Lustre

Hausarbeit von: Jan Harder

Universität Hamburg

Modul: Proseminar Speicher- & Dateisysteme

Betreuer: Dr. Michael Kuhn

Abgabe: 26.02.2019,

E-Mail: jan.harder37@gmail.com

Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

INHALT

1. EINLEITUNG

Der Name Lustre ist eine Zusammensetzung aus Linux und Cluster. Lustre beschreibt ein paralleles, verteiltes Dateisystem, welches für Linux-Systeme entwickelt wurde und speziell in Cluster-Computing-Umgeben eingesetzt wird. Diese setzen voraus, dass sehr viele Clients (100.000+)[1] auf einen Shared-Storage zugreifen können und, dass das Dateisystem auch bei hoher Auslastung verlässlich arbeitet. Lustre ist, wie es für Dateisysteme, welche unter Linux verwendet werden, üblich ist, POSIX-konform, wodurch es eine standardisierte Schnittstelle anbieten kann, welche nicht extra für jedes Betriebssystem angepasst werden muss. Lustre ist unter der Open-Source-GNU-GPL-Lizenz (Version 2) frei verfügbar und darf bearbeitet und auch wieder veröffentlicht werden.[2]

1.1 MOTIVATION

Der Anspruch an moderne Dateisysteme wächst immer weiter. Diese müssen immer größere Datenmengen in kürzerer Zeit verarbeiten können. Die Server haben jedoch einen beschränkten Durchsatz und eine beschränkte Kapazität. Lustre will dieses Flaschenhalsproblem, dass sich die Übertragung an einem Server staut, dadurch vermeiden, dass die Daten über mehrere Server verteilt werden. So kann der Durchsatz, also die übertragene Datenmenge in einer bestimmten Zeit deutlich erhöht werden. Wie Lustre dies genau umsetzt, wird im Folgenden erläutert.

1.1 DAS PROJEKT LUSTRE

Lustre wurde 1999 als Forschungsprojekt von Peter J. Braam an der Carnegie Mellon University ins Leben gerufen. Zwei Jahre später gründete er daraus das Unternehmen Cluster File Systems, welches vom US-Energieministerium gefördert wurde. 2007 wurde die Firma von Sun Microsystems aufgekauft, welche das eigene Dateisystem ZFS und ihr Betriebssystem mit Lustre optimieren wollte. Nachdem Sun Microsystems jedoch 2010 von Oracle aufgekauft wurde, wurde die Weiterentwicklung an Lustre eingestellt. Daraufhin setzten verschiedene Gruppen, wie Whamcloud oder Open SFS, die Weiterentwicklung fort, was durch die Open Source Lizenz möglich wurde. [2]

[1] Lustre; “Introduction to Lustre Architecture”; Oktober 2017: http://wiki.lustre.org/images/6/64/LustreArchitecture-v4.pdf
[2] Wikipedia: https://en.wikipedia.org/wiki/Lustre_(file_system)

2. ARCHITEKTUR

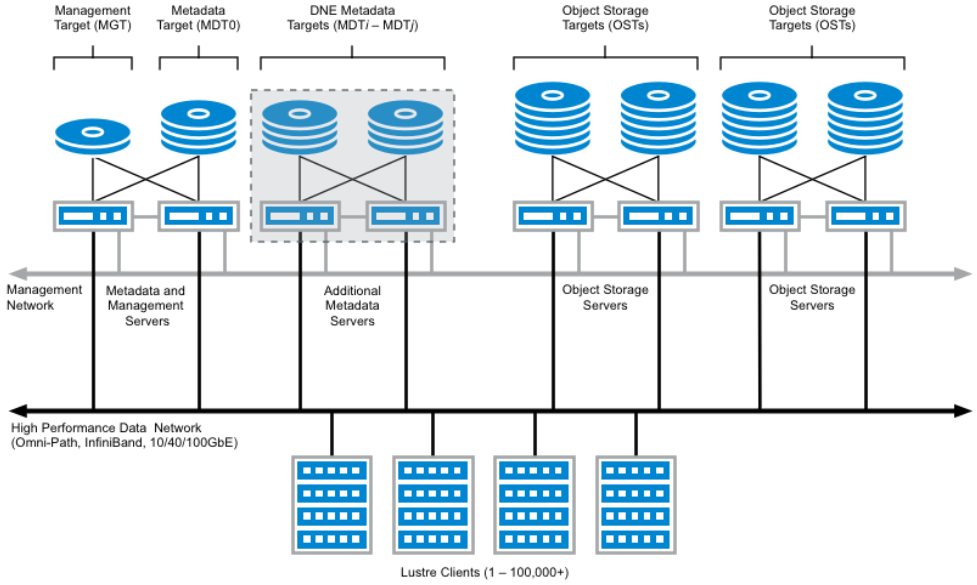

Abbildung 1: Aufbau eines Lustre Dateisystems von wiki.lustre.org [1]

Das Lustre-Dateisystem besteht aus Clients, Servern und Speichermedien (Targets). Die Kommunikation erfolgt über das L(ustre)-Net(working). Über dieses sind alle Komponenten miteinander verbunden. Dadurch müssen die Clients nicht mehr direkt mit den Speichermedien kommunizieren, sondern können Anfragen an die jeweiligen Server stellen, welche die Anfragen bearbeiten und das Ergebnis den Clients melden. So muss den Clients der genaue Speicherort der Dateien nicht länger bekannt sein. Es gibt drei verschiedene Server- und Targetarten. Jeweils Management-, Metadata- und Object Storage-Server und Targets. Die Funktionen dieser Server auf denen entsprechende Services laufen, werden im Folgenden weiter erläutert.

2.1 CLIENTS

Lustre ist für eine sehr hohe Anzahl an Clients konzipiert, kann jedoch auch für kleinere Systeme genutzt werden. Die Clients können durch den Global Name Space alle Daten sehen und benötigen keine eigenen Festplatten mehr, da alle Dateien verteilt gespeichert werden. Der Lustre-Client fasst die Meta- und Objektdaten POSIX-konform zusammen, wodurch die Anwendungen, die auf den Rechnern laufen, nicht speziell für Lustre geschrieben werden müssen.

2.2 MANAGEMENT SERVER / MANAGEMENT TARGET

Der Management Service (MGS) stellt die Konfigurationsinformationen über das Netzwerk bereit. Alle Komponenten des Netzwerkes müssen sich bei der ersten

Anmeldung bei dem Management Service registrieren, damit die Kommunikation zwischen den verschiedenen Komponenten möglich ist. Ein Lustre-Netzwerk hat nur genau einen Management Server. Die Konfigurationsinformationen werden von dem Management Service auf den Management Targets (MGT) gespeichert, wobei ein Management Service Dateien auf mehreren Management Targets speichern kann. Der Management Service ist nicht an den eigentlichen I/O-Operationen beteiligt. Ein Ausfall des Management Services ist besonders kritisch, da sich dann keine Server oder Clients mehr in dem Netzwerk registrieren könnten.[3]

2.3 METADATA SERVER / METADATA TARGET

Der Metadata Service (MDS) ist für die Bereitstellung von Metadaten zuständig. Die Metadaten werden in inodes (Index nodes) gespeichert und enthalten Informationen über die Datei, wie den Dateinamen, Zugriffsrechte, Sperren oder Informationen über die Aufteilung in Stripes. Die Sperren sind in verteilten, parallelen Dateisystemen dringend notwendig, um die Kohärenz der Dateien sicherzustellen. Die inodes werden auf Metadata Targets (MDT) gespeichert. Außerdem sind die Metadata Services für das Erstellen und Löschen von Dateien zuständig, da bei diesen Vorgängen Metadaten bearbeitet oder erzeugt werden müssen. Ohne die Metadata Services oder die Metadata Targets ist kein Zugriff auf die Dateien möglich, da der Speicherort nicht ermittelt werden kann. Im Gegensatz zu den Management Services ist die Anzahl der Metadata Services nicht begrenzt.

2.4 OBJECT STORAGE SERVER / OBJECT STORAGE TARGET

Der Object Storage Service (OSS) verwaltet die gespeicherten Dateien. Diese werden auf den Object Storage Targets (OST) gespeichert. Des Weiteren ist der Object Storage Service für I/O-Operationen, also für den lesenden oder schreibenden Zugriff zuständig. Untereinander agieren die Object Storage Server passiv, da die Verteilung der Dateien von den Metadata Services übernommen wird. Dateien werden in Stripes aufgeteilt und auf mehrere Objekte verteilt, dadurch muss eine Datei nicht unbedingt auf einem einzigen Server liegen. Die Kapazität des Netzwerkes lässt sich leicht durch Hinzufügen von Object Storage Targets erhöhen, wodurch die einfache Skalierbarkeit der Lustre Netzwerke realisiert wird. Die Kapazität des Netzwerkes wird durch die Summe der Kapazitäten der

[3] Lustre; “Introduction to Lustre Architecture”; Oktober 2017: http://wiki.lustre.org/images/6/64/LustreArchitecture-v4.pdf

einzelnen Object Storage Targets bestimmt. Diese sollten möglichst gleichmäßig auf die Object Storage Server verteilt werden, um eine maximale Performance zu ermöglichen.

2.5 OBJECT STORAGE DEVICES

Die verschiedenen Targets sind Object Storage Device (OSD) Instanzen. Diese können zwei verschiedene Dateisysteme in Lustre installiert haben. Die erste Möglichkeit ist LDISKFS, eine Weiterentwicklung von ext4. Für dieses muss jedoch der Kernel angepasst werden. Die Anzahl der Inodes, welche auf dem Target gespeichert werden können, wird bei der Formatierung des Speichers berechnet. Am Anfang wurde in Lustre nur LDISKFS für die Targets unterstützt. Ab der Version 2.3 von Lustre wurde dann auch das Dateisystem ZFS unterstützt. Für dieses muss der Kernel nicht angepasst werden, jedoch ist die Installation aufwendiger als bei LDISKFS. Die Anzahl der Inodes wird dynamisch berechnet. ZFS ermöglicht die Speicherung von Dateien mit größerer Dateigröße als bei LDISKFS, außerdem kann theoretisch insgesamt eine größere Datenmenge gespeichert werden. Diese ist aber natürlich auch durch die Eigenschaften des Speichermediums begrenzt. Kombinationen von ZFS und LDISKFS sind ebenfalls möglich.[2]

2.6 LUSTRE NETWORKING (LNET)

LNET dient als API und Kommunikationsprotokoll für Lustre. Es vermittelt zwischen den Clients und den Servern. Dementsprechend müssen alle Clients und Server für LNET konfiguriert sein. LNET unterstützt verschiedene Netzwerktypen, wie Ethernet, Quadrics oder Infiniband. Letzteres ist besonders für hohen Durchsatz ausgelegt. Durch die Verwendung von LNET Routern können mehrere LNET-Netzwerke verbunden werden und Schnittstellen zwischen den verschiedenen Netzwerktypen geschaffen werden.

2.7 KONFIGURATION

Der minimale Aufbau eines Lustre-Netzwerkes besteht aus jeweils einem Paar aus Management Server und -Target, Metadata Server und -Target und Object Storage Server und -Target. Um jedoch eine hohe Verfügbarkeit zu generieren werden der Metadata Server und der Management Server in einer Failover-Kombination verbunden. Failover-Kombinationen stellen eine zentrales Merkmal von Lustre dar, welches ich im nächstes Kapitel weiter erläutern werde. Durch die einfache Skalierbarkeit von Lustre lassen sich problemlos weitere Metadata Server und -Targets und Object Storage Server und -Targets hinzufügen,

wodurch die hohe Verfügbarkeit und Kapazität ermöglicht werden können. Bei herkömmlichen Dateisystem könnte durch das Hinzufügen weiterer Server zwar die Verfügbarkeit erhöht werden, aber nicht die Performance des Netzwerkes, da trotzdem nur ein Server auf den Speicher zugreifen könnte, es wird also nur ein zweites Dateisystem erzeugt, anstatt dass der zweite Server den ersten bezogen auf die Performance unterstützt.[1]

3. FUNKTIONEN

Zwei wesentliche Funktionen von Lustre sind Failover-Konfigurationen und Striping. Beide tragen zu großen Teilen den Hauptversprechen von Lustre bei. Durch sie sollen besonders die hohe Verfügbarkeit aber auch die einfache Skalierbarkeit und das Speichern von vielen und großen Daten ermöglicht werden.

3.1 FAILOVER

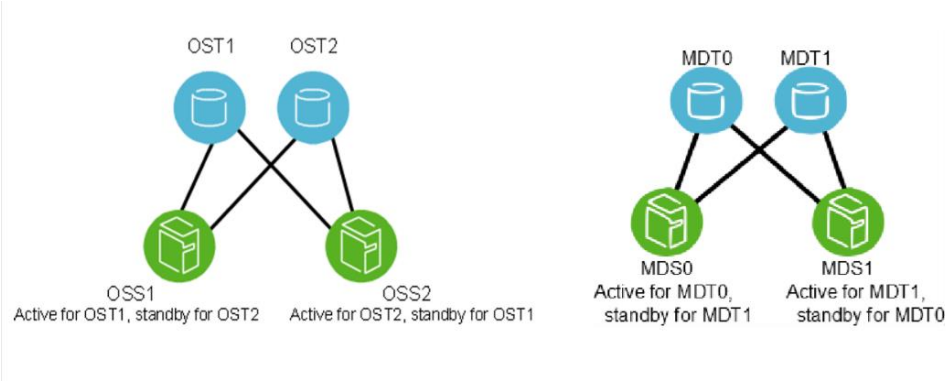

Abbildung 2: Aufbau von Failoverpaaren von doc.lustre.org [2]

Failover-Konfigurationen werden eingesetzt, um die Ausführung bei Serverausfällen sicherzustellen und somit auch eine hohe Verfügbarkeit zu gewährleisten. Die Failover-Konfiguration kann für alle verschiedenen Serverarten in Lustre angewandt werden. Es werden immer zwei Targets mit zwei Servern verbunden, so dass jeder der beiden Server eine Verbindung zu beiden Targets hat. Dabei wird zwischen aktiv/aktiv- und aktiv/passiv-Konfigurationen unterschieden. Bei den aktiv/aktiv-Konfigurationen sind beide Server aktiv, aber jeweils nur für einen der beiden Targets. Diese Kombination wird für Object Storage Server und -Target und für Metadata Server und -Target verwendet (siehe Abbildung 2). Wenn einer der beiden Server ausfallen würde, würde der andere noch aktive Server alle Anfragen an alle verbunden Targets der beiden Server übernehmen. Da der noch aktive Server dann dementsprechend mehr Anfragen bearbeiten müsste, würde sich jedoch auch die Zeit, die benötigt wird, um alle Anfragen zu bearbeiten, erhöhen. Bis zur Version 2.4 von Lustre wurden die Metadata Server und -Targets noch passiv/aktiv geschaltet, da es bis zu dieser Version nur einen

Metadata Server gab. Außerdem wäre der Ausfall des Metadata Servers besonders kritisch.[4]

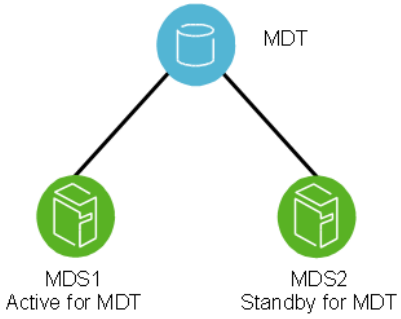

Abbildung 3: Aufbau von aktiv/passiv-Failoverpaaren von doc.lustre.org [3]

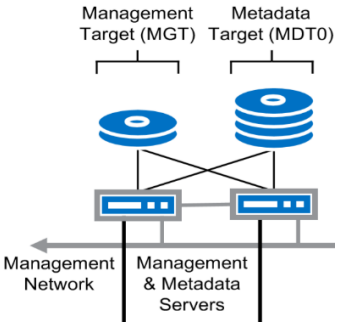

Abbildung 4: Aufbau von MGS/MDS-Failoverpaaren von wiki.lustre.org [4]

Die aktiv/passiv-Konfiguration besteht aus zwei Servern, welche beide nur mit demselben Target verbunden sind (siehe Abbildung 3). Hierbei ist ein Server gerade aktiv und der zweite Server ist passiv. Dieser wird nur aktiviert, wenn der erste Server ausfällt, in diesem Fall würde er alle Anfragen an den ausgefallenen Server übernehmen. So ist sichergestellt, dass die Targets immer angesprochen werden können. Da es nur einen Management Server gibt, kann dieser nicht mit einem anderen Management Server in Failover-Konfiguration installiert werden. Stattdessen wird dieser zusammen mit einem Metadata Server in Failover-Kombination installiert (siehe Abbildung 4). In der Regel ist dies der Root-Metadata Server und das entsprechendem Target (MDT0). Die Kombination ist aktiv/aktiv und im Falle eines Serverausfalles des Management Servers würde der Metadata Server auch Anfragen an die Management Targets bearbeiten und entsprechend andersherum bei einem Ausfall des Metadata Servers. Um sicherzustellen, dass alle Anfragen bearbeitet werden, auch wenn der entsprechende Server ausgefallen ist, werden alle Anfragen in einem Transaction-Log gespeichert. Wenn die Anfrage durch den Serverausfall nicht bearbeitet wurde, wird die Anfrage anschließend mit dem entsprechendem Failover-Server erneut ausgeführt.[4]

3.2 STRIPING

Um auch große Daten speichern zu können wird bei Lustre auf Striping gesetzt. Dabei werden die Dateien auf mehrere Stripes und Objekte aufgeteilt. Diese Stripes werden verteilt auf verschiedenen Object Storage Targets gespeichert. Alle Stripes einer Datei, die sich auf dem gleichen Object Storage Target befinden werden in dem gleichen Objekt gespeichert. Dabei werden die einzelnen Stripes fortlaufend nach dem Round-Robin-Algorithmus verteilt,

[4] Oracle; Intel Corporation; “Lustre Software Release 2.x - Operations Manual”; 2017: http://doc.lustre.org/lustre_manual.xhtml

wodurch eine gleichmäßige Verteilung und somit optimale Zugriffszeiten entstehen. *Als Beispiel zu Veranschaulichung des Algorithmus: Bei fünf Object Storage Targets und zwei Dateien mit jeweils 3 Stripes, würde die erste Datei auf die OST 1,2 und 3 aufgeteilt werden und die zweite Datei auf die OST 4,5,1.* Eine Datei kann auf bis zu 2000 verschiedene Object Storage Targets verteilt werden. Dadurch wird die Dateigröße nicht durch die Kapazität eines einzelnen OST limitiert. Außerdem werden durch das Striping das Lesen und das Schreiben von Dateien schneller, da dies parallel auf mehreren OST passieren kann. [5]

4. VERWENDUNG

Aufgrund der hohen Effizienz durch die Parallelität wird Lustre besonders stark für Supercomputer eingesetzt. Auch für Berechnungen mit sehr großen Datenmengen ist Lustre durch die einfache Skalierbarkeit gut geeignet. Daher wird es zum Beispiel bei aufwendigen Berechnungen, wie Wetterprognosen eingesetzt, aber auch bei zeitkritischen Berechnungen. 2016 haben 70 der 100 weltweit leistungsstärksten Supercomputer auf das Dateisystem Lustre zurückgegriffen. [1]

5. AKTUELLE FORSCHUNG

Dadurch, dass das Lustre Projekt ein Open Source Projekt ist, arbeiten viele verschiedene Gruppen, Universitäten oder Unternehmen an der Entwicklung von Verbesserungen. In der Liste von durchgeführten oder aktuellen Updates finden sich zum Beispiel die Universität Hamburg, Intel oder die Lustre Community. Des Weiteren sind noch viele geplante Projekte von unterschiedlichen Gruppen gelistet, wodurch zu erwarten ist, dass die Forschung an Lustre auch in Zukunft durch verschiedene Gruppen vorangetrieben wird. Ein aktuelles Forschungsprojekt ist zum Beispiel, die Komprimierung von Dateien, welche im Dateisystem gespeichert werden. So könnten auch unabhängig vom Hinzufügen von weiteren oder größeren Speichermedien, mehr Dateien gespeichert werden. Außerdem würden die Zeiten, welche für die Bearbeitung von Anfragen im Dateisystem benötigt werden, durch die kleineren Dateigrößen, minimiert werden. [6]

[5] Intel; „Lustre File Striping“: https://www.intel.com/content/dam/www/public/us/en/documents/training/lustre-file-striping.pdf

[6] Lustre: http://wiki.lustre.org/Projects

6. FAZIT

Lustre hat durch die Open-Source-Lizenz und durch die damit verbundene Weiterentwicklung von vielen unabhängigen Gruppen, eine vielversprechende Zukunft. Durch die Failover-Konfiguration wird eine hohe Verfügbarkeit garantiert und durch den parallelen Zugriff und das Striping der Dateien wird eine erhöhte Kapazität und ein erhöhter Durchsatz ermöglicht.

Quellenverzeichnis

Farber, Rob; „Lustre to DAOS: Machine Learning on Intel's Platform"; 23.05.2016: https://www.nextplatform.com/2016/05/23/lustre-daos-machine-learning-intels-platform/

Intel; „Lustre File Striping": https://www.intel.com/content/dam/www/public/us/en/documents/training/lustre-file-striping.pdf

Kuhn, Michael; „Parallele verteilte Dateisysteme - Hochleistungs-Ein-/Ausgabe"; 11.05.2015: https://wr.informatik.uni-hamburg.de/_media/teaching/sommersemester_2015/hea-15-parallele_verteilte_dateisysteme.pdf

Lustre: http://wiki.lustre.org/Projects

Lustre; "Introduction to Lustre Architecture"; Oktober 2017: http://wiki.lustre.org/images/6/64/LustreArchitecture-v4.pdf

Oracle; Intel Corporation; "Lustre Software Release 2.x - Operations Manual"; 2017: http://doc.lustre.org/lustre_manual.xhtml

Pelzer, Martin; Tersteegen, Hanno; „Lustre – ein High-Performance-Dateisystem"; 2005: http://berrendorf.inf.h-brs.de/lehre/ss05/parsys/s_Lustre.pdf

Riehm, Benedikt Johannes; „Das parallele Dateisystem Lustre" http://www.scc.kit.edu/scc/docs/Lustre/Riehm-Lustre.pdf

sysGen: https://www.sysgen.de/lustre-parallel-filesystem.html

Tennert, Oliver; Kobras, Daniel; „Das verteilte Dateisystem Lustre"; Linux-Magazin 11/2007: http://www.linux-magazin.de/ausgaben/2007/11/need-forspeed/7/

Wikipedia: https://en.wikipedia.org/wiki/Lustre_(file_system)

Wikipedia: https://en.wikipedia.org/wiki/Round-robin_scheduling

Letzter Aufruf der Webseiten jeweils am 26.02.2019

Abbildungen:

Lustre; “Introduction to Lustre Architecture”; Oktober 2017: http://wiki.lustre.org/images/6/64/LustreArchitecture-v4.pdf Seite 5 [1,4]

Oracle; Intel Corporation; “Lustre Software Release 2.x - Operations Manual”; 2017: http://doc.lustre.org/lustre_manual.xhtml [2,3]

Letzter Aufruf der Webseiten jeweils am 26.02.2019

ERKLÄRUNG

Ich versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt habe und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.