

Zukünftige Entwicklungen

Hochleistungs-Ein-/Ausgabe

Michael Kuhn

Wissenschaftliches Rechnen
Fachbereich Informatik
Universität Hamburg

2017-06-30



Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

1 Zukünftige Entwicklungen

- Motivation
- Hardware
- Software
- Zusammenfassung

2 Quellen

Burst Buffer...

- Hohe Kosteneinsparungen möglich
 - E/A-System muss nicht mehr für Lastspitzen ausgelegt werden
 - Dadurch ist u. U. ein kleineres Speichersystem ausreichend
- Eventuell können langsamere Technologien genutzt werden
 - Ethernet statt InfiniBand
 - 5.400 RPM statt 7.200 RPM
- Insgesamt erhöht sich die Ausnutzung der Hardware
 - Burst Buffer absorbieren auch ungünstige E/A problemlos
 - E/A wird „aufbereitet“ an das Speichersystem weitergegeben

Beschleuniger

- Zusätzliche Berechnungen teilweise aufwendig
 - Deduplikation, Kompression etc.
- GPUs ungeeignet, da PCIe-Bus die E/A beschränkt
 - Maximal ≈ 16 GB/s
- In Zukunft eventuell interessant
 - Seit Haswell QuickAssist mit DEFLATE und LZS
 - Xeon Phi in Zukunft gesockelt, direkt am RAM angebunden

Lustre [2]

- Das klassische parallele verteilte Dateisystem gilt als überholt
 - Teile davon sollen als Basis für zukünftige Systeme dienen
 - Außerdem momentan einzige verfügbare Lösung
- Für Lustre ist eine Vielzahl neuer Funktionen geplant
 - Verschlüsselung
 - Kompression
 - Komplexe Dateilayouts
 - Weitere E/A-Optimierungen

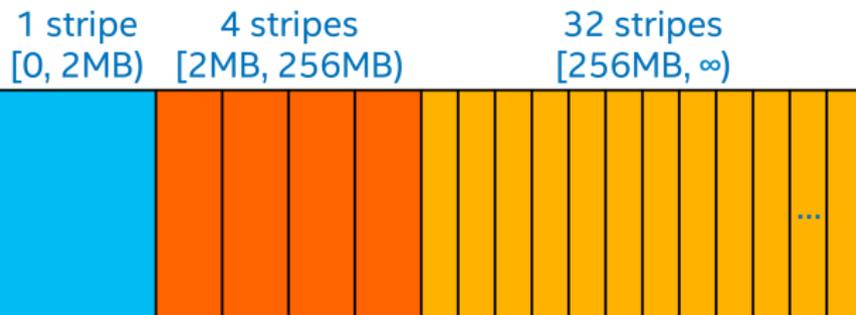
Lustre... [2]

- Verschlüsselung zunehmend wichtig
 - Regierungen, Militär etc.
- Mehrere Zugriffsstufen
 - Frei zugänglich, vertraulich, geheim, streng geheim
 - Kein Datentransfer zwischen unterschiedlichen Stufen
- Authentifizierung und Autorisierung
 - Z. B. via Kerberos
- Verschlüsselung der Kommunikation
 - Und in Zukunft auch der persistenten Daten

Lustre... [2]

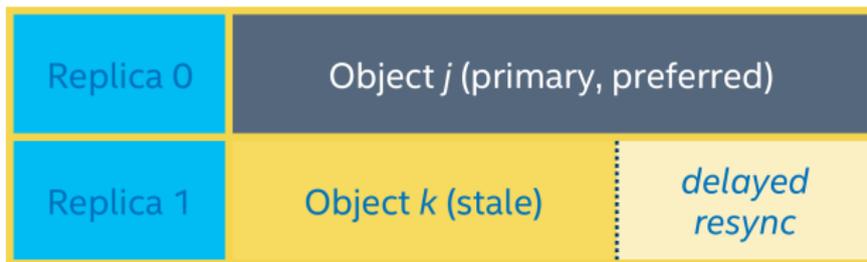
- Unterschiedliche Anforderungen an Dateiverteilung
 - Kleine Dateien nur auf wenige OSTs
 - Große Dateien auf möglichst viele
- Automatische Anpassung der Streifenparameter

Example progressive file layout with 3 components



Lustre... [2]

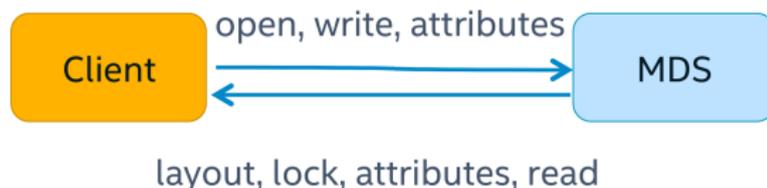
- Replikation von Dateien
 - Kann pro Datei festgelegt werden
 - High Availability, Robustheit, höhere Lesegeschwindigkeit, Migration zwischen Speicherklassen etc.



Overlapping (mirror) layout

Lustre... [2]

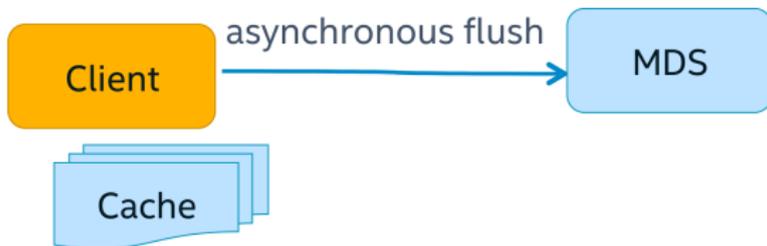
- Daten kleiner Dateien werden direkt auf dem MDT gespeichert
 - Spart Kommunikation mit OSTs
 - Erlaubt diverse Optimierungen wie z. B. Readahead
- MDTs sind üblicherweise für kleine Anfragen optimiert
 - Große Anfragen werden weiterhin über OSTs abgewickelt
- Daten müssen u. U. migriert werden, wenn Dateien wachsen



Small file IO directly to MDS

Lustre... [2]

- Metadatenoperationen sind typischerweise klein
 - Verursachen daher viel Netzwerk-Overhead
- Ein Cache erlaubt das Zusammenfassen mehrerer Operationen
 - Nur mit Sperren möglich, da sonst Konflikte auftreten können
 - Z. B. kann ein Verzeichnis gesperrt werden und darin viele Dateien angelegt werden



Client Metadata Operations Cache

POSIX

- POSIX stellt aber immer noch eine Limitierung dar
 - POSIX ist portabel, schränkt aber Leistung ein
- Einige Alternativen
 - MPI-IO meist aber doch auf POSIX
 - Häufig POSIX-Dateisysteme auf unterster Ebene
- Abkehr von POSIX
 - Object Stores oft ausreichend
 - Häufig Einschränkungen einiger Garantien

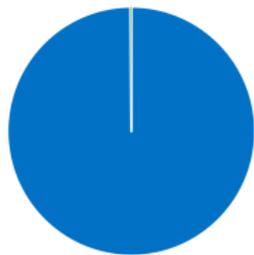
POSIX...

- Keine globale Kohärenz mehr
 - Stattdessen nicht-kohärente Zonen
 - Zum Beispiel durch Burst Buffer oder Forwarder
- Beispiel: Cache-Domains in BeeGFS
 - Anwendungen laufen in unterschiedlichen Domänen
 - Daten werden zuerst in nicht-kohärenten Cache geschrieben
 - Caches befinden sich beispielsweise auf lokalem NVRAM
 - Daten werden aus Cache ins Dateisystem migriert

DAOS

- Ganzheitlichere Ansätze in Arbeit
 - Distributed Application Object Storage (DAOS)
- DAOS unterstützt mehrere Speichermodelle
 - Arrays und Records als Basisobjekte
 - Objekt beinhalten Arrays und Records (Key-Array)
 - Container beinhalten Objekte
 - Storage Pools beinhalten Container
- Unterstützung für Versionierung
 - Operationen werden in Transaktionen durchgeführt
 - Transaktionen werden als Epochen persistiert
- Ausnutzung moderner Speichertechnologien

DAOS... [1]



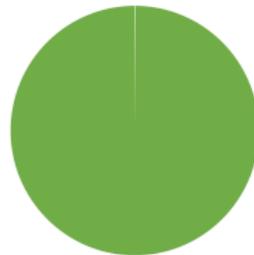
■ HDD ■ Software stack

msec vs. μ sec



■ NAND ■ Software stack

μ sec vs. μ sec



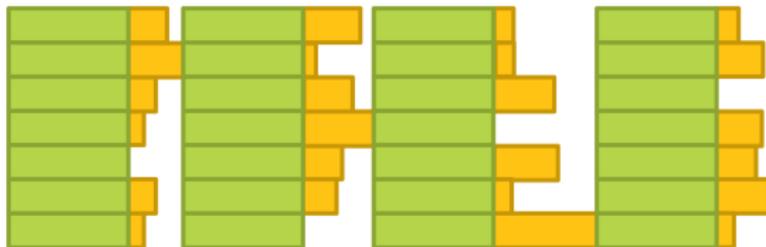
■ 3D XPoint™ ■ Software stack

nsec vs. μ sec

- E/A-Latenzen problematisch
 - Steigender Overhead durch Softwareschichten
- E/A-Granularität problematisch
 - Momentan 1 MiB, bald 16 MiB; zusätzliche Konflikte und Sperren
 - Bedingt durch darunterliegende Speichermedien
 - 3D XPoint bietet bytebasierten Zugriff

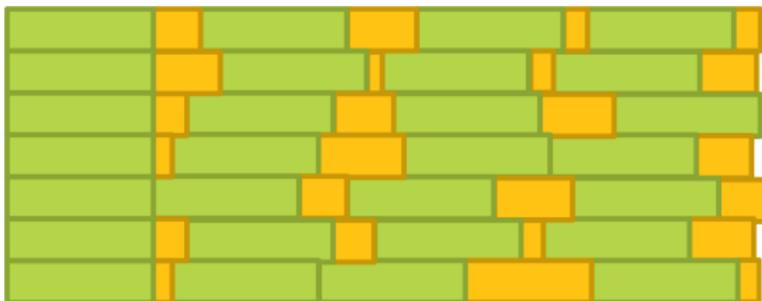
DAOS... [3]

- Üblicherweise synchrone E/A
 - Anwendungen werden durch den langsamsten ausgebremst
- Ähnliches Problem bei kollektiver Kommunikation
- E/A-Schwankungen sind der Normalfall
 - Speichersystem ist eine geteilte Ressource
 - Selten Quality of Service oder andere Garantien



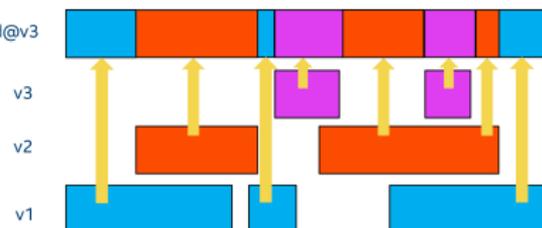
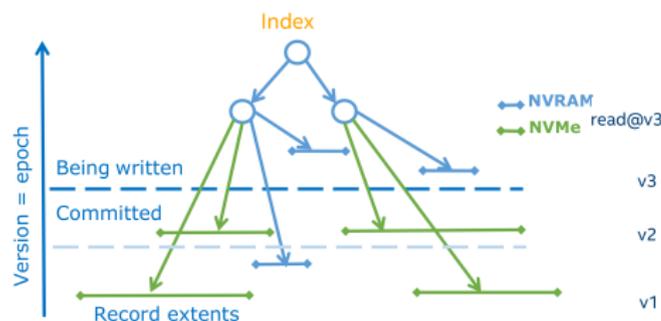
DAOS... [3]

- E/A sollte komplett asynchron stattfinden
 - Dadurch keine Wartezeiten mehr
- Frage: Wann ist die Datei konsistent?
 - Prozess könnte nächste Iteration schreiben während andere Prozesse noch mit der vorherigen beschäftigt sind



DAOS... [1]

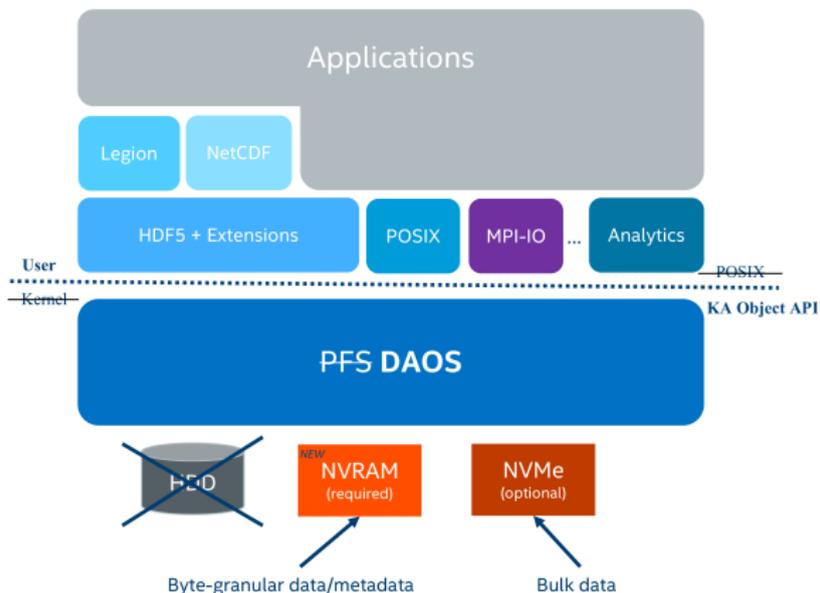
- Lösung: Transaktionen bzw. Epochen
 - Operationen finden in Transaktionen statt
 - Mehrere Transaktionen können zu einer Epoche zusammengefasst werden
- Epochen sind global konsistente Schnappschüsse
 - Epochen werden pro Objekt vergeben
 - Dadurch keine Kohärenzprobleme beim Lesen



DAOS... [3]

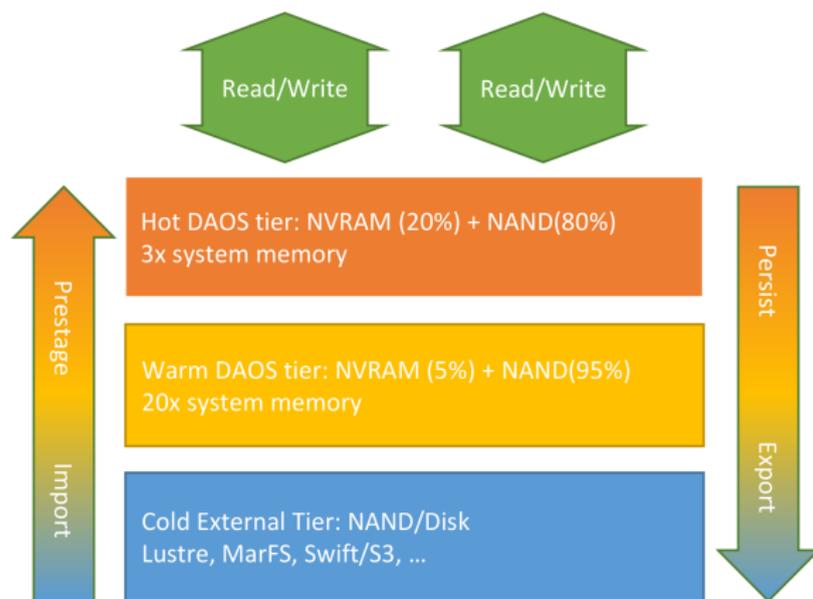
- Native Unterstützung für HDF5
 - Eine HDF5-Datei entspricht einem DAOS-Container
 - HDF-Datenstrukturen werden direkt in DAOS-Objekte überführt
 - Dadurch z. B. Umorganisation der Daten für effizienten Zugriff
- Unterstützung für unterschiedliche Anwendungen
 - POSIX und MPI-IO für Legacy-Anwendungen
 - Big-Data-Schnittstellen für MapReduce etc.
 - S3, NFS, Blockgeräte etc.

DAOS... [1]



- User-Space-Stack entwickelt für NVRAM und NVMe
 - Keine Kontextwechsel etc.

DAOS... [1]



- Prestage lädt Daten in den Cache, Persist speichert Daten
- Import/Export für HDD-basierte Speichersysteme

Big Data

- Big Data wird immer wichtiger
 - Häufig leistungstechnisch suboptimal
- Hadoop nutzt normalerweise HDFS
 - Daten werden auf lokale Speichergeräte kopiert
 - Kommunikation über HTTP
- Zunehmend Unterstützung für Big-Data-Anwendungen
 - Lustre, OrangeFS etc.

1 Zukünftige Entwicklungen

- Motivation
- Hardware
- Software
- Zusammenfassung

2 Quellen

Quellen I

- [1] Andreas Dilger. DAOS: Scale-out Object Storage for NVRAM.
<http://materials.dagstuhl.de/files/17/17202/17202.AndreasDilger.Slides.pdf>.
- [2] Brent Gorda. HPC Storage Futures – A 5-Year Outlook.
<http://lustre.ornl.gov/ecosystem-2016/documents/keynotes/Gorda-Intel-keynote.pdf>.
- [3] Brent Gorda. HPC Technologies for Big Data.
http://www.hpcadvisorycouncil.com/events/2013/Switzerland-Workshop/Presentations/Day_2/3_Intel.pdf.
- [4] DEEP-ER. DEEP-ER. <http://www.deep-er.eu/>.

Quellen II

- [5] Jian Huang, Karsten Schwan, and Moinuddin K. Qureshi. Nvram-aware logging in transaction systems. *Proc. VLDB Endow.*, 8(4):389–400, December 2014.
- [6] Jonas Bonér. Latency Numbers Every Programmer Should Know. <https://gist.github.com/jboner/2841832>.
- [7] N. Liu, J. Cope, P. Carns, C. Carothers, R. Ross, G. Grider, A. Crume, and C. Maltzahn. On the role of burst buffers in leadership-class storage systems. In *012 IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST)*, pages 1–11, April 2012.
- [8] Mike Vildibill. Advanced IO Architectures. <http://storageconference.us/2015/Presentations/Vildibill.pdf>.

Quellen III

[9] N/A. Latency Numbers Every Programmer Should Know.
<http://i.imgur.com/k0t1e.png>.