Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

FAKULTÄT
FÜR MATHEMATIK, INFORMATIK
UND NATURWISSENSCHAFTEN

Seminararbeit: Proseminar Speicher- und Dateisysteme

Georg Leander Siemund

georg.siemund@student.uni-hamburg.de
Studiengang Wirtschaftsinformatik
Matr.-Nr. 7051313
Fachsemester 3

Betreuer: Dr. Michael Kuhn

Abgabe: 03.2019

A distributed system is one where the failure of some
computer I've never heard of can keep me from getting my work done.
– *Leslie Lamport*

Inhaltsverzeichnis

1 Einleitung

Distributed Asynchronous Object Storage (DAOS) ist ein ganzheitliches, objektbasiertes Speichersystem für die parallelen, verteilten Exascale-Hochleistungssysteme der Zukunft. Ein Ziel dieses Speichersystems ist die optimale Nutzung des neuartigen NVRAMs; einem nichtvolatilen Speichermedium, das eine ähnlich geringe Latenz hat wie volatiler RAM. Heutige Speichersysteme benötigen viel Zeit, um Daten zu persistieren, was bei der vergleichsweise hohen Latenz der genutzten Speichermedien kein Problem darstellt. Bei Exascale-Systemen dagegen, die NVRAM als schnellen, persistenten Speicher nahe der Rechenknoten nutzen und ein Vielfaches der IOPS haben, nähme der heutige Softwarestack eine signifikante Zeit in Anspruch und würde das gesamte System verlangsamen. Um das Leistungspotenzial von NVRAM nutzen zu können, muss der Softwarestack, der zwischen persistentem Speichermedium und den Anwendungen liegt, überdacht werden. [2]

1.1 Motivation

Noch existieren keine Supercomputer, die eine Rechenleistung im Exaflopbereich besitzen. Doch das könnte sich bald ändern. Das Exascale Computing Project (ECP) zum Beispiel, ein Zusammenschluss des US Office of Science und der US National Nuclear Security Administration, hat es sich als Ziel gesetzt bis 2021 ein Exascale-System zu erbauen. Ein solches System könne zu neuen Erkenntnissen in der Wissenschaft führen, dadurch dass enorme Datenmengen in weniger Zeit analysiert und genauere Simulationen durchgeführt werden können (Exascale Computing Project - Exascale Computing Project, o.D.). Doch die Nutzung eines solchen Systems bringt Herausforderungen für Wissenschaftler, Ingenieure und Programmierer mit sich. Die gewaltigen Datenmengen und dessen Metadaten müssen in jedem Fall konsistent bleiben. Darüber hinaus verbessern sich die Simulationen und werden komplexer, weshalb HPC-Softwareentwickler und Wissenschaftler ihre Modelle überdenken müssen. [1]

1.2 Zielsetzung

Das Ziel der Arbeit ist es, herauszufinden, welche Vorteile DAOS im Vergleich zu heutigen Speichersystemen hat. Ebenso will ich die Herausforderungen aufzeigen und bewerten, inwieweit DAOS sich für Exascale-Systeme eignet.

2 Architektur

2.1 Grundsätzlicher Aufbau eines Exascale-Systems

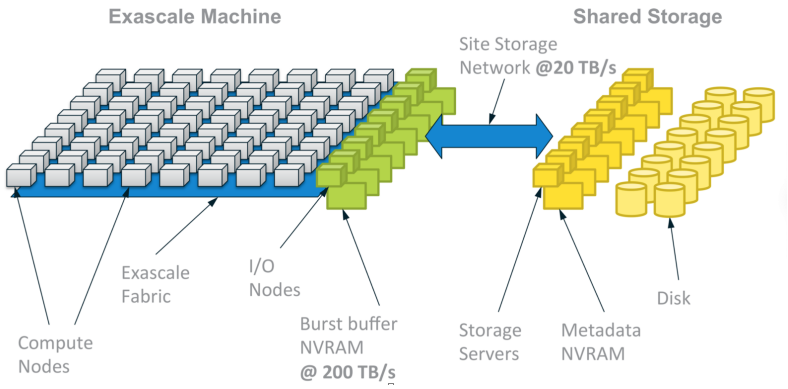

Abbildung 2.1: Aufbau eines Exascale-Computers [2]

Das Modell des Exascale-Systems, auf dem DAOS funktionieren soll, teilt sich in zwei Komponenten auf: Die rechnende Komponente (linke Seite in Abb. 2.1) und die Speicherkomponente (rechte Seite in Abb. 2.1). Die Komponenten sind direkt miteinander Verbunden, sodass der Rechner jederzeit auf Daten aus der entfernten Speicherkomponente zugreifen bzw. nicht weiter benötigte Daten dort ablegen kann. [2]

Die rechnende Komponente teilt sich auf in Rechenknoten (graue Quader in Abb. 2.1) und Eingabe-/ Ausgabeknoten bzw. den Burst Buffer (grüne Quader in Abb. 2.1). Diese grünen Quader sind mit den Rechenknoten gekoppelt. Die Compute Nodes können von diesem Speicher schnell Daten auslesen und voraussichtlich in naher Zukunft benötigte Daten ablegen. Dieser Speicher dient außerdem als Burst Buffer, kann also großen Datenmengen absorbieren und diese dann Stück für Stück in die Rechenknoten einspeisen. Der Burst Buffer entlastet die Rechenknoten in dem Fall, dass mehr Daten auf einmal zur Berechnung zur Verfügung gestellt werden, als von den Compute Nodes verarbeitet werden können. In den Eingabe-/ Ausgabeknoten wird hauptsächlich NVRAM eingesetzt. [2]

Die langsamere Speicherkomponente des Systems besteht aus Speichermedien mit großer Speicherkapazität und einer höheren Latenz. Zugriff von einem Rechenknoten auf die dort gespeicherten Daten dauert länger, weil die Daten erst einmal gelesen und dann in

den Burst Buffer geladen werden müssen. Deshalb sollten oft benutzte Daten möglichst nicht in der langsameren Speicherkomponente des Systems abgelegt werden. Die folgende Grafik veranschaulicht die Aufteilung der Daten in 3 Kategorien:

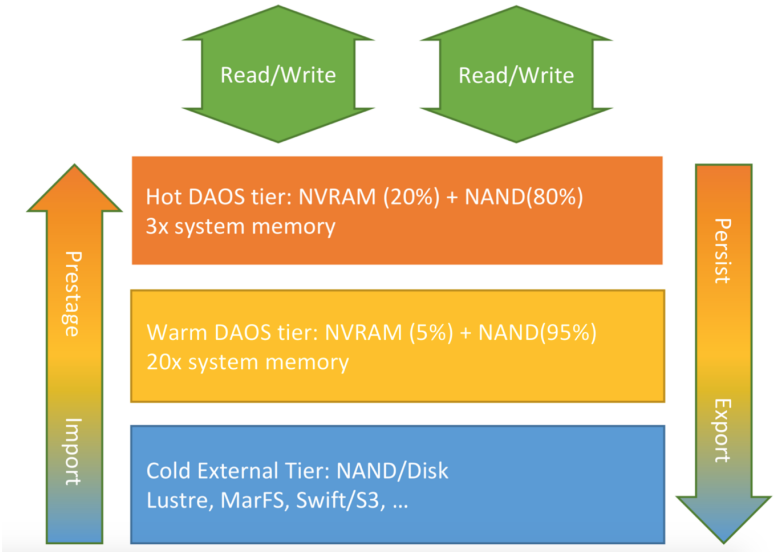

Abbildung 2.2: Kategorisierung der Daten in DAOS [5]

Das Hot DAOS Tier beinhaltet die Daten, die direkt an den Rechenknoten im Burst Buffer für eine schnelle Benutzung gespeichert sind. Dieser Speicher ist der kleinste, persistente Speicher bevor die Daten in den volatilen Hauptspeicher geladen werden. Er besteht zu 20% aus NVRAM und zu 80% aus schnellem NAND-Speicher. Dieses Speichertier nur etwa 3x so groß wie der Hauptspeicher.
Das Warm DAOS Tier liegt nicht direkt an den Rechenknoten, stellt dafür aber eine Speichereinheit mit etwa der 20-fachen Größe des Hauptspeichers bereit. Diese Einheit besteht zu 5% aus NVRAM und zu 95% aus NAND-Speicher.
Zuletzt gibt es das Cold DAOS Tier. Hier werden große Datenmengen, die von den Rechenknoten nur selten benutzt werden, langfristig und kostengünstig gespeichert. [5]

Diese Einteilung der Daten hat den Hintergrund, dass aus ökonomischen Gründen nicht der gesamte persistente Speicher aus NVRAM bestehen kann. Neben anderen Faktoren muss also der Tradeoff zwischen Kosten, Latenzvorteilen und Speicherkapazität abgewogen werden.

2.2 Objektbasierter Speicher

Die in Kapitel 2.1 behandelte Kategorisierung der Daten wird durch die objektbasierte Speicherung unterstützt, da dies die Analyse und Einteilung vereinfacht. Im Folgenden will ich kurz Dateisysteme und objektbasierte Speicherung der Daten gegenüberstellen und weitere Vorteile der objektbasierten Speicherung für ein Exascale-System nennen.

Im Gegensatz zu Dateisystemen existieren im objektbasierten Speichermodell keine Dateien, in denen Daten gespeichert sind. Die Objekte selbst sind die Daten, inklusive der jeweiligen Metadaten. Objekte liegen außerdem auf der gleichen Hierarchieebene, während Dateissysteme hierarchisch aufgebaut sind. [2]

Um Objekte trotz der fehlenden Struktur aufzufinden, können beliebig viele Metadaten in den Objekten gespeichert werden. Das ist ein entscheidender Vorteil von objektbasierter Speicherung gegenüber Dateisystemen in Exascale-Systemen. Die große Menge an Metadaten vereinfacht die Suche der Daten.

2.3 NVRAM

Non-Volatile Random-Access Memory (NVRAM), eine besondere Art des Random-Access Memorys, speichert Daten persistent, also ohne dass durchgängig eine Stromzufuhr benötigt wird. Dies steht im Gegensatz zu dem bekannteren DRAM, der Daten nur so lange speichert, wie eine Stromzufuhr vorhanden ist. NVRAM vereinigt die Vorteile des Random-Access Memorys und persistenter Speichermedien. [8]

Random-Access Memory hat den Vorteil, dass es im Vergleich mit den meisten persistenten Speichermedien eine sehr geringe Latenz hat. Darüber hinaus, bietet Random-Access Memory byte-granularen Zugriff auf die Daten. Bei anderen persistenten Speichermedien müssen immer Datenblöcke ausgelesen werden, die meistens größer sind als die tatsächlich benötigten Daten. Byte-Granularität bedeutet, dass diese Blöcke sehr viel granularer sind, was die zum Auslesen benötigte Zeit verringert. [8]

Ein spezifisches Beispiel für NVRAM ist die von Intel entwickelte 3D-XPoint Technologie. Den Einsatz dieser Technologie gucken wir uns im nächsten Beispiel an.

2.4 Latenzvorteil des NVRAMs

Die Grafik 2.3 veranschaulicht den Latenzunterschied zwischen 3 verschiedenen Speichermedien. Die y-Achse steht für die Latenz in Mikrosekunden, also die Zeit, die es ab der Datenabfrage von einer Anwendung dauert, bis ein Datenblock von dem jeweiligen Speichermedium ausgelesen wurde. Auf der x-Achse sehen wir die 3 Speichermedien, die wir vergleichen wollen:

1) Eine moderne SSD

2) Die Intel 3D XPoint Technologie, wie eine Festplatte peripher angeschlossen

3) Die Intel 3D XPoint Technologie, wie Hauptspeicher in der Zentraleinheit verbaut

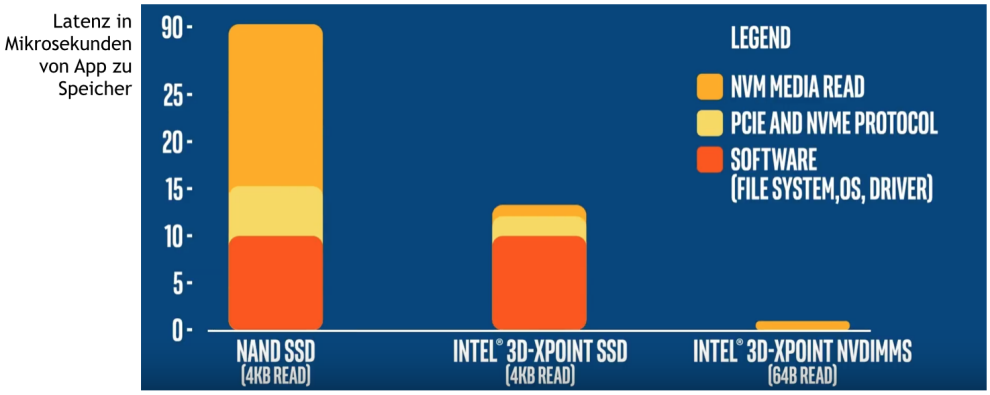

Abbildung 2.3: Latenzvergleich zwischen 3 Speichermedien [4]

Der gesamte Balken eines Speichermediums in der Grafik beschreibt die Zeit, die das Speichermedium benötigt, bis die von einer Applikation angefragten Daten ausgelesen wurden. Der orangefarbene Anteil des jeweiligen Balkens stellt die Zeit dar, die das Speichermedium zum tatsächlichen Auslesen der Daten benötigt. Der restliche Anteil des Balkens stellt die sonstige Verzögerung dar, verursacht durch das Dateisystem, das Betriebssystem, Treiber und die Protokolle. Es ist zu beachten, dass die y-Achse zwischen 25ms und 90ms einen Sprung macht. Bei Nand SSDs dauert also das tatsächliche Auslesen der Daten länger, als der erste Eindruck der Grafik vermittelt. [3][6]

Man sieht, dass die NAND SSD insgesamt am längsten braucht, um von einer Applikation angefragte Daten auszulesen, nämlich über 90 Mikrosekunden. Dabei wird die meiste Zeit mit dem tatsächlichen Auslesen der Daten von dem Speichermedium aufgebraucht. Das wird durch den großen Anteil des orangefarbenen Balkens an der Gesamthöhe des Balkens für das Speichermedium veranschaulicht. Würde man hier z. B. den Treiber oder das Dateisystem optimieren und beschleunigen, wäre der Unterschied auf die Gesamtlatenz kaum bemerkbar. Das liegt daran, dass Treiber bzw. Dateisystem auch vorher nur einen geringfügigen Anteil an der Gesamtlatenz hatten. [3][6]

Die Intel 3D XPoint SSD wird ebenfalls wie eine Festplatte peripher angeschlossen. Der Unterschied ist, dass diese SSD auf der 3D XPoint Technologie basiert, die um ein Vielfaches schnellere Zugriffszeiten auf die Daten verspricht. Hier teilt sich die Latenz zwischen dem Softwarestack und dem tatsächlichen Auslesen der Daten anders auf. Da das Auslesen der Daten um ein vielfaches schneller geht und der Anteil der durch Software und Protokoll verursachten Latenz höher ist, sticht die von Software und Protokoll verursachte Latenz mehr heraus. In dem Fall ist es durchaus so, dass ein Optimieren von Treiber oder Dateisystem einen merklichen Unterschied in der Gesamtlatenz macht. [3][6]

Im dritten Fall nutzen wir die 3D Xpoint Technologie wie volatilen Arbeitsspeicher.

Der entscheidende Unterschied ist, dass der Arbeitsspeicher nun persistent ist. Man sieht, dass in diesem Fall die durch Dateisystem, Betriebssystem, Treiber und Protokolle verursachte Latenz vollständig entfällt. Die Gesamtlatenz besteht also nur aus der Zeit, die für das tatsächliche Auslesen der Daten auf dem Speichermedium anfällt. Zusätzlich kann Arbeitsspeicher granularer ausgelesen werden. Während der periphere Speicher unabhängig von der tatsächliche benötigten Datenmenge nur Blockweise (oft in 4096-Byte-Blöcken, wie in Grafik 2.4) ausgelesen werden kann, ist bei Arbeitsspeicher sogenanntes byte-granulares auslesen möglich. Offensichtlich bietet diese Methode also die mit Abstand geringste Latenz, da die durch Software und Protokoll verursachte Latenz entfällt. [3][6]

2.4.1 Wieso die durch Software und Protokoll verursachte Latenz entfällt

In Kapitel 2.4 habe ich den Vorteil davon erläutert, die persistente 3D-XPoint Technologie als Arbeitsspeicher zu benutzen. Nun will ich behandeln, wieso die durch Software und Protokoll verursachte Latenz dadurch entfällt.

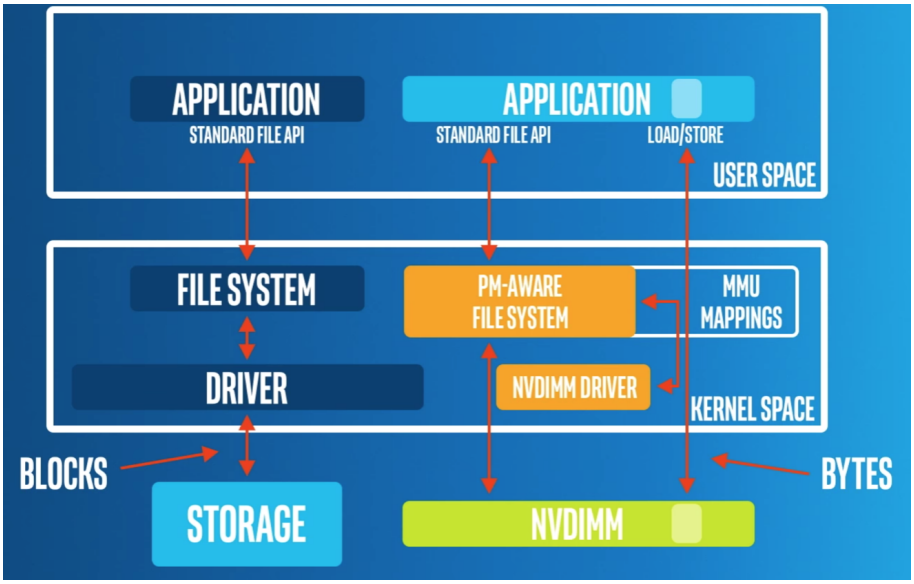

Abbildung 2.4: Wieso die Latenz entfällt [6]

Die Grafik zeigt auf der linken Seite, welche Vorgänge in Gang gesetzt werden, wenn eine Anwendung Daten persistent auf die Festplatte schreiben möchte. Die rechte Seite beschreibt die Vorgänge bei Lade- und Schreibvorgängen bei der Nutzung von Random Access Memeory bzw. Non-Volatile Random Access Memeory.

Will eine Anwendung auf die Festplatte schreiben, nutzt diese eine Filesystem-API, um die jeweiligen Files zu öffnen. Dann kann die Applikation die Files lesen und bearbeiten. Die Änderungen werden allerdings erst einmal nur im Arbeitsspeicher gehalten, der heutzutage nicht persistent ist. Die Daten müssen deshalb im nächsten Schritt in den persistenten Speicher, die Festplatte, geladen werden, damit Konsistenz der Daten ge-

währt ist. [6]

Damit auf die Festplatte geschrieben werden kann, muss erst einmal der Softwarestack in Abb. 2.4.1 durchlaufen werden, d. h. jedes laden und schreiben läuft über das Betriebssystem. Dieser Umweg fügt jedem Schreib- und Lesevorgang einige Millisekunden Latenz hinzu. Wie in Kapitel 2.4 erläutert, macht dieser Umweg selbst bei heutigen SSDs kaum einen merklichen Unterschied, weil das tatsächliche Auslesen der Daten auf dem Speichermedium ein Vielfaches der Zeit kostet. Bei dem im gleichen Kapitel vorgestellten NVRAM macht dieser Umweg allerdings einen Großteil der Gesamtlatenz aus. [6]

Der große Unterschied, den persistenter RAM macht, ist dass die Daten bereits im Arbeitsspeicher persistent sind. Da Applikationen direkt mit dem Arbeitsspeicher arbeiten, wird der Umweg und die durch das Betriebssystem verursachte Latenz umgangen. NVRAM ist schnell genug, um so nahe am Prozessor zu funktionieren. Darüber hinaus ist er byte-addressable, d. h. Daten werden granularer ausgelesen. [6]

Das schließt die Nutzung von volatilem RAM jedoch nicht aus. Dieser könnte als ein weiteres Cache-Level fungieren. Sehr oft genutzte Daten, die nicht unbedingt direkt persistiert werden müssen, können darin gespeichert werden.

2.5 Speichermodell

Dieser Abschnitt befasst sich mit der logischen Speicherung der Daten in DAOS. Wie die Bezeichnung des Speichersystems bereits aussagt (Distributed Asynchronous Object Storage), handelt es sich um Speicherung der Daten als Objekte. Das Speichermodell ähnelt eher einer Datenbank als einem Dateisystem.

Auf der untersten Ebene befinden sich die Objekte, bestehend aus key-value stores, mehrdimensionalen Arrays und Blobs. Blob steht für Binary Large Object und bezeichnet eine Menge binärer Daten, typischerweise Multimediadaten wie Bilder oder Audio. Key-value stores sind ein sehr einfaches Datenmodell, bei dem jeder Key genau einem Wert zugeordnet ist. Der Key ist meistens ein String, zum Beispiel ein Hashwert. Über diesen Key können die damit assoziierten Daten dann gefunden und ausgelesen werden. Mehrdimensionale Arrays sind Arrays mit beliebig vielen Dimensionen, also ein Array von einem Array von einem Array etc. [2]

Auf der Ebene über Objekten stehen Container, das heißt mehrere zusammenhängende Objekte bilden einen Container. Das Konzept eines Containers in der Objektspeicherung ähnelt damit einer Datei in einem Dateisystem, da zusammenhängende Daten darin abgelegt werden. Der Unterschied zu Dateisystemen ist, dass Container alle in der gleichen

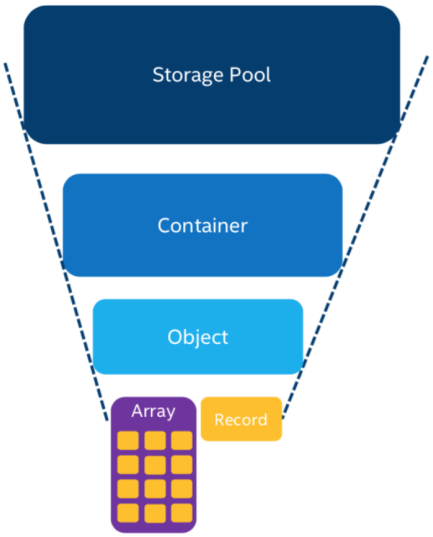

Abbildung 2.5: Speichermodell DAOS [5]

Hierarchieebene abgelegt sind. Das Konzept der Container ist für das Sicherstellen von der Konsistenz der Daten relevant (siehe. 2.6). Container wiederum sind in sogenannten Storage Pools organisiert. [2]

2.6 Versionierung und Asynchronität

Synchrone IOPS sind in Exascale-Systemen unrealistisch, weil das Warten auf den langsamsten Prozess das gesamte System ausbremst. In DAOS findet die Ein- und Ausgabe asynchron statt und Prozesse laufen unabhängig voneinander ab.

Eine große Herausforderung bei High Perfomance Computing im Exascale-Bereich ist, die Konsistenz der Daten bei hoher Parallelität sicherzustellen. Dafür ist es wichtig, dass es für Anwendungen einen global konsistenten Zustand gibt, falls ein Rollback durchgeführt werden muss. Container und Transaktionen bilden dabei die atomaren Einheiten, auf dessen Basis die Konsistenz festgestellt wird. Container bietet einen sogenannten ßingle access context". Das bedeutet, dass jedes Objekt nur über den jeweiligen Container angesprochen werden kann, in dem es liegt. Wir betrachten nun das Konzept der Transaktionen in DAOS. [2]

Transaktionen sind nummerierte Sammlungen von Prozessen, die auf einen Container angewendet werden und diesen verändern. Es kann mehrere Prozesse in einer Transaktion geben, die den jeweiligen Container durch den ßingle access contextmodifizieren. Das Hinzufügen/ Entfernen von Daten und modifizieren eines Containers wird nicht direkt auf einem Container ausgeführt, sondern in einer Transaktion zusammengefügt. Sobald diese Transaktion committed ist, werden die Veränderungen auf den Container

angewendet und persistiert. Dann ist die Transaktion abgeschlossen. Auf einem Container können gleichzeitig mehrere Prozesse laufen. [2]

Ein mit den Transaktionen verwandtes Konzept sind in DAOS Epochen. Eine Epoche repräsentiert die persistenten Änderungen auf Container. Das Konzept der Epochen existiert, weil nicht alle Transaktionen persistiert, sondern einige verworfen werden. Epochen übernehmen die ID der jeweiligen Transaktion, die zur letzten persistenten Änderung geführt hat. Sie sind demnach global konsistente Snapshots. [2]

2.7 Kompatibilität

2.7.1 HDF5

HDF5 ist ein wissenschaftliches Dateiformat, das in wissenschaftlichen Anwendungen mit großen Datenmengen genutzt wird. Eine Datei in dem Format besteht aus 2 Objekten. Erstens aus Datensätzen, die mehrdimensionale Arrays sind. Zweitens aus Gruppen, die aus Datensätzen und aus weiteren Gruppen besteht. Dadurch ist das Format hierarchisch und ähnelt einem Dateisystem. Die Besonderheit an dem HDF5-Format ist, dass Dateien beliebig groß sein können. Darüber hinaus sorgt die interne Struktur der Dateien dafür, dass Datensätze in beliebig großen Dateien gefunden werden können. Das macht HDF5 sehr viel effizienter als zum Beispiel das Speichern in Textdateien.[7]

HDF5 ist ein Standardformat, das in einer Vielzahl von wissenschaftlichen und betriebswirtschaftlichen Anwendungen genutzt wird. Damit diese und ähnliche Anwendungen mit DAOS weiterhin funktionieren, muss die Kompatibilität zwischen DAOS und HDF5 sichergestellt werden, damit Anwendungen wie gewohnt ihre Daten im HDF5-Format ablegen können. Wie in Kapitel 2.5 bereits erläutert, ähnelt das Konzept eines Containers dem einer HDF5-Datei, weil sowohl Container als auch Dateien zusammenhängende Daten beinhalten. Dadurch können HDF5-Dateien auf DAOS-Container gemappt werden und Anwendungen werden weiter unterstützt. [2]

Die Grafik 2.7.1 vergleicht ein heutzutage standardmäßiges HPC-System mit einem zukünftigen DAOS Exascale-HPC System und illustriert Unterschiede zwischen beiden Systemen. Die durchgestrichenen Elemente sind heutige Elemente und Standards. Daneben stehen dann gegebenenfalls die Elemente, durch die heutige Elemente und Standards ersetzt werden.

Die Grafik zeigt, dass heutzutage Anwendungen im Userspace laufen. Über weitere Softwareschichten arbeiten die Anwendungen auf Basis eines parallelen Dateisystems (PFS) und persistieren Daten auf HDD Festplatten. Das PFS läuft allerdings nicht im Userspace, was einen ständigen Umweg über das Betriebssystem erfordert, um Daten

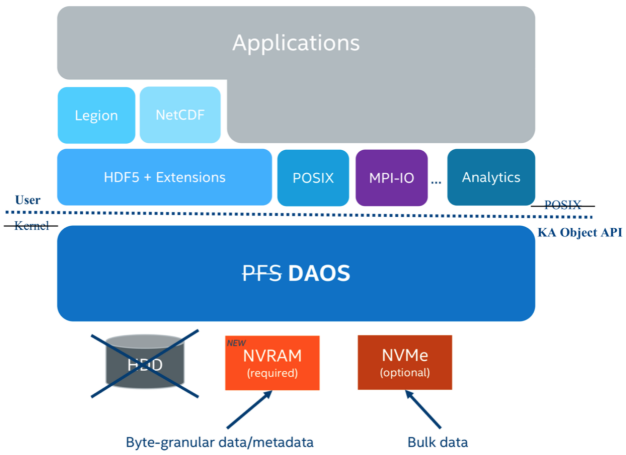

Abbildung 2.6: Mapping [5]

zu persistieren. Das wurde bereits ausführlich im Kapitel 2.4.1 ausführlich behandelt. DAOS eliminiert diesen Umweg mit dem Einführen von NVRAM als Speichermedium für oft genutzte Daten und den Burst Buffer. Die Notwendigkeit des Kernels entfällt also für das persistente Speichern von Daten. [6]

3 Bewertung

Distributed Asynchronous Object Storage (DAOS), ein ganzheitliches, objektbasiertes Speichersystem für die parallelen, verteilten Exascale-Hochleistungssysteme der Zukunft ist ein essenzieller Schritt für die Weiterentwicklung von High Performance Computing-Systemen. Im Folgenden möchte ich Vorteile und Herausforderungen der Architektur gegenüberstellen.

3.1 Vorteile von DAOS

Ein Vorteil der DAOS-Architektur ist die weitaus bessere Performance bzw. das Performancepotential durch die prozessornahe Speicherung auf Nicht-Volatilem RAM. Solche Systeme können zu neuen Erkenntnissen in der Wissenschaft führen, dadurch dass enorme Datenmengen in weniger Zeit analysiert und genauere Simulationen durchgeführt werden können. Ein weiterer Vorteil ist, dass trotz des veränderten Softwarestacks Applikationen nicht verändert werden müssen, um auf DAOS zu funktionieren.

3.2 Herausforderungen bei DAOS

Eine der größten Herausforderungen bei DAOS ist die richtige Nutzung des NVRAMs. Es müssen die hohen Kosten und Nutzen des NVRAMs abgewägt werden. Dies wird sehr ausschlaggebend für die Performance des Systems sein. Eine weitere Herausforderung ist, dass DAOS sich noch nicht in der Praxis bewährt hat, sondern ein Forschungsprojekt ist. Es ist außerdem schwierig zu zeigen, dass DAOS tatsächlich auf Exascale-Ebene funktionieren wird, da es solche Systeme noch nicht gibt. Es bleibt also offen, wie gut DAOS wirklich in der Praxis funktioniert.

4 Literatur

[1] Breitenfeld, M. Scot, et al. "DAOS for extreme-scale systems in scientific applications.ärXiv preprint arXiv:1712.00423 (2017).

[2] Lofstead, Jay, et al. "DAOS and friends: a proposal for an exascale storage system.SSC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2016.

[3] Foong, Annie, and Frank Hady. SStorage as fast as rest of the system."2016 IEEE 8th International Memory Workshop (IMW). IEEE, 2016.

[4] Intel Software, I. N. T. E. L. (2018, 29. Juni). High-Performance Software Defined Storage Over Persistent Memory. DAOS. Intel Software [Video]. Abgerufen 31. März, 2019, von $https://www.youtube.com/watch?v=bpXhH_jUGss$

[5] Dr. Micheal Kuhn. Zukünftige Entwicklung Hochleistungs-Ein-/Ausgabe. Abgerufen 31.März, 2019, von $https://wr.informatik.uni-hamburg.de/_media/teaching/sommersemester_2018/h18-zukuenftige_entwicklungen.pdf$

[6]Intel Software, A. N. D. Y. R. U. D. O. F. F. (2017, 13. Juli). What is Persistent Memory: Persistent Memory Programming Series. Intel Software [Video]. Abgerufen 31. März, 2019, von $https://www.youtube.com/watch?v=E2KYqdyZcQY$

[7]HDF Home - The HDF Group [Video]. (o.D.). Abgerufen 31. März, 2019, von $https://www.hdfgroup.org/$

[8]Wikipedia contributors. (2019, 25. März). Non-volatile random-access memory - Wikipedia [Video]. Abgerufen 31. März, 2019, von $https://en.wikipedia.org/wiki/Non-volatile_random-access_memory$