Zukünftige Entwicklungen

Hochleistungs-Ein-/Ausgabe

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Michael Kuhn

2019-06-25

Wissenschaftliches Rechnen
Fachbereich Informatik
Universität Hamburg

Zukünftige Entwicklungen

- Motivation
- Hardware
- Software
- Zusammenfassung

- Supercomputer werden immer leistungsfähiger
  - Speichersysteme werden immer größer
  - Höhere Anforderungen durch hohe Prozesszahlen
- Immer weiter wachsende Datenmengen
  - Supercomputer haben mehr als 1 PB RAM
  - Bei 1 TB/s mehr als 15 Minuten für einen Checkpoint
- Hardware wird immer schneller
  - Software muss Hardware effizient nutzen können
  - Traditionell schwergewichtige E/A-Stacks

Exascale...

- Häufig eine Datei pro Prozess
  - TaihuLight besitzt über 40.000 Knoten
    - Mehr als 10.000.000 Kerne
- POSIX immer noch weit verbreitet
  - Strikte Kohärenzanforderungen
  - Globale Sichtbarkeit aller Änderungen
- Traditionelles Dateisystemkonzept häufig über den Kernel
  - Andere Komponenten werden bereits ohne den Kernel abgewickelt
    - Z. B. InfiniBand mit Kernel Bypass

Speicherhierarchie

- Aktueller Stand
  - L1-, L2-, L3-Cache, RAM, SSD, HDD, Band
- Große Latenz-Lücke zwischen RAM und SSD
  - Hohe Leistungsverluste sobald Daten nicht im RAM liegen
  - Overhead durch das Netzwerk ist noch nicht eingerechnet
- Lücke ist auf Supercomputern besonders offensichtlich
  - Knotenlokal im RAM oder im parallelen verteilten Dateisystem
- Neue Technologien sollen die Lücke schließen
  - NVRAM, NVMe, 3D XPoint etc.
  - Dadurch eventuell auch zusätzliche Stufen im Supercomputer

Speicherhierarchie...

- Sehr niedrige Latenz zwischen CPU und Cache
  - Ein Taktzyklus entspricht ≈ 0,33–0,5 ns
  - RAM bereits deutlich langsamer

Stufe	Latenz
L1-Cache	≈ 1 ns
L2-Cache	≈ 5 ns
L3-Cache	≈ 10 ns
RAM	≈ 100 ns

Tabelle 1: Latenzen [8, 6]

Speicherhierarchie...

- Sehr niedrige Latenz zwischen CPU und Cache
  - Ein Taktzyklus entspricht ≈ 0,33–0,5 ns
  - RAM bereits deutlich langsamer
- Faktor von 1.000 zwischen RAM und SSD
  - In Supercomputern zusätzlich Netzwerklatenz

Stufe	Latenz
L1-Cache	≈ 1 ns
L2-Cache	≈ 5 ns
L3-Cache	≈ 10 ns
RAM	≈ 100 ns
SSD	≈ 100.000 ns
HDD	≈ 10.000.000 ns
Band	≈ 50.000.000.000 ns

Tabelle 1: Latenzen [8, 6]

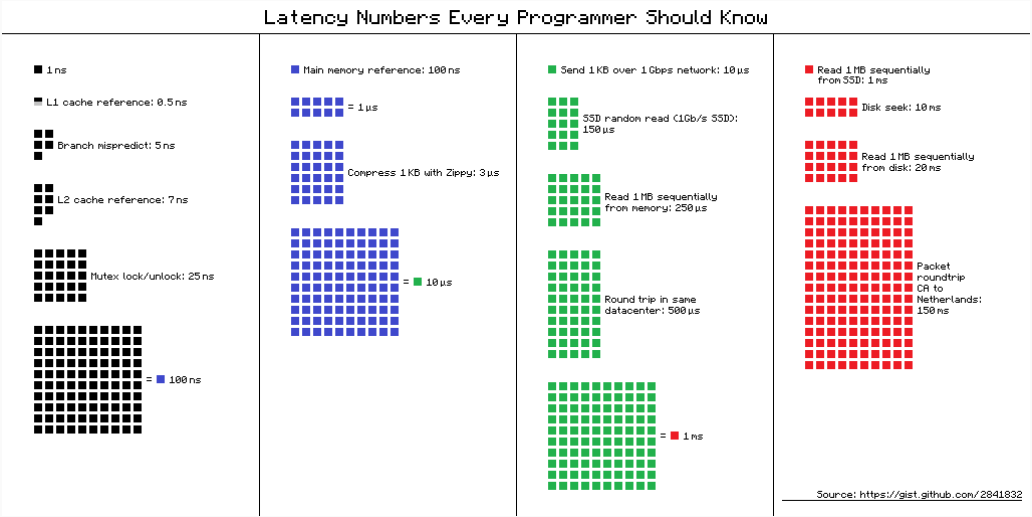
Speicherhierarchie...

- Sehr niedrige Latenz zwischen CPU und Cache
  - Ein Taktzyklus entspricht ≈ 0,33–0,5 ns
  - RAM bereits deutlich langsamer
- Faktor von 1.000 zwischen RAM und SSD
  - In Supercomputern zusätzlich Netzwerklatenz
- Zukünftig mehrere Zwischenstufen
  - Durch NVRAM außerdem neue Ansätze möglich

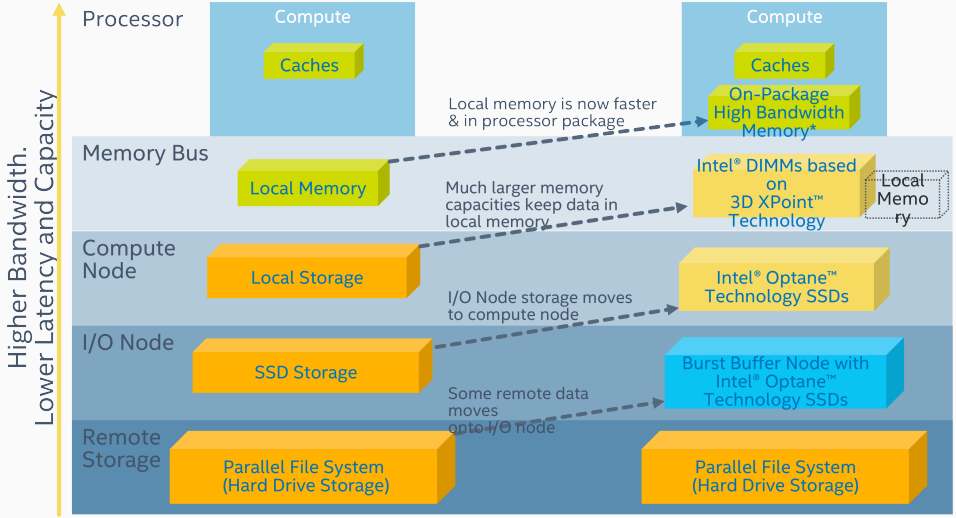
Stufe	Latenz
L1-Cache	≈ 1 ns
L2-Cache	≈ 5 ns
L3-Cache	≈ 10 ns
RAM	≈ 100 ns
NVRAM	≈ 1.000 ns
NVMe	≈ 10.000 ns
SSD	≈ 100.000 ns
HDD	≈ 10.000.000 ns
Band	≈ 50.000.000.000 ns

Tabelle 1: Latenzen [8, 6]

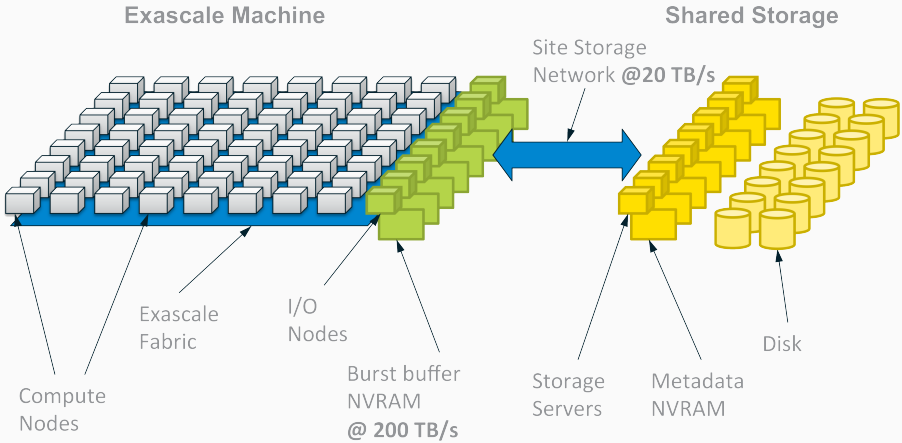
Speicherhierarchie... [8, 11]



Speicherhierarchie... [3]



Exascale [4]



- E/A-Knoten mit Burst Buffern nahe an Rechenknoten
- Langsameres Speichernetzwerk zu Dateisystem-Knoten

- E/A-Knoten können zusätzliche Aufgaben übernehmen
  - Auslagerung von Berechnungen und Transformationen
  - Umsortieren und Zusammenfassen der E/A-Anfragen
- Umorganisation der Daten
  - Z. B. explizites Speichern von Unterdatenmengen
  - Dafür aber Wissen über Datenformat notwendig
- Je näher an den Rechenknoten, desto mehr Rechenleistung
  - Andererseits müssen Daten zuerst übertragen werden
  - Bei schnellerer Hardware muss trotzdem auf Overhead geachtet werden

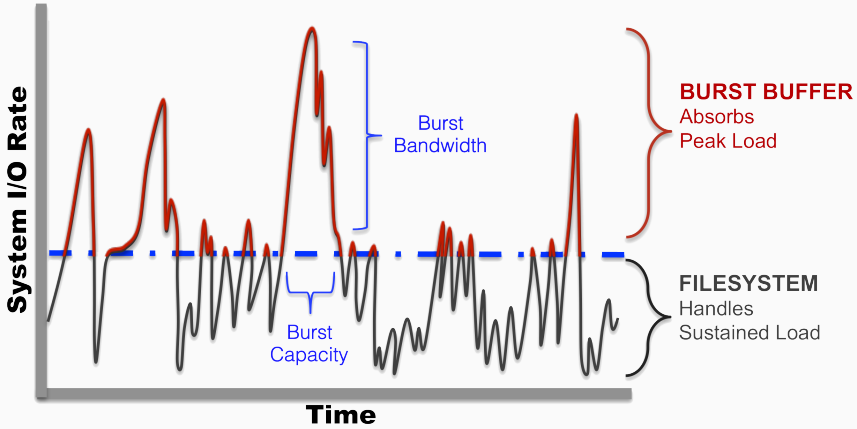
Burst Buffer

- E/A-Verhalten ist meist nicht gleichmäßig
  - Anwendungen rechnen, schreiben dann gemeinsam Checkpoint
  - Hohe E/A-Last während des Checkpoints, danach deutlich weniger
- Lastspitzen bremsen Anwendungen aus
  - Unter Umständen schreiben mehrere Anwendungen gleichzeitig
  - Speichersysteme oft unterdimensioniert
    - Beispiel Mistral: $\approx$ 20 TB/s (Rechenknoten) vs. $\approx$ 0,5 TB/s (Lustre)
- Hohe Kosten, um Durchsatz bereitzustellen
  - HDDs für Kapazität, SSDs für Durchsatz
  - Außerdem beschränkt durch das Netzwerk

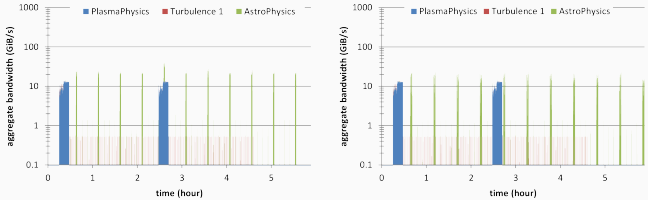
- Koordination zwischen Anwendungen nicht gegeben
  - Könnte genutzt werden, um Last auszugleichen
  - Ist allerdings auch schwierig zu realisieren
    - Benötigt Koordination von Anwendungen, Scheduler und Dateisystem
- Dateisystem ist geteilte Ressource
  - Alternative Ansätze mit Quality of Service für E/A
    - Anwendungen teilen dem Scheduler ihre E/A-Anforderungen mit

Analysis of a major HPC production storage system

- 99% of the time, storage BW utilization < 33% of max
- 70% of the time, storage BW utilization < 5% of max



Burst Buffer... [9]



(a) burst buffer turned off, full storage system in use, 128 file servers

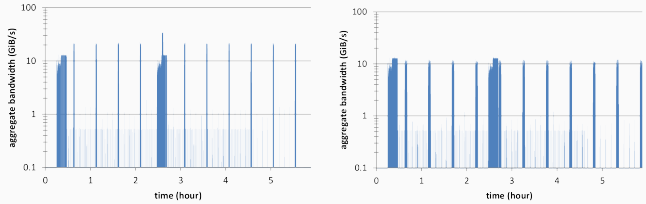
(b) burst buffer turned off, half storage system in use, 64 file servers

(c) burst buffer turned on, full storage system in use, 128 file servers

(d) burst buffer turned on, half storage system in use, 64 file servers

Fig. 5: Ten second average data transfer rate for the compute nodes observed during the multiapplication simulations.

Burst Buffer... [9]



(a) burst buffer turned off, full storage system in use, 128 file servers

(b) burst buffer turned off, half storage system in use, 64 file servers

(c) burst buffer turned on, full storage system in use, 128 file servers

(d) burst buffer turned on, half storage system in use, 64 file servers

Fig. 6: Ten second average data transfer rate for the external storage system observed during the multiapplication simulations.

- 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 (oder günstigeres InfiniBand)
  - Festplatten mit 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 häufig ungeeignet, da PCIe-Bus die E/A beschränkt
  - Maximal $\approx$ 16 GB/s (PCIe 3.0) bzw. $\approx$ 32 GB/s (PCIe 4.0)
- In Zukunft eventuell interessant
  - Seit Haswell QuickAssist mit DEFLATE und LZS
  - Xeon Phi und FPGAs gesockelt, direkt an RAM angebunden
  - PCIe 5.0 mit $\approx$ 64 GB/s

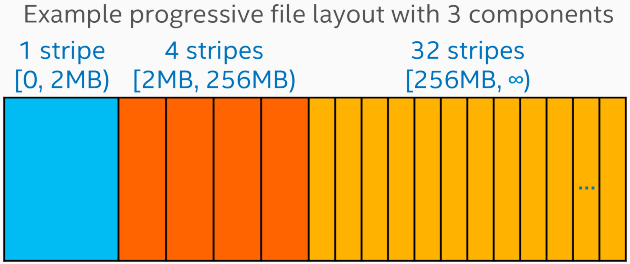
Lustre [3]

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

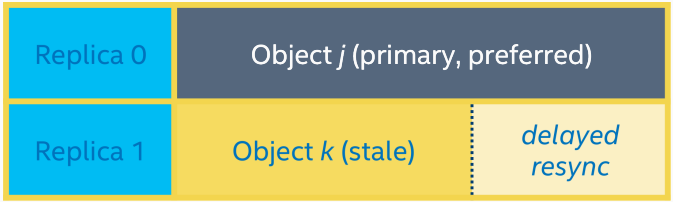
- Verschlüsselung zunehmend wichtig
  - Regierungen, Militär etc.
- Mehrere Zugriffsstufen
  - Frei zugänglich, vertraulich, geheim, streng geheim
  - Kein Datentransfer zwischen unterschiedlichen Stufen
- Authentifizierung und Autorisierung
  - Z. B. via Kerberos
- Verschlüsselung der Kommunikation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

Lustre... [3]

- Unterschiedliche Anforderungen an Dateiverteilung
  - Kleine Dateien nur auf wenige OSTs
  - Große Dateien auf möglichst viele
- Automatische Anpassung der Streifenparameter
  - Ziel: Minimierung des Overheads und Maximierung der Leistung

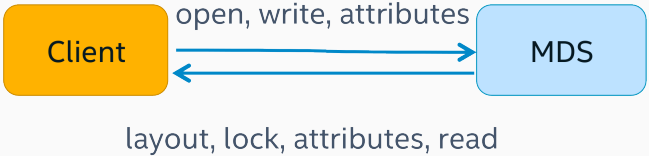


- Replikation von Dateien
  - Kann pro Datei festgelegt werden
  - High Availability, Robustheit, höhere Lesegeschwindigkeit, Migration zwischen Speicherklassen etc.
- Synchronisation geschieht im Hintergrund



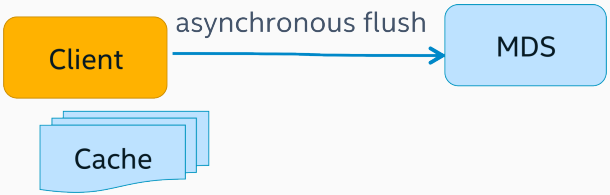
Overlapping (mirror) layout

- 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

- 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

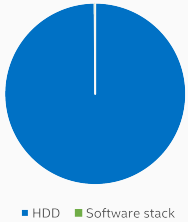
- POSIX stellt aber immer noch eine Limitierung dar
  - POSIX ist portabel, schränkt aber Leistung ein
- Einige Alternativen
  - MPI-IO setzt meist doch auf POSIX auf
  - Häufig POSIX-Dateisysteme auf unterster Ebene
- Abkehr von POSIX
  - Object Stores oft ausreichend
  - Häufig Einschränkungen einiger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

- Idee: 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

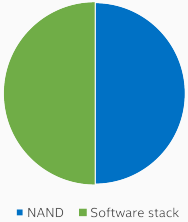
BeeGFS®
developed by Fraunhofer
Cache-Domain 1
Cache-Domain 2
...
Cache-Domain n
Non-coherent Subdomains for scalable Caching
non-coherent write_to_cache()
Per-Subdomain Caches on NVM
flush_to_global() / prefetch_to_cache()
Global parallel Storage Layer with traditional Hard-Disks

- Ganzheitlichere Ansätze in Arbeit
  - Distributed Application Object Storage (DAOS)
- DAOS unterstützt mehrere Speichermodelle
  - Arrays und Records als Basisobjekte
  - Objekte 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

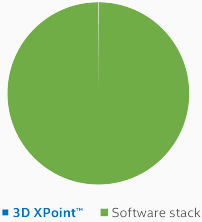
DAOS... [2]



msec vs. µsec



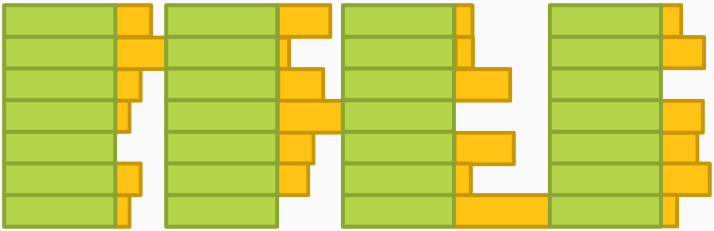
µsec vs. µsec



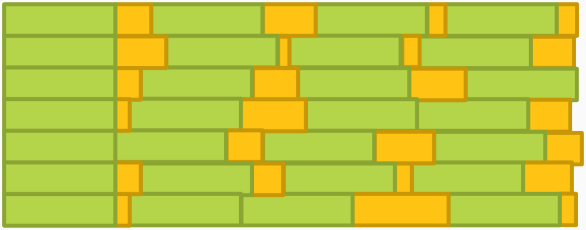
nsec vs. µsec

- 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

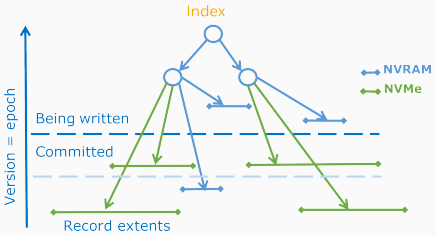
- Üblicherweise synchrone E/A
  - Anwendungen werden durch den langsamsten ausgebremst
    - Ähnliches Problem bei kollektiver Kommunikation
  - Dadurch hohe Wartezeiten, in denen nichts passiert
- E/A-Schwankungen sind der Normalfall
  - Speichersystem ist eine geteilte Ressource
  - Selten Quality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

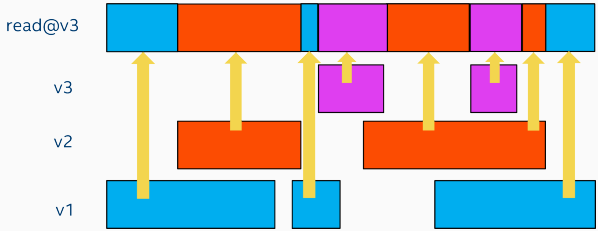


- E/A sollte komplett asynchron stattfinden
  - Dadurch keine Wartezeiten mehr
- Frage: Wann ist die Datei konsistent?
  - Bisher: Wenn alle Prozesse die (kollektive) E/A abgeschlossen haben



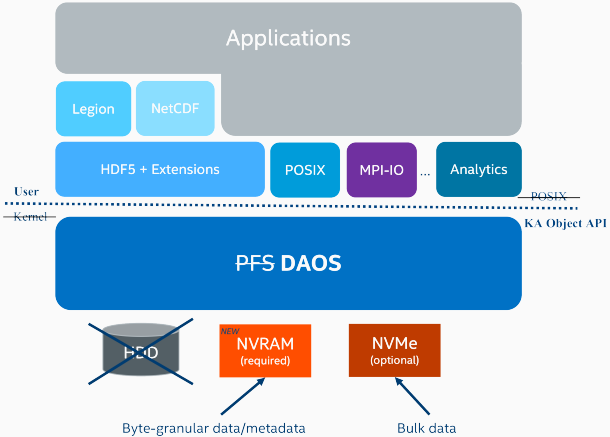
- Lösung: Transaktionen bzw. Epochen
  - Operationen finden in Transaktionen statt
  - Mehrere Transaktionen können zu einer Epoche zusammengefasst werden
- Epochen sind global konsistente Schnappschüsse
  - Epochen werden pro Objekt vergeben
  - Dadurch keine Kohärenzprobleme beim Lesen



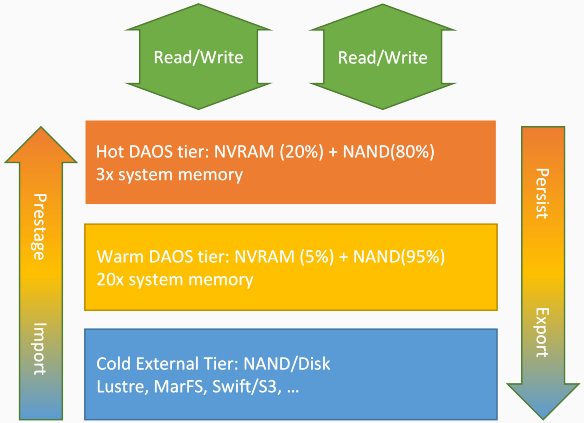


- Native Unterstützung für HDF5
  - Eine HDF5-Datei entspricht einem DAOS-Container
  - HDF-Datenstrukturen werden direkt in DAOS-Objekte überführt
    - Dadurch z. B. Umorganisation der Daten für effizienten Zugriff
- Unterstützung für unterschiedliche Anwendungen
  - POSIX und MPI-IO für Legacy-Anwendungen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

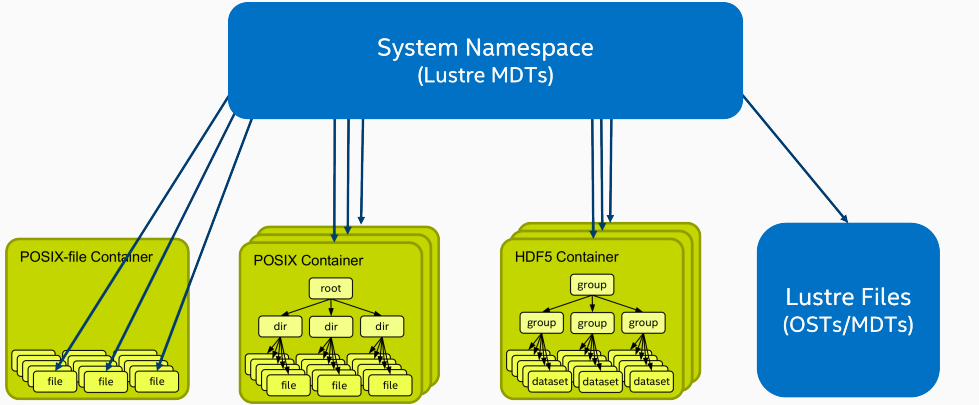
DAOS... [2]



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



- Prestage lädt Daten in den Cache, Persist speichert Daten
- Import/Export für externe Speichersysteme
  - Z. B. HDD-basierte Dateisysteme wie Lustre



- Lustre verwaltet exportierte Container

- 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.
- Problem: Bisher zwei völlig unterschiedliche Software-Stacks

Big Data... [7, 1]

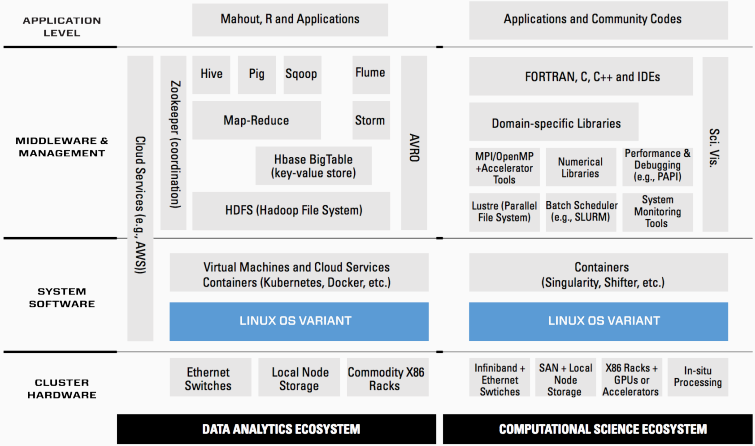
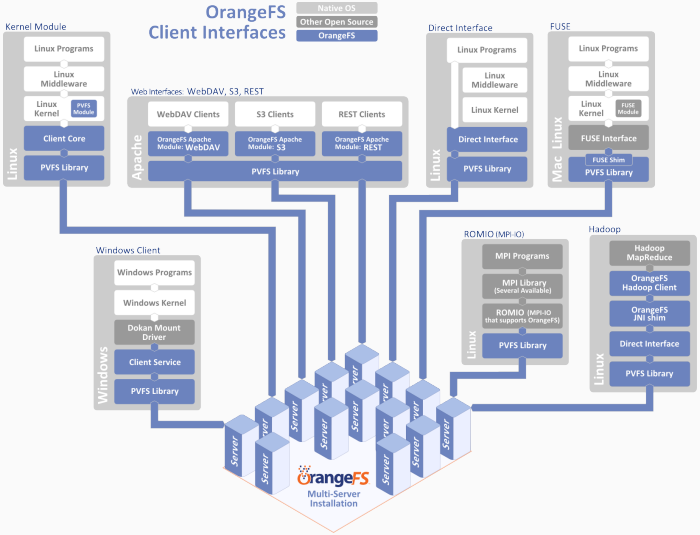


Figure 1: Different software ecosystems for high-end Data Analytics and for traditional Computational Science. [Credit: Reed and Dongarra [66]]

Big Data... [12]


OrangeFS Client Interfaces
Native OS
Other Open Source
OrangeFS
Kernel Module
Linux Programs
Linux Middleware
Linux Kernel
PVFS Module
Client Core
PVFS Library
Linux
Web Interfaces: WebDAV, S3, REST
WebDAV Clients
S3 Clients
REST Clients
OrangeFS Apache Module: WebDAV
OrangeFS Apache Module: S3
OrangeFS Apache Module: REST
PVFS Library
Apache
Direct Interface
Linux Programs
Linux Middleware
Linux Kernel
Direct Interface
PVFS Library
Linux
FUSE
Linux Programs
Linux Middleware
Linux Kernel
FUSE Module
FUSE Interface
FUSE Shim
PVFS Library
Mac Linux
Windows Client
Windows Programs
Windows Kernel
Dokan Mount Driver
Client Service
PVFS Library
Windows
ROMIO (MPI-IO)
MPI Programs
MPI Library (Several Available)
ROMIO (MPI-IO that supports OrangeFS)
PVFS Library
Linux
Hadoop
Hadoop MapReduce
OrangeFS Hadoop Client
OrangeFS JNI shim
Direct Interface
PVFS Library
Linux
Server
OrangeFS
Multi-Server Installation

- Andererseits interessante Ansätze aus dem Big-Data-/Cloud-Umfeld
  - Elastizität zur dynamischen Anpassung des Dateisystems
  - Zuschalten von zusätzlichen Dateisystem-Servern bei Bedarf
- Nutzung von Object Stores
  - Viele Anwendungen benötigen keine POSIX-Dateisysteme
  - MPI-IO-Funktionalität kann auf Object Stores abgebildet werden

Zusammenfassung

- Neue Hardware-Technologien werden den E/A-Stack verändern
  - Neue Hierarchieschichten durch NVRAM, NVMe und 3D XPoint
  - Systeme werden dadurch komplexer und leistungsfähiger
  - Gleichzeitig sind Kosteneinsparungen möglich
  - Datentransformationen immer weiter verbreitet (Kompression)
- Die aktuelle E/A-Software wird grundlegend überarbeitet
  - High-Level-Schnittstellen für Anwendungen
  - POSIX schränkt Leistung ein und wird nicht mehr benötigt
  - Neue Ansätze aus Datenbanken und Big Data

Quellen

[1] J.-C. Andre, G. Antoniu, M. Asch, R. Badia Sala, M. Beck, P. Beckman, T. Bidot, F. Bodin, F. Cappello, A. Choudhary, B. de Supinski, E. Deelman, J. Dongarra, A. Dubey, G. Fox, H. Fu, S. Girona, W. Gropp, M. Heroux, Y. Ishikawa, K. Keahey, D. Keyes, W. Kramer, J.-F. Lavignon, Y. Lu, S. Matsuoka, B. Mohr, T. Moore, D. Reed, S. Requena, J. Saltz, T. Schulthess, R. Stevens, M. Swany, A. Szalay, W. Tang, G. Varoquaux, J.-P. Vilotte, R. Wisniewski, Z. Xu, and I. Zacharov. **Big Data and Extreme-Scale Computing: Pathways to Convergence.** Technical report, EXDCI Project of EU-H2020 Program and University of Tennessee, 01 2018. `http://www.exascale.org/bdec/sites/www.exascale.org.bdec/files/whitepapers/bdec2017pathways.pdf`.

[2] Andreas Dilger. **DAOS: Scale-out Object Storage for NVRAM.** `http://materials.dagstuhl.de/files/17/17202/17202.AndreasDilger.Slides.pdf`.

Quellen ...

[3] Brent Gorda. **HPC Storage Futures – A 5-Year Outlook.** http://lustre.ornl.gov/ecosystem-2016/documents/keynotes/Gorda-Intel-keynote.pdf.

[4] Brent Gorda. **HPC Technologies for Big Data.** http://www.hpcadvisorycouncil.com/events/2013/Switzerland-Workshop/Presentations/Day_2/3_Intel.pdf.

[5] DEEP-ER. **DEEP-ER.** http://www.deep-er.eu/.

[6] Jian Huang, Karsten Schwan, and Moinuddin K. Qureshi. **Nvram-aware logging in transaction systems.** *Proc. VLDB Endow.*, 8(4):389–400, December 2014.

Quellen ...

[7] John Russell. **New Blueprint for Converging HPC, Big Data.** https://www.hpcwire.com/2018/01/18/new-blueprint-converging-hpc-big-data/.

[8] Jonas Bonér. **Latency Numbers Every Programmer Should Know.** https://gist.github.com/jboner/2841832.

[9] 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.

[10] Mike Vildibill. **Advanced IO Architectures.** http://storageconference.us/2015/Presentations/Vildibill.pdf.

Quellen ...

[11] N/A. **Latency Numbers Every Programmer Should Know.** http://i.imgur.com/k0t1e.png.

[12] OrangeFS Development Team. **OrangeFS Documentation.** http://docs.orangefs.com/.