelle Netzwerktopologie voraus, die möglichst so schnell ist wie eine interne Kommunikation. Verbindungstopologien, die häufig zum Einsatz kommen, sind Infiniband oder Myrinet, wobei der Marktanteil von Myrinet in den letzten Jahren aufgrund des proprietären Aufbaus immer weiter zurückgeht und Infiniband das am häufigsten verwendete Medium ist.

Auch wird bei diesem System mindestens ein **Hauptserver** benötigt, welcher von der Außenwelt angesprochen werden kann, um die Bereitstellung des Speicherplatzes bzw. der bereits vorhandenen Dateien regelt. Der Nutzer darf dabei nicht erkennen, ob es sich um ein reguläres oder verteiltes Dateisystem handelt, dieser bekommt in jedem Fall die Möglichkeit, eine Datei zu lesen, zu verändern oder abzulegen. Die Kommunikation über verschiedene Rechner erfordert Zeit, die bei besonders hoher Auslastung oftmals zum Flaschenhals der Kommunikation werden kann. Diese muss so schnell wie möglich sein, damit das verteilte Ablegen von Dateien zu einem Performancegewinn führen kann.

Ein weiterer großer Unterschied zwischen den Dateisystemen besteht in der unterschiedlichen Behandlung der Daten. Während bei regulären Systemen die Datei in verschiedene Objekte fester Chunkgröße zerlegt wird und anschließend in einen vorgeschriebenen Speicherbereich geschrieben werden können, wird eine Datei durch

einen Server zentral aufgespalten und über einen Verzeichnis- und Dateidienst an die entsprechenden Stellen geschrieben. Diese Aufsplittung der Daten führt zu einem sehr hohen Performancegewinn, da parallele Zugriffe möglich sind, um die Datei zu schreiben und wieder zusammensetzen zu können. Dadurch werden die Anfragen schneller bearbeitet.

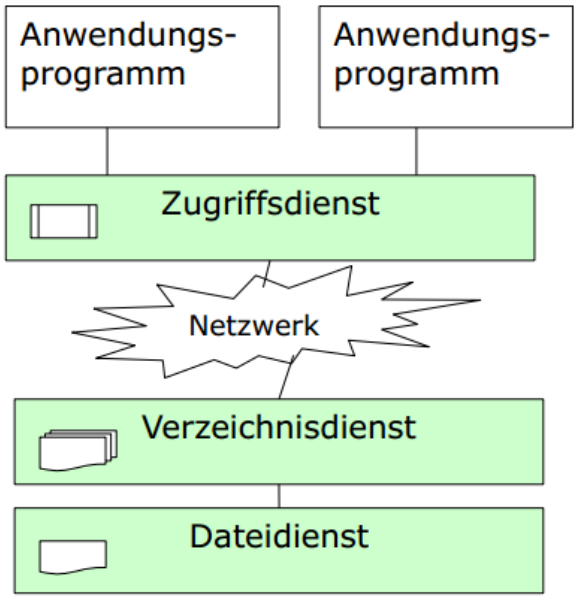

Abbildung 4.1: Grundsätzlicher Aufbau eines verteilten Dateisystems [23]

Der **Zugriffsdienst** stellt hierbei die zentrale Schnittstelle zwischen Andwendungsprogramm und Dateisystem. Dieser bildet die Aufrufe ab. Das gesamte System ist lediglich über den Zugriffsdienst von außen anzusprechen, die restliche interne Kommunikation bleibt verborgen.

Der Verzeichnisdienst verwaltet die Dateien und deren Metadaten. Dazu gehört sowohl auch die Zuordnung der Metadaten mit den Speicherorten der einzelnen Objekte, aus denen die Datei besteht, aber auch die Steuerung der Zugriffsrechte. Der Dateidienst regelt alle elementaren Operationen der gespeicherten Dateien. Dazu gehört das Lesen, Schreiben, das Setzen von Timestamps, die Allokation von Speicherbereichen und die Pufferung.

Die Gemeinsamkeit der Dateisysteme liegt heutzutage in der Behandlung der Daten. Die strikte Trennung zwischen Metadaten und den eigentlichen Werten ist in beiden Ansätzen vorhanden.
Der große Unterschied zwischen regulären und verteilten Dateisystemen besteht in der

Kommunikation. Während reguläre Systeme lediglich die physikalisch vorhandenen Speichermedien ansprechen können, ist es bei verteilten Dateisystemen möglich, mehrere Server und deren Datenträger zusammenzufassen. Die Kapazität erhöht sich somit für den Nutzer, der nun nicht mehr manuell seine Daten auf verschiedene Festplatten aufteilen muss, sondern diese zentral als eine Festplatte verwalten kann.

4.2 Kriterien zur Auswahl der Dateisysteme

Für das Universitätsrechenzentrum ist es wichtig, dass das zukünftige Dateisystem bestimmte Voraussetzungen erfüllt, damit dieses zur Auswahl stehen kann. Um eine Entscheidung für oder gegen ein Dateisystem treffen zu können, müssen Vergleichspunkte geschaffen werden, mit deren Hilfe die Unterschiede deutlich gemacht werden können. Diese sind allgemein für jedes System zur Datenspeicherung gültig.

Performance Die Performance des Dateisystems gehört zu den elementarsten Voraussetzungen von verteilten Dateisystemen. In diesem Punkt spielen sowohl die **Lese- und Schreibgeschwindigkeiten** eine Rolle, als auch die Zugriffszeiten auf eine bestimmte Datei. Viele Dateisysteme bieten Speicherkapazitäten von hunderten von Petabyte an, was die Suche nach einer speziellen kleinen Datei stark erschweren kann. Dafür muss ein effizientes **Metadatenkonzept** eingefügt werden, damit diese in möglichst geringer Zeit gefunden werden kann. Aus dem "Parallel File System Survey Report"[30] ist zu entnehmen, dass hauptsächlich in "High-Performance-Computing"Datencentern das Hauptaugenmerk auf die Performance des Gesamtsystems gelegt wird.

Zugriffsmöglichkeiten und -kontrolle Die gespeicherten Daten im Dateisystem müssen jedem Nutzer zur Verfügung gestellt werden können. Die Zugriffsmöglichkeiten beschreiben dabei, welche Möglichkeiten der Nutzer hat, diese zu verwenden. Die Zugriffskontrolle regelt zum einen die Sicherheitsaspekte, die die Rechte jeder Datei kontrollieren und die Kollisionsbehandlung, wenn mehrere Nutzer auf eine Datei zugreifen (Konsistenz). Dieses ist häufig entscheidend bei simultanen schreibenden Zugriffen.

Ortstransparenz Die Ortstransparenz gibt an, dass ein Nutzer eines Dateisystems keine Informationen über den tatsächlichen Speicherort der Datei erhalten muss. Ausreichend ist die Information über das Vorhandensein der Datei selbst.

Replikation Das Kriterium Replikation zeigt auf, ob es die Möglichkeit besteht, die Daten redundant durch eventuell mehrere Replikationen abzuspeichern, um einen

möglichen Datenverlust bei Ausfall eines Servers zu verhindern.

Fehlerverhalten Das Verhalten des gesamten Systems im Fehlerfall ist in diesem Kriterium zusammengefasst. Die Möglichkeiten reichen von automatisierten Neustarts einzelner Knoten, bis zur automatischen Wiederherstellung des Dateisystems nach Serverausfall.

Skalierbarkeit Skalierbarkeit bedeutet, dass die Arbeitsgeschwindigkeit des Clusters ansteigt, je mehr Server hinzugefügt werden. Dabei ist der Unterschied häufig groß, ob ein Metadatenserver oder Storageserver hinzugefügt wird. Das Ergebnis sollte eine Verbesserung der Zugriffsgeschwindigkeiten darstellen.

Heterogenität Dieses Kriterium beschreibt, wie das Dateisystem mit unterschiedlichen Hard- und Softwarekombinationen umgehen kann.

Administration Die Administration ist ein zentraler Punkt, der sich in mehrere Bereiche unterteilen lässt. Zum einen ist es wichtig, dass die Konfiguration des Dateisystems relativ leicht zu handhaben ist und einzelne Server ohne großen Aufwand hinzugefügt oder entfernt werden können. Zum anderen ist es positiv, wenn die Administration auch dezentral über einen entfernten Rechner durchgeführt werden kann.

Sicherheit Die Sicherheit der abgespeicherten Daten im Dateisystem darf niemals außer Acht gelassen werden. Es sollte sowohl eine Zugriffskontrolle existieren, optional ist eine direkte Verschlüsselung aller
Daten von Vorteil, um diese vor Zugriffen von nicht authorisierten Personen zu schützen.

Für das Universitätsrechenzentrum Greifswald sind selbstverständlich alle Punkte von großer Bedeutung, entscheidend sind jedoch die Kriterien Performance, Skalierbarkeit und Administration. Diese Punkte stehen neben der generellen Sicherheit im Vordergrund der Betrachtungen.

4.3 FhGFS

Das vom Fraunhofer Institut für Techno- und Wirtschaftsmathematik entwickelte verteilte parallele Dateisystem FhGFS wird als „high performance parallel file system“ [24] beworben. Es wurde entwickelt, um in High-Performance-Clusterverbunden eine hohe Datenübertragungsrate und Skalierbarkeit zu gewährleisten, und dabei höchst flexibel im administrativen Gebrauch sein. Die Nutzung des Systems ist kostenfrei, der Support wird kommerziell vom Fraunhofer Institut und deren internationalen Partnern angeboten.
Bereits in der heutigen Zeit arbeiten mehrere Supercomputer, welche in der Top500 Liste aufgeführt sind, mit diesem Dateisystem. Als Beispiel ist die Goethe-Universität in Frankfurt am Main aus Deutschland zu nennen, sie arbeitet bereits seit einigen Jahren erfolgreich damit.

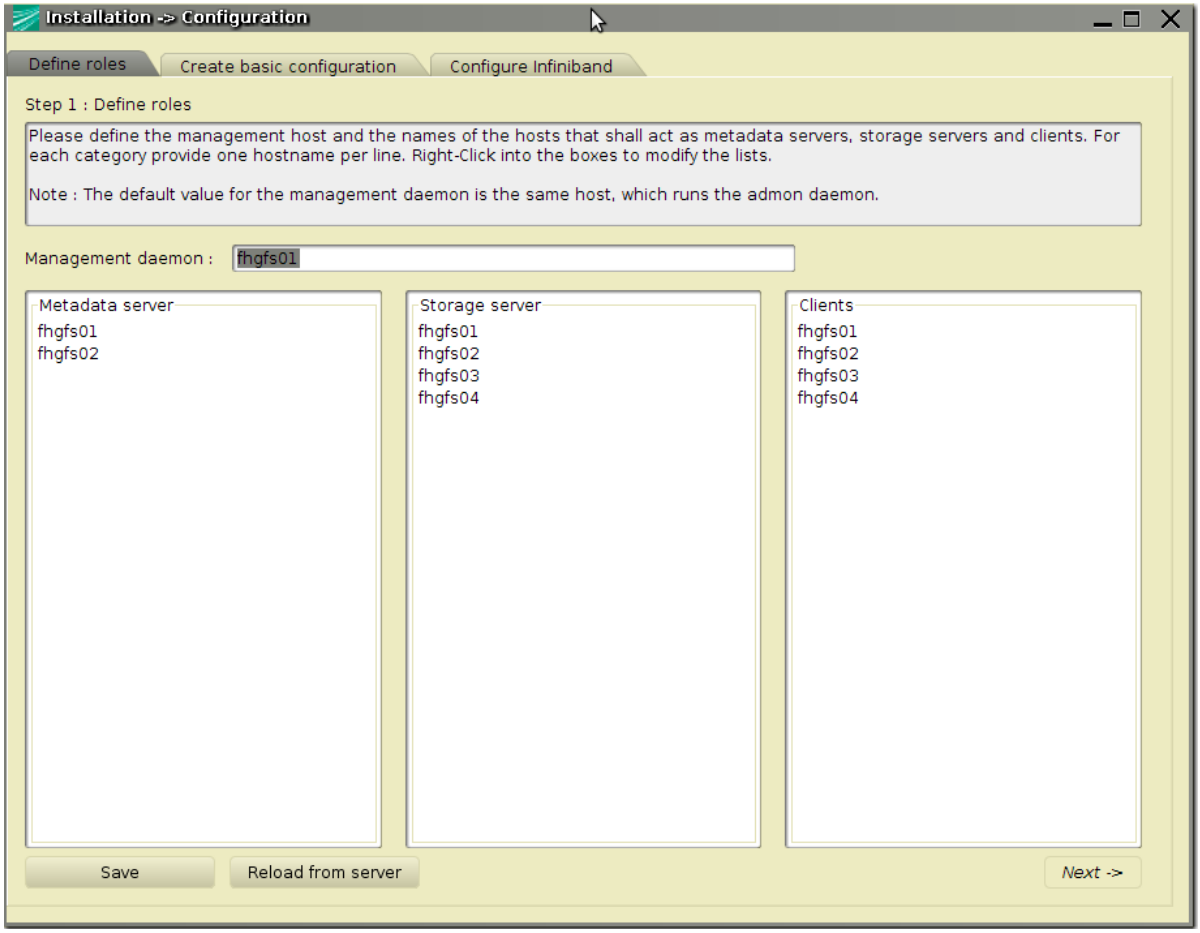

Abbildung 4.2: Administrationsoberfläche FhGFS [25]

Das Fraunhofer ITWM entwickelte ihr Dateisystem für Debian-, SuSe- und RedHat-Linux-Distributionen, ein Installation auf anderen Systemen wird nicht empfohlen, aber auch nicht völlig untersagt. Eine mögliche Test-Installation auf einem Gentoo-System war nicht möglich.

4.3.1 Interner Aufbau von FhGFS

Bei der Entwicklung des Dateisystems lag das Hauptaugenmerk auf die **Optimierung des Datendurchsatzes**. Dieser ist abhängig von der Anzahl der verfügbaren Clusterknoten, weil das System parallele Schreib- und Lesezugriffe auf mehrere verfügbare Knoten durchführen kann.

Beim Schreiben einer Datei wird diese in **„Chunks“** aufgeteilt, welche häufig eine festgelegte Größe haben, die am Kommunikationsprotokoll gekoppelt ist, um eine möglichst hohe Performance gewährleisten zu können. Es ist keine Fragmentierung beim Übertragen der Daten nötig. Häufig wird dabei Infiniband oder IPoIB (IP over Infiniband) verwendet. FhGFS bietet eine direkte Unterstützung von Infiniband an, ohne große zusätzliche Veränderungen durchzuführen.

Durch die parallele Nutzung mehrerer Server wird die Geschwindigkeit sowohl für Lese-, als auch für Schreibvorgänge erhöht. Das Ziel ist dabei ein **linearer Speed-Up** in beiden Richtungen. Spezielle Metadatenserver verwalten die Zuordnung einzelner Dateien. Durch diese Parallelität kann ein Speedup erfolgen.

Das Fraunhofer Filesystem arbeitet im Hintergrund mit verschiedenen Servern. Für eine Grundinstallation ist dabei ein Management-, ein Metadaten- und ein Storageserver nötig. Optional ist die Einrichtung eines Administrationsservers, welcher eine grafische Oberfläche zur Installation und Wartung einzelner Server zur Verfügung stellt.
Die Metadaten- und Storage-Server können beliebig während des laufenden Betriebes erweitert, bzw. aus dem Verbund entfernt werden. Dies ist über die Administrationsoberfläche oder über die Kommandozeile möglich. Die Clients müssen bei der Installation ein spezielles Programmpaket installiert haben, um auf das Dateisystem zugreifen zu können. Dieses Paket installiert einen Dienst, der automatisch das Verzeichnis mounted.

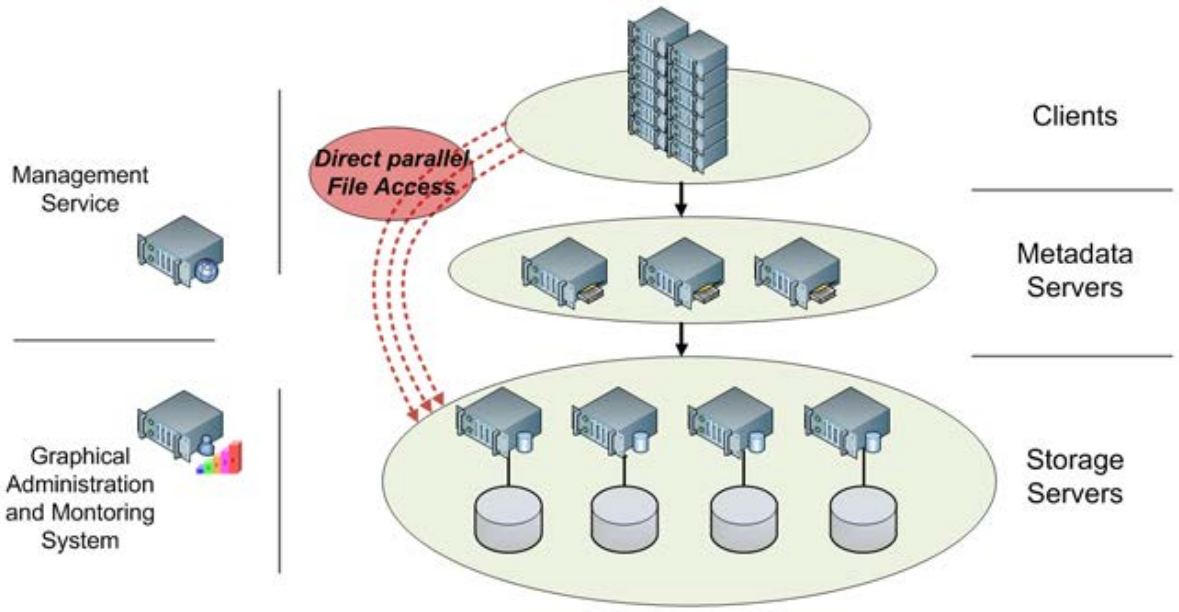

Abbildung 4.3: Kommunikation mit FhGFS [26]

Durch die dynamische Bereitstellung der angegebenen Server ist es möglich, diese nach den persönlichen Bedürfnissen einzurichten.
Es besteht die Möglichkeit, einen oder mehrere zentralisierte Metadatenserver einzurichten oder diese direkt an die einzelnen Storageservern zu binden. Dadurch wird der gesamte Aufbau sehr flexibel und vom theoretischen Standpunkt auch massiv skalierbar. Durch die interne Unterstützung von Infiniband als zusätzliche Netzwerkhardware erreicht man dynamischen Failover, wenn ein Infiniband- und Ethernetnetzwerk gleichzeitig verwendet werden, denn das Dateisystem bietet das automatische Umschalten und Auslasten beider Netzwerktopologien an.

Wichtig ist hierbei auch zu erwähnen, dass das System zum aktuellen Zeitpunkt zwei verschiedene Arten von **Caching** unterstützt. Der „buffered“-Modus kann einen kleinen Teil der Dateien lokal zwischenspeichern, die Größe beträgt allerdings nur wenige hundert KByte. Die zweite Möglichkeit ist der „paged“-Modus. Dieser greift auf den Linux pagecache zu und kann die Größe des vorhandenen Arbeitsspeichers annehmen. Dieser Modus ist allerdings vom Fraunhofer Institut noch als experimentell gekennzeichnet. Die verschiedenen Typen vom Caching können in der „fhgfs-client.conf“ angepasst werden. Im vorliegenden Testfall wurden diese ausgeschaltet, damit keine Verfälschungen in den Ergebnissen durch Caching möglich sind.

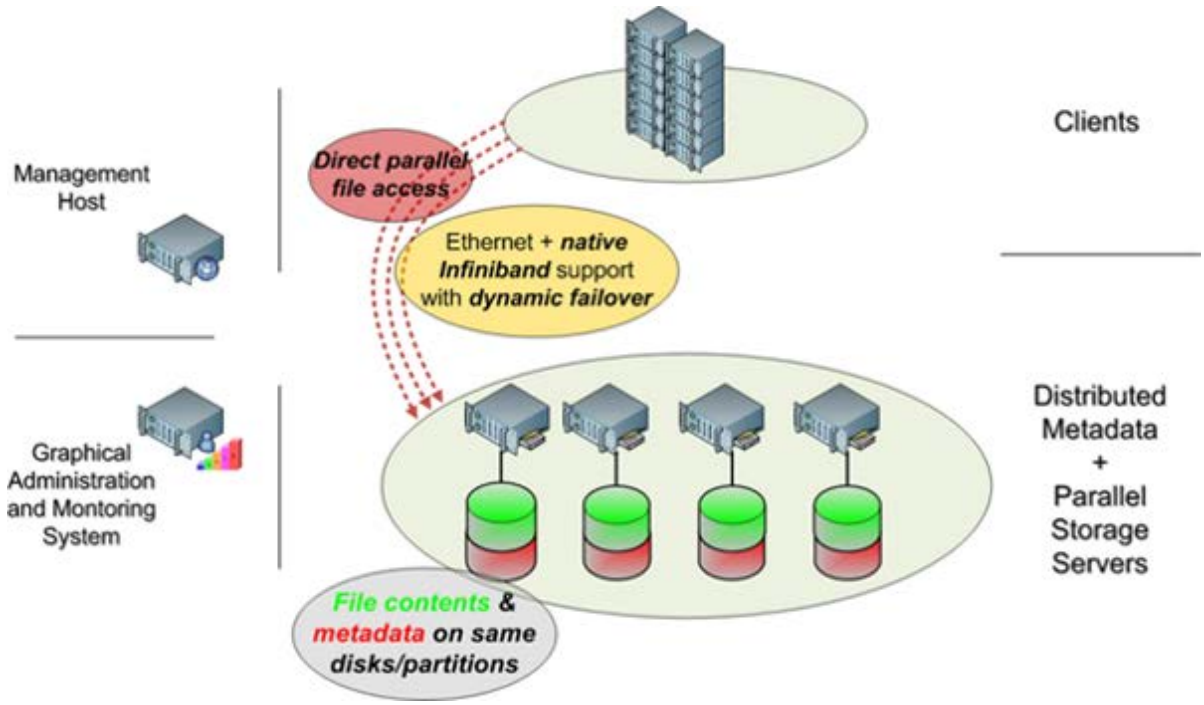

Abbildung 4.4: FhGFS möglicher Aufbau [29]

FhGFS verändert das interne Dateisystem jedes einzelnen Knotens nicht, das macht es möglich, spezielle Konfigurationen der Festplatten, wie zum Beispiel ein RAID-Verbund, auszunutzen. Diese Methode hat den Vorteil, dass Sicherheitsaspekte (z.B. RAID-Mirroring pro Server) nicht verloren gehen und Geschwindigkeitsvorteile (z.B. RAID-Striping pro Server) ausgenutzt werden können. Nachteilig ist, dass diese zusätzlich verwaltet werden müssen und die optimale Kombination für die persönlichen Anwendungen muss im Vorfeld festgelegt werden.

Das FhGFS ein verteiltes Dateisystem ist, zeigt sich im internen Aufbau. Durch das Zusammenfassen mehrerer Server und der Darstellung als einzelne Storage-Ressource nach außen, sowie die strikte Trennung zwischen Metadaten und den eigentlichen Dateiinhalten führen zu einer sehr guten Skalierbarkeit, hoher Flexibilität und hoher Arbeitsgeschwindigkeit.

Das Fraunhofer Institut verspricht in späteren Releases die Implementation von Replikationsservern, die Verfügbarkeit der Daten trotz eines Serverausfalls erhöhen. Zum aktuellen Zeitpunkt verfügt FhGFS nicht über die Möglichkeit, Mirroring automatisch durchzuführen. Lediglich durch die Ausführung vorgefertigter Skripte können Kopien des Metadaten- und Storageservers erstellt werden. Diese sind noch nicht automatisiert verfügbar und müssen vom Administrator manuell gepflegt werden.

4.3.2 Administration

Ein wichtiges Entscheidungskriterium für das Universitätsrechenzentrum ist der Punkt der Administration. Wichtig hierbei ist die einfache Bedienung.
Da das Fraunhofer Institut eine grafische Oberfläche zur Installation und Administration anbietet, kann nach der Grundinstallation ohne besondere Kenntnisse die gewünschte Konfiguration eingerichtet werden. Die Flexibilität zeigt sich besonders in diesem Punkt, da neue Clients und Server hinzugefügt werden können, ohne das System herunterfahren zu müssen.
Die Oberfläche bietet zudem eine Zustandsüberwachung, sowie Nutzungsstatistiken an, die sofort abrufbar sind. Diese können für jeden Knoten einzeln oder in der Gesamtübersicht der einzelnen Serverstrukturen abgefragt werden.
Wichtig ist hierbei anzumerken, dass die automatisierte Installation durch die grafische Oberfläche nur möglich ist, wenn dieser Server, der den Dienst bereit stellt, „passwordless ssh“ unterstützt und zwar für den Root-Nutzer. Zudem ist eine Internetverbindung nötig, da die benötigten Pakete durch den internen Paketmanager des Linux-Betriebssystems automatisch heruntergeladen werden. Diese Pakete werden für Suse-, RedHat-Enterprise- und Debian-Distributionen angeboten. Besondere Arten werden nicht direkt unterstützt, die Installation wird vom Fraunhofer Institut nicht empfohlen, aber auch nicht verboten.

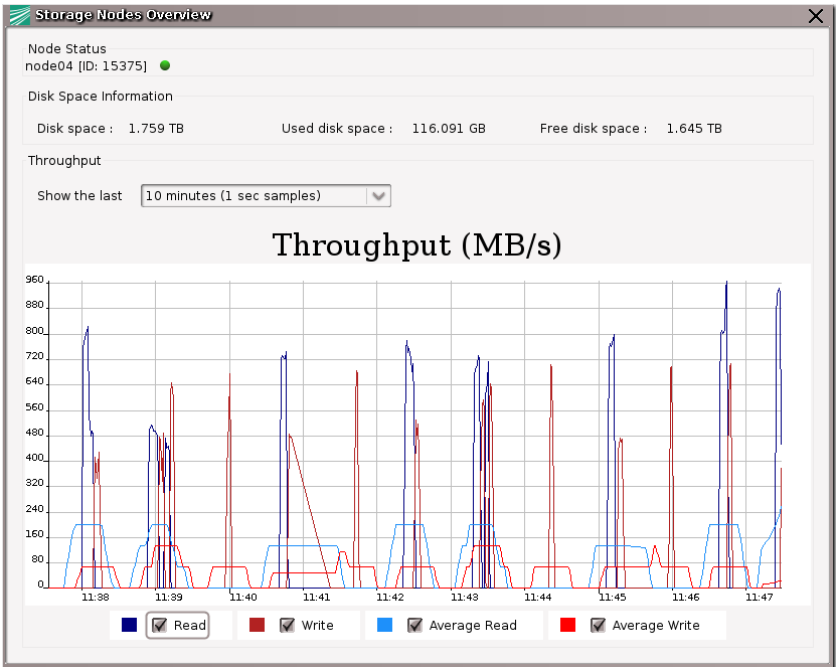

Abbildung 4.5: Administrationsoberfläche von FhGFS am Beispiel Datendurchsatz

4.3.3 Vor- und Nachteile von FhGFS

Das Fraunhofer Filesystem hat den Vorteil, dass die Verteilung der Aufgaben völlig flexibel und schnell einzurichten ist. Während des laufenden Betriebs können zusätzliche Knoten hinzugefügt und entfernt werden. Dies gilt für die Metadaten- und Storageserver, selbstverständlich auch für die Clients. Die Aufteilung der Daten erfolgt automatisch im Hintergrund.

Auch die Installation der einzelnen Server kann automatisiert durchgeführt werden. Durch die zur Verfügung stehende grafische Oberfläche ist es möglich, einzelne Server anzusprechen, denen eine bestimmte Rolle zuzuweisen und automatisch die benötigten Daten zu überspielen.
Allerdings benötigt der Hauptserver, wo der Administrationsdienst gestartet wurde, für die Installation eine funktionierende Internetverbindung, da dieser die Pakete einzeln herunterlädt. Manuell ist die Installation auch möglich, die Installationsdateien liegen für RedHat-Enterprise-, Debian- und Suse-Linux-Distributionen vor. Die Installation der einzelnen Pakete auf einem Gentoo-Betriebssystem war nicht möglich, laut Aussage des Fraunhofer Institutes wird allerdings an einer einheitlichen Umsetzung gearbeitet.

Die Kombination verschiedener Betriebssysteme ist ebenso möglich, wie die unterschiedliche Konfiguration und Partitionierung vorhandener Massenspeicher. FhGFS arbeitet dabei mit verschiedenen RAID-Kombinationen zusammen und unterstützt alle regulären Dateisysteme wie ext4, XFS und BTRFS als Untergrund.

Durch die grafische Oberfläche bekommt der Administrator einen guten Überblick über die vorhandenen Metadaten- und Storageserver. Es können sowohl die Betriebszeiten, die Performance der einzelnen Server, als auch die Auslastung derer überblickt werden. Dadurch können Engpässe schnell erkannt werden. Außerdem kann bei Anbindung von FhGFS an einen Mailserver dieser automatische Nachrichten verschicken, wenn es zu Fehlern kommt. FhGFS zwingt diese grafische Umsetzung nicht auf, die einzelnen Server können zusätzlich über Konsolenbefehle administriert werden.

Zusätzlich ist es möglich, den Metadatenserver vollständig im Arbeitsspeicher unterzubringen. Durch die fehlende Replikationsmöglichkeit, die noch nicht vollständig implementiert wurde, ist davon allerdings abzuraten, da bei einem Stromausfall diese vollständig zerstört wären und die Daten im System unbrauchbar sind.

4.4 MooseFS

MooseFS ist ein open-source-Projekt, was damit wirbt „A File System for highly reliable petabyte storage“ [33] zu sein. Im Jahr 2009 ist die erste offizielle Version 1.6.5 der Öffentlichkeit vorgestellt worden und wird seit dem als open-source-Projekt gepflegt und weiterentwickelt. Das Hauptaugenmerk liegt auf die **Hochverfügbarkeit**, sowie auf die hohe Flexibilität in der Einrichtung und Erweiterung des Dateisystems.

Hochverfügbarkeit wird bei MooseFS durch die automatische Replikationsmöglichkeit gegeben, indem Daten zunächst auf einem Server abgelegt werden, dieser aber selbstständig Replikationen anlegt.

Es stehen alle Dateioperationen zur Verfügung, wie sie von regulären Dateisystemen auch angeboten werden. Dazu gehören die Operationen Lesen, Schreiben und Ausführen.
Bei der Entwicklung wurde auch großen Wert auf die Sicherheit der Daten gelegt. Durch die Limitierung der Zugriffsrechte auf bestimmte Dateien und Verzeichnisse auf Basis von IP-Adressen oder festgelegten Passwörtern erschwert dieses Dateisystem die Zugriffe von fremden Nutzern. Hinzu kommt, dass gelöschte Dateien für eine definierte Zeit in einem Dateisystem-Papierkorb verschoben werden, bevor diese endgültig gelöscht werden. Dadurch ist eine Wiederherstellung in einer bestimmten Zeit möglich.

In Europa findet man MooseFS derweil in verschiedenen Einsatzgebieten vor. Die offizielle Homepage www.moosefs.org bietet einen Service an, mit dessen Hilfe die Verbreitung des Dateisystems in offiziellen Unternehmen und Universitäten auf der ganzen Welt aufzeigt. Als Beispiel arbeitet in Deutschland die Firma „Adrodev GmbH“ mit dem Dateisystem in einem großen Fileserver, von anderen Unternehmen wird das Dateisystem als Backup-Lösung oder im Webhostingbereich eingesetzt.

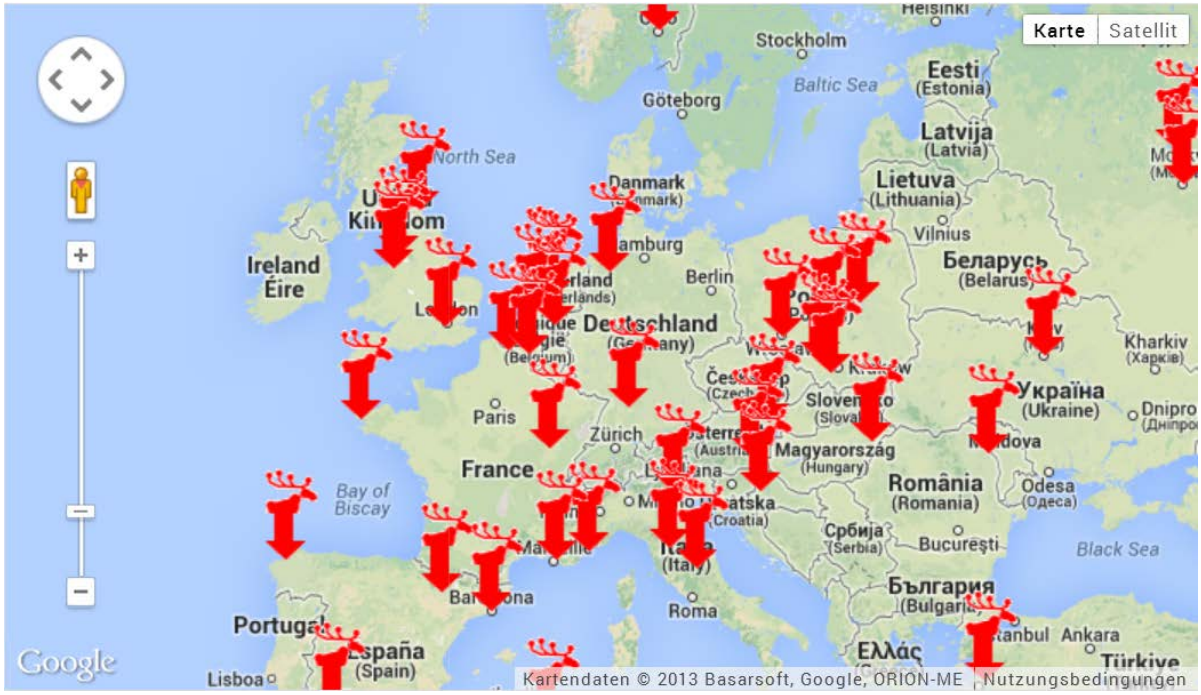

Abbildung 4.6: Einsatz von MooseFS in Europa [34]

4.4.1 Interner Aufbau von MooseFS

Der Grundaufbau von MooseFS besteht aus vier verschiedenen Komponenten. Der „Master-Server“ dient als Managementserver, welcher die gesamten Metadaten aller gespeicherten Dateien beinhaltet. Dazu gehört sowohl die Dateigröße, als auch die verschiedenen Attribute und der Speicherort. Darüber hinaus werden alle benötigten Informationen über die gesamte Architektur gespeichert, wie zum Beispiel die einzelnen Verzeichnisse, die unterschiedlichen Netzinterfaces und die angeschlossenen Geräte.

Die zweite Komponente sind die Datenserver, hier „chunk servers“ genannt. Diese sind zuständig für die Speicherung aller Verzeichnisse und Dateien. Diese erfüllen noch eine weitere wichtige Eigenschaft. Wenn mehrere Server zur Verfügung stehen, können diese selbstständig bei geringer Last die Synchronisierung einzelner Dateien durchführen, die auf mehreren Replikationsservern gespeichert werden sollen. Die neuen weiteren Speicherziele müssen mit einer kurzen Nachricht im „Master-Server“ bekannt gemacht werden.

Um Replikationen durchführen zu können, existieren bei MooseFS Metadata-Backup-Server - „megalogger server“. Diese Server speichern die Veränderungen der Metadaten, außerdem machen diese in regelmäßigen Abständen

eine Sicherheitskopie des Metadatenservers. Dadurch ist ein Weiterarbeiten möglich, auch wenn der „Master-Server“ ausfällt.

Die vierte Komponente in der Architektur ist der Client - „mfsmount“, welcher den Zugriff auf das Dateisystem ermöglicht und das Verzeichnis auf dem Client-Computer mounted. Außerdem regelt es die Kommunikation mit dem Master-Server für die Übertragung der Metadaten, sowie mit den Chunkservern, die die Dateien und Verzeichnisse zur Verfügung stellen. Zu den Operationen gehört das Schreiben und Lesen von Dateien, das Auslesen von Verzeichnissen, das Verändern vorhandener Dateien, sowie die Änderung vorhandener Dateiattribute.

Damit dies möglich ist, benötigt der Client zusätzlich das **„FUSE“**-System. „FUSE“ steht dabei für „Filesystem in Userspace“ und dient für die Kommunikation. Das ermöglicht die Nutzung des Clients auf Linux, FreeBSD und MacOSX-Betriebssystemen. „FUSE“ ist dabei ein Kernelmodul, welches ermöglicht, die Dateisystemoperationen aus dem Kernel in den Userspace zu portieren, sodass Programme auch solche Operationen ausführen können.

Für die Kommunikation von „mfsmount“ ist eine direkte Kommunikation mit dem „Master-Server“ und den vorhandenen Chunkservern nötig.

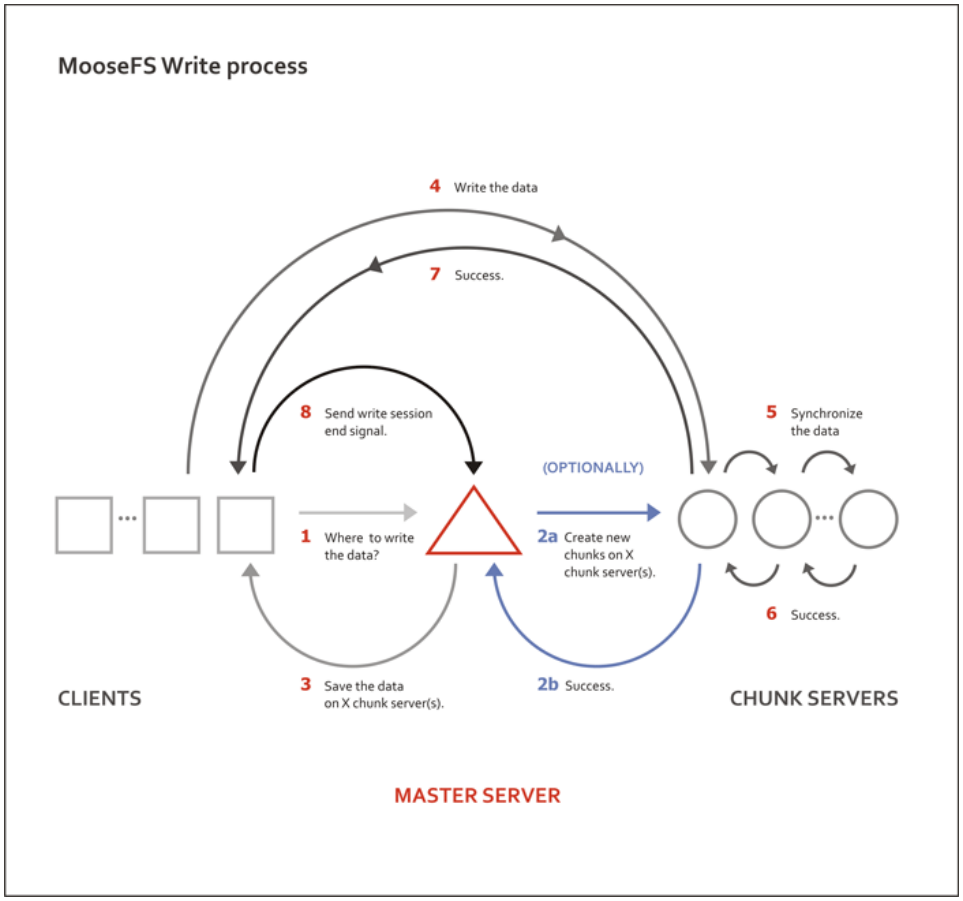

Abbildung 4.7: Schreibzugriff auf MooseFS [33]

Wenn ein Client eine Datei auf das Laufwerk schreiben möchte, welches von MooseFS bereitgestellt wird, wird eine Anfrage an den „Master-Server“ gestellt, welcher die Informationen über die Architektur und die Metadaten beinhaltet.
Dieser allokiert auf den verfügbaren Chunkservern die benötigte Anzahl von „Chunks“. Hierbei erfolgt sofort eine Verteilung, wenn mehrere Chunkserver verfügbar sind. Anschließend wird dem Client die Information bereitgestellt, auf welchen Servern dieser die Daten ablegen kann. Die Kommunikation erfolgt nun ausschließlich zwischen Client und den Chunkservern.

Anschließend erfolgt die automatische Replikation, wenn Replikationsserver eingerichtet wurden. Nach erfolgreichem Schreiben bekommt der Client die Nachricht, dass die Datei erfolgreich gespeichert wurde. Der Client sendet abschließend die Informationen an den „Master-Server“, der die Metadaten verändert. Bei einer Veränderung einer bereits vorhandenen Datei wird die neue Dateigröße, sowie das Attribut der letzten Veränderung abgespeichert.

Durch die automatische Verteilung der Daten auf verschiedene Server ist davon auszugehen, dass eine hohe Schreibperformance möglich ist, da bereits vom „Master-Server“ der benötigte Bereich allokiert wurde und somit die Daten mit höchstmöglicher Geschwindigkeit übertragen werden können.

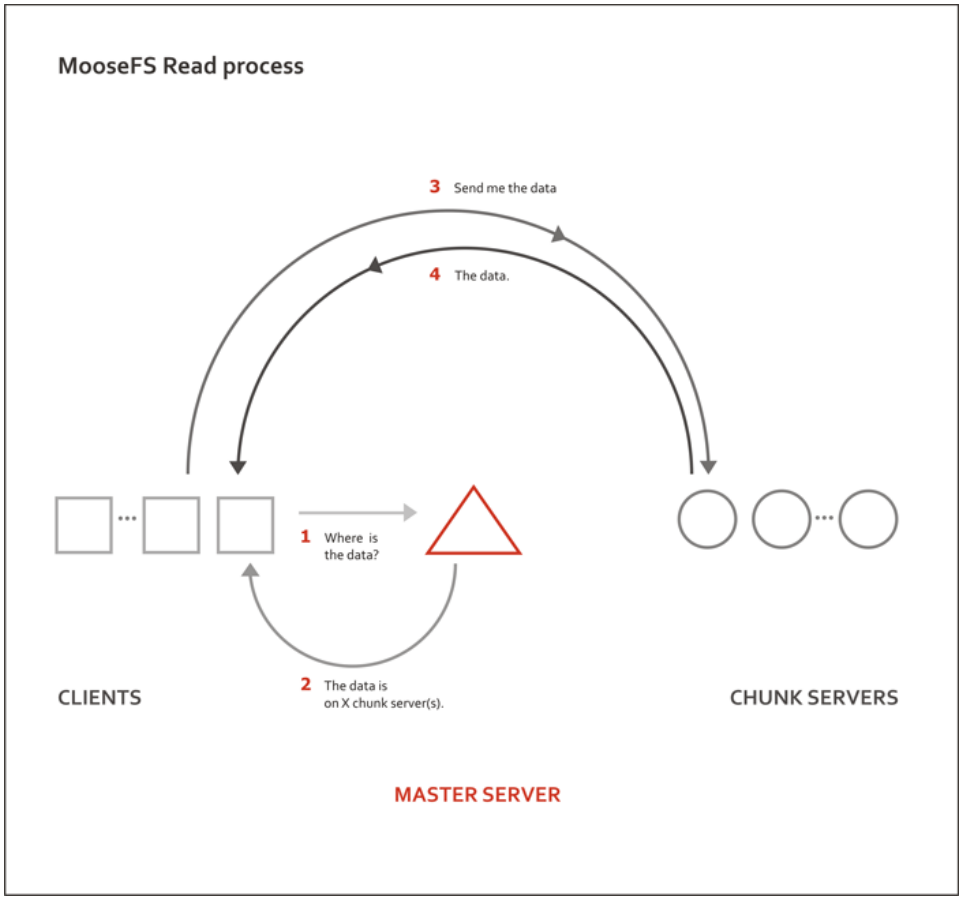

Abbildung 4.8: Lesezugriff auf MooseFS [33]

Der Lesezugriff ist einfacher aufgebaut. Der Client fragt beim Master-Server eine Datei an, dieser kennt bereits den korrekten Speicherort, gibt diesen an den Client zurück, sodass dieser eine erneute Anfrage an den Chunkserver stellt mit den Informationen über den genauen Speicherort. Dieser schickt die geforderte Datei an den Client.

Der Aufbau von MooseFS sieht zunächst sehr kompliziert aus, zeigt in der Realität viele positive Eigenschaften. Insbesondere der Leseprozess ist sehr einfach, da der Metadatenserver den genauen Speicherort der Datei kennt und diese Informationen dem Client bereitstellt, sodass dieser direkt mit den Chunkservern direkt kommunizieren kann.

Mit der Hilfe von „FUSE“ werden alle Operationen des Betriebssystems übersetzt in das korrekte Format, welches mit „mfsmount“ an die entsprechenden Server übertragen wird. Dabei ist das „FUSE“ unersetzlich, da sonst die Kommunikation nicht zu stande kommt.

Bei der Einrichtung des Dateisystems ist es direkt möglich, die Anzahl von Replikationen einzurichten. Diese können selbstverständlich nur so groß sein, wie die Anzahl der vorhandenen Chunkserver. Dieses **„goal“**, die Anzahl der Kopien, kann auf Verzeichnisse und einzelne Dateien angewendet werden.

Durch das Heraufsetzen dieses Wertes wird eine automatische Fehlertoleranz gewährleistet, da die Daten nun mehrfach abgespeichert werden und im Fall eines Ausfalles einzelner Hardwarekomponenten trotzdem zur Verfügung stehen.

Somit müssen Hardwareausfälle in einem einzelnen Knoten oder teilweise Netzausfälle nicht dazu führen, dass die Dateien nicht mehr verfügbar sind. In dem Fall, dass ein Server ausfällt, wird dieses vom „Master-Server“ registriert und automatisch der Replikationsserver angesprochen. Dies führt dazu, dass der Nutzer keine Ausfälle mitbekommt.

Wenn diese angeforderte Datei nun verändert zurückgeschrieben wird, wird automatisch die Datei zunächst auf einem anderen Server abgelegt, sodass immer die Anzahl der Replikationen eingehalten wird. Wenn der defekte Server wieder verfügbar ist, dann erhält dieser die neue Datei und ist wieder der Hauptchunkserver.
Der temporäre Chunkserver erhält die Nachricht, dass die Datei gelöscht werden kann, weil diese nicht mehr benötigt wird. Außerdem werden die Metadaten des Master-Servers aktualisiert.

Der Punkt des **Cachings** wurde in MooseFS noch nicht weiter verarbeitet. Ein Client hat keine Möglichkeiten, bestimmte Werte zwischenzuspeichern. Das bedeutet, dass für jede Datei erneut eine Anfrage an den Server gestellt werden muss.

Das Dateisystem der einzelnen Clusterknoten wird mit MooseFS nicht verändert. Es arbeitet mit ext3/4, XFS und auch BTRFS zusammen. Die Daten werden in einem speziellen Verzeichnis gespeichert und sind direkt auf dem Server nicht als reine Datei zu erkennen, da die Zerlegung in Objekte - Chunks - bereits vom Client vorgenommen und verteilt verschickt wurde. Auch die Nutzung interner RAID-Konfigurationen ist

unabhängig von MooseFS. Offiziell exisiteren dabei keine Begrenzungen. Die Begrenzung der Dateigröße liegt bei 2TB, die Begrenzung des gesamten Dateisystems liegt bei 16EiB.

4.4.2 Administration

Ähnlich wie FhGFS bietet MooseFS eine administrative Oberfläche, die per HTTP-Port direkt angesprochen werden kann. Daher ist eine spezielle Installation eines weiteren Dienstes nicht nötig, es muss lediglich auf dem „Master-Server“ der entsprechende Dienst „mfscgiservice“ gestartet werden. Dabei ist jeder HTTP-Port möglich, standardmäßig ist „9425“ geöffnet.

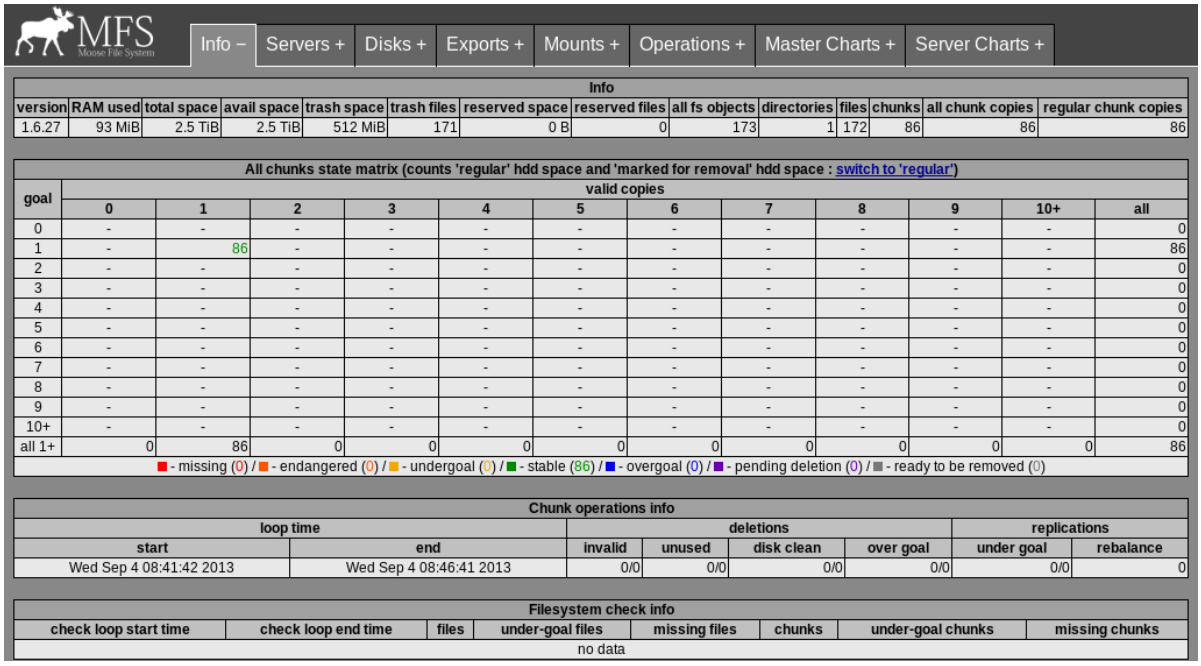

Abbildung 4.9: Administrationsübersicht von MooseFS [33]

Der Zugriff auf das System wird über einen üblichen Browser angesteuert. Dies setzt voraus, dass der „mfscgiservice“ gestartet wurde. Eine automatisierte Installation ist nicht vorgesehen, die Anzeigen dienen lediglich der Information über den aktuellen Zustand, sowie der Verteilung der Daten und der Auslastung der Festplattenkapazitäten auf den verschiedenen Chunkservern. Defekte werden zudem angezeigt, sodass die Ausfallzeit eines Knotens möglichst gering gehalten werden kann.

Die grafische Oberfläche bietet zwar keine Einstellmöglichkeiten, sie zeigt detaillierte

Informationen über einzelne Hardwarekomponenten aller verfügbaren Knoten an. Während einzelne Festplatten und Partitionen angezeigt werden können, erhält der Nutzer auch eine Übersicht über die einzelnen Verteilungen der Daten auf den unterschiedlichen Partitionen. Auch die Geschwindigkeit einzelner Komponenten wird angezeigt, außerdem wird eine Übersicht gegeben, wie viele Chunks aktuell gespeichert sind.

Die Installation einzelner Server muss manuell für jeden einzelnen Knoten eingerichtet werden. Da aktuell lediglich ein Installationspaket vorhanden ist, muss die Konfiguration entsprechend für jeden einzelnen Server angepasst werden. So muss für eine Storageserverinstallation darauf geachtet werden, dass dieser keine Metadaten verwaltet soll und auch die grafische Oberfläche nicht installiert wird. Auch die Konfiguration muss manuell getätigt werden, da der Dateiname der Konfigurationsdateien manuell verändert werden muss.
Der Umstand, dass die „configure“-Datei mit entsprechenden Befehlen bereits vor der Installation angepasst werden muss, sowie die manuell benötigte Umbenennung der einzelnen Konfigurationsdateien führt dazu, dass die Installation sehr umständlich wird. Insbesondere bei der Installation auf besonders vielen Knoten eines Clusters ist der Aufwand größer als bei der autmatisierten Installation von FhGFS.

Bei der Entfernung eines Storageservers muss die entsprechende „mfs_mounts.conf“ Datei angepasst werden, dass dieser Pfad nicht mehr zur Verfügung steht. Nach dem Neustart erkennt dies der „Master-Server“ und verteilt die vorhandenen Daten auf andere Knoten. Anschließend kann der Server entfernt werden, ohne das es zu einem möglichen Datenverlust kommt. Allerdings erfolgt keine Meldung, wenn dieser Prozess abgeschlossen ist, der Administrator muss eine entsprechende Zeit warten bis dass alle Daten korrekt übertragen wurden.

4.4.3 Vor- und Nachteile MooseFS

Auf dem Papier wirkt MooseFS als ein sehr performantes, sicheres, flexibles und durchdachtes Dateisystem. Die Möglichkeit der direkten Implementation von Replikationen dient der Sicherung von Daten und erhöht die Hochverfügbarkeit durch automatischen Failover. Bei einem Hardwareausfall werden die Replikationsserver angesprochen und die Veränderungen temporär und selbstständig auf einen anderen Server verteilt. Beim Wiedererscheinen des Hauptchunkservers werden die Daten eigenständig auf den aktuellen Stand gebracht.

Die Grundüberlegungen über die Zugriffsmöglichkeiten sind sehr durchdacht und ermöglichen die Sicherstellung von Dateien und verbietet den Zugriff Fremder, wenn dieses nicht gestattet ist.

Das interne Dateisystem der einzelnen Knoten des Compute-Clusters wird nicht verändert, alle standardmäßigen Dateisyteme werden dabei unterstützt. Dabei können auch alle verfügbaren RAID-Konfigurationen, die entweder Geschwindigkeits- oder Sicherheitsvorteile anbieten, genutzt werden. Die Daten sind trotz virtuellem System noch verfügbar.

Die Flexibilität des Dateisystems wird dadurch gewährleistet, dass zur Laufzeit bestimmte Server hinzugefügt oder entfernt werden können.

Es ist zu erwähnen, dass das Kernelmodul „FUSE“ genutzt werden muss, welches die Kommunikation mit einem Dateisystem aus dem Kernel in den Userspace umlegt. Dieses ist für die Verbindung mit dem „Master-Server“ elementar wichtig. Der Punkt ist als neutraler Kritikpunkt anzusehen, weil die Nutzung eines zusätzliches Programms zur Benutzung des Dateisystems nicht immer positiv ausfallen muss. Die Anpassungen einzelner Teile der Kommunikation müssen auf beiden Seiten eingepflegt werden. Wenn die Entwickler von „FUSE“ generelle Änderungen vornehmen, müssen die in MooseFS auch angepasst werden.

4.5 CephFS

CephFS ist ein paralleles verteiltes Dateisystem, welches Dateien in Objekte zerlegt und diese auf verschiedene Storageserver ablegt. Dabei soll dieses eine sehr gute Performance, Skalierbarkeit und Sicherheit bieten. Ceph ist ein Open-Source-Projekt, welches frei verfügbar und einsetzbar ist. Im Jahr 2006 wurde das erste Paper über Ceph veröffentlicht, seit 2010 ist dieses offiziell in den Linux-Kernel übernommen worden.
Im August 2013 wurde eine aktualisierte Version zur Verfügung gestellt. Laut des Herstellers ist ein produktiver Einsatz allerdings noch nicht empfohlen, weil nicht verhindert werden kann, dass in dem frühen Entwicklungsstadium, in dem sich Ceph befindet, Fehler auftreten und es somit zu Datenverlust kommen kann.
Trotz dieses Umstandes ist Ceph ein großes Thema und zählt als potentieller Primus aller bereits bekannten verteilten Dateisysteme.

4.5.1 Interner Aufbau von CephFS

Das Dateisystem von Ceph besteht aus mindestens vier Komponenten. **OSD** steht für "Ceph OSD Daemonünd ist ein Storageserver. Dieser speichert die Daten, verwaltet die Replikationen und kann automatisch Recovery-Befehle ausführen, damit Daten bei einem Serverausfall weiter zur Verfügung stehen. **„Ceph Monitor“** verwaltet die Struktur der einzelnen Server, das bedeutet, dass Maps angelegt werden, welcher Server wofür zuständig ist und speichert die Änderungen ab. Die **„Ceph Metadata Server“** dienen der Speicherung der Metadaten der Dateien, die auf dem Storageserver gespeichert wurden. Die Clients mounten mit Hilfe von „FUSE“ („Filesystem in Userspace“) ein für sie vorgegebenes Netzlaufwerk.

Diese können mit Hilfe des Befehls „CEPHX“ verschlüsselt auf bestimmte Bereiche auf dem Dateisystem zugreifen. Die Verschlüsselung wird mit dem Austausch eines geheimen Schlüssels durchgeführt, welcher von jedem Client angelegt und dem Monitor bekannt gemacht werden muss.

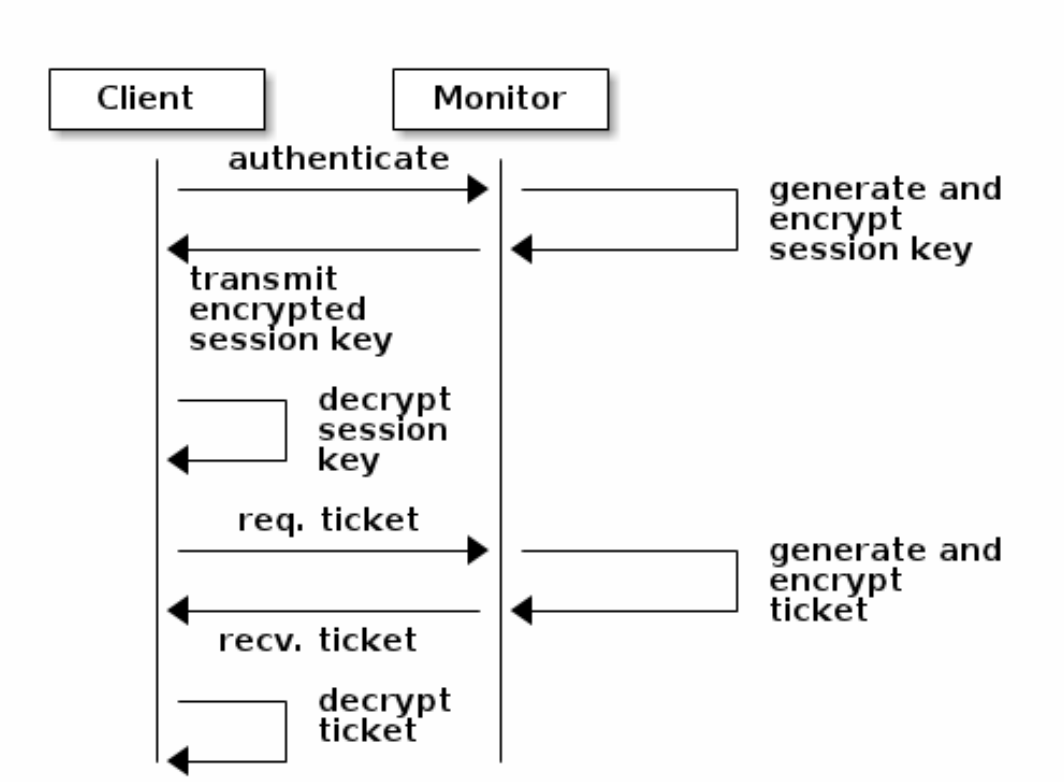

Abbildung 4.10: Authentifizierung eines Clients bei CephFS [35]

Mit Hilfe von mehreren Metadatenservern ist es möglich, die Aufgaben zu verteilen, standardmäßig übernehmen diese die Speicherung von Unterordnern des Dateisystems. Im Gegensatz zu der Konkurrenz berechnet Ceph die Blockgröße einer Datei, die geschrieben werden soll und allokiert diese sowohl im eigenen Metadatenspeicher, als auch auf dem Storageserver.

Durch die integrierte Möglichkeit, Replikationen anzulegen, die sowohl für Metadaten- und Storageserver möglich sind, ist eine automatische Lastverteilung und zugleich die Sicherung der Daten gewährleistet, da diese auf mehreren Servern gespeichert und zur Verfügung gestellt werden können.

Genau wie MooseFS kann Ceph mit FUSE auf das Dateisystem zugreifen, allerdings steht hierbei eine angepasste Version zur Verfügung, diese verspricht eine bessere Performance und Ausfallsicherheit.
Die Unterscheidung zwischen Metadaten- und Storageserver ist typisch für verteilte

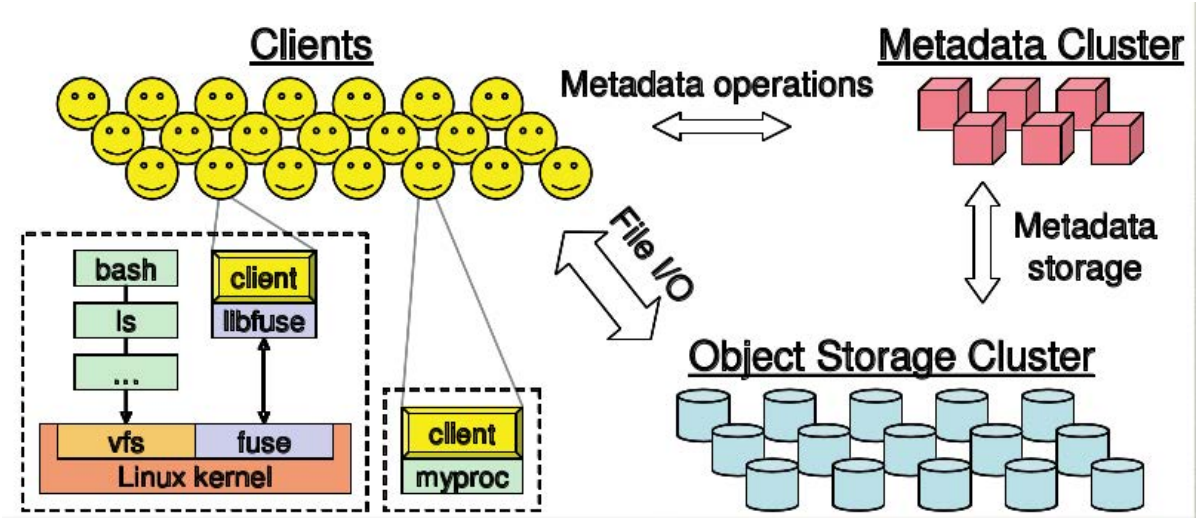

Abbildung 4.11: Transaktionsübersicht CephFS [36]

Dateisysteme. Die Metadatenoperationen werden dabei ausschließlich zwischen dem MDS und dem Client ausgehandelt. Bei einem Schreibzugriff wird der reservierte Block vom Metadatenserver an den Client übergeben, dieser kann seine Daten direkt in diesem Bereich schreiben.

Beim Lesen verhält es sich dabei ähnlich. Die Anfrage wird an den MDS gestellt, dieser gibt den konkreten Speicherort zurück und die Kommunikation der Daten erfolgt aussschließlich über Client und OSD. Bei einem einfachen Dateilisting wird lediglich der Metadatenserver angesprochen, da dieser die gesamte Struktur gespeichert hat. Um diese Einträge möglichst schnell zu finden, verwendet Ceph einen Suchalgorithmus, der **„CRUSH“** genannt wird. „CRUSH“ ist eine Art Hashingfunktion, die mit Hilfe der inode-Nummer die Einträge wesentlich schneller finden kann. Auch wenn mehrere MDS zur Verfügung stehen ist die Performance beim Dateilisting sehr gut.

Die Replikationsmöglichkeit beschränkt sich nicht nur auf die Storageserver, auch die Metadatenserver können abesichert werden. Ceph arbeitet als „Copy-on-Write“-System, das bedeutet, wie auch bei BTRFS, dass zunächst Kopien einer Datei angelegt werden, bevor diese auf dem gesamten System geschrieben und die Metadaten entsprechend angepasst werden.

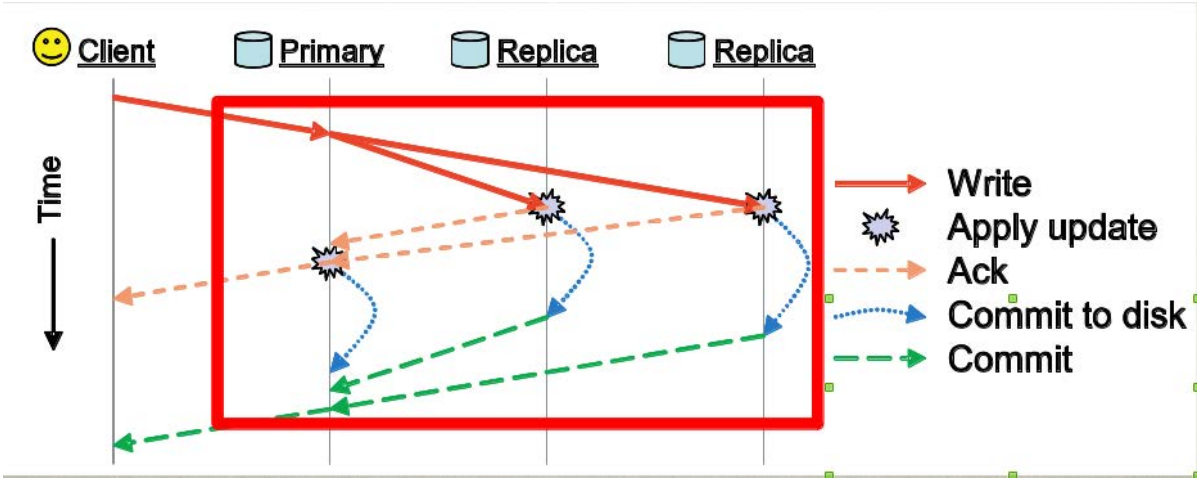

Abbildung 4.12: Replikationsaufbau Ceph [36]

Die Daten vom Client werden nur auf einem Server geschrieben, anschließend werden die Daten über die angelegten Replikationsserver verteilt. Dabei werden beim Schreiben der Replikationsdaten **Checksummen** eingeführt, die gewährleisten, dass die Daten korrekt übermittelt wurden.

Auch bei einem Ausfall des Hauptstorageservers übernimmt der nächste Replikationsserver die Arbeit, ohne dass ein Eingreifen des Administrators nötig ist oder der Benutzer etwas davon spürt. Die Metadaten werden entsprechend angepasst, sodass auch der Metadatenserver weiß, welchen Server er aktuell ansprechen muss. Dieser überprüft auch ständig die Verfügbarkeit der einzelnen Knoten und kann selbstständig die Daten von Replikationsservern abrufen. Ein solcher Server ist nicht dafür vorgesehen, die Performance zu verbessern, es dient der Erhöhung der Hochverfügbarkeit.

Als darunter liegendes Dateisystem empfiehlt Ceph die Nutzung von BTRFS, welches die Probleme von ext4 und XFS ausmerzen soll. Während XFS hauptsächlich die die Nutzung für große Dateien sehr gut funktioniert, möchte Ceph mit allen Arten von Dateigrößen performant arbeiten können. Da sich aber genau dieses System selbst noch in der Entwicklung befindet, ist die Sicherheit der Daten nicht gewährleistet.

4.5.2 Administration von CephFS

Die Administration von Ceph bietet zum aktuellen Zeitpunkt keine grafische Oberfläche an, die einen Überblick über die vorhandene Infrastruktur geben könnte. Durch Kommandozeilenbefehle ist es möglich, die Erreichbarkeit einzelner Server und Dateien zu ermitteln, sowie deren Replikationsstatus. Das erfordert Wissen über die einzelnen Befehle, zudem muss der generelle Aufbau des Systems bekannt sein.

Durch das Installieren des „ceph-deploy“-Paketes ist es möglich, automatisierte Installationen durchzuführen. Dies setzt eine aktive Internetverbindung, sowie den passwortlosen ssh-root-Zugang zu den einzelnen Knoten voraus.

Dieses Paket steht aktuell nur für RedHat-Linux-Distributionen zur Verfügung, eine Installation auf einem Gentoo-System war nicht möglich, da bestimmte Abhängigkeiten nicht aufgelöst werden konnten.

Durch diese automatisierte Installation hat der Administrator bei der Einrichtung eines Systems zwar den Vorteil, dass die Einstellungen auf jedem Knoten gleichmäßig gespeichert werden, allerdings müssen spezielle Änderungen immernoch manuell für jeden Knoten einzeln getätigt werden. Dies macht die Installation umständlich und zeigt sich im Vergleich zu anderen verteilten Dateisystemen als schwierig zu konfigurieren.

4.5.3 Vor- und Nachteile CephFS

Der große Vorteil von Ceph ist der modulare Aufbau des Dateisystems. Es ist durch spezielle Kommandozeilenbefehle möglich, die gesamte Infrastruktur einzurichten. Zusätzlich besteht die Möglichkeit, dass jeder Server nur die Pakete erhält, die auch gebraucht werden. Die Replikationsmöglichkeiten der Storage-, sowie der Metadatenserver machen Ceph zu einem sicheren, verteilten Dateisystem. Die Updatemöglichkeiten auf neuere Versionen ist unkompliziert möglich, diese können durch einen Update-Befehl durchgeführt werden, wenn das Dateisystem nicht in Benutzung ist.

Das eigentliche Hauptaugenmerk von CephFS liegt dabei in der optimalen Nutzung der Festplattenkapazitäten, um Performance und Skalierbarkeit zu erreichen. Wenn alle diese Punkte zusammen ein einheitliches Ergebnis zusammentragen, ist Ceph eine sehr gute Alternative zu anderen, auch kommerziellen, Dateisystemen.

Auch CephFS nutzt das Kernelmodul „FUSE“, allerdings in einer speziellen abgewandelten Version, die vom Hersteller zur Verfügung gestellt wird. Daher ist der Umstand, dass Änderungen am eigentlichen Programm erst dann nötig sind, wenn der Hersteller selbst diese auch am eigenen „Ceph-FUSE“-Paket einpflegt.

Es existieren einige gravierende Nachteile. Die fehlende administrative Übersicht über alle vorhandenen Knoten machen die Einrichtung und Kontrolle kompliziert und zum Teil sehr umständlich. Diese Art der Einrichtung jedes einzelnen Servers stellt dabei eine Herausforderung dar, da keine gute und verständliche Dokumentation für die Installation des Systems vorhanden ist. Der größte Nachteil ist allerdings, dass sich das System aktuell noch in der Entwicklung befindet und der Hersteller explizit darauf verweist, dass es zu Datenverlusten kommen könnte und dass ein produktiver Einsatz des Systems nicht empfohlen wird. Das ist ein ausschlaggebendes Kriterium für die Betrachtung, welches Dateisystem produktiv zum Einsatz auf den neuen Compute-Cluster des Universitätsrechenzentrums installiert werden soll.

4.6 GlusterFS

GlusterFS ist ein Open-Source paralleles, verteiltes Dateisystem, welches von der Firma Red Hat entwickelt und weitergeführt wird. Aktuell in der Version 3.4.0 bietet das System einen modularen Aufbau an, indem die einzelnen Komponenten flexibel eingesetzt werden können.

GlusterFS soll ein extrem performantes Dateisystem sein, welches perfekte Skalierbarkeit bietet, sowie mit Storagegrößen umgehen kann, die über die Petabytegrenze hinaus gehen. Durch die Einrichtung von Replikationsservern können Daten automatisiert gesichert werden, die Hochverfügbarkeit kann somit gewährleistet werden. Zudem hält sich GlusterFS an generelle Namenskonventionen, sodass dieses in verschiedenen Umgebungen eingesetzt werden kann.

GlusterFS kann durch seinen modularen Aufbau auf unterschiedliche Arten verwendet werden. Entweder stellt dieses ein einzelnes Netzlaufwerk über das Netzwerk zur Verfügung, indem Benutzer lesen und schreiben können oder, wenn mehrere Server zur Verfügung stehen, können diese verteilt („distributed“) abgelegt werden. Durch die Möglichkeit, Replikationen einzurichten, kann GlusterFS auch als "Replicated Storageängesehen werden. Außerdem sind verschiedene Kombinationen einzelner Komponenten möglich.

Dieses System ist bereits sehr weit verbreitet und wird von vielen Firmen produktiv verwendet. Die stetige Entwicklung von RedHat und der gesamten Gluster-Community zeigt sich in den Updates, sowie der Möglichkeit, wenn Fehler aufgetreten sind, dass diese direkt über die Internetadresse gemeldet werden können.

4.6.1 Interner Aufbau von GlusterFS

Die Komponenten von GlusterFS bestehen aus **Datenservern** und **Replikationsservern**. Der große Unterschied zwischen den anderen bereits beschriebenen Dateisystemen ist die Behandlung der einzelnen Dateien. Diese werden nicht in einzelne Objekte zerlegt, sondern die gesamte Datei wird auf einem Server abgelegt.
Dabei sind diese standardmäßig nicht verschlüsselt und für den Administrator frei zugänglich. Eine eigene Verschlüsselungsmethode kann bei der Installation zusätzlich mit eingerichtet werden. Die Auswahl, welche Datei auf welchem Server abgelegt wird, ist nicht möglich.

Genau wie CephFS und MooseFS benutzt GlusterFS die Kommunikationsschnittstelle „FUSE“, welche die Dateisystemoperationen aus dem Kernel in den Userspace verlagert, sodass Applikationen darauf zugreifen können. Das darunter liegende Dateisystem kann dabei sowohl in ext3/4 partitioniert sein, auch XFS wird direkt angeboten. Durch die Nutzung von „FUSE“ werden nahezu alle Linux-Distributionen unterstützt, sowie FreeBSD, OpenSolaris und Mac OS X.

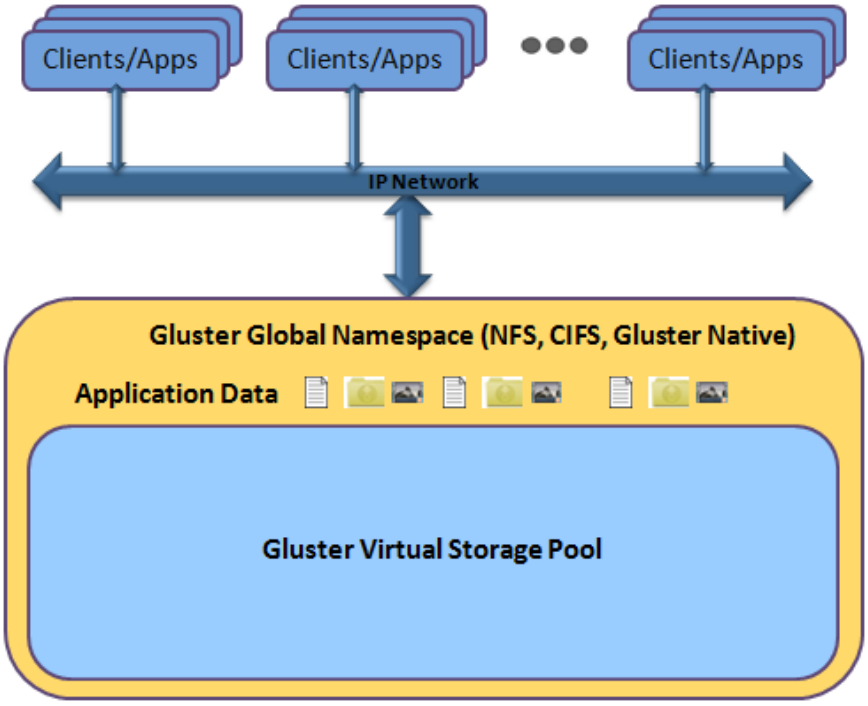

Abbildung 4.13: Übersicht über GlusterFS [37]

Das virtuelle Dateisystem wird über ein vorhandenes gesetzt und die Daten werden in zuvor festgelegte Ordner abgelegt. Damit ist es für einen Administrator möglich, diesen zu durchsuchen, weil die gesamte Datei gespeichert wird. Die Nutzung von RAID-Konfigurationen ist direkt möglich, da dieses unabhängig vom virtuellen vorhandenen System ist.

Zusätzlich bietet GlusterFS die Nutzung von Infiniband an, dafür wird ein eigens entwickeltes RDMA-Tool zur Verfügung gestellt. Dieses kann in den Kommunikationsdateien festgelegt werden, auch eine Redundanz über zusätzliche Netzwerkschnittstellen ist möglich. Es wird standardmäßig die schnellste Verbindung gewählt. Wenn diese nicht zur Verfügung steht, dann wird automatisch auf die nächste Verfügbare umgeschaltet, sodass ein Fail-Over gewährleistet werden kann.

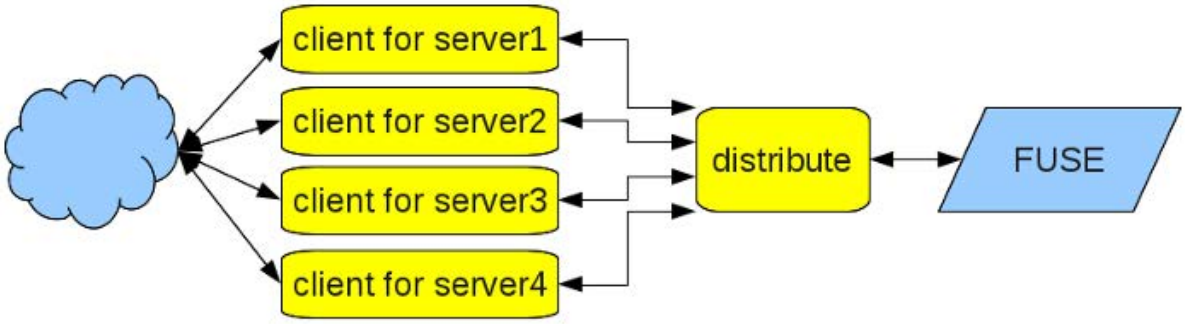

Abbildung 4.14: Datenverteilung über GlusterFS [38]

Während andere verteilte Dateisysteme eine Trennung zwischen Metadaten und eigentlichen Daten vornimmt, unterscheidet GlusterFS sich in diesem Fall. Es wird generell die gesamte Datei auf einem Server gespeichert. Wenn ein Programm mehrere Dateien benötigt, können diese von mehreren Servern gelesen werden, ansonsten wird jede Datei nur von einem zur Verfügung gestellt. Dies hat auch den Nachteil, dass die Dateien nicht zur Verfügung stehen, wenn ein Server nicht erreichbar ist und kein Replikationsserver vorhanden ist.

Dies kann zu Problemen führen, insbesondere bei sensiblen Daten. Diese Art der Verteilung von Daten hat insbesondere bei Zugriffen auf viele verschiedene Dateien große Vorteile, da die Anfragen stets verteilt werden können. Hinzu kommt, dass jeder Client auch als Server dienen und seinen verfügbaren Speicher zur Verfügung stellen kann. Es muss aber darauf geachtet werden, dass die Datenübertragung von sehr großen Dateien nicht die Größe einzelner Festplatten- und Festplattenkonfigurationen

überschreitet, zum Beispiel beim Schreiben einer 2TByte großen Datei, wenn nur 1TByte große Festplatten zur Verfügung stehen. Diese Dateiübertragung schlägt fehl. GlusterFS bietet zwar praktisch unendlich Speicherplatz an, die einzelne Dateigröße ist allerding limitiert auf die Festplattenkapazitäten einzelner Knoten im Cluster.

Bei einem Hinzufügen von neuen Knoten in ein bereits existierendes System, gibt es eine **„Rebalance“**-Funktion, welche automatisch Daten auf diesen Server mit auslegt und dementsprechend die Geschwindigkeit erhöht werden kann.

Die Möglichkeit des **Zwischenspeicherns** von Daten auf den lokalen Festplatten der Clients ist mit GlusterFS möglich. Die Caching-Konfigurationen sind dabei sehr weitläufig. Zum einen kann die generelle Größe angegeben werden, es ist aber auch möglich, eine Mindest- und Höchstgrenze einzustellen. Das ist insbesondere bei Daten von Vorteil, die häufig gelesen werden müssen.
Da dieser Punkt für das Universitätsrechenzentrum allerdings nicht von Vorteil ist, da die Berechnungen auf dem Compute-Cluster erfolgen und die Ergebnisse immer in einer Datei aktualisiert werden, wird das Caching für die folgenden Testdurchläufe deaktiviert. Dies führt auch dazu, dass es zu einem besseren Vergleich der einzelnen Dateisysteme kommen kann.

Generell bietet GlusterFS die Möglichkeit der **Sicherungen** an, die auf IP-Basis ausgelegt ist. Die Zugriffskontrolle kann sowohl beim Client, als auch beim Server eingestellt werden. Zusätzlich ist die Möglichkeit gegeben, dass bestimmte Clients auf bestimmte Verzeichnisse innerhalb des Dateisystems zugreifen dürfen.

Bei der Installation des Clients müssen alle vorhandenen Storageserver bekannt gemacht werden, sodass dieser die Aufgabe der Verteilung der Daten automatisch übernimmt. Dadurch, dass keine Trennung zwischen Metawerten und den einzelnen Daten durchgeführt wird, ist es nicht nötig, einen eigenen Server anzusprechen. Jedoch wird immer ein Hauptserver ausgewählt, welcher immer zuerst angesprochen wird. Um einen Nutzer entsprechende Kapazitäten zur Verfügung zu stellen, bietet GlusterFS sogenannte **„Subvolumes“** an, welche mit einer gewünschten Dateigröße angelegt werden kann.

4.6.2 Administration

Die Administration von GlusterFS ist nicht mit anderen Systemen zu vergleichen. Es fehlt generell eine grafische Oberfläche, allerdings werden bei der Installation Pakete installiert, die spzezielle Gluster-Kommandozeilenbefehle einrichten, mit dessen Hilfe die Infrastruktur aufgebaut, verwaltet und wieder aufgelöst werden kann.
Die Möglichkeit des Anlegens von Replikationen wird bereits beim Erstellen des „subvolumes“ eingerichtet. GlusterFS verarbeitet automatisch die Daten und führt die Replikationen aus, ohne dass der Benutzer davon etwas bemerkt. Diese doch recht komplizierte Möglichkeit, das Dateisystem zu administrieren, ist verbesserungswürdig, insbesondere weil Beispiele der Konkurrenz aufzeigen, dass eine einfachere und übersichtliche Art der Verwaltung möglich ist und auch Sinn macht.

4.6.3 Vor- und Nachteile GlusterFS

Durch den grundlegenden Aufbau und der Verteilung der Dateien auf verschiedene Server, ohne dass diese selbst verändert werden, kann das System sehr flexibel und performant machen. Auch durch den Verzicht auf die Nutzung eines speziellen Metadatenservers wird Speicherplatz und Performance eingespart. Bei anderen Dateisystemen wird mindestens ein gesamter Knoten nur für die Verwaltung der Metadaten verwendet, dieser kann bei GlusterFS als zusätzlicher Storageserver verwendet werden. Allerdings kann, insbesondere bei sehr großen Dateien, der Geschwindigkeitsvorteil nicht mehr vorhanden sein, weil beim Zugriff immer ein einzelner Server angesprochen wird und damit eine direkte Datenübertragung zwischen einem Server und dem Client stattfindet.

Positiv anzumerken ist, dass GlusterFS mit jeder Distribution genutzt werden kann, die „FUSE“ unterstützt und somit sehr weitreichend verwendet werden kann. Die Performance sollte sich insbesondere beim Lesen oder Schreiben vieler Dateien beim Hinzufügen mehrerer Server verbessern, dies ist vom Grundaufbau des Systems zu erwarten.
GlusterFS bietet zudem eine eigene Variante von „FUSE“ an, welche besser auf das Dateisystem abgestimmt ist.

GlusterFS bietet keine grafische Oberfläche, die den Aufbau und die Verteilung der Server aufzeigen könnte. Die Installation wird mit Kommandozeilenbefehlen durchgeführt. Durch die Einrichtung des **„glusterfs-utils“**-Pakets ist es möglich, die Erreichbarkeit der anderen Server zu überprüfen.

Für jeden Server muss eine eigene Konfigurationsdatei geschrieben werden, wo exakt aufgeführt wird, welcher Client darauf zugreifen kann und welche weiteren Storageserver sich im Netzwerk befinden. Dieser Umstand ist in der ersten Einrichtung relativ kompliziert, allerdings können verschiedene Ketten aufgebaut werden. Ketten bedeutet, dass wenn „Server1“ „Server2“ kennt und dieser „Server3“, dann kennen sich auch „Server1“ und „Server3“. Diese Möglichkeit vereinfacht die Einrichtung des Dateisystems.

Zudem bietet GlusterFS an, lediglich eine bestimmte Größe des vorhandenen Dateisystems zur Verfügung zu stellen, indem eine virtuelle Partition eingerichtet wird. GlusterFS arbeitet dabei mit allen gängigen regulären Dateisystemen zusammen, empfohlen wird die Nutzung von ext4 bei besonders kleinen und XFS bei großen Dateien. Die Nutzung von BTRFS wird zwar unterstützt, allerdings durch den frühen Entwicklungsstand noch nicht empfohlen.

4.7 Entscheidung

Die vorliegende Arbeit erläutert den Aufbau der Systeme FhGFS, MooseFS, Ceph und GlusterFS genauer. Dabei existieren weitere, auch kommerzielle, Systeme, die einen ähnlichen internen Aufbau vorweisen können und eine ähnliche Struktur der Daten beinhaltet.

Diese Dateisysteme wurden mit Bedacht ausgewählt, weil diese jeweils einen unterschiedlichen Grundgedanken haben und auf unterschiedliche Kerneigenschaften eines Dateisystems spezialisiert sind. Andere Dateisysteme, wie zum Beispiel „Lustre“, welches offiziell im Linux-Kernel aufgenommen wurde, zählen zu großen Vertretern dieser Arten. „Lustre“ ist vom Grundaufbau vergleichbar mit „MooseFS“, allerdings bietet dieses noch keine direkten Replikationsmöglichkeiten an. Andere Systeme bieten spezielle Eigenschaften an, die für das Universitätsrechenzentrum nicht von Bedeutung sind und somit nicht getestet werden.

Für die Entscheidung, welches Dateisystem für den neuen Compute-Cluster verwendet werden sollte, werden im Folgenden die Systeme FhGFS, MooseFS und GlusterFS auf einem zur Verfügung gestellten Test-Cluster installiert und auf unterschiedliche Eigenschaften hin überprüft. Es wird auf die Installation von CephFS verzichtet, da sich das System, ähnlich wie BTRFS als reguläres Dateisystem, noch in einer relativ frühen Entwicklungsphase befindet und daher für einen produktiven Einsatz noch ungeeignet ist. Allerdings sind erste Testberichte sehr vielversprechend, sodass CephFS bei der Betrachtung für die Zukunft nicht vergessen werden darf.

	FhGFS	MooseFS	GlusterFS
max. Größe	unbegrenzt	16EiB	unbegrenzt
max. Dateigröße	keine Angabe	2TB	Größe der kleinsten Festplatte
Verbreitung	hoch	gering	sehr hoch
RAID-Konfigurationen	alle	alle	alle
Kommunikationsmöglichkeiten	Ethernet, Infiniband	Ethernet	Ethernet, Infiniband
Trennung Metadaten - Storage	ja	ja	nein
Performance	++	o	+
Skalierbarkeit	++	o	++
Sicherheit	o	++	+
Stabilität	++	+	++
Heterogenität	+	++	++
Replikation	--	++	++
Fehlerverhalten	-	++	o
Administration	++	+	-

Abbildung 4.15: Zusammenfassung Eigenschaften verteilter Dateisysteme

Wie aus der Abbildung 4.15 zu entnehmen, besitzen alle vorgestellten Dateisysteme die Möglichkeit, sehr große Datenmengen zusammenzufassen. FhGFS und GlusterFS bieten sogar theroetisch unbegrenzten Speicherplatz an, die verteilt auf einer sehr großen Anzahl von Servern liegen können. Dabei werden von allen Systemen alle unterliegenden regulären Dateisysteme und die dazugehörigen RAID-Konfigurationen unterstützt.

Die Tabelle zeigt dabei die unterschiedlichen Gewichtungen der grundlegenden Eigenschaften, die an ein verteiltes Dateisystem gelegt werden. Dabei bedeutet „++“, dass das System auf diesen Punkt spezialisiert ist, **„o“**, dass das System dies zwar anbietet, aber nicht als Hauptargument anzusehen ist und „–“, dass dieser Eigenschaft vom Dateisystem nicht erfüllt wird. Dabei gibt es noch Abstufungen **(„+“, „-“)**, die aufzeigen, dass diese Eigenschaft unterstützt wird, allerdings die Umsetzung im Vergleich zu den anderen Systemen komplizierter und unkomfortabler gelöst wurde.

Die **Verbreitung** ist der einzelnen Dateisysteme ist unterschiedlich. Während GlusterFS und FhGFS auf zahlreichen großen Clusterverbunden zum Einsatz kommen, ist MooseFS ein relativ unbekanntes Dateisystem, welches bis zur heutigen Zeit nur vereinzelt eingesetzt wird. FhGFS und GlusterFS sind Dateisysteme, die auch auf Supercomputern zu finden sind, die in der Top500-Liste gelistet werden. Dieser Umstand zeigt, dass diese beiden Dateisysteme sehr ausgereift sein müssen und eine hohe Performance und Skalierbarkeit aufweisen.

Unterschiede gibt es in der Auswahl der **Netzwerktopologien**. MooseFS bietet lediglich den Umstand, über TCP-Verbindungen mit den einzelnen Servern zu kommunizieren. Dabei ist der Umweg über Infiniband-over-IP möglich. Die direkte Implementierung von Infiniband bieten FhGFS und GlusterFS an, dazu können spezielle Pakete installiert werden, die Erkennung erfolgt automatisch.

In den zuvor festgelegten **Kriterien** zur Auswahl der Dateisysteme gibt es große Gegensätzlichkeiten, da sich bereits die grundlegenden Eigenschaften völlig unterscheiden. Während das Fraunhofer Dateisystem auf absolute Performance, Skalierbarkeit und Stabilität ausgelegt ist, dazu eine sehr gute administrative Oberfläche anbietet, liegt das Hauptaugenmerk bei MooseFS auf die Sicherheit und Replikationsmöglichkeit. Diese Punkte werden bei FhGFS nur sekundär behandelt und sollen in folgenden Versionen mit eingeführt werden. MooseFS steht nicht für hohe Performance, allerdings für Stabilität und Zugriffskontrolle. GlusterFS ist genau zwischen diesen beiden Konkurrenten einzuordnen. Es wirbt für eine gute Performance und ausgezeichnete Stabilität, bietet zudem noch die Sicherheit und Replikationsmöglichkeiten an, die auch bei MooseFS eingearbeitet wurden und bei FhGFS fehlen. Dafür existiert keine administrative Oberfläche für dieses System, die eine einfache und unkomplizierte Installation und Wartung ermöglichen könnte.

Alle vorgestellten Dateisysteme haben die Eigenschaft, in **heterogenen Netzwerken** zu funktionieren. Während MooseFS und GlusterFS auf allen Linux-Distributionen, sowie OpenSolaris, FreeBSD und Mac OS X genutzt werden kann, stehen für das FhGFS Installationen für RedHat-Enterprise, Debian- und SuSe-Distributionen zur Verfügung. Die Verbindung unterschiedlicher Hardware und Betriebssystemen ist in allen vorgestellten Beispielen gegeben.

Die Eigenschaft **„Fehlerverhalten“** zeigt allerdings große Unterschiede zwischen den Dateisystemen. Das Fraunhofer Dateisystem kann im Falle eines Ausfalles einzelner Knoten keine automatischen Replikationsmöglichkeiten anbieten. Daher ist damit zu rechnen, dass das gesamte Dateisystem kurzzeitig nicht zur Verfügung steht. Dieses Feature soll in späteren Versionen hinzugefügt werden, um Hochverfügbarkeit erreichen zu können. MooseFS und GlusterFS bieten sowohl Replikationsmöglichkeiten an, als auch die Eigenschaft, sich automatisch zu rekonfigurieren, sobald ein Server nicht zur Verfügung steht. Diese können bei entsprechender Anzahl von Replikationen automatisch auf den nächsten Server umschalten, um die Daten weiter verfügbar zu halten. Allerdings ist es auch hier nicht möglich die Daten zu retten und

wiederherstellen zu können, wenn keine Sicherungen angelegt sind. MooseFS bietet zudem einen Dateisystempapierkorb an, welcher für einen gewissen Zeitraum gelöschte Daten vorbehält, um eine Rekonstruktion durchführen zu können. Dieser Zeitraum ist flexibel einstellbar.

Zusammenfassend ist zu sagen, dass die Eigenschaften aller vorgestellten Dateisysteme aufzeigen, dass es bei verteilten Dateisystemen nicht dazu kommen kann, dass ein Vertreter alle nötigen Eigenschaften erfüllt. CephFS ist hierbei ausgeklammert worden, weil dieses noch nicht als System anzusehen ist, welches in einem produktiven Einsatz installiert werden sollte. Die folgenden Testdurchläufe werden mit FhGFS, MooseFS und GlusterFS durchgeführt, um einen konkreten Überblick über die Performance, Skalierbarkeit, Sicherheit, Stabilität und Administration zu erhalten. Dies geschieht, damit die Entscheidung für ein bestimmtes Dateisystem für den neuen Compute-Cluster des Universitätsrechenzentrums getroffen werden kann.

Kapitel 5

Tests

Um ein Dateisystem auf Performance und Stabilität testen zu können, exisiteren mehrere Möglichkeiten und Programme, die automatisiert verschiedene Verfahren ablaufen lassen. Bei vielen Programmen werden Möglichkeiten angeboten, diese nach eigenen Bedürfnissen zu konfigurieren. Insbesondere der Datendurchsatz, der sich durch das Zusammenschalten von mehreren Servern und dem verteilten Abspeichern der Daten erhoffen lässt, soll dabei untersucht werden. Außerdem werden Langzeittests durchgeführt. Diese bestehen aus dem dauerhaften Schreiben und Lesen vom Daten, die in dem Dateisystem hinterlegt sind bzw. werden. Die Stabilität ist insbesondere im dauerhaften Einsatz unablässig. Ein möglicher Ausfall zieht unweigerlich das Stilllegen des gesamten Clusters nach sich, da keine Rechenoperationen mehr gespeichert, ausgeführt und verarbeitet werden können. Selbst wenn diese noch ablaufen würden, wäre die Gefahr, ein falsches Ergebnis zu bekommen, zu hoch, als das die Geschwindigkeit von Vorteil wäre. Insbesondere bei gleichzeitigen Zugriffen ist auch die Auslastung des Servers wichtig. Wenn dieser bereits mit einer Schreiboperation ausgelastet ist, ist ein Zusammenbruch bei vielen Zugriffen bereits vorprogrammiert. Daher ist die Speicher- und CPU-Auslastung während der Testphase immer im Auge zu behalten.

5.1 Testsystem

Das vorliegende Testsystem besteht aus drei Servern, die mit Infiniband, sowie Gigabit-Ethernet miteinander verbunden sind. Jeder Server besteht aus zwei Intel Xeon X5500 CPU's mit 4x2.6GHz und 48GB Arbeitsspeicher. Als Datenträger kommen pro Server 1TB Festplatten der Marke Samsung zum Einsatz. Diese Festplatten verfügen über einen SATA2-Anschluss, drehen mit 5400 Umdrehungen pro Minute und verfügen über 8MB Cache. Laut Testberichten [39] erreicht die Festplatte im Lesemodus eine maximale Geschwindigkeit von 110MByte/s, im Durchschnitt liegt die Leseperformance bei 85MByte/s. Die maximale Schreibrate war geringfügig langsamer, im Durchschnitt liegt diese bei 83MByte/s.

Für den Metadatenserver wurden zwei solcher Festplatten zu einem RAID-0-Verbund zusammengefasst. Das bietet einen zusätzlichen Geschwindigkeitsvorteil, allerdings sind bei einem möglichen Ausfall einer Festplatte die gesamten Metadaten unbrauchbar. Für einen produktiven Einsatz ist diese Kombination nicht zu empfehlen, da sollte zusätzlich auf eine Sicherung auf dieser Ebene geachtet werden, als Beispiel wäre ein Raid-10-Verbund zu nennen, welches die gleichen Geschwindigkeiten anbietet und zusätzlich eine Sicherung der Daten durchführt. Für die Durchführung der Tests ist nicht von einem Defekt der Hardware auszugehen, daher kommt die Konstellation mit einem RAID-0 Verbund zum Einsatz. Die beiden weiteren Server werden jeweils mit einer 1TB Festplatte installiert.

Nach Rücksprache mit den Nutzern des aktuellen kleineren Clusters werden die Dateigrößen entsprechend angepasst. Die laufenden Programme erzeugen häufig Dateien, die im Größenbereich von 128KByte bis 100MByte liegen. Selten werden Bereiche von über 1GByte überschritten. Die Ergebnisse der Programme werden wiederrum in das Dateisystem geschrieben, diese übersteigen aber zum aktuellen Zeitpunkt nicht die 1GByte-Grenze, sodass eine Performancemessung mit Dateien, die größer als 2GB sind, für das Universitätsrechenzentrum nicht von Bedeutung sind. Dabei ist sowohl die Lese-, als auch Schreibgeschwindigkeit von Interesse. Die komplexen Berechnungen werden auf dem Compute-Cluster ausgeführt, die jeweils lesende und schreibende Zugriffe auf die Festplatten durchführen. Die Ergebnisse werden anschließend in eine bestimmte Datei geschrieben. Nach weiteren Durchläufen können die Programme entweder neue Dateien schreiben oder die Ergebnisse an die vorhandene Datei anfügen.

Die durchgeführten Tests werden jeweils als One-Node- und Two-Node-Installation durchgeführt. Dies geschieht, damit eine mögliche Skalierbarkeit durch das Hinzufügen eines zusätzlichen Storageservers erkennbar gemacht wird.

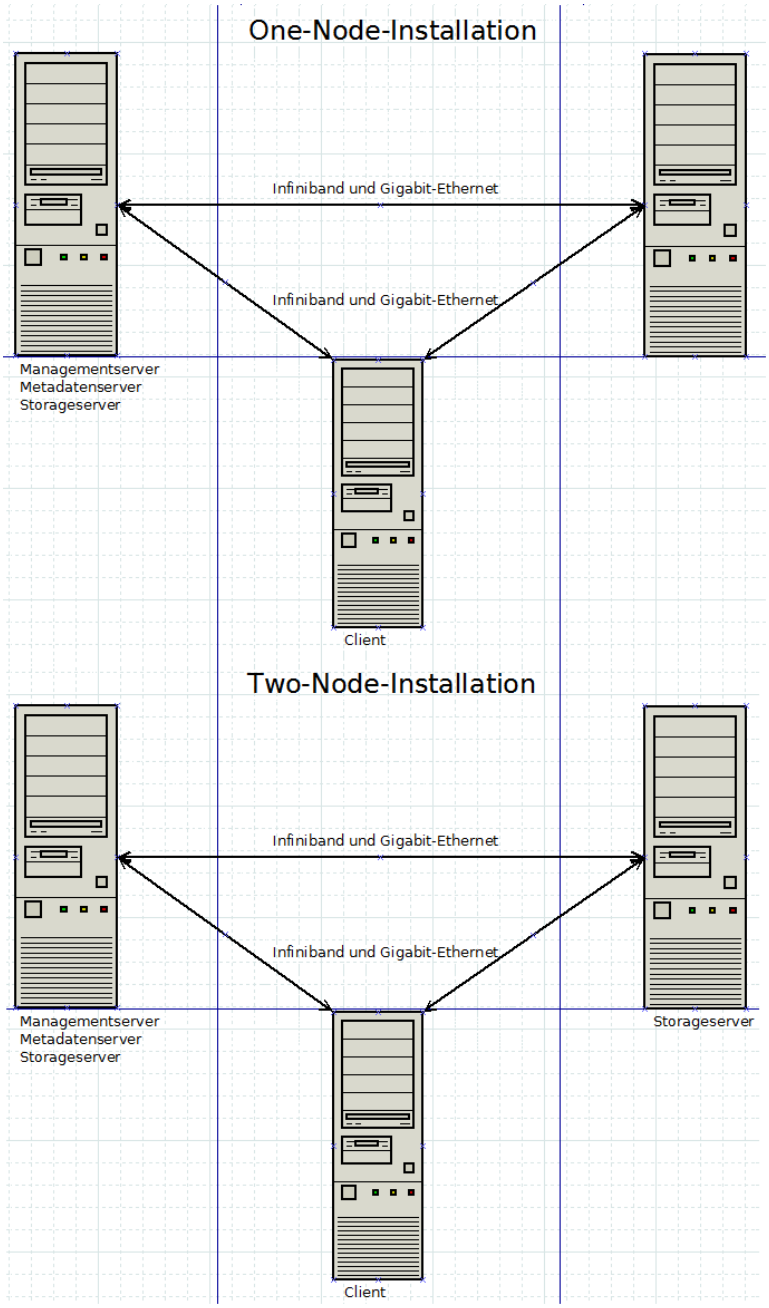

Abbildung 5.1: Aufbau des Testsystems

5.2 Durchgeführte Tests

Auf der vorhandenen Testumgebung werden mit Hilfe des Programmes „Iozone“ verschiedene Testverfahren durchgeführt, die zur Performancemessung, sowie zur Validierung der Systemstabilität genutzt werden. Dieses Benchmarking-Tool wurde in der Programmiersprache C geschrieben, generiert gewünschte Dateigrößen und misst sowohl die Lese- als auch die Schreibrate im Dateisystem.
Im "Parallel File System Survey Reportäus dem Jahr 2010 [30] ist zudem zu entnehmen, dass „Iozone“ am häufigsten verwendet wird, da dieses viele Möglichkeiten bietet, die Performance eines Parallelen Dateisystems messen zu können. Die Messungen und Methoden sind mittlerweile sehr umfangreich geworden, daher sind diese in verschiedene Szenarien unterteilt. Als Ergebnis erhält der Anwender eine Auswertung der einzelnen Testfälle entweder direkt auf der Konsole oder in einem speziellen Excel-Format, um diese grafisch auswerten zu können. Das Tool wurde ausgewählt, weil es international Anerkennung in allen größeren Rechenzentren gefunden hat und die Individualisierung auf die persönlichen Bedürfnisse und die nahezu perfekte Auswertungsmöglichkeit eine sehr gute Übersicht über die Gesamtperformance eines parallel verteilten Dateisystems gibt.

Write-Mode In diesem Modus wird eine Datei erzeugt, die eine gewünschte Größe besitzt und unterschiedlich fragmentiert ist. Diese wird anschließend in das Dateisystem übertragen, dabei wird die Geschwidigkeit der Erstellung der Datei, sowie der Erstellung der Metadaten gemessen. Bei einer großen Datei, die in viele kleine Pakete unterteilt ist, sind die Metadaten entsprechend groß, wenn ein Dateisystem diese einzeln abspeichert. Um das zu verhindern, verwenden einige verteilte Dateisysteme sogenannte Ëxtents", die das Zusammenfassen von bestimmte Bereichen ermöglichen. Dieser Testmodus zeigt auf, ob dies funktioniert oder die Datenverwaltung entsprechend größer wird, da die Schreibgeschwindigkeit abnehmen sollte.

Re-Write-Mode Mit Hilfe dieses Testszenarios wird eine Datei, die bereits existiert, erneut übertragen. Bei intelligenten Systemen sollte dieses schneller verlaufen, da sich die Metadaten nicht ändern und daher dieser Overhead nicht geschrieben werden muss.

Read-Mode Die zuvor geschriebene Datei wird anschließend aus dem Dateisystem heraus gelesen, sodass man die Lesegeschwindigkeit messen kann. Dabei ist enscheidend, wie schnell die Zuordnung von Metadaten und der eigentlichen Datei gefunden werden kann, sodass eine Übertragung durchgeführt wird. Es ist auch zu

erwarten, dass die Geschwidigkeit zunimmt, wenn die Datei verteilt auf mehreren Servern liegt, die parallel arbeiten.

Re-Read-Mode In diesem Modus wird eine Datei erneut gelesen. Ziel ist es, dass die Performance ansteigen sollte, da sich die Daten noch im Cache des Dateisystems befinden sollte und somit nicht erneut gesucht und eingelesen werden müssen.

Random-Read-Mode Eine bereits zuvor geschriebene Datei wird in diesem Modus an unterschiedlichen Stellen gelesen. Insbesondere für verteilte Dateisysteme ist dies ein sehr wichtiger Test für die Performance. In größeren Ausführungen wird eine Datei verteilt auf mehrere Festplatten und Servern abgelegt. Wenn nur ein bestimmter Bereich der Datei angefordert wird, muss dieser entsprechend aus den Metadaten gefunden werden und angefordert werden. Dabei spielt die Anzahl der Server und Festplatten, sowie die Größe des Caches des Systems eine entscheidene Rolle.

Random-Write-Mode Analog zum Random-Read-Mode führt dieser Änderungen an bestimmten Stellen einer Datei aus. Dabei wird diese nicht vollständig neu übertragen, sondern die Änderungen werden verschickt und das Dateisystem muss diese an den entsprechenden Stellen verändern.

Random-Mix-Mode Dieser Modus ist eine Kombination aus Random-Read- und Random-Write-Modus. Durch die parallele Ausführung von Lese- und Schreibanfragen wird die Performance des Systems aufgezeigt, da auch hierbei nur bestimmte Teile einer Datei angefordert und abgespeichert werden.

Backwards-Read-Mode In diesem Modus wird eine Datei, die zuvor im System abgelegt wurde, rückwärts gelesen. Die Möglichkeit, dass so ein Fall effektiv eintreten kann, ist im produktiven Einsatz zwar eher selten, allerdings nicht unmöglich. Insbesondere wenn eine Datei mit Messdaten erstellt wurde und die aktuellsten immer ans Ende der Datei geschrieben wurde und diese zunächst wichtig sind, kann eine Datei rückwärts übertragen werden, da die wichtigen Bereiche als erstes verschickt werden. Dateisysteme sind häufig nicht für einen solchen Lesemodus ausgelegt, wenn eine Datei angefordert wird, wird diese immer vorwärts übertragen, bzw. bei verteilten Dateisystemen werden die einzelnen Objekte vorwärts übertragen. Um dieses zu bewerkstelligen, müssen die Anfragen entsprechend häufig gestellt werden, damit ein Rückwärtslesen möglich gemacht wird. Das Ergebnis der Performancemessung gibt

zusätzlichen Aufschluss darüber, wie das System auch mit vielen Anfragen umgehen kann.

Fwrite/Fread-Mode Ähnlich zum Write- und Read-Mode wird eine Datei mit einer bestimmten Größe und Fragmentierung erstellt. Allerdings unterscheiden sich diese Modi dadurch, dass die Bibliothek fwrite() genutzt wird. Diese führt zwischengespeicherte Schreiboperationen durch. Dieser Zwischenspeicher sollte im Arbeitsspeichers des Client-Computers liegen. Insbesondere bei großen Dateien, die stark fragmentiert sind, kann ein Zwischenspeicher diese zusammenfassen und gebündelt übertragen. Dadurch werden die Systemaufrufe reduziert und die Übertragung kann schneller durchgeführt werden. Durch diesen Modus werden sogenannte "Jumbo-Framesërzeugt, die gebündelt übertragen werden.

Frewrite/Freread-Mode Auch dieser Modus führt eine erneute Lese- bzw. Schreiboperation einer bereits existierenden Datei durch. Diese sollte zu einem Performanceanstieg führen, da die Metadaten zwar übertragen aber nicht mehr ausgewertet werden müssen, wenn sich die Datei nicht verändert hat. Beim Lesevorgang ist die Größe des verfügbaren Caches für die Geschwindigkeit wichtig, da erhofft wird, dass sich die Datei noch im Zwischenspeicher befindet.

Mmap Die Funktion mmap() adressiert eine bestimmte Datei aus einem Dateisystem in den Arbeitsspeicher des Empfängers. Anschließend wird die Datei zunächst in den Arbeitsspeicher übertragen, dieses System wird von vielen Programmen genutzt, um mögliche benötigte Dateien bereits im Hintergrund laden zu können, damit die Zugriffszeit verringert werden kann. Um diese Dateien als solche zu identifizieren, werden diese mit bestimmten Flags versehen, die häufigsten sind „MS_SYNC“ und „MS_ASYNC“.

In der vorliegenden Konfiguration werden die Dateisysteme im Automatik-Modus getestet. Dieser führt insgesamt 13 Tests automatisiert durch. Aufgerufen wird das Programm mit der Angabe des Pfades, wo das Dateisystem eingehangen ist und der Angabe der gewünschten Obergrenze der Datei. Zusätzlich können die Ergebnisse in einer Datei abgelegt werden, die anschließend grafisch ausgewertet werden können. Ein Beispiel für einen solchen Aufruf lautet:
$./iozone - f /mnt/fhgfs/test - Raboutput.wks - g2G$

KB	reclen	write	rewrite	read	reread	random read	random write	bkwd read	record rewrite	stride read	fwrite	frewrite	fread	freread
64	4	2293572	2785914	2463162	3564887	81644	26058	18018	20375	165816	2282308	2571066	181759	203842
64	8	4026451	3562993	4245577	4290473	68674	65702	32669	47022	185034	3789562	2911872	211984	225444
64	16	5773407	4946780	4965770	5328946	190470	138495	46274	143844	223862	3183501	3769321	190519	231006
64	32	3370521	4279118	4953555	5773407	235297	3533973	51203	226915	205741	2910007	4233447	183946	267862
64	64	4585690	7193208	4903977	7069626	209787	3201558	86717	4608138	159585	4596195	5790251	183946	180816
128	4	3468851	3278551	342298	606689	95952	29391	24945	33534	163269	2847287	2607028	247620	253000
128	8	5104248	4564427	412750	549169	93987	47673	38095	74552	401268	3868454	4398590	250995	214765
128	16	4142908	4404511	410278	436884	102726	122361	64130	127487	432358	3773513	3646840	280679	337826
128	32	5831744	5333187	472203	490251	165590	270556	75340	237929	498035	4114075	3364714	257581	254990
128	64	4431690	7569262	6389423	6725456	441327	6118218	150399	429611	438212	2848519	4406046	281361	465549
128	128	5338081	6070053	7106411	7089183	446162	5790070	154583	4400544	452132	3124917	7516431	484894	544522
256	4	3277134	4913732	684653	587208	111492	46034	45237	49144	279767	2941341	2325150	361109	503907
256	8	4407486	6732437	600961	729437	143737	55662	54911	71828	295271	5014944	5131298	442097	453847
256	16	3821204	2533290	785273	727166	186450	130009	81974	136540	988450	4127860	3876802	403805	472408
256	32	3085460	7779222	648097	785312	207794	265540	124207	236180	757145	4662364	4130613	406966	416315
256	64	5320375	5030224	6238203	7112159	550515	430242	149446	386726	588407	3936569	4416746	509017	602206
256	128	5446604	8520536	6935470	7319925	513057	5230028	192637	723130	611126	3121900	5577892	531110	701360
256	256	5338976	6241142	6751122	6919191	625754	5123145	280978	5458352	520243	5434986	6757387	635094	577930
512	4	329057	569582	728388	827267	136605	47437	57143	52218	454288	2976598	2858844	509927	580487
512	8	369128	607282	885826	896590	142655	73699	82780	101669	510010	3304816	5227268	606565	622781
512	16	379249	622096	738704	750748	184572	139431	95060	146961	339724	3218701	3761946	528956	589141
512	32	393517	673600	689052	867645	213331	169759	136859	208630	1433784	3122020	3368627	574020	651441
512	64	356076	656483	5959500	7424441	249636	304400	182140	300461	632101	5627957	6087084	707273	750797
512	128	370765	700483	6480695	7317941	665746	600993	217877	488981	745263	5947324	4971290	638488	759555
512	256	332488	591241	6919649	7023231	649841	639915	289747	1106043	703241	3301638	7750325	615303	739868
512	512	12831	692800	739892	814010	834006	534510	358812	727248	774642	824463	713059	664875	710073
1024	4	377432	566969	711643	737812	35640	34920	42242	37371	56051	971610	967044	553517	585775
1024	8	400946	587473	721154	849079	85920	89205	89973	88290	60916	1113102	1036421	635212	661916
1024	16	373997	552939	745856	751825	118599	121787	118435	187371	83238	1343628	959775	608831	664113
1024	32	359309	638004	787750	741005	185576	257937	213248	268633	210523	1039515	1015835	590218	626642
1024	64	387434	599185	1404721	1380159	231258	365971	263172	334098	1216156	948959	1099864	700899	633283
1024	128	410757	621008	1261261	1607578	333345	509177	361585	428283	1414307	1209047	1155686	778652	798749
1024	256	362621	679919	1352705	1526098	487853	748013	416107	682195	1242861	1104774	941959	674559	705771
1024	512	442901	622892	790794	875892	870757	781134	481859	878191	879734	876736	1134048	810735	821902
1024	1024	434264	694723	844864	837228	894308	718552	498556	807640	806996	718113	765330	754585	885009
2048	4	4089	557268	699679	699675	45324	46932	46927	44293	57257	798718	749613	599335	705008
2048	8	449232	616305	693762	701362	77394	79753	84376	75261	62040	773678	759627	653461	674323

Abbildung 5.2: Iozone Aufzeichnung eines Testdurchlaufes

Dieser Aufruf führt den automatischen Test durch („-a“), im Verzeichnis */mnt/fhgfs/test* („-f“) und gibt die Ergebnisse in der Datei *output.wks* („-Rb“) aus. Das Verzeichnis ist optional anzugeben, es ist auch möglich, iozone direkt in das Verzeichnis zu kopieren und von dort zu starten. Im vorliegenden Beispiel wurde auf dem Client das Fraunhofer Filesystem in diesem Verzeichnis gemounted.

Durch die Einstellung $-g$ kann man angeben, welche Größe die maximal getesteten Dateien haben sollen. Wenn diese nicht angegeben wurde, liegt die maximale Größe bei 512MB, welche in der heutigen Zeit nicht mehr zeitgemäß sind, da der Arbeitsspeicher heutzutage wesentlich größer ist und somit eine sehr genaue Betrachtung der einzelnen Testergebnisse nicht transparent ist.

Das Tool bietet noch zusätzliche Einstellungsmöglichkeiten und Parameter an, die angepasst werden können. Durch die Auswahl der unterschiedlichen Dateigrößen ist dieses für den vorliegenden Testfall allerdings nicht weiter nötig, da sowohl die I/O-Performance bei sehr kleinen Dateien, als auch bei sehr großen Dateien mit entsprechender Fragmentierung überprüft wird. Somit wird ein sehr guter Überblick über die möglichen Übertragungsraten gegeben.

Kapitel 6

Testauswertung

Nach der Ausführung der unterschiedlichen Testszenarien folgt eine entsprechende Auswertung der Ergebnisse. Diese sind zunächst unterteilt in die verschiedenen vorausgesetzten Anforderungen des Universitätsrechenzentrums. Insbesondere die Punkte Skalierbarkeit und Performance können mit Messergebnissen verdeutlicht werden. Stabilität ist ein sehr wichtiger Punkt, welcher in der heutigen Zeit elementar ist. Allerdings gibt es für diesen Testpunkt keine explizite Möglichkeit, diese zu überprüfen. Die Verwaltbarkeit ist ein subjektiver Aspekt, welcher bei der Installation der unterschiedlichen Dateisysteme aufgefallen ist und in diesem Punkt betrachtet wird.

6.1 Performance

Performance ist in Clusterverbunden das wichtigste Kriterium, worin sich die Dateisysteme unterscheiden könnten, sodass die Entscheidungsfindung maßgeblich beeinflusst wird. Versprochen wird von jedem Hersteller die höchstmögliche Performance, in der Realität unterscheiden sich diese aber oftmals sehr stark.

Insbesondere der Unterschied zwischen Lese- und Schreibgeschwindigkeiten, sowie die Zugriffszeiten sind dabei teilweise eklatant groß. Dabei existieren große Unterschiede bei unterschiedlichen Dateigrößen und deren unterschiedliche Fragmentierung. Eine 1GB große Datei, die in 4KByte große Blöcke unterteilt wird, benötigt normalerweise 65536 Einträge im Metadatenserver. Diese können zunächst schnell abgelegt werden, allerdings bei Dateisystemen, die mit sehr vielen besonders kleinen Daten umgehen müssen, ist das Auffinden der Metadaten häufig der Flaschenhals.

Jeder Hersteller von parallelen Dateisystemen arbeitet dabei mit unterschiedlichen Ansätzen, diese zu vereinheitlichen und so die Geschwindigkeit zu erhöhen. Ziel ist es, dass die Fragmentierung keine Unterschiede darstellt, ob das Dateisystem mit vielen oder wenigen Daten umgehen muss.

Lese- und Schreibgeschwindigkeiten mit iozone „Iozone“ ist ein Tool, mit dessen Hilfe die Lese- und Schreibgeschwindigkeiten eines Dateisystems untersucht werden können. Dabei werden unterschiedliche Dateigrößen mit unterschiedlichen Chunkgrößen betrachtet. Das Ergebnis sollte möglichst eine einheitliche Geschwindigkeitsdarstellung aufzeigen, ohne das die Dateigröße und Fragmentierung dabei eine Rolle spielt.

Insbesondere bei verteilten Dateisystemen, wo Metadatenserver und Storageserver von einander getrennt sind, ist dies nur mit sehr viel Aufwand zu bewerkstelligen. Eine Datei von 128KByte Größe, welche in einem Chunk gespeichert werden kann, ist theoretisch schneller abzulegen, als eine 128GByte große Datei, die in 128Kbyte Chunks aufgeteilt wird. In diesem Test wird die reine Lese- und Schreibgeschwindigkeit gemessen und ausgewertet.

6.1.1 FhGFS

Das Fraunhofer Institut wirbt mit der hohen Performance und der sehr guten Skalierbarkeit ihres Dateisystems. Im vorliegenden Testfall wird zunächst die Installation des Management-, Metadaten- und Storageservers auf einem einzigen Clusterknoten ausgeführt. Der Client ist ein weiterer Knoten, welcher über ein Gigabit-Netzwerkswitch und Infiniband mit dem Hauptknoten verbunden ist. Außerdem steht ein weiterer Knoten zur Verfügung, der einen weiteren Datenserver zur Verfügung stellt und im weiteren Testfall Metadaten- und Storageserver darstellt. Somit werden die alle möglichen verfügbaren Konstellationen im Testfall behandelt.

One-Node-Installation mit Ethernetinterface Bei der Installation aller Server auf einem Cluster-Knoten wird erwartet, dass die Geschwindigkeit sich kaum von einem regulären Dateisystem unterscheidet, da eine direkte Verteilung der Daten nicht möglich ist, lediglich die Unterscheidung zwischen Meta- und Storagedaten stellt einen Unterschied dar. Auf einem weiteren Knoten wird mit Hilfe des zur Verfügung stehendem Clients das Netzlaufwerk gemounted.

Der Test mit „iozone“ zeigt die Lese- und Schreibgeschwindigkeiten von Dateigrößen zwischen 512Kbyte und 4GByte an, die auf dem Netzlaufwerk gespeichert werden. Diese werden in unterschiedlichen Chunkgrößen gemessen, zwischen 4KByte und 16MByte. Ab einer Dateigröße von 32MByte wird ab einer Chunkgröße von 64KByte gemessen. Dabei werden alle verfügbaren Tests des Programmes angeschaltet.

Zunächst wird ein Testlauf mit diesen Werten durchlaufen, wobei das Infiniband ausgeschaltet ist und nur die Gigabit-Netzwerkschnittstelle zur Verfügung steht. Das resultierende Ergebnis zeigt auf, dass die Transferraten zwischen 95MByte und **105MByte** pro Sekunde liegen, sowohl im Lese- als auch Schreibmodus. Die Gigabit-Netzwerkschnittstelle stellt eine theroetische maximale Transferrate von 125MByte/s zur Verfügung, daher ist anzunehmen, dass die Netzwerkschnittstelle der begrenzende Faktor ist.

KB	reclen	write	rewrite	read	reread	random read	random write	bkwd read	record rewrite	stride read	fwrite	frewrite	fread	freread
512	256	104129	105457	7016251	5627028	103020	104299	64695	188861	100609	3046056	7420958	100353	103790
512	512	105675	102093	106579	107182	105069	107382	93395	107610	102030	106622	105829	103370	105045
1024	4	91575	100778	101195	103907	8370	14617	14807	19114	6423	171151	171011	89809	97971
1024	8	95070	101336	97887	109437	20804	26125	20782	26224	6721	179048	191434	91014	98595
1024	16	86978	90990	93421	109390	29809	42888	39886	43190	8631	196244	197531	98688	100568
1024	32	102574	92670	92636	109344	34633	57363	51377	58308	32917	186043	170837	86471	98888
1024	64	103456	103550	198529	200508	43261	76343	72168	77900	199178	200354	197412	101779	103018
1024	128	104234	103434	195267	197993	44669	102944	70469	84203	201176	198948	201138	100797	102998
1024	256	103831	92619	177962	197831	68627	128367	75863	123327	179683	200234	201258	94482	101425
1024	512	105229	97765	92862	98376	104055	88634	90331	100847	99080	194124	200274	90228	103926
1024	1024	109812	110236	109273	109098	110263	110690	102575	110438	109942	110667	110392	109635	110108
2048	4	99965	98908	99100	106102	12533	15593	16002	13487	6510	117890	128120	95260	101905
2048	8	102155	101316	101255	105817	20717	25741	21475	25740	6982	121723	133690	97158	101900
2048	16	103142	97180	96644	106699	39474	44072	39833	43639	9414	124528	55084	100599	101900
2048	32	102548	103272	101050	106163	42390	56585	52948	58249	42878	129203	134198	92194	102175
2048	64	100846	99848	128402	137468	62429	24683	74448	76931	49132	137119	136432	98253	103444
2048	128	96640	105654	125404	137082	72387	90933	83360	90057	386560	135566	127093	96794	103902
2048	256	97995	101577	129065	137181	100004	101250	87972	102385	394760	134958	135908	96512	103335
2048	512	106700	106394	104896	105637	104913	107101	95339	104292	104821	129637	98141	102713	105235
2048	1024	111129	86566	108913	107937	102165	110936	104579	110804	109613	197929	211942	107241	107062
2048	2048	109959	111419	110012	109401	110072	110244	106135	111008	109566	110024	111226	7883	109519

Abbildung 6.1: FhGFS - One-Node-Installation per Gigabit-Ethernet

Two-Node-Installation mit Ethernetinterface Im zweiten Testfall wird ein zusätzlicher Knoten des Clusters als weiterer Storageserver hinzugeschaltet. Dadurch soll aufgezeigt werden, dass die Daten möglichst gleichmäßig verteilt werden. Durch die unterschiedliche Größe der verfügbaren Festplatten ist eine direkte Verteilung nicht möglich, da auf dem weiteren Storageserver lediglich 50% der Festplattenkapazität zur Verfügung stehen.

Auch das Hinzuschalten eines zweiten Storageservers, sodass eine Verteilung der Daten erfolgt, zeigte bei der Nutzung des Ethernetinterfaces keine Unterschiede. Es ist also davon auszugehen, dass die Netzwerkkarte die maximale Geschwindigkeit von 1000Mbit/s ausreizen kann, diese aber als begrenzender Faktor für die Performancemessung des Dateisystems anzusehen ist. Mit Hilfe der grafischen Auswertungsmöglichkeiten, die von der grafischen Oberfläche angeboten werden, ist eine maximale Transferrate von durchschnittlich **105MByte/s** zu erkennen, wenn nur ein Storageserver hinterlegt ist, bei zwei verfügbaren Knoten erkennt man sehr gut,

dass die Verteilung der Daten gleichmäßig auf beide Server erfolgt, diese allerdings nur mit jeweils 55MByte/s lesen und schreiben können. Die Verteilung funktioniert also einwandfrei, lediglich die Übertragungsrate begrenzt die Geschwindigkeit.

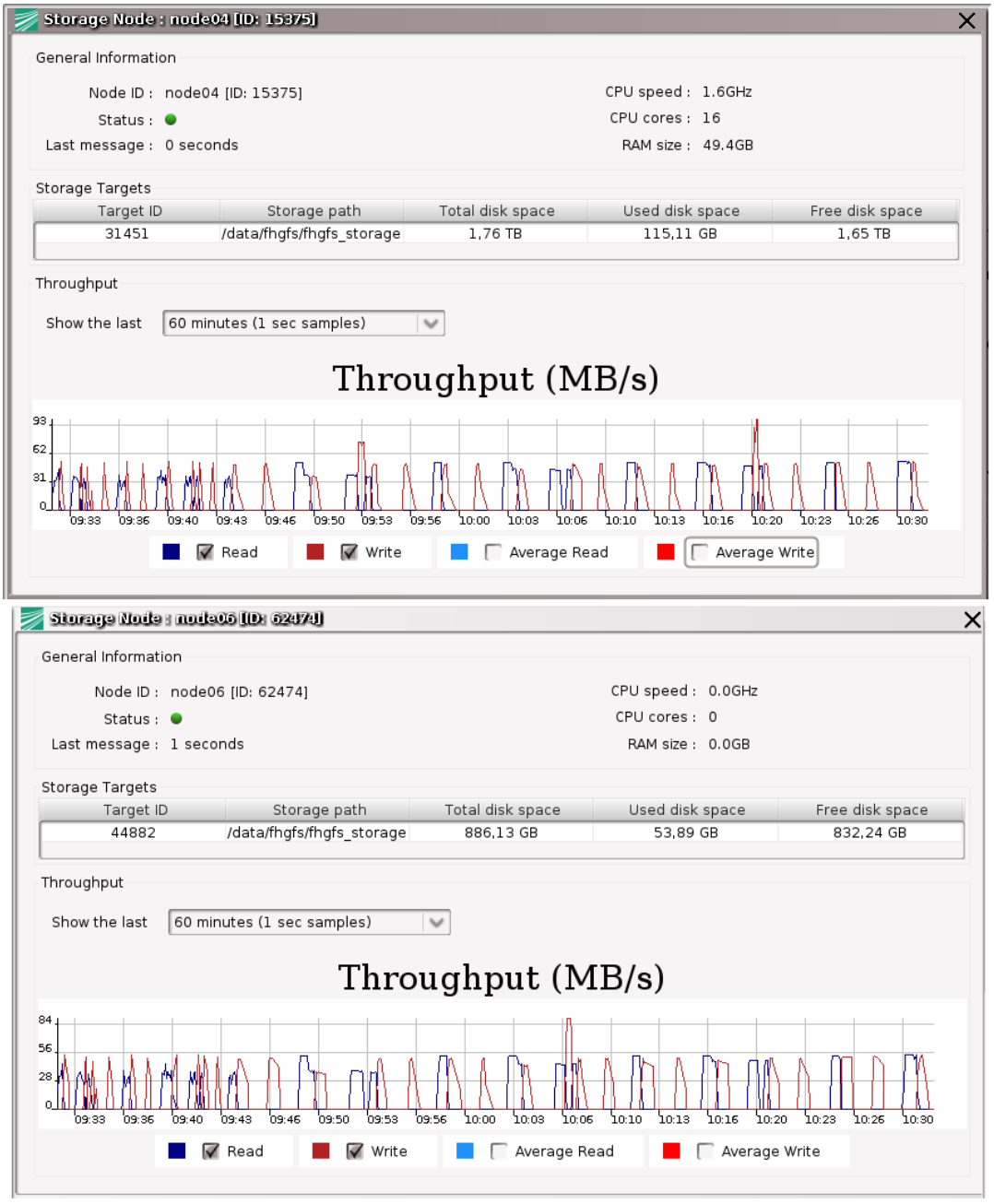

Abbildung 6.2: FhGFS - Lastverteilung FhGFS-Admin-Übersicht

Um den Flaschenhals der begrenzten Netzwerkhardware auszuschalten, wurde in weiteren Testdurchläufen das **Infiniband** auf allen Knoten eingerichtet und aktiviert. Das Fraunhofer Dateisystem erkennt automatisch, dass dieses nun zur Verfügung steht und versucht von nun an, den Datenverkehr über diese Schnittstelle zu regeln. Das Abschalten der Ethernetverbindung ist dabei nicht nötig, diese dient als Failover, wenn das Infiniband nicht zur Verfügung steht. Diese Hardware bietet eine theoretische Geschwindigkeit von 12GBit/s an, sodass diese eine mehr als Verzehnfachung der eigentlichen Geschwindigkeit darstellt. In den folgenden Testdurchläufen ist auch eine wesentliche Erhöhung der Geschwindigkeit zu erkennen, allerdings liegen diese Werte weit unter der möglichen Übertragungsrate von Infiniband, sodass dieser Flaschenhals nicht mehr exisitert.

One-Node-Installation mit Infiniband Die weitere Netzwerkschnittstelle soll dafür sorgen, dass die korrekte Performance des Systems gemessen werden kann. Die Erwartungshaltung ist dementsprechend so, dass die Lese- und Schreibgeschwindigkeit sich auf dem Niveau befindet, die die vorhandene Hardware bietet. Im vorhandenen Cluster arbeiten zwei Standard-2,5“-Festplatten in einem RAID-0 Verbund, welcher über einen zusätzlichen RAID-Controller bewerkstelligt wird. Die Geschwindigkeit liegt dabei bei durchschnittlich **160-180MByte/s** im Lese- und Schreibmodus.

Writer Report				Chunkgröße			
Dateigröße	4KB	8KB	16KB	32KB	64KB	128KB	
512KB	140,24	177,81	177,37	187,33	192,66	191,45	
1024KB	144,03	179,50	170,33	166,03	171,41	173,37	
2048KB	169,21	173,03	173,12	161,63	173,04	176,36	
4096KB	172,05	171,45	175,97	178,15	175,19	175,33	
8192KB	173,97	172,92	175,31	175,57	175,60	173,80	
16384KB	174,72	173,77	175,95	176,20	176,25	175,17	
32768KB	0,00	0,00	0,00	0,00	176,13	176,28	
65536KB	0,00	0,00	0,00	0,00	176,92	154,71	
131072KB	0,00	0,00	0,00	0,00	176,77	169,92	
262144KB	0,00	0,00	0,00	0,00	176,19	140,48	
524288KB	0,00	0,00	0,00	0,00	165,29	176,77	
1048576KB	0,00	0,00	0,00	0,00	168,07	170,21	
2097152KB	0,00	0,00	0,00	0,00	164,95	158,91	
4194304KB	0,00	0,00	0,00	0,00	170,95	170,75	
Durchschnitt:	162,37	174,75	174,67	174,15	174,24	170,25	
Schreibgeschwindigkeit in MByte/s							
Dateigröße	256KB	512KB	1024KB	2048KB	4096KB	8192KB	16384KB
512KB	187,91	188,78					
1024KB	154,16	170,39	172,44				
2048KB	173,63	176,69	207,49	206,10			
4096KB	175,60	185,32	197,80	208,50	206,14		
8192KB	176,22	179,44	205,15	200,54	205,64	204,83	
16384KB	175,86	181,24	135,98	126,89	223,30	207,54	203,62
32768KB	175,48	180,28	205,20	203,96	204,09	209,68	209,56
65536KB	176,34	138,65	209,48	209,53	175,78	180,63	209,48
131072KB	143,90	181,22	209,73	209,69	204,42	216,72	203,88
262144KB	176,74	180,10	171,01	163,23	204,66	209,86	190,15
524288KB	176,33	171,43	200,54	198,97	141,76	181,76	197,05
1048576KB	165,99	176,85	205,83	199,27	162,28	207,97	199,73
2097152KB	165,84	160,88	186,62	204,70	187,79	197,46	199,49
4194304KB	165,31	174,62	192,92	201,09	198,65	201,90	203,32
Durchschnitt:	170,66	174,71	192,32	194,37	192,23	201,83	201,81
Schreibgeschwindigkeit in MByte/s							

Abbildung 6.3: Schreibgeschwindigkeit mit einem Knoten - Auswertung

Wie zu erwarten war liegt die Schreibperformance im absoluten Oberlimit der möglichen Geschwindigkeit der Hardware. Im Durchschnitt liegt die Geschwindigkeit bei **180MByte/s**. Dabei ist ein Anstieg der Geschwindigkeit bei größeren Chunks zu

erkennen. Größere Dateien mit kleinen Chunks müssen häufig den Metadatenserver ansprechen, welcher Teil der Datei wo auf dem Server gespeichert werden soll. Dies kann allerdings nicht als Performanceeinbußen gemessen werden, denn die Hardware stellt keine höheren Transferraten zur Verfügung.

Es ist somit ein weiterer Flaschenhals auszumachen, der nur durch schnellere Festplatten ausgemerzt werden kann, allerdings ist dies im vorliegenden Testfall nicht möglich. Insbesondere kleine Dateien können auch kaum gemessen werden, weil diese sich häufig noch im Zwischencache der Festplatten befinden und entsprechend schnell gelesen werden können, daher wurden Dateien, die kleiner als 512Kbyte sind, nicht in die Bewertung mit einbezogen.

Reader Report				Chunkgröße			
Dateigröße	**4KB**	**8KB**	**16KB**	**32KB**	**64KB**	**128KB**	**256KB**
512KB	144,19	175,68	162,71	171,77	172,88	164,00	194,94
1024KB	154,34	191,60	159,98	186,50	162,88	159,14	194,10
2048KB	170,94	170,66	161,82	169,09	157,40	166,86	183,12
4096KB	168,03	173,16	175,66	164,20	172,06	176,18	178,44
8192KB	171,16	151,51	173,16	174,20	172,18	175,74	175,48
16384KB	174,67	175,19	176,05	177,19	184,35	177,01	175,18
32768KB	0,00	0,00	0,00	0,00	183,31	183,78	183,05
65536KB	0,00	0,00	0,00	0,00	181,79	179,98	182,70
131072KB	0,00	0,00	0,00	0,00	181,59	181,87	182,05
262144KB	0,00	0,00	0,00	0,00	181,15	180,92	180,76
524288KB	0,00	0,00	0,00	0,00	180,78	180,98	181,05
1048576KB	0,00	0,00	0,00	0,00	180,62	181,28	180,80
2097152KB	0,00	0,00	0,00	0,00	180,64	180,97	180,55
4194304KB	0,00	0,00	0,00	0,00	180,85	180,65	180,83
Durchschnitt:	163,89	172,97	168,23	173,82	176,60	176,38	182,36

Lesegeschwindigkeit in MByte/s						
Dateigröße	**512KB**	**1024KB**	**2048KB**	**4096KB**	**8192KB**	**16384KB**
512KB	144,65					
1024KB	174,43	208,50				
2048KB	176,46	206,38	202,61			
4096KB	173,75	205,78	205,89	206,11		
8192KB	151,88	205,68	205,42	205,56	205,28	
16384KB	181,87	208,87	209,09	209,07	208,93	208,58
32768KB	182,64	210,77	210,61	210,66	210,73	210,35
65536KB	184,82	211,58	211,59	211,55	211,58	211,62
131072KB	180,48	211,98	212,03	212,07	211,79	212,11
262144KB	185,71	212,18	211,88	212,27	212,36	212,09
524288KB	185,76	212,15	212,39	212,43	212,41	212,33
1048576KB	185,68	212,28	212,21	212,44	212,39	212,21
2097152KB	186,14	212,35	212,38	212,41	212,27	212,16
4194304KB	186,21	212,36	212,40	212,42	212,39	212,37
Durchschnitt:	177,18	210,07	209,88	210,64	211,01	211,54

Abbildung 6.4: Lesegeschwindigkeit mit einem Knoten - Auswertung

Auch bei der Lesegeschwindigkeit sind keine Überraschungen aufgetreten. Auch hier liegt die durchschnittliche Geschwindigkeit bei etwa **180MByte/s**. Dieser Wert kommt zu Stande, weil kleinere Chunkgrößen mit 160-170MByte/s gelesen werden, größere Chunks mit bis zu 215MByte/s. Auch hier ist das gleiche Phänomen zu erkennen wie beim Schreibtest. Die maximale Performance der vorhandenen Hardware ist hier der begrenzende Faktor, welcher keine höheren Datenübertragungsraten ermöglicht.

Die Performance des Dateisystems hat bei dieser Konstellation alle Erwartungen erfüllen können, auch die Performance bei großen Dateien mit sehr kleinen Chunkgrößen ist entsprechend als sehr gut zu betrachten, weil die Performance nur durch die gegebene Hardware begrenzt wird. Dies führt dazu, dass davon auszugehen ist, dass beim Hinzufügen eines weiteren Servers als Datenserver die Performance steigen sollte, weil eine Verteilung der Daten erfolgt und somit höhere Übertragungsgeschwindigkeiten erreicht werden können.

Two-Node-Installation mit Infiniband Die Erwartungshaltung bei der Verteilung der Daten ist im Vergleich zu der One-Node-Installation so, dass die Performance sich möglichst verdoppelt, da nun zwei Server parallel angesprochen werden können. Bei mehreren Testdurchläufen ist eine Erhöhung der Datenübertragungsrate zu erkennen, allerdings liegt diese etwas unter der Erwartungshaltung. Dies ist dadurch zu erklären, dass bei dem weiteren Server nicht die gleiche Performance zu erwarten ist, weil dort nur eine einzelne Festplatte eingebaut wurde, die nicht so hohe Datenübertragungsraten bietet, wie im Hauptknoten. Die Schreibperformance bei der One-Node-Installation lag im Durchschnitt bei 180MByte/s, die interne Festplatte hat durchschnittlich eine Übertragungsrate von 80-100MByte/s, daher ist eine Performance von **260-280MByte/s** zu erwarten.

Writer Report				Chunkgröße			
Dateigröße	4KB	8KB	16KB	32KB	64KB	128KB	256KB
512KB	174,89	218,15	218,72	227,38	236,86	288,83	255,77
1024KB	265,24	268,90	274,88	256,94	280,43	277,01	253,68
2048KB	248,57	240,12	254,91	273,56	233,21	244,41	256,31
4096KB	267,61	254,22	273,50	263,50	245,11	262,49	269,76
8192KB	239,81	242,12	241,40	266,78	267,69	237,28	264,49
16384KB	243,45	249,33	246,60	261,43	251,24	243,93	260,99
32768KB					257,10	265,65	268,15
65536KB					255,78	263,95	251,72
131072KB					263,03	188,01	275,75
Durchschnitt	239,93	245,47	251,67	258,27	254,50	252,40	261,84

Schreibgeschwindigkeit in MByte/s						
Dateigröße	512KB	1024KB	2048KB	4096KB	8192KB	16384KB
512KB	235,50					
1024KB	241,67	243,60				
2048KB	251,26	264,97	282,21			
4096KB	274,75	278,49	272,38	257,10		
8192KB	275,04	279,38	265,40	256,99	277,83	
16384KB	253,96	255,15	259,21	260,02	268,28	270,07
32768KB	267,88	269,19	275,18	268,25	272,93	269,83
65536KB	279,29	273,04	275,95	273,57	273,37	272,98
131072KB	319,80	284,85	297,95	291,23	278,93	273,40
Durchschnitt	266,57	268,58	275,47	267,86	274,27	271,57

Abbildung 6.5: Schreibgeschwindigkeit mit zwei Knoten - Auswertung

Das Ergebnis des durchgeführten Tests zeigt auf, dass auch hier die maximale Performance beider Server erreicht werden konnte. Durchschnittlich lag die Schreibgeschwindigkeit bei **260MByte/s**. Dies sind genau die Werte, die von der Hardware maximal zur Verfügung gestellt werden können. Die Auslastung beider Server ist somit erreicht. Auch hier ist zu erkennen, dass ein Anstieg der Geschwindigkeit bei größeren Chunks erfolgt. Während bei 4KByte großen Fragmenten

die durchschnittliche Schreibrate bei 240MByte/s lag, wurde bei einer Größe von 2048KByte die maximale Rate von 275,47MByte/s gemessen. Dies ist eine signifikante Änderung der Übertragungsgeschwindigkeit, was darauf schließen lässt, dass das Dateisystem mit größeren Chunks besser umgehen kann. Diese können auch für kleinere Dateien eingesetzt werden, weil diese mit weniger Metadaten geschrieben werden müssen.

Reader Report				Chunkgröße			
Dateigröße	**4KB**	**8KB**	**16KB**	**32KB**	**64KB**	**128KB**	**256KB**
512KB	212,31	320,08	299,77	205,30	407,28	409,81	417,70
1024KB	236,19	254,33	256,48	268,09	391,33	417,61	480,76
2048KB	251,32	276,24	280,66	263,78	369,83	323,10	351,81
4096KB	261,54	249,95	281,14	283,02	343,08	318,57	352,19
8192KB	304,03	290,73	311,97	291,38	325,54	336,32	317,74
16384KB	311,51	336,41	309,99	316,52	337,99	357,88	335,04
32768KB	0,00	0,00	0,00	0,00	338,74	341,71	340,71
65536KB	0,00	0,00	0,00	0,00	342,17	356,03	332,45
131072KB	0,00	0,00	0,00	0,00	345,88	339,61	325,01
Durchschnitt:	262,82	287,96	290,00	271,35	355,76	355,63	361,49

Lesegeschwindigkeit in MByte/s						
Dateigröße	**512KB**	**1024KB**	**2048KB**	**4096KB**	**8192KB**	**16384KB**
512KB	261,22					
1024KB	284,90	326,68				
2048KB	302,97	278,79	313,53			
4096KB	314,24	312,18	281,93	299,94		
8192KB	287,35	325,16	276,50	305,80	314,03	
16384KB	340,60	353,55	330,64	349,21	336,97	349,02
32768KB	366,89	344,99	377,53	358,56	334,09	305,90
65536KB	298,16	376,83	370,33	348,52	342,12	347,84
131072KB	352,25	377,37	376,33	347,31	346,99	387,13
Durchschnitt:	312,06	336,94	332,40	334,89	334,84	347,47

Abbildung 6.6: Lesegeschwindigkeit mit zwei Knoten - Auswertung

Die Leseperformance lag bereits im vorigen Test mit einem Server höher als die Schreibgeschwindigkeit. Es wird erwartet, dass sich im Durchschnitt die Geschwindigkeit des One-Node-Tests multipliziert mit dem Faktor 1,5 ergibt, da sich im zweiten Server lediglich eine statt zwei Festplatten befindet. Daraus ergibt sich ein **Erwartungswert von 320MByte/s**.

Nach der Durchführung des Tests zeigt sich deutlich, dass auch dieses Ergebnis erreicht werden konnte. Während Dateien mit kleinen Chunkgrößen mit 260-290MByte/s gelesen werden konnten, wurde die maximale Rate von 361MByte/s mit einer Größe von 256KByte erreicht, durchschnittlich liegt die Lesegeschwindigkeit bei **320MByte/s**. Die Ergebnisse zeigen auf, dass auch in diesem Fall die maximale

Performance der Server erreicht wurde und das Dateisystem mit größeren Datenraten umgehen kann.

Die grafische Oberfläche, welche bereitgestellt wird, zeigt auf, dass die Daten auf beiden Servern gleichzeitig geschrieben werden, dass die Übertragungsgeschwindigkeit erreicht werden kann. Die Auslastung, welche während des Tests mit dem Befehl „$du - sh$“ auf beiden Knoten berechnet wurde, zeigt auch auf, dass die Daten auf 1MByte exakt genau aufgeteilt werden, sodass die Performance so stark verbessert werden konnte.

Als Ergebnis der Performancemessung des Fraunhofer Dateisystems zeigt auf, dass die Begrenzung der Geschwindigkeit nicht am Dateisystem, sondern an der Performance der Hardware liegt. Dies ist damit zu begründen, dass für die Testzwecke Standard-Festplatten im 2,5Format und 5400U/min verbaut wurden. Diese sind für einen reinen Servereinsatz nicht konzipiert und bieten nicht die Übertragungsraten an, die Hochleistungsfestplatten mit SAS-Anschluss („Serial Attached Storage“) und 15.000U/min anbieten können. Damit wäre eine weitere Verbesserung der Lese- und Schreibraten möglich.

Auch das Thema der Skalierbarkeit wurde im Test aufgezeigt, dass die Geschwindigkeiten sich soweit verbessert haben, wie es die Performance einzelner Server zulässt. Durch das Hinzuschalten weiterer Datenserver und die Möglichkeit, die Metadaten aufzuteilen führt dazu, dass die Performance beim neuen Compute-Cluster des Universitätsrechenzentrums wesentlich verbessert werden wird und somit das Dateisystem eine sehr gute Wahl im Punkt Performance darstellt.

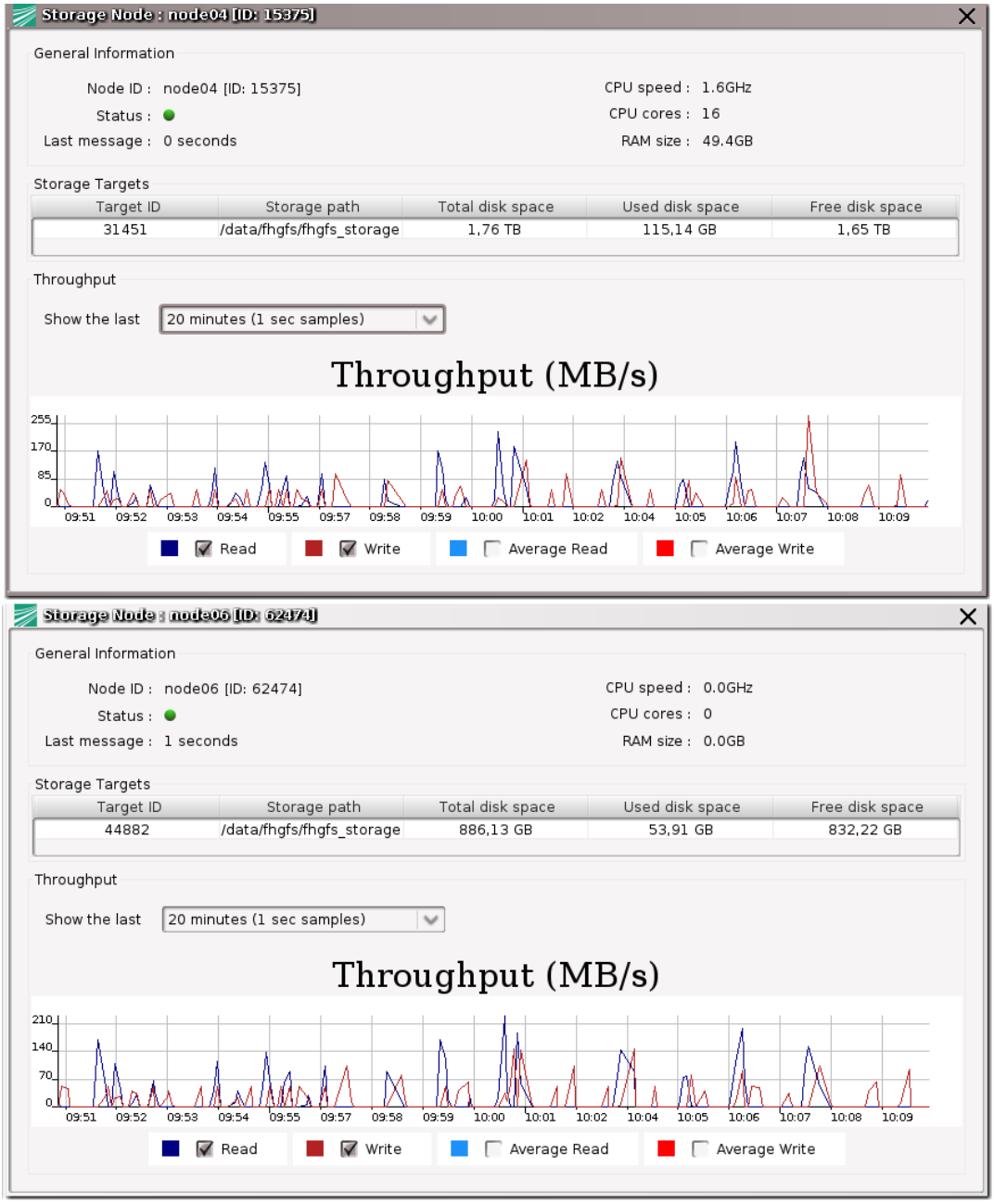

Abbildung 6.7: FhGFS - Two-Node-Installation per Infiniband

6.1.2 MooseFS

Während der Betrachtung über die Arbeitsweise von MooseFS ist aufgefallen, dass das gesamte System sehr durchdacht ist und eine hohe Performance bieten könnte. Die Geschwindigkeiten wurden zunächst mit einem Master-, einem Storageserver und einem Client durchgeführt. Anschließend erfolgte eine Messung, indem ein weiterer Storageserver hinzugeschaltet wurde.

Die Erwartungshaltung war so, dass die Geschwindigkeit möglichst die Grundperformance der genutzten Festplatten ausreizen konnte, sowie die Verbesserung, wenn ein weiterer Server hinzugeschaltet wird.

Aufgrund der Begrenzung von Gigabit-Ethernet und der fehlenden Unterstützung von Infiniband, wurde IPoIB genutzt. Das bedeutet, dass einzelne Server mit einer IP-Adresse über Infiniband angesprochen werden können. Dieses erreicht nicht die volle theroetische Geschwindigkeit von 12GBit/s, im vorliegenden Testsystem wird eine maximale Übertragungsrate von 1-1,5Gbit/s erreicht.

MooseFS bietet die Möglichkeit, dass Storageserver als **Replikationsserver** eingesetzt werden können. Die Geschwindigkeit wird dabei abnehmen, denn die Bestätigung, dass die Daten korrekt geschrieben wurden, erreicht den Client erst dann, wenn die Daten korrekt auf alle Server übertragen wurde. Dies würde das Ergebnis im Vergleich zum Fraunhofer Filesystem verändern, da dieses keine Replikationsmöglichkeiten anbietet.

Performance mit einem Storageserver Die Auswertung der Testergebnisse ist sehr enttäuschend. Während FhGFS mit der Installation des Metadaten- und Storageservers auf einem einzelnen Knoten dessen maximale Performance erreichen konnte, bricht die Geschwindigkeit bei MooseFS stark ein.
Aus den Testergebnissen ist zu entnehmen, dass die Schreibgeschwindigkeit unabhängig

				Chunkgröße			
	4KB	8KB	16KB	32KB	64KB	128KB	256KB
1MB	10,44	18,51	19,19	15,49	17,66	21,55	18,83
2MB	10,35	14,29	13,04	15,92	19,85	18,00	17,79
4MB	10,42	12,84	14,35	15,28	15,55	16,12	16,63
8MB	10,59	12,41	14,62	15,65	16,45	17,35	18,38
16MB	9,96	12,66	14,79	15,95	18,50	16,70	19,56
32MB	12,88	13,29	15,58	16,72	14,60	15,99	17,19
64MB	10,32	13,38	14,18	15,33	18,89	16,52	17,62
128MB	10,40	13,46	14,49	14,47	20,16	14,51	6,52
256MB	10,64	8,84	5,48	9,43	12,48	2,42	9,75
512MB	6,52	15,97	14,86	16,40	4,49	8,64	8,20
1024MB	8,86	7,66	14,03	9,32	7,47	7,45	12,70
2048MB	9,06	8,54	15,32	8,24	9,47	11,07	9,38

Dateigröße in MByte

	512KB	1024KB	2048KB	4096KB	8192KB	16384KB
1MB	18,12	17,57	0,00	0,00	0,00	0,00
2MB	19,01	18,35	18,58	0,00	0,00	0,00
4MB	16,62	16,47	15,50	17,52	0,00	0,00
8MB	26,72	17,18	16,44	15,30	16,27	0,00
16MB	17,91	16,62	17,09	16,89	17,20	16,42
32MB	17,00	17,83	16,59	16,78	17,22	16,45
64MB	17,35	17,36	17,72	15,66	16,26	17,38
128MB	14,50	14,44	14,50	14,45	14,51	14,52
256MB	6,18	11,03	38,62	24,74	6,07	8,58
512MB	8,76	18,18	5,38	9,09	6,01	11,00
1024MB	15,10	8,96	10,58	8,63	7,86	10,41
2048MB	8,70	8,06	14,51	14,01	14,27	12,56

Abbildung 6.8: MooseFS - Schreibgeschwindigkeit mit einem Storageserver

von der Chunkgröße **zwischen 10 und 20MByte/s** liegt. Dieses Ergebnis liegt weit entfernt von der theoretischen Leistung. Da der Storageserver auf dem Knoten mit nur einer Festplatte installiert wurde, liegt die theroetische Geschwindigkeit bei 80-100MByte/s. Besonders der Umstand, dass die Geschwindigkeit bei Dateigrößen über 512MByte weiter einbricht, zeigt, dass das System mit den Dateigrößen nur bedingt umgehen kann. Die Geschwindigkeit liegt dabei teilweise unter **10Mbyte/s**, welches selbst mit einem 100Mbit-Ethernetinterface erreicht werden könnte. Dieses Ergebnis ist absolut inakzeptabel für das Universitätsrechenzentrums, daher wurde die Leseperformance nicht weiter getestet.

Performance mit zwei Storageservern Nach den enttäuschenden Ergebnissen des Dateisystems mit einem einzelnen Storageserver ist die Erwartungshaltung entsprechend gering, dass sich die Performance beim Hinzufügen eines weiteren Storageservers weiter verbessern könnte. In der Tat ist die Geschwindigkeit überhaupt nicht angestiegen, sodass es zu keinen Geschwindigkeitsänderungen kam.

			Chunkgröße in KByte				
	4,00	**8,00**	**16,00**	**32,00**	**64,00**	**128,00**	**256,00**
1,00	12,28	11,03	12,07	13,04	17,86	18,75	18,06
2,00	12,56	13,46	15,59	16,32	16,75	18,04	18,19
4,00	10,09	12,02	14,37	15,38	16,08	16,49	16,73
8,00	10,40	12,56	14,16	21,34	18,77	16,32	16,35
16,00	11,61	12,59	15,11	17,25	17,47	16,68	17,09
32,00	10,59	14,90	15,94	16,38	15,47	16,06	17,01
64,00	10,88	13,54	14,92	14,70	6,50	16,66	16,94
128,00	10,83	13,31	14,86	14,88	14,87	14,87	14,89
256,00	4,57	12,44	5,26	12,76	12,76	12,62	4,62
512,00	8,69	71,34	5,62	5,95	11,52	9,58	11,88
1024,00	6,27	56,33	6,17	5,46	5,15	7,98	6,59
2048,00	5,41	50,41	5,16	5,89	5,31	7,15	6,34

Dateigröße in MByte						
	512,00	**1024,00**	**2048,00**	**4096,00**	**8192,00**	**16384,00**
1,00	16,51	16,60	0,00	0,00	0,00	0,00
2,00	18,66	17,76	16,83	0,00	0,00	0,00
4,00	16,96	11,45	15,87	17,04	0,00	0,00
8,00	16,57	16,43	20,73	17,37	21,34	0,00
16,00	17,11	16,80	16,75	16,21	16,70	16,30
32,00	16,74	17,49	16,48	18,58	18,51	16,22
64,00	16,95	17,20	17,15	17,38	16,68	16,22
128,00	14,88	14,88	14,88	14,89	14,88	14,87
256,00	5,48	4,86	12,71	12,35	12,76	7,80
512,00	5,51	8,80	5,58	4,55	11,13	10,82
1024,00	8,79	6,14	6,75	8,28	9,25	5,45
2048,00	5,64	5,82	4,90	4,70	5,51	5,99

Abbildung 6.9: MooseFS - Schreibgeschwindigkeit mit zwei Storageserver

Wie auch die Ergebnisse der Testdurchläufe mit einem Storageserver sind auch diese Werte mehr als enttäuschend. Hier sticht lediglich der Testbereich mit einer Chunkgröße von 8KByte heraus, dort werden Werte von **50 - 70MByte/s** erreicht. Ansonsten liegen die Werte wieder im Bereich von **5-15MByte/s**. Diese Werte sind absolut inakzeptabel, daher wurde auch hier auf einen Test der Lesegeschwindigkeit verzichtet.

MooseFS zeigt im internen Aufbau sehr gute Eigenschaften auf, die auf eine gute Performance schließen lassen könnten. Allerdings sind in der Realität die Werte so schlecht, dass der Test abgebrochen wurde und das Dateisystem nicht weiter zur Auswahl steht. Der Einbruch eines verteilten Dateisystems, insbesondere bei Dateigrößen ab 256MByte, auf solch niedrige Werte ist absolut inakzeptabel. Daher werden die Werte zwar erwähnt, allerdings steht dieses System nicht mehr zur Auswahl für das Universitätsrechenzentrum.

6.1.3 GlusterFS

Anders als die bereits vorgestellten Dateisysteme verfolgt GlusterFS eine andere Herangehensweise an den grundsätzlichen Aufbau und der Verteilung der Daten. Es erfolgt keine Aufteilung zwischen Metadaten und den eigentlichen Daten, sondern diese werden zentral pro Server abgelegt. Der Hauptknoten stellt somit automatisch auch seine verfügbaren Festplattenkapazitäten als Storageserver zur Verfügung.

Die einzelnen Server müssen im Client bekannt gemacht werden, sodass bereits auf Clientseite eine Verteilung der Daten stattfinden kann. Es ist kein zusätzlicher Metadatenserver nötig.

Die Installation von GlusterFS wurde daher mit einem Server und einem Client durchgeführt. Anschließend wurde ein zweiter Knoten hinzugeschaltet, welcher auch im Client bekannt gemacht wurde. Die Verteilung der Daten erfolgte somit automatisch, ohne dass größere Umstrukturierungen durchgeführt werden mussten.

Es muss dabei darauf geachtet werden, dass der Client selbst keine Kapazitäten zur Verfügung stellt, dieses muss in den Konfigurationsdateien entsprechend ausgespart werden.

Aufgrund der Limitierung der Geschwindigkeit des Gigabit-Ethernet-Interface wurde direkt Infiniband eingerichtet, sodass dieser Flaschenhals nicht zu verfälschten Ergebnissen führen kann.

One-Node-Installation Die Installation von GlusterFS ist rein auf Kommandozeilen beschränkt worden. Es existiert keine grafische Oberfläche. Auch die Installationsanweisung und die Dokumentation auf der offiziellen Homepage *www.gluster.org* sind nicht vollständig. Die Bekanntgabe der einzelnen Storageserver und Clients ist nicht beschrieben worden. Diese sind für den funktionierenden Betrieb allerdings elementar.

Bei der One-Node-Installation wurde der Hauptknoten des vorhandenen Clusters, welcher mit zwei Standard-2,5“-Festplatten im RAID-0-Modus ausgestattet ist, als Storageserver ausgewählt. Der Client bietet keine zusätzliche Speicherkapazität an, sodass eine Verbindung nur auf einem Server gegeben ist.

Writer Report				Chunkgröße			
Dateigröße	**4KB**	**8KB**	**16KB**	**32KB**	**64KB**	**128KB**	**256KB**
1024KB	69,35	81,64	81,70	91,07	92,42	103,30	90,00
2048KB	73,24	83,58	89,37	91,28	95,65	95,65	95,46
4096KB	74,74	84,48	103,04	94,31	97,28	97,90	97,65
8192KB	116,87	123,19	91,89	96,05	98,43	99,09	99,48
16384KB	94,98	126,80	101,21	96,26	98,55	141,36	135,08
32768KB	118,59	110,35	112,51	154,61	124,74	129,94	132,55
65536KB	115,73	142,38	145,81	142,44	136,26	142,38	163,37
131072KB	126,84	137,52	140,44	152,86	159,70	159,08	143,88
262144KB	126,35	139,91	153,69	160,84	167,13	160,66	164,64
524288KB	130,73	147,60	157,81	161,86	165,30	169,85	171,86
1048576KB	133,75	149,63	162,21	167,88	170,32	171,96	175,57
2097152KB	133,86	150,39	160,26	168,21	172,99	174,41	175,75
Durchschnitt:	109,59	123,12	125,00	131,47	131,57	137,13	137,11

Schreibgeschwindigkeit in MByte/s						
Dateigröße	**512KB**	**1024KB**	**2048KB**	**4096KB**	**8192KB**	**16384KB**
1024KB	86,58	82,03	0,00	0,00	0,00	0,00
2048KB	96,81	93,55	91,12	0,00	0,00	0,00
4096KB	97,84	97,23	115,04	92,89	0,00	0,00
8192KB	133,62	98,59	97,32	138,57	92,57	0,00
16384KB	97,85	123,67	96,58	93,99	115,21	124,82
32768KB	139,79	122,24	123,18	141,72	130,45	128,78
65536KB	138,92	163,31	152,52	132,99	139,27	128,18
131072KB	158,61	154,75	157,98	155,30	140,67	150,53
262144KB	165,89	167,91	161,66	157,60	153,00	149,91
524288KB	169,56	171,11	168,28	162,80	154,72	154,36
1048576KB	173,97	173,47	170,87	167,41	158,06	156,01
2097152KB	176,84	175,85	173,46	169,26	160,06	159,24
Durchschnitt:	136,36	135,31	137,09	141,25	138,22	143,98

Abbildung 6.10: GlusterFS - Schreibperformance mit einem Knoten

Die Schreibgeschwindigkeit von GlusterFS liegt im Durchschnitt bei ca. **135MByte/s**. Dies liegt unterhalb der möglichen Geschwindigkeit, die von der Hardware zur Verfügung gestellt wird.
Insbesondere der Umgang mit besonders kleinen Chunkgrößen stellt GlusterFS vor größeren Herausforderungen, die Geschwindigkeit liegt deutlich unter der von größeren Chunks. Die Geschwindigkeitseinbußen kommen hauptsächlich vom Schreiben von Dateien die Kleiner als 4096KByte sind. Dabei ist allerdings zu beachten, dass die exakte Bestimmung der Geschwindigkeit aufgrund der sehr schnellen Bearbeitung kaum möglich ist, die Werte dienen lediglich als Richtwerte. Die Performance steigt auch mit zunehmender Dateigröße an, dort ist eine exaktere Bestimmung der Geschwindigkeit möglich. Diese liegt im Durchschnitt bei **160MByte/s**, welches eher der maximal möglichen Performance des Systems entspricht.

Reader Report				Chunkgröße			
Dateigröße	**4**	**8**	**16**	**32**	**64**	**128**	
1024	131,09	147,67	148,42	143,53	143,94	144,16	
2048	132,38	143,73	145,50	142,70	153,38	158,44	
4096	152,91	160,75	146,62	145,79	147,28	147,97	
8192	136,64	141,09	144,63	151,64	151,96	146,34	
16384	147,18	144,30	149,05	160,48	163,31	159,89	
32768	142,32	158,90	158,55	161,12	161,00	149,53	
65536	137,25	157,59	155,79	147,50	160,55	149,02	
131072	144,18	144,06	161,28	161,76	154,76	154,73	
262144	150,71	162,75	161,32	164,04	163,79	159,66	
524288	151,21	165,66	165,67	163,02	164,12	165,69	
1048576	153,47	165,40	162,43	170,12	170,35	167,33	
2097152	156,29	168,02	168,40	170,41	172,21	168,05	
Durchschnitt:	144,64	154,99	155,64	156,84	158,89	155,90	
	256	**512**	**1024**	**2048**	**4096**	**8192**	**16384**
1024	148,14	151,16	151,40	0,00	0,00	0,00	0,00
2048	154,95	153,21	153,66	147,28	0,00	0,00	0,00
4096	154,96	148,16	163,02	149,34	147,53	0,00	0,00
8192	156,66	153,06	156,57	148,45	152,04	139,17	0,00
16384	146,35	145,76	158,93	156,22	151,97	142,94	138,66
32768	144,59	159,11	144,60	141,26	145,11	135,39	143,10
65536	158,98	147,77	157,79	156,10	152,52	143,54	143,32
131072	161,08	158,70	151,34	151,43	153,87	140,50	143,19
262144	157,70	159,54	155,12	153,74	151,77	143,45	143,67
524288	164,66	158,77	162,01	159,87	156,54	145,13	144,23
1048576	163,87	164,58	163,76	160,77	156,63	145,28	144,46
2097152	165,61	164,99	164,38	161,91	158,71	145,87	144,60
Durchschnitt:	156,46	155,40	156,88	153,31	152,67	142,36	143,15

Abbildung 6.11: GlusterFS - Leseperformance mit einem Knoten

Die Leseperformance zeigt eine deutlich gleichmäßigere Performance. Sowohl bei kleinen, als auch bei großen Chunkgrößen beträgt der Unterschied maximal 10MByte/s. Auch hier ist zu erkennen, dass große Dateien schneller übertragen werden konnten, als sehr kleine.

Grundsätzlich liegt die Leseperformance mit einem Storageknoten im Durchschnitt bei **150MByte/s**. Bei großen Dateien ist bei Chunkgrößen von 32 und 64KByte zu erkennen, dass die Geschwindigkeit auf über 170MByte/s anwächst, dies ist in etwa die Performance, die die Hardware erreichen kann.

Im Vergleich zu MooseFS und FhGFS liegt dieses Dateisystem im Mittelfeld. Die Geschwindigkeit sowohl im Lese- als auch im Schreibmodus ist für die vorhandene Hardware als gut anzusehen, allerdings liegt die Performance von FhGFS noch ein wenig höher und kann somit die Hardware noch besser ausnutzen. Die Performance von MooseFS ist dagegen sehr schlecht und kann sich nur ganz weit unten positionieren.

Two-Node-Installation Aufgrund des Aufbaus von GlusterFS ist ein Anstieg der Geschwindigkeit im Lesen und Schreiben einzelner Dateien nicht zu erwarten. Durch den veränderten Aufbau wird eine einzelne Datei lediglich auf einem Knoten abgespeichert. Selbst beim Hinzufügen zusätzlicher Storageserver ist ein Anstieg beim Lesen und Schreiben einer einzelnen Datei nicht zu erwarten.

Writer Report			Chunkgröße in KByte				
Dateigröße	**4**	**8**	**16**	**32**	**64**	**128**	
1024	70,77	78,86	87,18	85,53	90,66	85,17	
2048	71,43	79,94	87,15	90,50	106,44	91,41	
4096	73,31	85,58	89,34	92,72	95,78	97,09	
8192	74,31	104,64	101,33	104,52	96,37	98,12	
16384	109,21	85,66	142,97	97,13	137,30	98,57	
32768	106,54	114,31	118,36	151,69	152,13	121,08	
65536	114,62	131,63	136,79	151,94	145,17	161,56	
131072	129,61	143,63	151,85	148,15	163,94	161,57	
262144	127,69	146,81	153,45	154,29	165,47	164,77	
524288	130,79	147,16	158,21	162,57	168,61	169,65	
1048576	131,39	150,10	159,83	166,31	169,17	171,68	
2097152	133,82	150,69	159,81	165,16	171,04	172,31	
Durchschnitt:	106,13	118,25	128,85	130,88	138,51	132,75	
	256	**512**	**1024**	**2048**	**4096**	**8192**	**16384**
1024	87,87	84,74	119,63	0,00	0,00	0,00	0,00
2048	92,59	91,65	91,83	88,26	0,00	0,00	0,00
4096	95,81	95,74	95,95	93,74	93,22	0,00	0,00
8192	98,66	97,32	99,45	96,03	101,99	92,73	0,00
16384	115,14	99,45	99,57	149,95	129,00	134,58	117,59
32768	150,65	120,55	131,56	149,80	143,94	147,25	110,14
65536	154,15	164,60	155,77	148,25	154,43	138,28	138,53
131072	159,06	158,08	155,54	148,90	146,44	146,00	138,85
262144	168,00	164,40	165,95	160,87	158,56	147,97	151,39
524288	167,76	171,84	171,59	167,92	165,03	153,15	155,10
1048576	172,52	174,70	174,26	169,88	167,15	156,99	155,88
2097152	171,34	174,78	173,88	172,11	169,93	158,25	158,88
Durchschnitt:	136,13	133,15	136,25	140,52	142,97	141,69	140,80
Schreibgeschwindigkeit in MByte/s							

Abbildung 6.12: GlusterFS - Schreibperformance mit zwei Knoten

In der Abbildung 6.12 ist zu erkennen, dass sich die Schreibperformance gegenüber dem vorherigen Ergebnis mit einem Knoten nicht verändert hat. Die Geschwindigkeit liegt im Durchschnitt bei **135MByte/s**. Auch hier ist eine Zunahme der Geschwindigkeit bei größeren Dateien unabhängig von der Chunkgröße zu erkennen, insbesondere bei kleinen Chunkgrößen ist die Schreibgeschwindigkeit fast doppelt so hoch. Dies scheint daran zu liegen, dass Geschwindigkeit sehr hoch ist und die Zeit, um die Datei zu übertragen, so gering, dass geringste Abweichungen schon zu einem großen Unterschied führen. Bei besonders großen Dateien liegt die durchschnittliche

Schreibrate bei **165MByte/s**.

Es ist also auch hier ein Unterschied bei kleineren und sehr großen Chunkgrößen zu erkennen. Um die möglichst optimale Schreibperformance zu erhalten, sollte daher eine Chunkgröße zwischen 128 und 1024KByte gewählt werden.

Reader Report				Chunkgröße			
Dateigröße	4KB	8KB	16KB	32KB	64KB	128KB	256KB
1024KB	155,48	170,06	106,94	161,82	125,15	190,23	190,74
2048KB	100,69	171,15	99,52	202,71	154,39	185,57	144,27
4096KB	193,39	175,24	188,95	271,18	225,62	236,23	232,49
8192KB	142,74	151,76	130,51	152,18	144,82	211,11	194,54
16384KB	124,01	135,17	161,35	125,68	254,25	251,97	161,11
32768KB	185,28	148,67	122,03	148,04	229,45	255,71	138,54
65536KB	124,51	105,46	169,59	117,51	216,00	153,90	129,29
131072KB	160,80	115,34	179,84	160,21	215,50	212,94	104,83
262144KB	170,73	163,33	180,12	140,20	199,64	201,54	222,35
524288KB	168,77	184,59	165,31	221,70	231,29	242,61	213,87
1048576KB	166,24	107,74	120,77	240,64	219,27	210,29	216,46
2097152KB	101,93	102,14	150,07	249,29	252,69	252,65	245,52
Durchschnitt:	149,55	144,22	147,92	182,60	205,67	217,06	182,84

Lesegeschwindigkeit in MByte/s						
Dateigröße	512KB	1024KB	2048KB	4096KB	8192KB	16384KB
1024KB	111,87	158,88	0,00	0,00	0,00	0,00
2048KB	161,79	187,52	202,97	0,00	0,00	0,00
4096KB	124,80	126,21	149,08	156,49	0,00	0,00
8192KB	178,71	167,98	149,34	133,04	182,24	0,00
16384KB	213,68	163,96	138,48	123,85	164,83	161,57
32768KB	207,41	106,65	177,46	149,64	164,96	162,35
65536KB	197,89	122,63	156,83	157,23	120,42	154,18
131072KB	181,40	226,98	221,48	131,95	117,17	115,77
262144KB	230,49	242,91	217,55	166,29	156,87	142,64
524288KB	220,73	243,66	249,71	163,81	174,76	172,63
1048576KB	211,91	235,07	231,15	202,36	188,97	176,81
2097152KB	253,41	241,99	252,38	220,95	189,92	176,40
Durchschnitt:	191,17	185,37	195,13	160,56	162,24	157,79

Abbildung 6.13: GlusterFS - Leseperformance mit zwei Knoten

Im Gegensatz zur Schreibrate hat sich die Geschwindigkeit beim Lesen leicht erhöht. Der Durchschnitt liegt hier bei ca. **190MByte/s**. Dies ist die maximale Performance, die vom Hauptknoten zur Verfügung gestellt wird. Die Unterschiede in den verschiedenen Lesegeschwindigkeiten sind allerdings auch höher. Daraus kann man erkennen, dass die Daten verteilt wurden und auch der zweite Storageserver benutzt wurde. Dieser bietet lediglich Transferraten von maximal **100MByte/s** an, welche mit einigen Dateien auch erreicht wurden.

Die Performancemessung mit zwei verschiedenen Knoten ist beim Aufbau von GlusterFS sehr schwierig, da immer eine einzelne Datei auf einem einzelnen Knoten abgelegt wird. Die Geschwindigkeit kann daher mit diesen Testverfahren nicht korrekt ermittelt werden.

Insgesamt bietet GlusterFS ein durchschnittliches Ergebnis an. Während FhGFS die Daten verteilt auf mehrere Server speichern kann und damit sowohl einen Schreib- als auch Lesegeschwindigkeitsvorteil erhält, so kommt es bei GlusterFS nur zu einer Verteilung bei der Nutzung vieler Dateien. Erst dann kann die Geschwindigkeit stark ansteigen. Trotzdem ist GlusterFS im vorliegenden
Testfall immer schneller als MooseFS. Beide bieten die Möglichkeit von Replikationen an.

6.1.4 Zusammenfassung Performance

Zusammenfassend im Punkt Performance ist eindeutig zu erkennen, dass **FhGFS** die höchste Geschwindigkeit anbietet. GlusterFS ist aufgrund der Arbeitsweise nicht in der Lage die Geschwindigkeit zu erhöhen, weil im vorliegenden Testfeld immer nur eine Datei geschrieben wird, es ist allerdings noch schneller als MooseFS. Die Performance von diesem System ist absolut inakzeptabel, sodass dieses für das Universitätsrechenzentrum nicht weiter zur Auswahl steht.

6.2 Stabilität

Damit aussagekräftige Ergebnisse zu dem Thema Stabilität von Dateisystemen gemacht werden können, müssen verschiedene Punkte beachtet werden. Stabilität bedeutet, dass das Dateisystem zum einen stets erreichbar ist, dass die einzelnen Serverkomponenten keine internen Fehler zum Vorschein bringen und dass die gespeicherten Daten zu jeder Zeit verfügbar sind.
Stabilität im Fehlerfall ist ebenfalls ein sehr wichtiges Thema, da insbesondere bei Clustersystemen viel Hardware verbaut wird und diese dementsprechend wahrscheinlicher für einzelne Ausfälle gegenüber Einzelservern sind. Ein Dateisystem gilt als stabil, wenn die gespeicherten Daten zu jeder Zeit verfügbar sind und wenn im Falle eines Ausfalls einzelner Komponenten diese auch weiterhin zur Verfügung stehen.

6.2.1 FhGFS

Das Fraunhofer Filesystem zeichnete sich im Testfeld als zunächst sehr stabil aus. Es kam zu keiner Zeit zu Ausfällen der einzelnen Serverkomponenten, sodass die Stabilität softwareseitig gegeben ist. Im Testszenario wurden Dauertests mit gleichzeitigen Lese- und Schreibzugriffen über **72 Stunden** durchgeführt. Dabei kam es zu keiner Zeit zu einem Fehler.

Auch die Geschwindigkeit trotz hoher Datenmengen und hoher Anzahl an Metadateneinträgen wurde nicht beeinträchtigt. Allerdings fehlt dem Dateisystem die Möglichkeit von Replikationen. Es besteht lediglich der Grundansatz, dass ein Administrator zu bestimmten Zeiten kleinere **Snapshots** einzelner Festplatten durchführen kann, bei einem abrupten Ausfall einzelner Komponenten kommt es allerdings unweigerlich zu Datenverlusten, bzw. zu Systemausfällen, wo bestimmte Dateien nicht zur Verfügung stehen.

FhGFS verändert das interne Dateisystem von einzelnen Knoten im Cluster nicht, sodass Festplattenfehler mit Hilfe von anderen Hilfsmitteln, wie zum Beispiel bestimmte RAID-Konfigurationen, abgefangen werden und somit der Datenverlust verhindert werden kann. Allerdings bietet das Dateisystem keine Möglichkeit, automatische Replikationen anzufertigen.

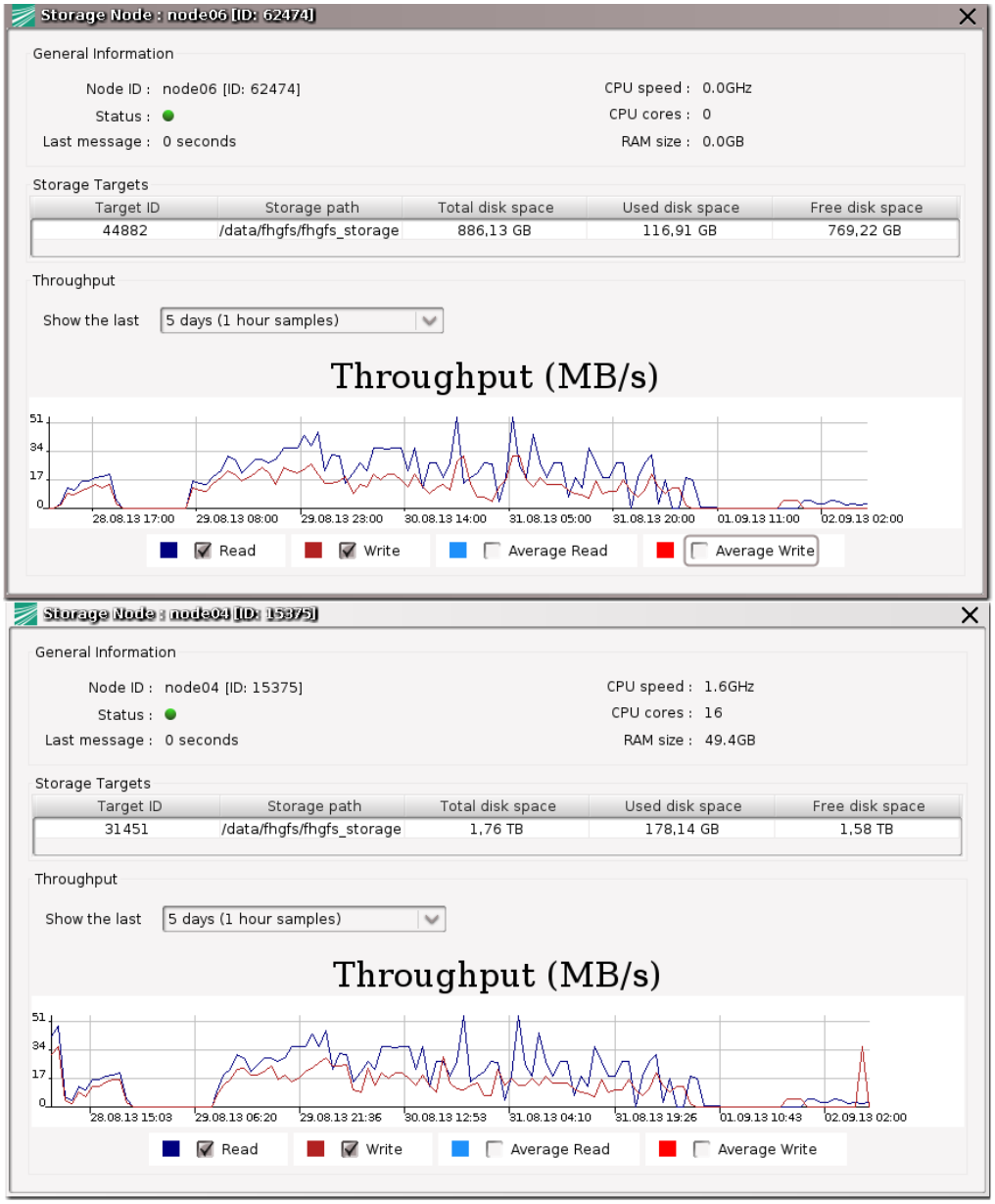

Abbildung 6.14: FhGFS - Lasttest über 72 Stunden

Das keine Fehler bei einem Lasttest auftreten, ist bei einem verteilten Dateisystem elementar wichtig. Dieses Feature konnte das Fraunhofer Dateisystem erfüllen, sodass dieser Punkt positiv zu rechnen ist für die Entscheidungsfindung des Universitätsrechenzentrums für die Auswahl des Dateisystems.

6.2.2 MooseFS

MooseFS wird als ein sehr ausfallsicheres und stabiles Dateisystem angepriesen. In der vorgegebenen Testkonfiguration wurde ein Dauertest durchgeführt, welcher aufzeigen sollte, ob es zu Fehlern kommt. Dieser wurde allerdings nach **24 Stunden** abgebrochen, weil die Performance für die gegebenen Verhältnisse zu schlecht war, als das das Thema Stabilität noch eine Rolle spielen würde. Innerhalb der 24 Stunden konnten keine Fehler festgestellt werden, sodass zumindest für einen kurzen Zeitraum das Dateisystem als stabil anzusehen ist. Dies ist jedoch weniger aussagekräftig, weil der Test kurzfristig abgebrochen wurde.

6.2.3 GlusterFS

Im vorliegenden Testfeld wurde ein GlusterFS Dateisystem mit 2 Servern und einem Client aufgebaut. Der Client führte einen Dauertest von **72 Stunden** durch. Dafür wurde das Testprogramm „iozone“ fünfmalig ausgeführt. Dieses führte sowohl Lese- als auch Schreiboperationen durch.
GlusterFS zeigte sich als sehr stabiles Dateisystem, welches zu keiner Zeit Fehler aufzeichnete. Dieses wäre von „iozone“ bemerkt worden. Zusätzlich ist durch die Möglichkeit der **Selbstrekonfiguration** die Möglichkeit gegeben, dass ein Weiterarbeiten möglich ist, auch im Falle eines Defektes eines Serverknotens.

Während des Lasttests wurde der zusätzliche Storageserver vom Netzwerk getrennt, sodass dieser nicht mehr zur Verfügung stand. Die Geschwindigkeit ist dann kurzzeitig eingebrochen, allerdings waren keine Fehler zu erkennen. GlusterFS ist ein sehr ausgereiftes und stabiles Dateisystem, welches durch die automatische Rekonfiguration es ermöglicht, trotz Hardwareausfällen weiter zu arbeiten.

6.2.4 Zusammenfassung Stabilität

Zusammenfassend ist zu erkennen, dass alle Dateisysteme zum aktuellen Zeitpunkt sehr stabil sind. Dies wird durch die weite Verbreitung von GlusterFS und FhGFS unterstützt, auch der Einsatz von MooseFS ist laut Berichten sehr zuverlässig.

Für den Punkt Sicherheit ist GlusterFS am besten geeignet, da es durch die Replikationsmöglichkeiten, sowie die automatische Rekonfiguration es ermöglicht, dass

es trotz Hardwareausfällen zu keiner Störung des laufenden Betriebes kommen kann. Auch MooseFS bietet diese Möglichkeiten an, allerdings ist wurde auf ein Lasttest verzichtet, da die Grundperformance nicht gegeben ist und das System nicht mehr zur Auswahl steht.
FhGFS bietet ein sehr stabiles Dateisystem an, allerdings sind keine Möglichkeiten gegeben, dass das System auch nach einzelnen Hardwareausfällen weiter funktioniert. Die Möglichkeiten der Sicherung von Metadaten- und Storageservern ist zwar gegeben, müssen aber manuell vom Administrator geplegt werden. Die Sicherung besteht hauptsächlich auf Festplattenebene durch verschiedene RAID-Konfigurationen. Diese können allerdings auch bei anderen Dateisystemen eingesetzt werden.

Dadurch, dass das Universitätsrechenzentrum einen Compute-Cluster hauptsächlich für wissenschaftliche Berechnungen benutzen wird, ist auf den Punkt Stabilität relativ viel Wert zu legen. Wenn ein Knoten zwischenzeitlich ausfällt, kann es passieren, dass beim Einsatz von FhGFS die gesamten Berechnungen erneut durchgeführt werden müssen, da die Ergebnisse verloren sind. Die kurzzeitigen Ausfälle könnten allerdings im laufenden Betrieb in Kauf genommen werden, da die Performance höher liegt. Rein für den Punkt Stabilität wird eine Empfehlung für GlusterFS gegeben.

6.3 Skalierbarkeit

Das Thema Skalierbarkeit tritt insbesondere in großen „High-Performance-Computing“-Clusterverbunden auf. Durch das Hinzufügen zusätzlicher Hardware wird erwartet, eine möglichst **lineare Skalierbarkeit** zu erreichen. Dieses muss vom Dateisystem auch möglich gemacht werden. In der aktuellen Testkonstellation stehen dafür zwei Server zur Verfügung, die als Storageserver dienen. Um eine mögliche Skalierbarkeit aufzuzeigen, werden die gleichen Testverfahren zunächst mit einem Storageserver durchgeführt, bei einem weiteren Durchlauf wird der zweite hinzugeschaltet, sodass eine Erhöhung der Lese- und Schreibperformance zu erkennen sein sollte. Die Daten sollen verteilt auf die einzelnen Server geschrieben werden, um Skalierbarkeit zu erreichen.

6.3.1 FhGFS

Das Thema Skalierbarkeit ist ein großes Aushängeschild in der Werbung des Fraunhofer Dateisystems. Es verspricht **massive Skalierbarkeit** unabhängig von der Hardware. Der Punkt, dass die Metadaten- und Storageserver in jeder Umgebung eingerichtet werden können und dass diese auch gleichzeitig auf einem Clusterserver laufen können, machen das System sehr flexibel einsetzbar. Bei der Einrichtung mehrerer Metadatenserver ist es vom Fraunhofer Institut so eingerichtet worden, dass ein Server immer als Hauptserver angesprochen wird und dieser selbstständig die Daten verteilt.

Der Test mit Hilfe von „iozone“ wurde zunächst auf einem Knoten, der Management-, Metadaten- und Storageserver in einem war, durchgeführt. Der Client wurde anschließend auf einem weiteren Knoten installiert. Die beiden Knoten wurden mit Infiniband verbunden, sodass der Flaschenhals der Gigabit-Netzwerkkarten nicht zum Vorschein kam. Um eine Skalierbarkeit erkennen zu können, wurde ein weiterer Knoten als Storageserver hinzugefügt, sodass die Verteilung der Daten auf zwei Servern durchgeführt werden konnte.

One-Node-Installation Bei der Einrichtung des Dateisystems auf einem Knoten sind Lese- und Schreibgeschwindigkeiten zu erwarten, die sich nur gering vom normalen Dateisystem unterscheiden. In dem vorliegenden Testfall wurde das Programm „iozone“ im Lese- und Schreibmodus durchgeführt, jeweils von Dateigrößen von 512KByte bis 4GByte. Der Vorteil dieses Test liegt darin, dass die geschriebenen Daten durch

anschließendes Lesen überprüft werden.

Installiert wurde das System auf zwei Standard-SATA2-Festplatten im 2,5“-Format, die durch einen RAID-Controller in einem RAID-0-Verbund zusammenarbeiten. Das hat Geschwindigkeitsvorteile, da bereits hier eine direkte Verteilung der Daten auf beide Festplatten von statten geht. Die normale Lese- und Schreibgeschwindigkeit der Festplatten in diesem Verbund liegt bei durchschnittlich **180MByte/s**.

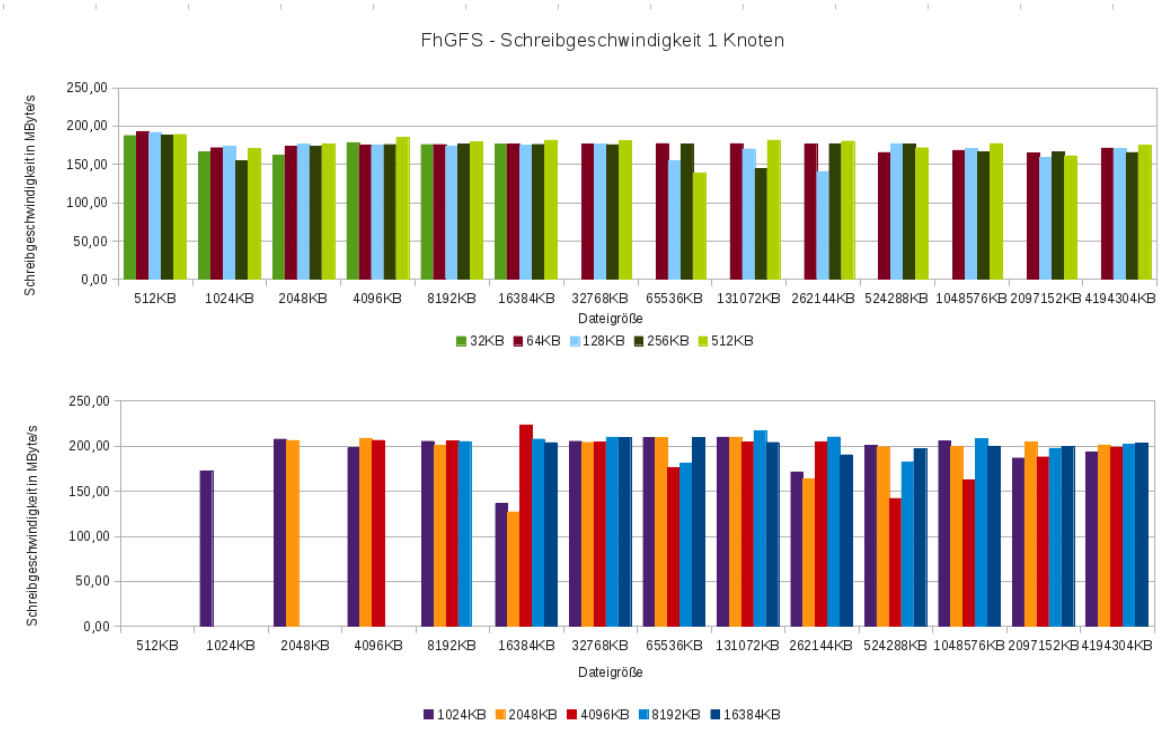

Abbildung 6.15: FhGFS - Schreibgeschwindigkeit mit einem Knoten - Diagramm

Die Schreibgeschwindigkeit wurde gemessen für unterschiedliche Dateigrößen mit unterschiedlicher Fragmentierung. Insbesondere Dateien mit großen Chunks wurden schneller geschrieben, als viele kleinere Dateien. Dies war auch zu erwarten, da die Anzahl der Metadateneinträge bei besonders kleinen Dateifragmenten zunimmt. Im Durchschnitt wurden Dateien mit besonders kleinen Chunkgrößen mit **160-170MByte/s** geschrieben, Dateien mit angepasster Chunkgröße konnten mit etwa **190-200MByte/s** geschrieben werden.

Bei der Lesegeschwindigkeit verhält es sich sehr ähnlich. Die im Test zuvor geschriebenen Daten werden sofort vom Programm wieder ausgelesen, um die Performance messen zu können. Als Ergebnis ist wiederum eine Geschwindigkeit zu

erwarten, die bei durchschnittlich **180MByte/s** liegen sollte.

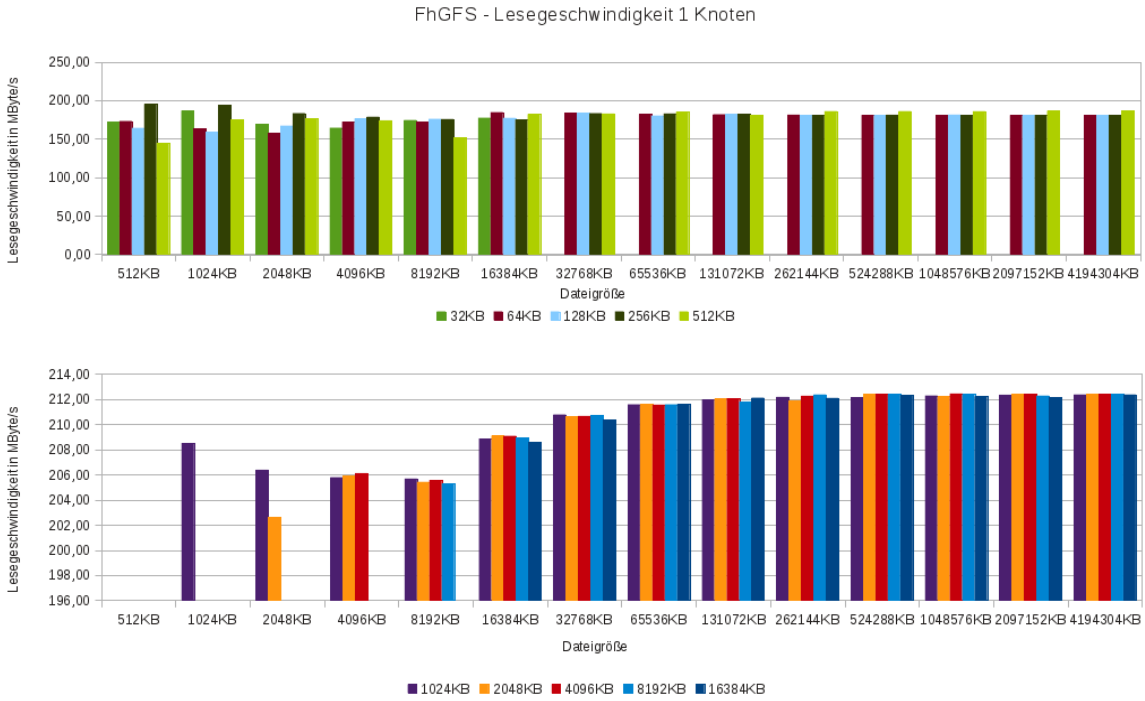

Abbildung 6.16: FhGFS - Lesegeschwindigkeit mit einem Knoten - Diagramm

Die Lesegeschwindigkeit weicht nur unwesentlich von der Schreibgeschwindigkeit ab, allerdings ist zu erkennen, dass die Performance bei größeren Chunks sich kaum noch verändert. Überschreiten diese die Größe von 1MByte/s ist kein Anstieg mehr zu erkennen. Dies lässt darauf schließen, dass die Metadaten bei solchen Größen stets zusammengefasst werden und die Geschwindigkeit somit nicht mehr veränderbar ist. Bei kleineren Chunkgrößen liegt die Lesegeschwindigkeit bei **160-170MByte/s**, ab einer Größe von 1Mbyte liegt diese konsequent im Durchschnitt bei **210MByte/s**. Diese Ergebnisse zeigen auf, dass die Lesegeschwindigkeit höher liegt als die Schreibgeschwindigkeit und die maximale Performance der vorhandenenen Hardware erreicht ist.

Two-Node-Installation Beim Hinzuschalten eines weiteren Storageservers ist eine direkte Verteilung der Daten gegeben. Für eine optimale Skalierbarkeit wäre ein linearer Anstieg der Geschwindigkeit perfekt und in der Realität gewünscht, allerdings kann diese kaum erreicht werden, da andere Faktoren, wie die Übertragung der einzelnen Daten auf andere Server und die Berechnung der Chunkgrößen eine Rolle spielen, die eine Linearität nicht ermöglichen.

Wenn eine solche lineare Skalierbarkeit erreicht werden würde, müssten die Schreibgeschwindigkeiten im Durchschnitt bei **350-400MByte/s** liegen. Die Erwartungshaltung ist dementsprechend, dass die Geschwindigkeit im Durchschnitt **300MByte/s** erreichen sollte, dann gilt das System als sehr gut skalierbar, dieser Wert entspricht **90%** der optimalen Performance.

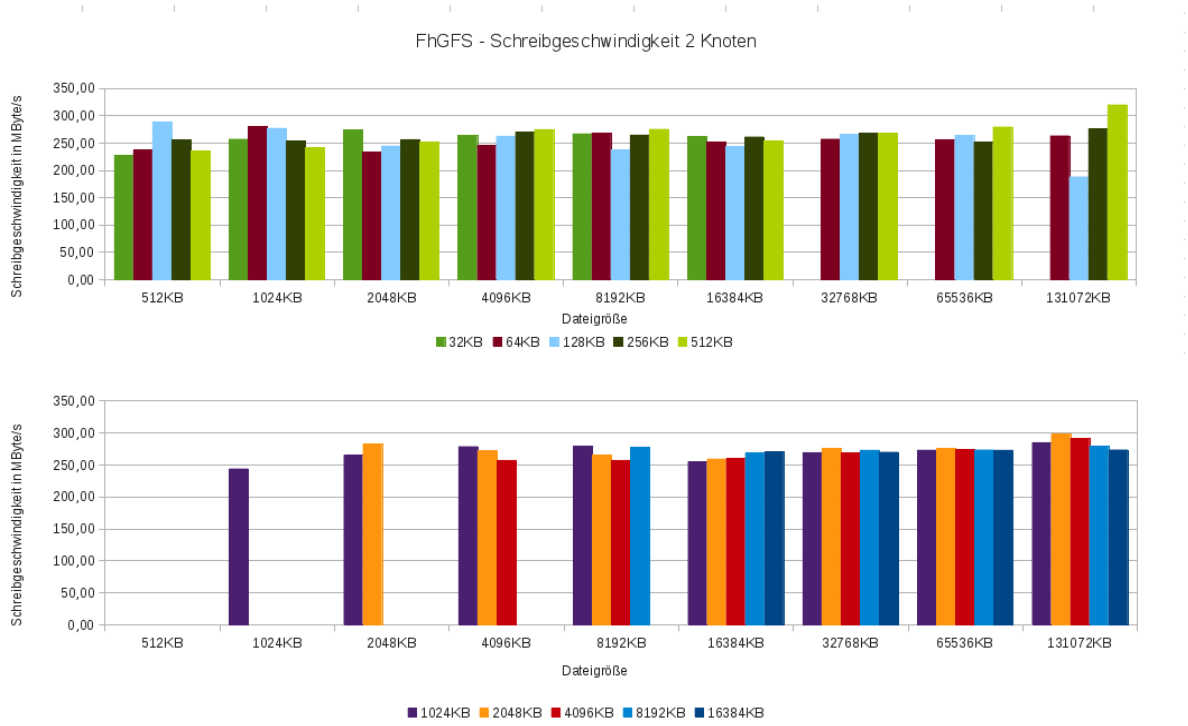

Abbildung 6.17: FhGFS - Schreibgeschwindigkeit mit zwei Knoten - Diagramm

Die Schreibgeschwindigkeit liegt im Durchschnitt bei **260MByte/s**. Dabei ist zu erkennen, dass die Geschwindigkeit zwar stark erhöht, allerdings bleibt diese unter den Erwartungen zurück.
Wenn man dazu betrachtet, dass der zweite Server lediglich mit einer einzigen 2,5“-Festplatte ausgerüstet wurde, die maximal eine Geschwindigkeit von **80MByte/s** erreichen kann, ist dieses Ergebnis trotzdem sehr beachtlich. Denn die Geschwindigkeiten dieser magnetischen Festplatten verhindern, dass sich die Geschwindigkeit verdoppeln kann.

Das zeigt sich auch in dem Punkt, dass die Chunkgrößen insbesondere bei größeren Dateien kaum noch Unterschiede aufzeigen, bis auf einige Ausreißer. Dass die Geschwindigkeit höher liegen kann, zeigt sich in der Dateigröße von 1GByte und der

Chunkgröße von 512KByte. Die gemessene Geschwindigkeit lag bei **320MByte/s**. Bei performanterer Hardware ist also eine bessere Skalierbarkeit zu erwarten.

Bei der Lesegeschwindigkeit verhält es sich etwas anders. Wie schon bei der One-Node-Installation liegt diese höher als die Schreibgeschwindigkeit. Dies ist auf die verbaute Hardware zurückzuführen, dass Leseoperationen schneller durchgeführt werden als Schreibzugriffe.

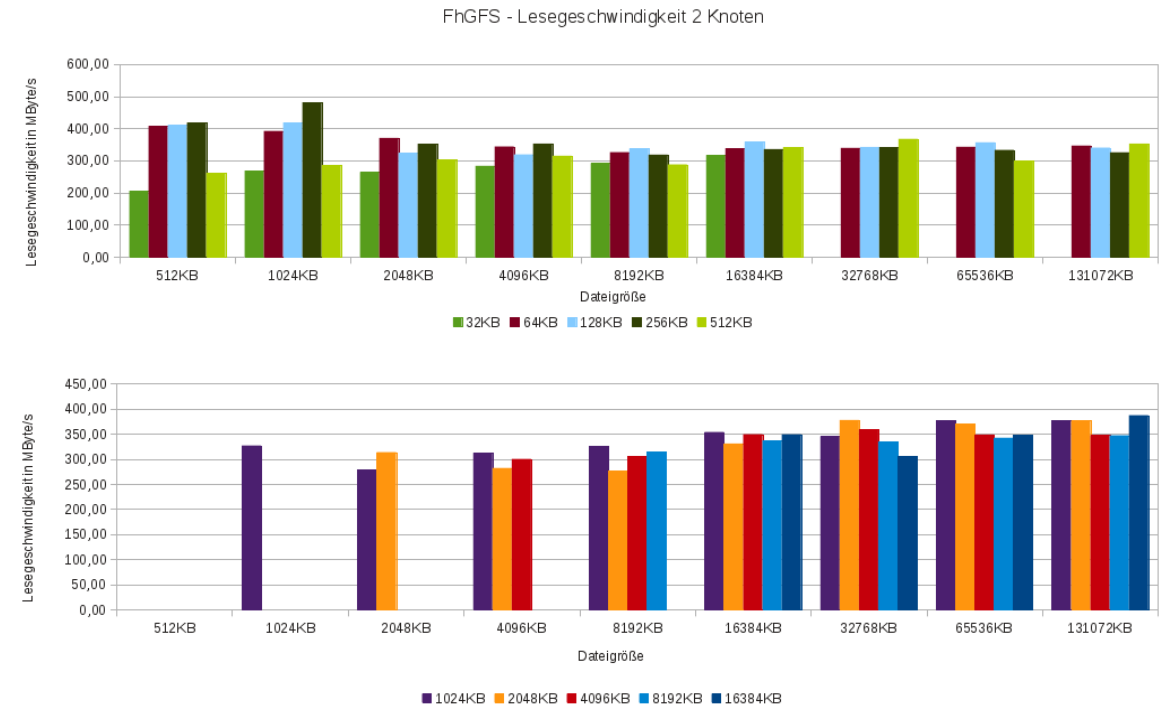

Abbildung 6.18: FhGFS - Lesegeschwindigkeit mit zwei Knoten - Diagramm

Die Lesegeschwindigkeit liegt im Durchschnitt bei **320MByte/s**. Auch hier exisiteren insbesondere bei kleinen Dateien mit kleiner Chunkgröße starke Ausreißer nach oben und unten. Bei einer Chunkgröße von 1024KByte schwankt die Messung zwischen 236MByte/s und 480MByte/s, dabei ist gerade der maximale Wert bei sehr kleinen Dateigrößen gemessen worden.

Die Geschwindigkeit liegt im Durchschnitt bei der maximalen Performance der vorhandenen Hardware, sodass zwar keine prozentuale Aussage darüber getroffen werden kann, ob das System gut skalierbar ist, allerdings ist ein großer Anstieg von Lese- und Schreibgeschwindigkeiten bei allen Chunkgrößen zu erkennen. Es ist davon auszugehen, dass die Geschwindigkeiten bei schnelleren Festplatten auch diese

entsprechend ausreizen kann. Das Fraunhofer Dateisystem zeigt also im vorliegenden Testfall eine sehr gute Skalierbarkeit. Das Ergebnis stellt einen maßgeblichen Anteil an der Entscheidungsfindung dar, welches Dateisystem auf dem neuen Compute-Cluster benutzt werden soll. Der Punkt der Skalierbarkeit ist mit der Performance darum so elementar wichtig, weil der neue Rechnerverbund auf mehreren hundert Knoten bestehen wird und daher die Möglichkeit gegeben sein muss, dass die Geschwindigkeit möglichst linear ansteigt, wenn neue Knoten hinzugeschaltet werden. In dem durchgeführten Testfall ist dies zumindest in diesem Ausmaß mit dem FhGFS absolut möglich.

6.3.2 MooseFS

Das Hauptaugenmerk von MooseFS wurde auf Sicherheit und Verfügbarkeit der gespeicherten Daten gelegt. Dies ist auch bei den Testergebnissen über die Skalierbarkeit zu erkennen. Die Performance des Dateisystems ist im Vergleich zum Fraunhofer Dateisystem als mieserabel anzusehen.

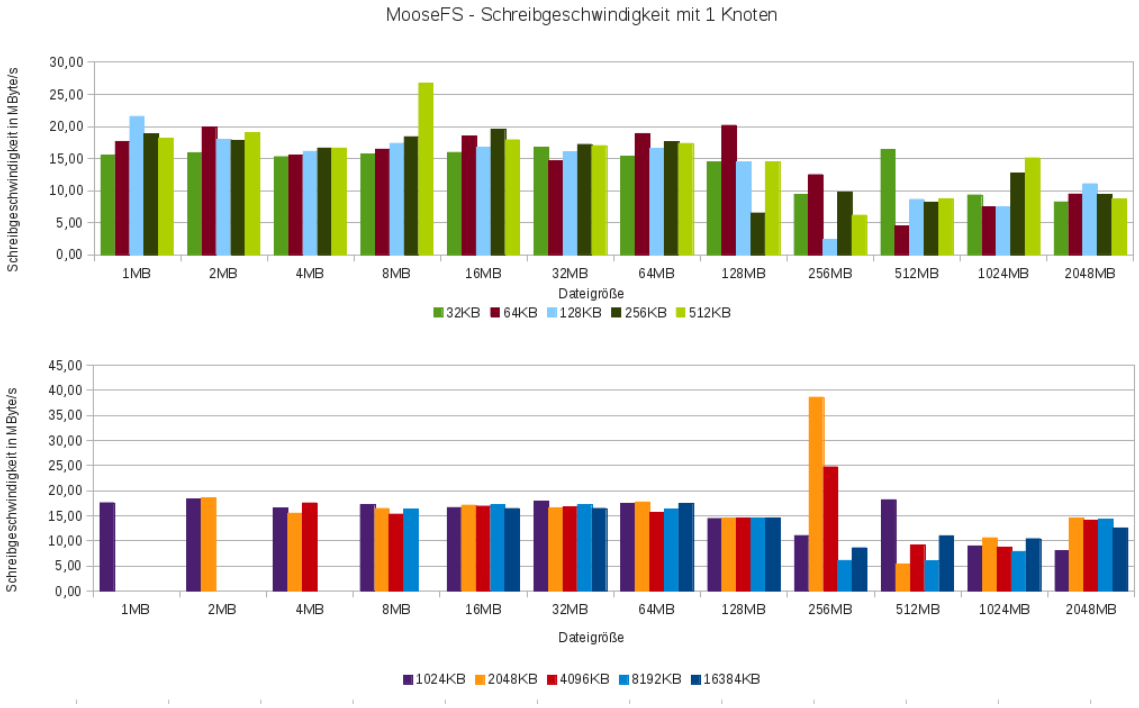

Abbildung 6.19: MooseFS - Schreibgeschwindigkeit mit einem Storageserver

Die Ergebnisse zeigen auf, dass die Geschwindigkeit sich generell auf einem sehr niedrigen Niveau befindet. Mit einem Storageserver ist zudem zu erkennen, dass es bei Dateigrößen zwischen 1 und 64MByte kaum Unterschiede macht, mit welchen Chunkgrößen diese Dateien übertragen wurden. Bei höheren Dateigrößen ist zudem meist ein Abfall der Geschwindigkeit zu erkennen, lediglich die Geschwindigkeit bei einer Dateigröße von 256MByte und einer Chunkgröße von 2048KByte ist ein Ausreißer zu erkennen, der anstelle von durchschnittlich 15-20Mbyte/s ca. **38Mbyte/s** erreichen konnte.

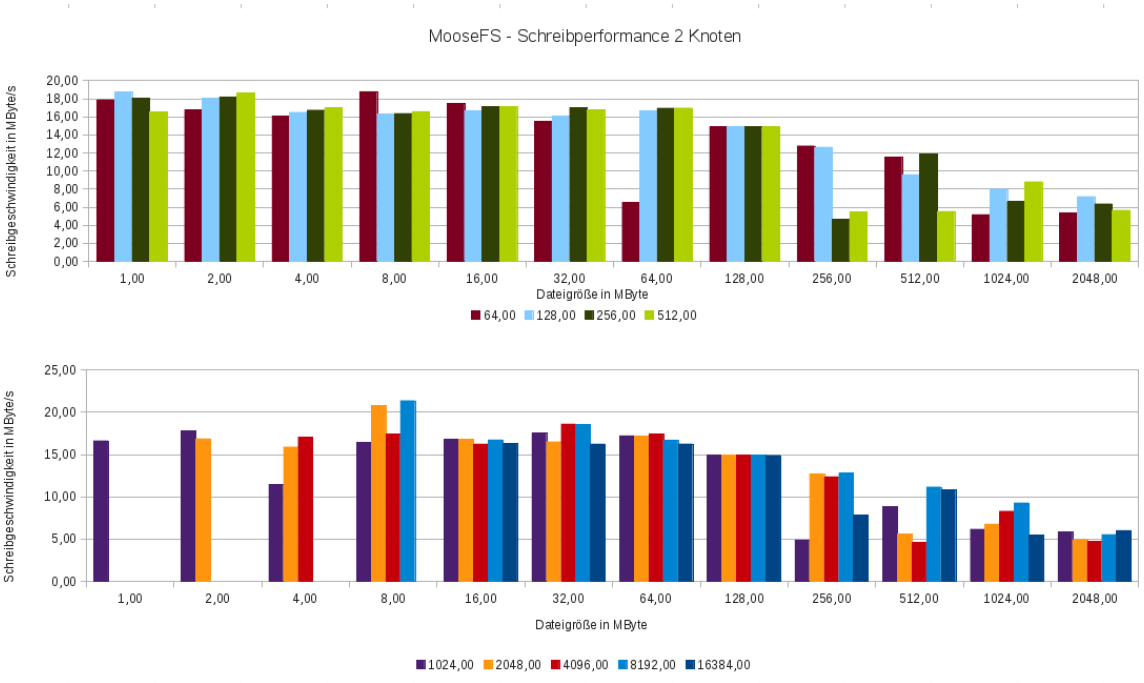

Abbildung 6.20: MooseFS - Schreibgeschwindigkeit mit zwei Storageservern

Um eine möglichst optimale Skalierbarkeit erkennen zu können, müssten die Schreibraten bei dem Hinzufügen eines weiteren Storageservers wenigstens mit dem Faktor 2 multipliziert werden können, das bedeutet eine Geschwindigkeit von durchschnittlich **35-40Mbyte/s**.
Im Ergebnis ist zu erkennen, dass die Geschwindigkeit sich insbesondere bei größeren Dateien sich sogar verschlechtert hat. Die Übertragungsgeschwindigkeit von Dateien zwischen 1Mbyte und 128MByte liegt durchschnittlich bei **16Mbyte/s**, welches vergleichbar ist mit der Performance von MooseFS mit einem Storageserver. Aufgrund der sehr schlechten Ergebnisse wurde auf eine Messung der Skalierbarkeit im Lesemodus verzichtet, weil die Geschwindigkeit völlig inakzeptabel ist.

MooseFS zeigt auch bei dem sehr wichtigen Punkt Skalierbarkeit eine **schlechte Performance**. Die Geschwindigkeit konnte durch das Hinzuschalten eines weiteren Servers nicht gesteigert werden, bei größeren Dateien ist diese sogar noch geringer. Es kann hiermit keine Empfehlung für den Einsatz auf dem neuen Compute-Cluster des Universitätsrechenzentrums gegeben werden.

6.3.3 GlusterFS

GlusterFS wirbt mit sehr guter Skalierbarkeit. Durch den grundlegenden Aufbau, dass jede Datei vollständig auf einem Server vorhanden ist, wird deutlich, dass bei bestimmten Konstellationen eine Skalierung nicht stattfinden kann.

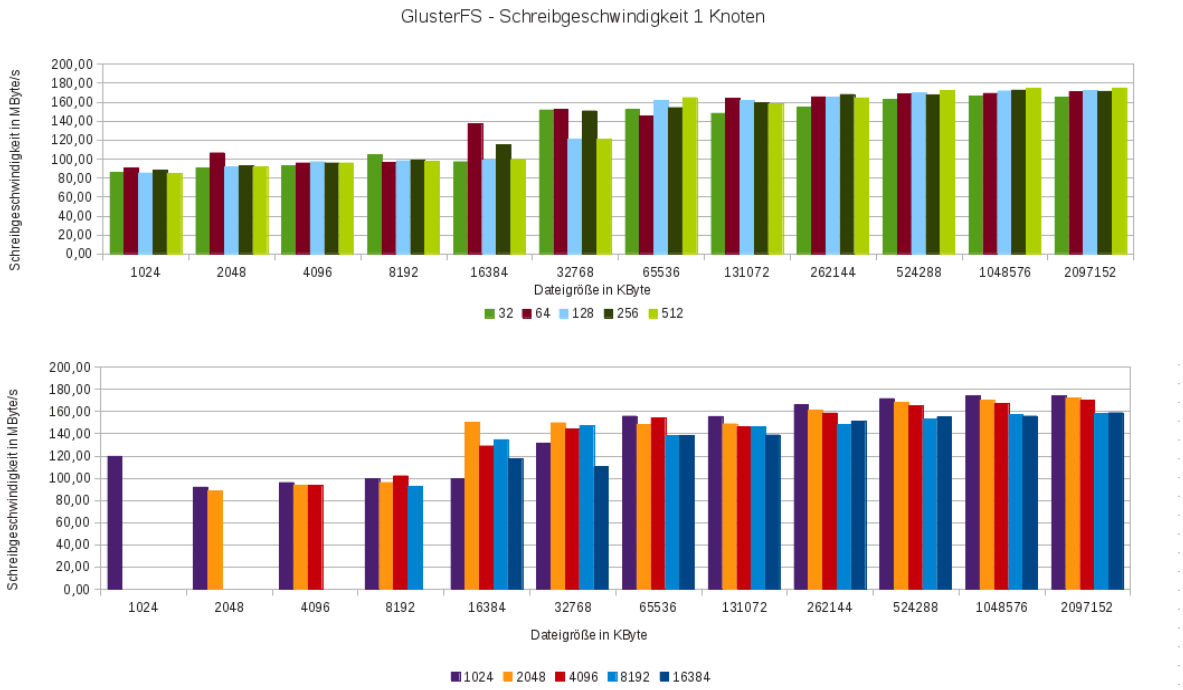

Abbildung 6.21: GlusterFS - Schreibgeschwindigkeit mit einem Storageserver

Die Schreibgeschwindigkeit mit einem Knoten liegt bei durchschnittlich **135MByte/s**. Dabei erhöht sich die Geschwindigkeit mit Zunahme der Dateigröße. Bei der Auswahl der Chunkgröße ist die beste Performance bei 256 bis 1024KByte gegeben.

Der Testdurchlauf mit Hilfe von „iozone“ führt Schreibtests mit durch, indem eine Datei mit einer bestimmten Größe und Fragmentierung geschrieben wird. Diese wird auf jeweils einen vorhandenen Storageserver geschrieben. Eine Skalierung kann daher nicht stattfinden.

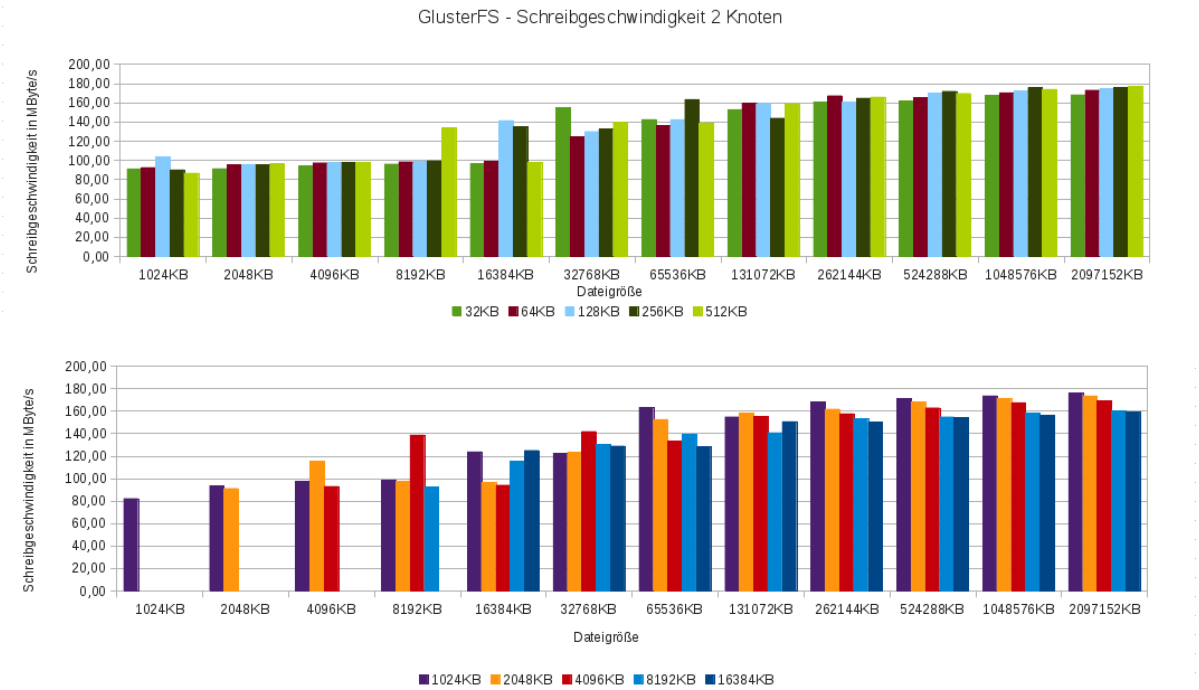

Abbildung 6.22: GlusterFS - Schreibgeschwindigkeit mit zwei Storageservern

Die grundlegende Geschwindigkeit beim Schreiben hat sich nicht verändert. Es ist zu erkennen, dass einige Ausreißer nach unten existieren, es ist dadurch zu erklären, dass der zweite Storageserver eine geringere Performance besitzt als der erste. Dies liegt daran, dass dieser mit nur einer Festplatte ausgestattet ist.

Wie in Abbildung 6.23 zu erkennen ist, verändert sich erwartungsgemäß die Leseperformance von GlusterFS im vorliegenden Testfall auch nicht durch das Hinzufügen eines weiteren Storageservers.

Die Leseperformance von GlusterFS mit einem Storageserver liegt auf dem Niveau der vorhandenen Hardware. Ein weiterer Anstieg der Geschwindigkeit ist nicht zu ermöglichen. Beim Zuschalten eines weiteren Storageservers ist kein Anstieg der Geschwindigkeit zu erwarten, eher ein Einbruch bei einzelnen Werte aufgrund der Verteilung auf unterschiedliche Hardware sollte zu erkennen sein.

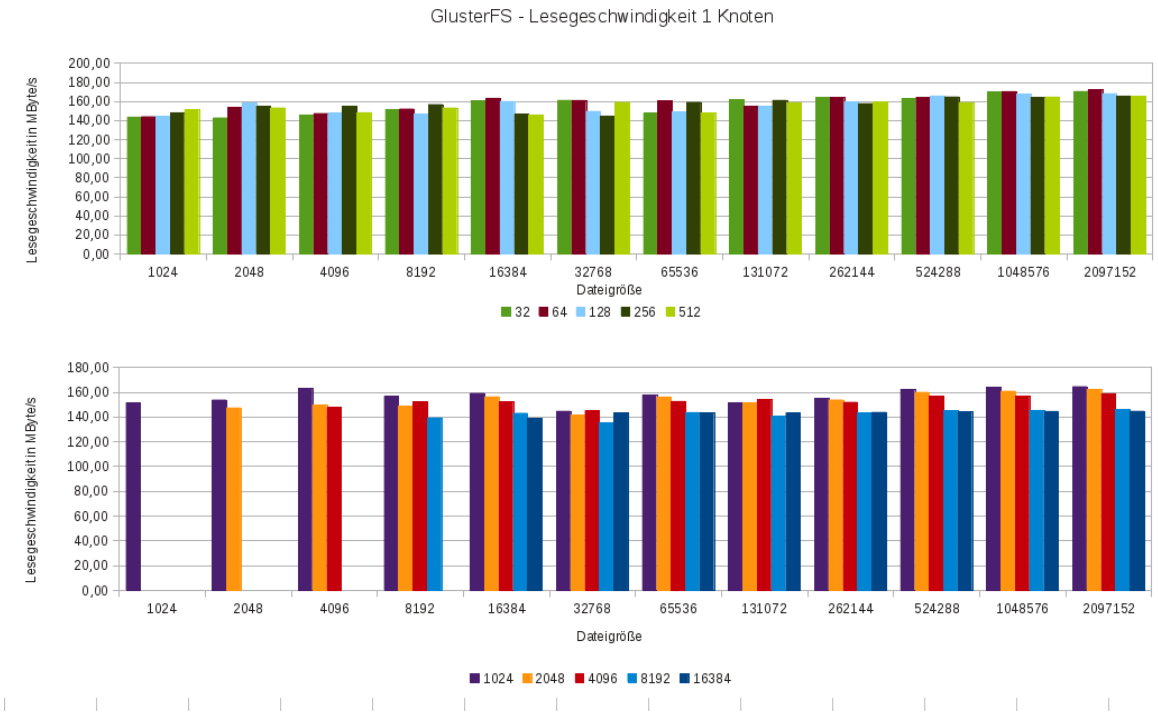

Abbildung 6.23: GlusterFS - Lesegeschwindigkeit mit einem Storageserver

In Abbildung 6.24 ist die Leseperformance von GlusterFS mit zwei Storageservern aufgezeichnet. Die erwarteten Einbrüche bei einzelnen Testdurchläufen sind eindeutig zu erkennen.
Der Grund dafür liegt in der Verteilung der Daten auf unterschiedliche Server. Während der erste Server zwei Festplatten im RAID-0-Verbund aufweisen kann und damit eine direkte Verteilung der Daten, beinhaltet der zweite Server nur eine Festplatte. Die Geschwindigkeitseinbußen sind daher vom zweiten Server zu erklären, der eine maximale Lesegeschwindigkeit von ca. **100MByte/s** zur Verfügung stellt.

Im vorliegenden Testfall konnte eine Verteilung der Daten zwar nachgewiesen werden, allerdings ist keine Skalierung erkennbar. Grund dafür ist der Aufbau von GlusterFS. Während andere verteilte Dateisysteme die einzelnen Dateien in Objekte zerlegen und diese unterschiedlich auf alle verfügbaren Storageserver aufteilen, speichert GlusterFS die gesamte Datei auf einem Server ab. Die Verteilung erfolgt erst, wenn mehrere Dateien zum Einsatz kommen. Das System ist für diesen Anwendungsfall nicht skalierbar, allerdings beim praktischen Einsatz, wo immer meherere Dateien erzeugt werden, erfolgt eine Verteilung und somit eine Erhöhung der Arbeitsgeschwindigkeit.

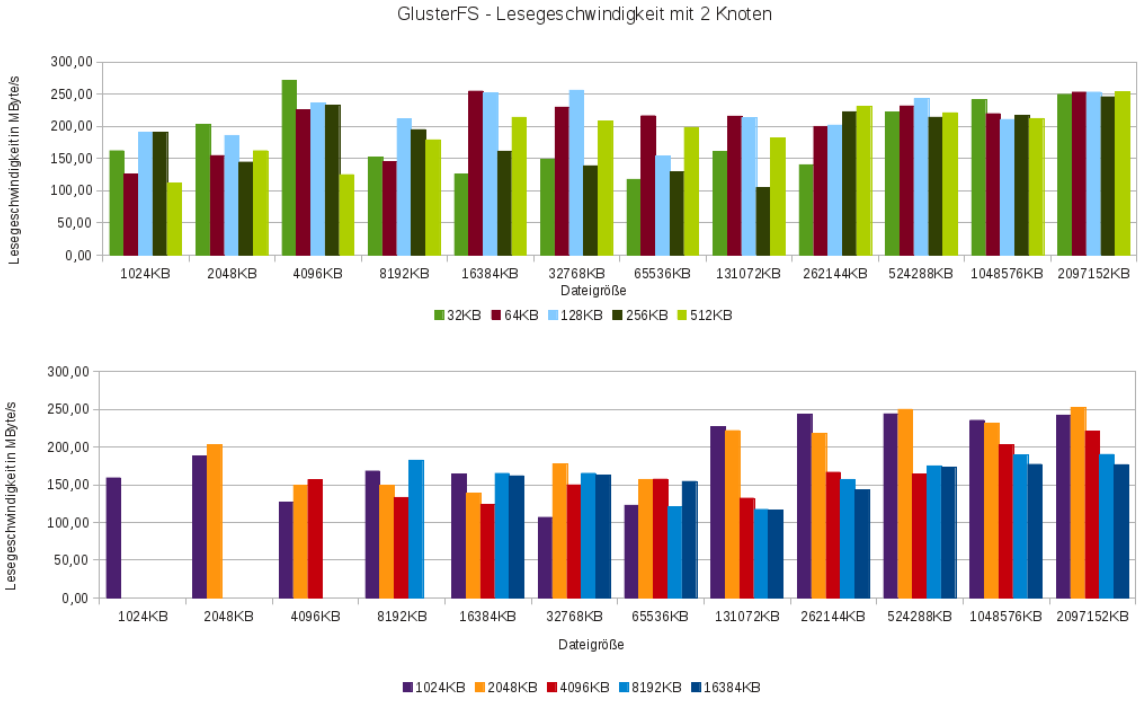

Abbildung 6.24: GlusterFS - Lesegeschwindigkeit mit zwei Storageservern

6.3.4 Zusammenfassung Skalierbarkeit

Skalierung bedeutet, dass beim Hinzufügen mehrerer Storageserver die Lese- und Schreibgeschwindigkeit möglichst linear zunehmen sollte. Im durchgeführten Test konnte dies nur das **Fraunhofer Dateisystem** aufzeigen. Die maximale Performance von beiden vorhandenen Storageservern konnte erreicht werden, sodass eine Linearität zu erkennen war.

MooseFS zeigte bereits im Schreibmodus sehr schlechte Ergebnisse, die sogar zu einer Verschlechterung bei großen Dateien beim verteilten Schreiben geführt haben. Der Test wurde daher abgebrochen und der Lesemodus nicht weiter untersucht.

GlusterFS konnte mit dem durchgeführten Testfall auch keine Skalierung aufweisen. Das ist damit zu begründen, dass die Dateien generell auf einem Server geschrieben werden. Die Verteilung der Daten konnte besonders im Lesemodus erkannt werden, sodass angenommen werden kann, dass in der praktischen Anwendung, wo generell mehrere Dateien verwendet werden, eine Skalierung erfolgen kann.

6.4 Verwaltbarkeit

Die Verwaltung von parallelen verteilten Dateisystemen ist ein großer Bestandteil des Testfeldes. Das Universitätsrechenzentrum benötigt ein System, welches möglichst leicht zu konfigurieren ist, sodass Erweiterungen im Nachhinein gut einzupflegen sind. Auch die administrativen Möglichkeiten müssen leicht verständlich sein, damit im Nachhinein die Verwaltung möglichst ohne großen Aufwand zu bewältigen ist.

Viele Dateisysteme bieten dafür eine grafische Administrationsoberfläche an, die unterschiedliche Aufgaben bewältigen und anzeigen kann. Dafür ist der Vergleich hinsichtlich der Verwaltbarkeit nötig, um einen konkreten Überblick zu bekommen.

6.4.1 FhGFS

Die Verwaltung des Fraunhofer Filesystems ist ein sehr gutes Beispiel, auf möglichst einfache Art ein sehr komplexes Programm administrieren zu können. Zum einen besteht die Möglichkeit, die Installationen und Überwachungsmethoden direkt auf der Konsole auszuführen. Die Pakete werden über ein spezielles Repository zur Verfügung gestellt oder direkt als Download angeboten. Für eine grafische Administration wird mit der Installation des „Admon“-Paketes eine Schnittstelle zwischen vorhandener JAVA-Anwendung und Management-Server hergestellt. Dieser ist sehr Ressourcenfreundlich, sodass eine Installation direkt auf dem Management-Server möglich ist und vom Fraunhofer Institut auch explizit angeboten wird.

Hierbei ist allerdings zu beachten, dass die JAVA-Anwendung zwar betriebssystemunabhängig programmiert wurde, aber das „Admon“-Paket, wie alle verfügbaren Daten lediglich für Suse-, RedHat- und Debian-Distributionen und deren Derivate direkt zur Verfügung steht. Die Pakete sind nur in dem Punkt unterschiedlich, dass diese jeweils die betriebssysteminternen Paketmanager aufrufen. Eine Installation von FhGFS war auf einer Gentoo-Plattform nicht möglich, da bestimmte Pakete, die von einzelnen Servern vorausgesetzt werden, nicht verfügbar waren.

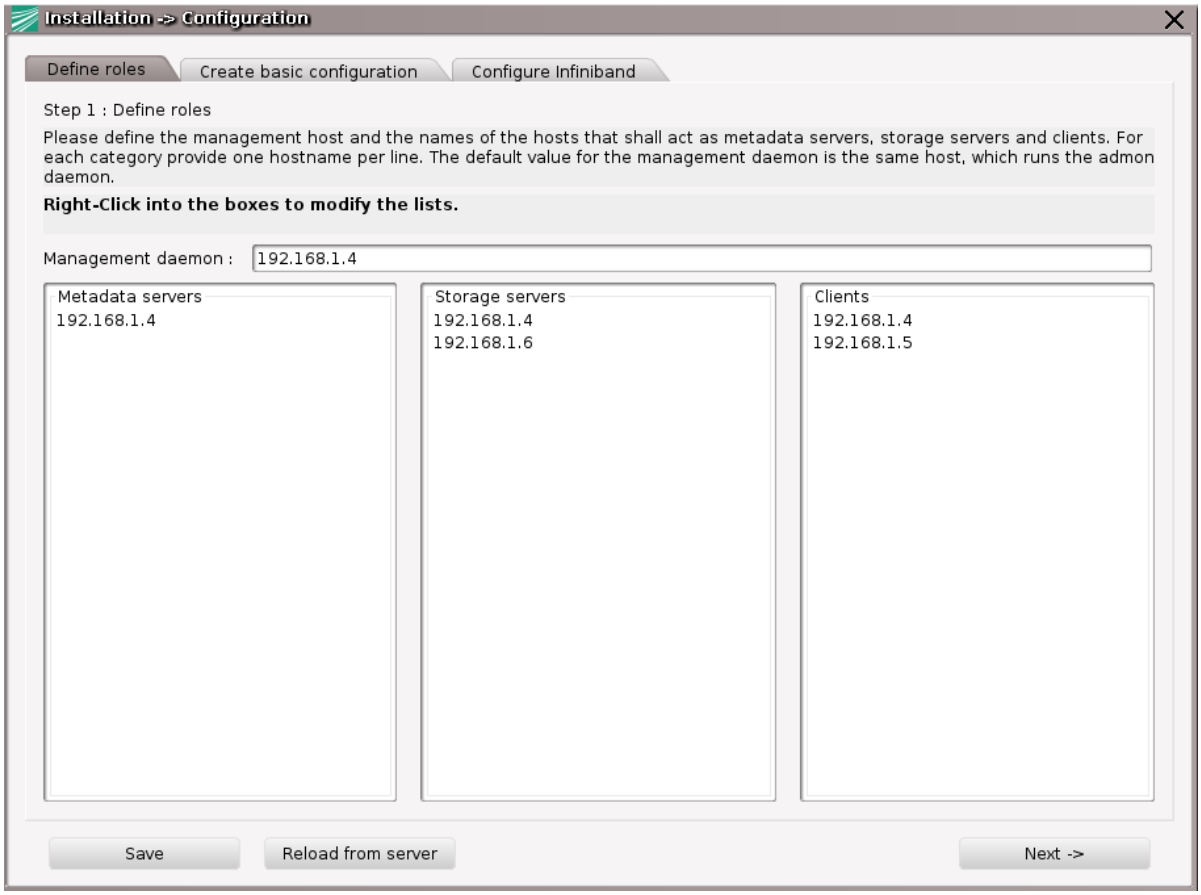

Abbildung 6.25: Konfiguration FhGFS im Testfall

Mit Hilfe der Administrationsoberfläche hat ein Nutzer die Möglichkeit, einen kompletten Überblick über die einzelnen Knoten zu bekommen, unterteilt in Metadaten- und Storageservern. Dabei wird eine direkte Anzeige der einzelnen Server gegeben, sowie einen gemeinsamen Überblick über alle verfügbaren.

Außerdem besteht die Möglichkeit der automatisierten Installation neuer Knoten mit speziellen Anforderungen. Beim Hinzufügen neuer Server müssen diese lediglich über die Oberfläche bekannt gemacht werden und mit dem Befehl „Install“ werden alle benötigten Pakete automatisch mit dem betriebssysteminternen Paketmanager heruntergeladen und installiert, sodass dieser nach kurzer Zeit verfügbar ist, ohne weiteres Zutun. Auch die Installation der Clients verläuft in diesem Fall problemlos und stellt keinen großen Aufwand für den Administrator dar. Durch diese Oberfläche wird auch angezeigt, ob bestimmte Knoten eventuell nicht zu erreichen sind. Zudem ist die automatische Benachrichtigung bei Fehlern per E-Mail möglich, sodass automatisch Nachrichten verschickt werden, wenn es zu Problemen mit der Hardware kommt.

Es besteht in der Gesamtansicht auch ein gravierendes Sicherheitsrisiko. Damit alle Server miteinander kommunizieren können, benötigen diese einen „passwordless ssh“ Zugang als *root*. Außerdem ist für die automatisierte Installation eine Internetverbindung Voraussetzung, zumindest für den Server, der den „Admon“-Dienst startet.
Mit dieser Methode wird dem Programm root-Zugriff auf das gesamte Betriebssystem gegeben. Dieser Punkt muss bei der Entscheidungsfindung mit einfließen.

Wenn bestimmte Voraussetzungen, wie der gesicherte Zugang zum Server, gewährleistet wird, dass dies keine großen Sicherheitsrisiken darstellt, zeigt das FhGFS auf, wie schnell, einfach und performant ein mächtiges Dateisystem zu administrieren ist.

6.4.2 MooseFS

Mit Hilfe der grafischen Oberfläche, welche sich mit jedem Browser anzeigen lässt, bekommt ein Administrator eine sehr gute Übersicht über die aktuelle Auslastung des Dateisystems. Dabei wird die aktuellen Anzahl der gespeicherten Daten angezeigt und wie viele sich davon im Dateisystempapierkorb befinden. Zudem ist eine Auslastung der Hardware im Hinblick auf CPU- und RAM-Auslastung zu erkennen, sowie die einzelnen vorhandenen Festplatten einzelner Knoten. Diese werden regelmäßig auf deren Verfügbarkeit überprüft. Außerdem ist eine sehr gute Übersicht über die Auslastung einzelner Festplatten vorhanden.

Die Anzeigen dienen für einen Administrator lediglich der Information. Man hat keine Einstellungsmöglichkeiten, diese müssen per Kommandozeilenbefehl auf den einzelnen Knoten ausgeführt werden. Die vorhandene Dokumentation über die Installation einzelner Serverkomponenten ist zwar gut, allerdings war es nicht möglich in irgendeiner Datei Werte, die zur Optimierung des Dateisystems zuständig sein könnten, zu verändern. Daher war keine Möglichkeit gegeben, die Performance zu verbessern.

MooseFS besitzt eine sehr gute grafische Oberfläche, die Informationen über den aktuellen Zustand der Hardware bereitstellt, allerdings fehlen dort die nötigen Einstellmöglichkeiten für bestimmte Befehle.

6.4.3 GlusterFS

Im Gegensatz zu den vorher beschriebenen Dateisystemen bietet GlusterFS **keine grafische Oberfläche** zur Verwaltung des Systems an. Dies geschieht völlig auf Kommandozeilenebene.
Dabei hat der Administrator die Möglichkeit, von jedem einzelnen Knoten die Erreichbarkeit des anderen zu überprüfen. Die Verwaltung einzelner Knoten muss dabei manuell durchgeführt werden. Dazu gehört die Installation der benötigten Pakete, die Einrichtung und Anpassung der Konfigurationsdateien, sowie die Bekanntmachung des Servers, um diesen in den Pool der Storageserver aufzunehmen. Alle Storageserver müssen zusätzlich einen Eintrag für einen neuen Client erhalten, damit die Zugriffsrechte gewährleistet werden können.

Die Installation und Verwaltung von GlusterFS ist lediglich für erfahrene Administratoren zu empfehlen. Es existiert keine Möglichkeit, bestimmte Punkte zentral zu bearbeiten. Jeder Knoten muss dabei einzeln konfiguriert werden. Das macht die Ersteinrichtung sehr kompliziert und oftmals unübersichtlich.

6.4.4 Zusammenfassung Verwaltbarkeit

Die Verwaltbarkeit ist für das Universitätsrechenzentrum ein sehr wichtiger Punkt bei einem verteilten Dateisystem. Da der neue Compute-Cluster aus mehreren hundert Knoten bestehen wird, muss diese möglichst einfach und übersichtlich gehalten werden. Zudem wäre es von Vorteil, wenn eine zentrale Möglichkeit besteht, die vorhandenen Server zu kontrollieren.

Von den vorgegebenen Dateisystemen werden die Eigenschaften nur von einem System erfüllt: **FhGFS**. Dieses bietet eine grafische Oberfläche zur Installation und Verwaltung an und zeigt zudem zusätzliche Informationen über den aktuellen Status aller verfügbaren Knoten. Außerdem ist es dabei möglich, einzelne Knoten aus dem Verbund zu entfernen und vom Master-Server aus neue Knoten zu installieren und hinzuzufügen.

Die Verwaltung von **MooseFS** ist nicht so stark. Es wird zwar auch eine grafische Oberfläche angeboten, allerdings verfügt diese nur über inhaltlichen Charakter. Die Auswertungen für einzelne Server sind sehr weitläufig, auch die Verteilung und Auslastung der Knoten ist sehr gut zu erkennen, allerdings existiert keine Möglichkeit der automatischen (De-)Installation von Knoten. Diese müssen manuell einzeln

hinzugefügt bzw. entfernt werden.

Das Gegenteil dieser Ansätze stellt **GlusterFS** dar. Dieses bietet keine grafische Oberfläche an und kann nur über Kommandozeilenbefehle verwaltet werden. Dabei ist dieser Umstand sehr kompliziert und muss für jeden einzelnen Knoten durchgeführt werden. Es existiert auch keine zentrale Einheit, die Informationen über einzelne Knoten und deren Auslastung angibt.

Die beste Verwaltung bietet das Fraunhofer Dateisystem FhGFS an, gefolgt von MooseFS und GlusterFS.

6.5 Zugriffskontrolle

Das Thema Zugriffskontrolle ist in Einsatzgebieten nötig, wo verschiedene Nutzer auf die gleichen Daten zugreifen müssen. Diese müssen entsprechend der festgelegten Richtlinien zur Verfügung gestellt werden.
Die verschiedenen Zugriffsmöglichkeiten - lesen, schreiben, ausführen - müssen dabei für jede Datei und jeden Ordner einzeln angewendet werden können. Auch muss es möglich sein, gesamte Benutzerordner vor anderen zu verstecken.

Dabei existieren grundsätzliche Ansätze, die die Rechtevergabe klassifizieren. Zum einen wird in UNIX-Umgebungen sogenannte „Discretionary Access Controls“ [31] benutzt, die die Zugriffskontrolle den einzelnen Benutzern überlassen. Wenn eine zentrale Rechteverwaltung eingesetzt wird, spricht man von „Mandatory Access Controls“. Als dritte Möglichkeit exisitert die rollenbasierte Rechtevergabe, „Role Based Access Control“, die die Rechteverwaltung mit Hilfe von verschiedenen Gruppen übernimmt.

Da diese Vergabe immer auf Dateiebene durchgeführt werden muss, werden solche Punkte, die meist in Verbindung auftreten, vom Dateisystem übernommen.

6.5.1 FhGFS

Beim verteilten Dateisystem des Fraunhofer Institutes exisitieren verschiedene Ansätze der Rechteverwaltung. Zum einen kann jeder Client ein bestimmtes Verzeichnis mounten, welches exklusiv für ihn zur Verfügung gestellt werden kann. Dieses kann nur von anderen Personen mit benutzt werden, die auch das entsprechende Verzeichnis mounten.

Außerdem besteht die Möglichkeit, dass die Zugriffsrechte mit „Discretionary Access Controls“ verändert werden können, das bedeutet, dass jeder Nutzer selbstständig für die Sicherheit der eigenen Daten zuständig ist. Dies ist insbesondere in Umgebungen wichtig, wo generell das gesamte Dateisystem zur Verfügung steht und jeder Benutzer die Möglichkeit besitzt, in Verzeichnisse lesend bzw. schreibend zugreifen zu können.

Die Zugriffskontrolle ist also vorhanden, allerdings hat ein Administrator keine

Möglichkeit, zusätzlich die Rechte zu verwalten.

Es ist eindeutig zu erkennen, dass die Rechteverwaltung nicht das wichtigste Merkmal des Dateisystems darstellt. Außerdem kann jeder Computer, der die Adressen des Management-Servers kennt, sowie „passwordless ssh“-Zugriff besitzt, den FhGFS-Client installieren und sofort auf das alle Dateien zugreifen.

Es existiert keine Möglichkeit, den Zugriff auf bestimmte Nutzer oder Computer zu beschränken. Daher muss bei der Einrichtung des Servers darauf geachtet werden, dass mögliche Angriffe direkt auf Betriebssystem- bzw. Netzwerkebene abgefangen werden können.

Da diese Rechteverteilung für die Nutzung des neuen Compute-Clusters für das Universitätsrechenzentrum ausreicht, kann in dem Punkt Zugriffskontrolle das Fraunhofer Dateisystem empfohlen werden.

6.5.2 MooseFS

Die Rechteverwaltung von MooseFS ist sehr komplex aufgebaut. Zum einen besteht die Möglichkeit, den Zugriff auf spezielle Nutzer zu beschränken oder auf bestimmte IP-Adressen oder -Bereiche. Außerdem kann der Zugriff eingeschränkt werden, es stehen die Standardrechte Schreiben, Lesen und Ausführen zur Verfügung. Zusätzlich kann für jeden Benutzer ein spezieller Ordner zur Verfügung gestellt werden, auf den nur dieser Benutzer zugreifen kann. Dies geschieht in der „mount“-Datei des Clients.

Genau wie das Fraunhofer Dateisystem benötigt MooseFS Zugriffsmöglichkeiten per "passwordless sshßu allen Knoten des Clusters, die Speicherplatz oder Rechenzeit für das Dateisystem bereitstellen sollen. Im direkten Vergleich ist das Thema Zugriffskontrolle allerdings besser gelöst, weil eine spezielle Benutzerauthentifizierung eingeführt wurde, sodass ein Zugriff auf fremde Daten erschwert wird.

6.5.3 GlusterFS

Während beim Fraunhofer Dateisystem lediglich die „Discretionary Access Controls“ existieren, womit jeder Benutzer die Rechtevergabe selbstständig bestimmen kann und bei MooseFS zusätzlich die Zugriffskontrolle auf Basis von IP-Adressen eingeführt wird, bietet auch GlusterFS diese Möglichkeiten an. Außerdem kann jeder „mount“ mit Hilfe von Benutzernamen und einem festgelegten Passwort gesichert werden. Dies führt

dazu, dass die Zugriffsmöglichkeiten für Fremde noch weiter erschwert werden.

6.5.4 Zusammenfassung Zugriffskontrolle

Die Zugriffskontrolle spielt für das Universitätsrechenzentrum zwar eine untergeordnete Rolle, darf allerdings nicht außer Acht gelassen werden.
Durch verschiedene Verfahren wird der neue Compute-Cluster bereits von der Außenwelt abgeschnitten und nur bestimmte Personen erhalten Zugriffsmöglichkeiten zur Nutzung. Daher ist eine sehr starke Rechteverwaltung nicht nötig, lediglich die einzelnen Benutzerordner sollten vor anderen geschützt werden können.

Das **Fraunhofer Dateisystem** bietet die Möglichkeit an, dass jeder Benutzer seinen speziellen Ordner direkt mounten kann. Dies hat den Vorteil, dass dieser andere Benutzerordner in der gleichen Struktur nicht sehen kann und somit nicht die Möglichkeit hat, darauf zugreifen zu können. Außerdem hat jeder die Möglichkeit, durch „Discretionary Access Controls“ die Rechte seiner eigenen Daten festzulegen. Das macht das System sehr einfach in der Bedienung, ein Administrator hat allerdings keine Möglichkeiten mehr, diese Dateien anzuzeigen bzw. zu verändern.

MooseFS beinhaltet mehr Möglichkeiten, die Rechte einzelner Clients einzuschränken. Auch hier hat der Benutzer die Möglichkeit, die Zugriffsrechte selbst bestimmen zu können. Ein Administrator kann zudem die Einschränkungen auch aufgrund von IP-Adressen vergeben.

Diese Punkte sind auch in **GlusterFS** zu finden. Darüber hinaus bietet das System noch zusätzlich die Rechtevergabe über einen speziellen Benutzernamen und Passwort an.

Insgesamt bietet **GlusterFS** die besten Möglichkeiten an, ein System vor fremden Zugriffen zu schützen, das Fraunhofer FhGFS allerdings am wenigsten, die aber für das Universitätsrechenzentrum bereits ausreichend sind.

6.6 Replikation

„Als Replikation wird der Prozess zur mehrfachen Datenspeicherung und Datenpflege in (verteilten) Datenbank- und Informationssystemen bezeichnet.“ [32]

Sowohl bei Datenbanken als auch bei der Abspeicherung von Metadaten, die im Hintergrund auch in Tabellen abgespeichert werden, bedeutet Replikation, dass die Möglichkeit bestehen muss, diese Daten automatisiert zu sichern und entweder im Notfall oder für die Lastverteilung zur Verfügung zu stellen.

Für Dateisysteme bedeutet dies, dass Replikationen für die Metadaten- und die Storageserver zur Verfügung gestellt werden sollten, um möglichst Hochverfügbarkeit erreichen zu können. Dies schließt mit ein, dass im Falle eines Systemausfalles die Daten, die dort gespeichert sind, weiter zur Verfügung stehen und somit kein Systemausfall zur Folge hat.

6.6.1 FhGFS

Das Fraunhofer Dateisystem bietet **keine automatischen Replikationsmöglichkeiten** für die Absicherung einzelner Server an. Dieses Feature wurde angekündigt, dass dieses in späterer Entwicklung mit eingepflegt werden soll. Zum aktuellen Zeitpunkt (Versionsnummer: 2012.10.r8) hat man lediglich die Möglichkeit, **Sicherungen** von einzelnen Servern durch Ausführung eines Befehls auf der Konsole durchzuführen. Allerdings bietet dies lediglich eine Sicherung und keine Replikationsmöglichkeit. Die Daten können auf einem anderen Server wiederhergestellt werden, es muss die Sicherung jedoch möglichst aktuell gehalten werden und ein Administrator muss diese manuell einpflegen.

Wenn FhGFS dieses Feature einpflegen würde und dabei die Performance, Stabilität und Sicherheit weiterhin gewährleistet, dann wäre das System vom Funktionsumfang noch größer und interessanter für noch mehr Unternehmen, so **fehlt ein sehr wichtiger Punkt**, der in der Wirtschaft häufiger gebraucht wird, als die reine Performance.

6.6.2 MooseFS

Das Thema Replikation ist ein großes Aushängeschild von MooseFS. Während der Nutzung des Dateisystems kann für **jede Datei und jedem Ordner** einzeln eingestellt werden, wie viele Replikationen davon angelegt werden sollen. Der Benutzer hat somit selbst die Möglichkeit, die Wichtigkeit der Daten einzustellen.

Es exisiteren auch globale Einstellungen. Der Wert wird bei MooseFS „goal“ genannt und kann zwischen einem und der maximal zur Verfügung stehenden Anzahl von Servern gewählt werden. Allerdings leidet die Performance unter der Replikation. Während des Schreibprozesses werden alle Daten auf einem Server abgelegt, dieser repliziert diese auf weitere Server. Erst wenn alle Daten übertragen wurden, bekommt der Client die Nachricht, dass die Datei ordnungsgemäß übertragen wurde. Insbesondere bei größeren Dateien kann es zu extremen Leistungseinbußen kommen. Durch die vorliegende Testkonfiguration und den bereits enttäuschenden Performanceergebnissen ohne Replikationen, wurde dieser Punkt nicht weiter behandelt.

Neben der automatischen Replikation bietet MooseFS weitere Sicherungsmöglichkeiten an. Zum einen ist es möglich, eine direkte Verteilung der Daten auf andere Server zu markieren, indem einzelne "mounts"von einem Knoten markiert werden. Die Daten werden somit auf andere Festplatten übertragen, sodass diese aus dem System entfernt werden kann. Außerdem können Snapshots eingerichtet werden, sowohl für die Storage- als auch für den Master-Server.

6.6.3 GlusterFS

Bei MooseFS wird sehr viel Wert auf die Sicherheit der Daten gelegt. Dies ist auch bei GlusterFS der Fall. Während die Zugriffsrechte bereits am besten verwaltet werden können, bietet dieses Dateisystem auch die Möglichkeit, **automatische Replikationen** von einzelnen Dateien und Ordnern anzulegen. Dies geschieht völlig automatisch im Hintergrund.
Es müssen keine speziellen Server zur Verfügung gestellt werden, die nur als Backup-Server dienen, sondern die Daten werden automatisch auf andere vorhandene Storageserver verteilt. Dies führt zu der Möglichkeit, dass bei einem Ausfall einzelner Hardwarekomponenten ein Weiterarbeiten möglich ist. Diese Verteilung geschieht automatisch im Hintergrund ohne das es zu Leistungseinbußen in der Performance kommt.

6.6.4 Zusammenfassung - Replikation

Replikationen sind insbesondere in der Sicherung der Dateien unumgänglich. Bei einem Ausfall einzelner Hardwarekomponenten kann es zu einem vollständigen Absturz des Dateisystems führen. So ist es im vorliegenden Testfall bei **FhGFS** der Fall. Durch den „Sanitycheck“ wird verhindert, dass die Verbindung vom Client zum Dateisystem hergestellt wird, wenn es zu einem Ausfall eines einzelnen Knotens kommen sollte. **MooseFS** und **GlusterFS** bieten durch die Replikationsmöglichkeit enorme Vorteile. Die Daten sind zum einen noch weiter verfügbar, zum anderen ist es generell möglich, weiter mit dem System zu arbeiten.

6.7 Fehlerverhalten

Das Fehlerverhalten eines Dateisystems ist sehr weitläufig. In diesem Kriterium wird betrachtet, wie das Dateisystem sich verhält, wenn ein Knoten nicht verfügbar ist, welche Möglichkeiten existieren, Server neu zu starten und die Einstellungen zu verändern.

6.7.1 FhGFS

In jedem Client ist standardmäßig der sogenannte „Sanitycheck“ eingerichtet, welcher beim Start des Client-Dienstes sicherstellt, dass alle zuvor eingerichteten Storageserver exisiteren und erreichbar sind.
Wenn ein einzelner Server nicht zur Verfügung steht, ist die Verbindung nicht möglich und der „mount“ des Verzeichnisses wird verhindert. Dies kann bei besonders großen Compute-Clustern zu großen Problemen führen, da die Ausfallwahrscheinlichkeit bei vielen Hardwarekomponenten höher ist als bei einem einzelnen Knoten.

Wenn ein einzelner Storageserver nun nicht erreichbar ist, dann ist eine Arbeit mit anderen Servern überhaupt nicht möglich. Um dies zu gewährleisten muss der „Sanitycheck“ ausgeschaltet werden.

Trotzdem ist es im Testfall dazu gekommen, dass dieser nicht gestartet werden konnte, wenn ein einzelner Knoten nicht zur Verfügung stand. Dies konnte nur durch ein erneutes kompilieren des Clients behoben werden, weil die einzelnen Server in einer versteckten Datei gespeichert wurden.

Im Testfall wurden die Server manuell installiert, laut Aussage des Fraunhofer Institutes ist dieser Umstand kein Problem, wenn die Installation **automatisch** mit Hilfe der grafischen Oberfläche durchgeführt wurde. Aufgrund der vorhandenen Konfiguration, ohne Internetzugriff, war diese Methode leider nicht möglich.

Dadurch, dass jeder Knoten über einen „paswordless ssh“ Zugriff zu jedem anderen Knoten zur Verfügung gestellt haben muss, ist es direkt möglich, dass einzelne Server hinzugefügt, abgeschaltet und neu gestartet werden können. Daher ist die Kontrolle vom Hauptserver sehr gut gegeben.

Im Großen und Ganzen steht das Fraunhofer Dateisystem im Punkt Fehlerverhalten nicht sehr gut dar. Die Möglichkeit, dass bestimmte Server zu dem Verbund hinzugefügt und daraus auch entfernt werden können ist zwar positiv, allerdings muss dies geschehen, bevor Benutzer dieses Dateisystem verwenden können.
Anschließend ist eine Veränderung der Struktur nur mit großen Umstellungen auf Clientseite möglich, zumindest bei der manuellen Installation einzelner Server. Auch beim Ausfall eines einzelnen Knotens kann es passieren, dass kurzzeitig das gesamte Dateisystem nicht mehr zur Verfügung steht. Erst beim vollständigen Wiederherstellen ist ein Weiterarbeiten möglich.

6.7.2 MooseFS

MooseFS wurde entwickelt, um eine sehr gute Replikationsmöglichkeit zu gewährleisten. Wenn die Daten auf mehrere Knoten aufgeteilt werden, hat das System die Möglichkeit, bei einem Ausfall eines Servers einen anderen anzusprechen, damit die **Verfügbarkeit** weiter gewährleistet werden kann. Bei einem drohenden Ausfall einer Festplatte wird der Administrator gewarnt, dieser kann diese markieren, sodass das Dateisystem die Daten noch auf andere Knoten verteilen kann.

Wenn ein plötzlicher Ausfall einer Festplatte eines einzelnen Knotens im laufenden Betrieb auftritt, ohne das Replikationen angelegt wurden, dann sind die Daten verloren und können nicht mehr wiederhergestellt werden, da die Verbindung zwischen Metadaten und den eigentlichen Daten nicht mehr gegeben ist. Allerdings ist es weiterhin möglich, mit dem System zu arbeiten, der „Master-Server“ verwaltet selbstständig das Umschalten.

6.7.3 GlusterFS

Das Fehlerverhalten von GlusterFS kann vom Administrator manuell eingestellt werden. Wenn keine Replikationen einer Datei vorhanden sind und der Server, auf dem diese gespeichert wurde, ausfällt, dann arbeitet das Dateisystem weiter, die Datei steht allerdings **nicht weiter zur Verfügung**. Es kommt dabei zu keinem Ausfall des gesamten Systems. Wenn Replikationen eingestellt werden, dann wird automatisch auf den anderen Server verwiesen, der eine Kopie der Datei beinhaltet. Dies macht GlusterFS zu einem sehr sicheren und ausgewogenem System, welches auch im

Fehlerfall noch weiter verfügbar ist und die Benutzer weiter arbeiten können.

Durch den generellen Aufbau des Systems, wo jede Datei vollständig auf einem Server abgelegt wird, hat es den Vorteil, dass bei einem Ausfall bestimmter Knoten keine „Datenleichen“ auf anderen Servern vorhanden sind, die nur aus Teilen der Datei bestehen. Andere Dateisysteme müssen solche Objekte selbstständig identifizieren und entfernen können, bei GlusterFS ist dieser Umstand nicht nötig.

6.7.4 Zusammenfassung - Fehlerverhalten

Fehlerverhalten ist ein sehr weitläufiger Begriff. Insbesondere bei verteilten Dateisystemen wird mit diesem Ausdruck der Umstand beschrieben, wie das System auf einen Ausfall einzelner Hardwarekomponenten reagiert und ob dieses anschließend noch weiter zur Verfügung steht.

Das **Fraunhofer Dateisystem** schneidet in diesem Punkt eher schlecht ab. Es existiert (noch) keine Möglichkeit automatische Sicherungen oder Replikationen von Servern anzulegen. Das führt dazu, dass im Falle eines Defektes das Dateisystem nicht zur Verfügung steht und es zu Datenverlust kommen kann.

Bessere Sicherungsmöglichkeiten bieten **MooseFS** und **GlusterFS** an. Diese können im Falle eines Knotenausfalls weiterarbeiten. Die automatische **Rekonfiguration** führt dazu, dass der defekte Server nicht weiter angesprochen wird und die Verteilung der Daten auf andere ausgelagert wird. Beide beherrschen den Umgang mit Replikationen. Im Falle eines Defektes werden die Dateien von anderen Servern gelesen. Dies erhöht die **Hochverfügbarkeit** eines Compute-Clusters.

6.8 Heterogenität

Die Eigenschaft Heterogenität für ein Dateisystem bedeutet, dass dieses auf unterschiedlichen Hardwarekomponenten installiert werden kann und mit diesen umgehen kann. Das Gegenteil wäre die Homogenität. Die Heterogenität beschränkt sich nicht nur auf unterschiedliche Hardware, sondern auch auf die verschiedenen Betriebssysteme und die, wenn nötig, unterschiedlichen Dateisysteme einzelner Knoten.

6.8.1 FhGFS

Das verteilte Dateisystem des Fraunhofer Institutes zeichnet sich durch nahezu **perfekte Heterogenität** aus. Es besteht die Möglichkeit, alle Arten von Hardware einzusetzen, die in einem Clusterverbund zusammengefasst werden. Die einzelne Peformance wird dabei vom Hauptserver analysiert und die Verteilung der Daten erfolgt entsprechend der Geschwindigkeit und Festplattenkapazitäten gleichmäßig.

Auf **Netzwerkebene** unterstützt dieses System automatisch die Nutzung von Ethernet- und Infiniband-Verbindungen. Diese müssen bei anderen Systemen meist händisch oder über Umwege hinzugefügt gefügt werden.

Auf **Betriebssystemebene** werden Installationsroutinen und Dienste für drei große Arten von Linux-Distributionen angeboten: RedHat-Linux, SuSe-Linux und Debian-GNU-Linux-Distributionen. Außerdem besteht die Möglichkeit, die Daten direkt als Paket zu erhalten, allerdings wird keine Garantie dafür gegeben, dass jede Distribution unterstützt wird.

Auch auf Client-Seite muss eine solche Distribution zur Verfügung gestellt werden, eine Implementation direkt für Mac- oder Windows-Systeme ist nicht verfügbar. Es konnten auch keine Informationen über solche Überlegungen gefunden werden.

FhGFS benötigt bereits vorhandene Dateisysteme auf den einzelnen Clients. Dabei werden **ext3/4**, sowohl mit Extents und ohne unterstützt, aber auch **XFS** und **BTRFS** können verwendet werden. Die Verwendung von BTRFS wird allerdings noch nicht empfohlen, weil sich dieses noch im Entwicklungsstadium befindet und somit das Dateisystem noch Fehler aufweisen kann, die die Konsistenz des verteilten Dateisystems beeinflussen kann. Auch die Mischung verschiedener Dateisysteme ist

möglich. Das hat den Vorteil, dass das Fraunhofer Dateisystem sehr flexibel einsetzbar ist und in Testumgebungen mit verschiedenen Systemen getestet werden kann.

6.8.2 MooseFS

MooseFS kann auf unterschiedlichen Betriebssystemen installiert werden. Sowohl nahezu **alle Linux-Distributionen**, sowie FreeBSD, OpenSolaris und als Client auch Mac OS X werden unterstützt. Durch die Nutzung von „FUSE“ ist auch eine entsprechende Implementation für Windows geplant, wenn das Projekt „WinFUSE“ als stabil gekennzeichnet wird.

Auf der **Netzwerkebene** unterstützt MooseFS zur Zeit nur IP-basierte-Verbindungen. Im Testfall wurde dieses durch Infiniband-over-IP genutzt.

Die regulären Dateisysteme auf jedem einzelnen Knoten, sowie die unterschiedlichen RAID-Konfigurationen, können vom Administrator frei gewählt werden, MooseFS unterstützt **ext3/4, XFS, BTRFS und andere**. Es ist auch nicht zwingend erforderlich, dass die Hardware möglichst gleich gehalten wird. Bei der Anmeldung eines neuen Storageservers wird vom Master-Server ein kurzer Festplattentest durchgeführt, sodass dieser die Geschwindigkeit, sowie den verfügbaren Festplattenspeicherplatz kennt und das Dateisystem entsprechend angepasst werden kann.

6.8.3 GlusterFS

Das Dateisystem steht sowohl als Archiv zur Verfügung, als auch als vorgefertigtes Image für Debian-Distributionen. Der Server kann auf **alle Arten von Linux-Distributionen** installiert werden, der Client zusätzlich, durch die Nutzung von „FUSE“, auf OpenSolaris, FreeBSD und auf Mac OS X.

Durch das Bereitstellen eines Paketes, welches zur Nutzung von Infiniband benötigt wird, kann der **Netzverkehr** sowohl über Ethernet-, als auch Infiniband-Verbindungen geregelt werden.

GlusterFS kann sowohl mit **verschiedenen Dateisystemen** im Untergrund umgehen, als auch mit deren unterschiedlichen RAID-Konfigurationen. Dies macht GlusterFS zu

einem sehr benutzerfreundlichen Dateisystem, welches in heterogenen Netzwerken eingesetzt werden kann.

6.8.4 Zusammenfassung - Heterogenität

Heterogenität bedeutet, dass das verteilte Dateisystem mit unterschiedlicher Hardware, den darunterliegenden Dateisystemen und Distributionen umgehen kann. Alle getesteten Systeme können mit allen Arten von regulären Dateisystemen umgehen, sowie die verschiedenen RAID-Konfigurationen. Diese können in unterschiedlichen Konstellationen und auf verschiedenen Hardwarekomponenten eingerichtet werden. Lediglich der Umstand, welche Distribution für die Installation der Server nötig ist, darin unterscheiden sich die Systeme sehr leicht. Während **MooseFS** und **GlusterFS** als gepacktes Archiv zur Verfügung stehen und auf jede Distribution installiert werden können, bietet **FhGFS** die Installation offiziell nur für RedHat-Enterprise-, SuSe- und Debian-Distributionen an. Im vorliegenden Testfall war eine Installation auf einem Gentoo-System nicht möglich. Der Client wird bei FhGFS auch nur für diese Systeme bereitgestellt, die anderen nutzen „FUSE“ und können auf allen Linux-Distributionen installiert werden, die diese Programmbibliothek unterstützen, zudem auch auf OpenSolaris, FreeBSD und Mac OS X Systemen.
Daher sind diese Systeme leicht flexibler in der Distributionsauswahl, als das Fraunhofer Filesystem.

Kapitel 7

Fazit

Für das Universitätsrechenzentrum der Hansestadt Greifswald wird ein verteiltes Dateisystem gesucht, welches den Ansprüchen in Sachen Performance, Skalierbarkeit und Stabilität gerecht wird. Dieses System soll auf dem geplanten neuen Compute-Cluster genutzt werden, welcher meherere hundert Knoten beinhalten wird. Das Dateisystem muss daher performant und skalierbar sein, um die höchstmögliche Geschwindigkeit der einzelnen Knoten zur Verfügung zu stellen. Diese Arbeit dient als Hilfe zur Ausarbeitung eines optimalen Nutzungskonzepts.
Das verteilte Dateisystem soll neben der Performance möglichst einfach zu konfigurieren und administrieren sein. Der neue Compute-Cluster wird für komplexe Berechnungen im naturwissenschaftlichen Bereich genutzt.

Um ein möglichst optimales Dateisystem herauszufinden, wurden bestimmte **Kriterien** festgelegt. Grundentscheidend ist die gesamte **Performance**, sowie die Möglichkeit der **Skalierbarkeit** muss gegeben sein. Zudem ist der Punkt **Stabilität** großer Bedeutung, allerdings können einzelne Systemausfälle hingenommen werden, solange das System nicht zu ständigen Ausfällen neigt. Der Punkt der **Verwaltbarkeit** ist besonders wichtig und kann bei sehr engen Entscheidungen großen Einfluss auf das Ergebnis nehmen.

Durch die Anwendung auf einem großen Compute-Cluster ist der Punkt der Heterogenität untergeordnet, weil die einzelnen Knoten mit den gleichen Komponenten und Distributionen genutzt werden.

Bei der Entscheidung für ein Dateisystem wurden im Vorfeld viele verteilte Dateisysteme miteinander verglichen. In die nähere Auswahl kam dabei das Fraunhofer Dateisystem FhGFS, welches optimale Performance und massive Skalierbarkeit bei einfachster Verwaltbarkeit verspricht. FhGFS hat zudem das Alleinstellungsmerkmal, dass alle genutzten Dienste und Bibliotheken vom Fraunhofer Dateisystem zur Verfügung gestellt werden. Es ist keine Nutzung zusätzlicher Programme oder Module nötig.

Außerdem wurde sich für MooseFS, CephFS und GlusterFS entschieden. Alle besitzen

die Eigenschaft, dass Hochverfügbarkeit durch die Nutzung von **Replikationen** angestrebt wird. Diese drei Dateisysteme nutzen zur Kommunikation des Clients mit dem Server das Kernelmodul „FUSE“, was ermöglicht, dass diese auch mit OpenSolaris-, FreeBSD-, und Mac OS X -Betriebssystemen arbeiten können.

Für die Testphase wurden weitere Entscheidungen getroffen. Auf dem zur Verfügung gestellten Servern des Universitätsrechenzentrums wurde das Betriebssystem *CentOS*6.4 installiert, partitioniert wurden alle Festplatten im *ext*4-Format. Dieses versprach die beste Performance besonders bei kleinen Dateien. Es wurde auf das Format BTRFS verzichtet, da dieses zwar im Linux-Kernel vorhanden, allerdings noch nicht als stabil gekennzeichnet wurde.

Der gleiche Umstand galt auch für die Nutzung von CephFS. Dieses befindet sich noch im Entwicklungsstadium und aufgrund dessen wird darauf hingewiesen, dass dieses System noch nicht im praktischen Einsatz genutzt werden sollte. Daher wurden die verteilten Dateisysteme FhGFS, MooseFS und GlusterFS miteinander verglichen.

	FhGFS	MooseFS	GlusterFS
Performance	1	6	3
Skalierbarkeit	1	6	4
Sicherheit	3	2	1
Stabilität	1	2	1
Heterogenität	2	1	1
Replikation	4	1	2
Fehlerverhalten	3	2	1
Administration	1	3	5
Gesamt:	**16**	**23**	**18**
Punktevergabe anhand von Schulnoten, weniger ist besser			

Abbildung 7.1: Zusammenfassung der Testergebnisse

Die beste Performance und Skalierbarkeit wurde vom **Fraunhofer Dateisystem** gezeigt, zusätzlich auch die Punkte Stabilität und Administration. Allerdings gaben die Eigenschaften Sicherheit, Fehlerverhalten und Replikation Grund zu Nachbesserungen. Diese sind vom Fraunhofer Institut bereits angekündigt und sollen zu einem späteren Zeitpunkt eingepflegt werden.

MooseFS bot zwar sehr gute Sicherungsmöglichkeiten und viele Eigenschaften für die Zugriffskontrolle an, die Testdurchläufe mit diesem System wurden allerdings abgebrochen, weil die Performance absolut indiskutabel war. Die Geschwindigkeit lag oftmals bei 5-10MByte/s, wohingegen das Fraunhofer Dateisystem Geschwindigkeiten von 300MByte/s erreichen konnte.
Die Installation wurde exakt nach den Vorgaben des Herstellers durchgeführt, auch verschiedene Durchläufe konnten keine anderen Ergebnisse aufzeigen. Es konnten keine Möglichkeiten der Performanceoptimierungen gefunden werden, da in den Konfigurationsdateien keine Parameter existieren, die zu einer Verbesserung der Geschwindigkeit hätten führen können.
Da das Dateisystem auf einigen Servern effektiv im Einsatz ist, bleibt die Frage offen, warum dieses System auf dem Cluster nicht vernünftig funktionierte. Trotzdem fällt MooseFS aus der Entscheidungsfindung heraus.

GlusterFS verfolgte einen anderen Ansatz, was die Verteilung der Daten betrifft. Diese wurden nicht zerlegt und getrennt voneinander auf verschiedenen Servern gespeichert, sondern die Dateien blieben vollständig bestehen und wurden auf einem einzelnen Server abgelegt. Die Verteilung erfolgte nur dann, wenn mehrere Dateien vorlagen. Diese wurden dann auf verschiedene Server verteilt und konnten dementsprechend auch verteilt gelesen werden.

Für das Universitätsrechenzentrum ist es entscheidend, die maximale **Performance** aus dem neuen Compute-Cluster zu erreichen. Die durchgeführten Tests belegen, dass dieses nur vom **Fraunhofer Dateisystem** erreicht wurde. Da die Punkte Replikation und die verschiedenen Sicherheitsaspekte eine eher untergeordnete Rolle spielen, konnte nur FhGFS der Gesamtsieger aus diesen Testverfahren werden. Aufgrund der professionellen grafischen Oberfläche wurde ein sehr guter Überblick über die vorhandene Hardware gegeben.

GlusterFS eignete sich auch für die Nutzung auf dem neuen Compute-Cluster, allerdings sind die Art der Verteilung und die fehlende Flexibilität im Bereich der Verwaltbarkeit zwei wichtige Faktoren, die vom Universitätsrechenzentrum vorausgesetzt werden. Der große Vorteil der Sicherungsmöglichkeiten spielte eine untergeordnete Rolle, sodass dieses nur als zweiter Sieger genannt werden kann.

Aufgrund der schlechten Performance von **MooseFS**, die nicht zu erklären ist, kann

keine Empfehlung dafür gegeben werden. Trotz zahlreicher Testdurchläufe lag die Schreibgeschwindigkeit lediglich bei 5-10% des Wertes, welcher von FhGFS zur Verfügung gestellt wurde. Auf einem „High-Performance-Cluster“ ist dies ein Wert, der als Ausschlusskriterium anzusehen ist.

Abbildungsverzeichnis

Selbstständigkeitserklärung

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel sind angegeben. Die Arbeit hat mit gleichem bzw. in wesentlichen Teilen gleichem Inhalt noch keiner Prüfungsbehörde vorgelegen.

Greifswald , 16. September 2013

Ort, Datum, Felix Janke

Literaturverzeichnis

[1] Name, Vorname (Jahr der Veröffentlichung), in: Thema, URL: http://... (Stand: Datum des Raussuchens).

[2] Büst, René (2010), in: Was ist Cluster Computing?, URL: http://clouduser.de/grundlagen/was-ist-cluster-computing-158 (Stand: 10.06.2013).

[3] Juelich, Forschungszentrum (2013), in: „JUQUEEN Juelich Blue Gene/Q“, URL: http://www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUQUEEN/JUQUEEN/JUQUEEN_node.html (Stand: 10.06.2013).

[4] Erb, Pascal (2008), in: Cluster Computing, URL: http://www.informatik.uni-mainz.de/lehre/cg/SS2008_PA/data/V09b_ClusterComputing_Erb.pdf (Stand: 11.06.2013).

[5] Held, Andrea (2012), in: Grundlagen der Hochverfügbarkeit, URL: http://www.tecchannel.de/sicherheit/management/429794/grundlagen_der_hochverfuegbarkeit/ (Stand: 11.06.2013).

[6] Held, Andrea (2005), in: Hochverfügbarkeit, Kennzahlen und Metriken, URL: http://www.tecchannel.de/test_technik/grundlagen/430342/hochverfuegbarkeit_kennzahlen_und_metriken/index13.html (Stand: 12.06.2013).

[7] Jelinek, Gerd (2007), in: High Availability Cluster, URL: http://www.ha-cc.org/uploads/pics/Cluster_mit_LB_und_Webserver_04.gif (Stand: 12.06.2013).

[8] Jelinek, Gerd (2007), in: High Availability Cluster, URL: http://www.ha-cc.org/uploads/pics/Load_Balancing_04.gif (Stand: 12.06.2013).

[9] Jelinek, Gerd (2007), in: High Availability Cluster, URL: http://www.ha-cc.org/uploads/pics/Beowulf_Cluster_09.gif(Stand: 12.06.2013).

[10] Microsoft Corp. (2012), in: Failoverclustering: Übersicht, URL: http://technet.microsoft.com/de-de/library/hh831579.aspx (Stand: 12.06.2013).

[11] WEBfactory GmbH (2013), in: WEBfactory Cluster - Höchste Verfügbarkeit durch redundante Server, URL: http://www.webfactory-world.de/images/produkte/webfactory-cluster-hochverfuegbarkeit.jpg?sfvrsn=12 (Stand: 12.06.2013).

[12] Microsoft Corp. (2013), in: Skalierbarkeit, URL: http://msdn.microsoft.com/de-de/library/aa292172(v=vs.71).aspx (Stand: 12.06.2013).

[13] Microsoft Corp. (2013), in: Überblick über Skalierbarkeit, URL: http://i.msdn.microsoft.com/dynimg/IC75058.gif (Stand: 12.06.2013).

[14] Luther, Jörg, Vilsbeck, Christian, Haluschak, Bernhard (2011), in: RAID im Überblick, URL: http://www.tecchannel.de/storage/extra/401665/raid_sicherheit_level_server_storage_performance_festplatten_controller/ (Stand: 12.06.2013).

[15] Patterson, David A., Gibson, Garth, Katz, Randy H., (1987) in: A Case for Redundant Arrays of enexpensive Disks (RAID), URL: http://www.cs.cmu.edu/~garth/RAIDpaper/Patterson88.pdf (Stand: 12.06.2013).

[16] Apple Inc. (2008), in: Mac OS X: About file system journaling, URL: http://support.apple.com/kb/ht2355 (Stand: 19.06.2013).

[17] Fehlau, Michael (2013), in: Linuxfibel - System-Administration - Dateisysteme, URL: http://de.linwiki.org/wiki/Linuxfibel_-_System-Administration_-_Dateisysteme (Stand: 19.06.2013).

[18] Hellwig, Christoph (2009), in: XFS: the big storage file system for linux, URL: http://oss.sgi.com/projects/xfs/papers/hellwig.pdf (Stand: 06.08.2013).

[19] Silicon Graphics Inc. (2006), in: XFS Overview und Internals 02 - Overview, URL: http://oss.sgi.com/projects/xfs/training/xfs_slides_02_overview.pdf (Stand: 06.08.2013).

[20] Kaiser, Nicolas (2011), in: Ext4, URL: http://kernelnewbies.org/Ext4 (Stand: 08.08.2013).

[21] Dr. Diedrich, Oliver (2009), in: The Ext4 Linux File system, URL: http://www.h-online.com/open/features/The-Ext4-Linux-file-system-746579.html (Stand: 08.08.2013).

[22] Dr. Diedrich, Oliver (2009), in: Das Dateisystem Btrfs, URL: http://www.heise.de/open/artikel/Das-Dateisystem-Btrfs-221863.html (Stand: 09.08.2013).

[23] Prof. Dr. Schütte, Alois (2012), in: Verteilte Dateisysteme, URL: http://www.fbi.h-da.de/~a.schuette/Vorlesungen/VerteilteSysteme/Skript/6_VerteilteDateisysteme/VerteilteDateisysteme.pdf (Stand: 09.08.2013).

[24] Fraunhofer Institut, in: FhGFS Wiki: Overview (o.J.), URL: http://www.fhgfs.com/wiki/wikka.php?wakka=Overview (Stand: 12.08.2013).

[25] Fraunhofer Institut, in: FhGFS Wiki: GUIConfigureFhGFS (o.J.), URL: http://www.fhgfs.com/wiki/images/config_roles.png (Stand: 12.08.2013).

[26] Fraunhofer Institut, in: FhGFS Wiki: SystemArchitecture (o.J.), URL: http://www.fhgfs.com/wiki/images/sysarch2.png (Stand: 12.08.2013).

[27] Norcott, William D., in: Iozone Filesystem Benchmark (o.J.), URL: http://www.iozone.org/docs/IOzone_msword_98.pdf (Stand: 22.08.2013).

[28] Professor Dr. rer. nat. Koch, Michael (2012), in: Parallelverarbeitung Einschub: einfache Kenngrößen, Vorlesungsfolien, m_par2_250912.pdf (Stand: 26.08.2013).

[29] Goetz, Tobias (2013), in: FhGFS: A Fast and Scalable Parallel Filesystem, URL: http://www.clustermonkey.net/FileSystems/fhgfs-a-fast-and-scalable-parallel-filesystem-crafted-in-germany.html (Stand: 27.08.2013).

[30] Wilson, Tracey, Stutz, Al, Dr. Buerger, Paul, Panton, Roger, Parker, Michelle, Ezekielian, Armen (2010), in: Parallel File System Survey Report, URL: http://www.avetec.org/applied-computing/dice/reports/docs/PFS_Survey_Report_2010.pdf (Stand: 26.08.2013).

[31] Dr. Oevel, Gudrun (2009), in: Vorlesung IT-Sicherheit, SS09, Kapitel 3, URL: http://imt.uni-paderborn.de/fileadmin/imt/personen/ vorlesung-it-sicherheit-ss-09/3-Zugriffskontrolle.pdf (Stand: 03.09.2013).

[32] Adler, Yves (2008), in: Replikation in Datenbanken, URL: http://www.imn.htwk-leipzig.de/ kudrass/Lehrmaterial/ DB2-VL/DB2-08/14A-Vortrag.pdf (Stand: 03.09.2013).

[33] Core Technology sp.z o.o (2013), in: About MooseFS, URL: http://www.moosefs.org/about-mfs.html (Stand: 03.09.2013).

[34] Core Technology sp.z o.o (2013), in: Who is using MooseFS, URL: http://www.moosefs.org/who-is-using-moosefs.html (Stand: 03.09.2013).

[35] Inktank Storage, Inc. (2013), in: Ceph Documentation, URL: http://ceph.com/docs/master/cephfs/ (Stand: 04.09.2013).

[36] Weitzel, Derek (2012), in: Ceph: A Scalable, High-Performance Distributed File System, URL: cse.unl.edu/ ylu/csce990/notes/Ceph.pptx (Stand: 04.09.2013).

[37] Red Hat, Inc. (2013), in: GlusterFS About, URL: http://www.gluster.org/about/ (Stand: 05.09.2013).

[38] Red Hat, Inc. (2013), in: GlusterFS Concepts, URL: http://www.gluster.org/community/documentation/ index.php/GlusterFS_Concepts (Stand: 05.09.2013).

[39] Schmelzle, Michael (2011), in: Samsung Spinpoint M8 1TB (HN-M101MBB) im Test, URL: http://www.pcwelt.de/produkte/ Samsung-Spinpoint-M8-1TB-HN-M101MBB-2-5-Zoll-Festplatte-Test-3265935.html (Stand: 11.09.2013).