

# Parallele verteilte Dateisysteme

## Hochleistungs-Ein-/Ausgabe

Michael Kuhn

Wissenschaftliches Rechnen  
Fachbereich Informatik  
Universität Hamburg

2016-04-29

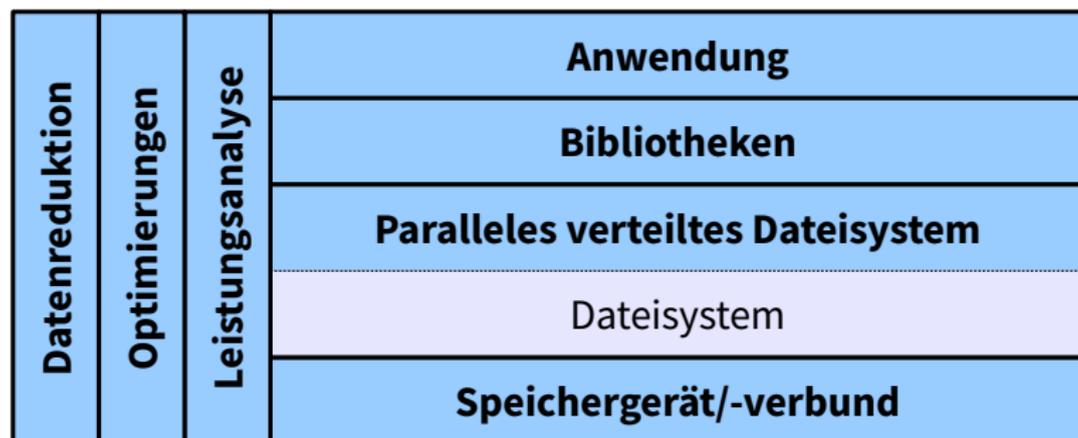


- 1 Parallele verteilte Dateisysteme
  - Orientierung
  - Konzepte
  - Leistungsüberlegungen
  - Lustre
  - OrangeFS
  - Leistungsanalyse
  - Ausblick und Zusammenfassung
  - Lustre-Tutorial

- 2 Quellen



# E/A-Schichten



**Abbildung:** E/A-Schichten

# Definition

- Parallele Dateisysteme
  - Erlauben parallelen Zugriff auf gemeinsame Ressourcen
  - Zugriff soll möglichst effizient erfolgen können
- Verteilte Dateisysteme
  - Daten und Metadaten sind über mehrere Server verteilt
  - Einzelne Server haben keine vollständige Sicht
- Benennung uneinheitlich
  - Oft nur „paralleles Dateisystem“ oder „Clusterdateisystem“

# Definition...

- Im Hochleistungsrechnen werden üblicherweise immer parallele verteilte Dateisysteme eingesetzt
  - NFS nur für nicht leistungskritische Anwendungen
- Lokale Dateisysteme erlauben auch parallelen Zugriff
  - Sperren z.B. via flock oder lockf

- Home-Verzeichnisse können beispielsweise über NFS realisiert werden
- Paralleler Zugriff ist in lokalen Dateisystem relativ einfach zu handhaben
  - Alle Zugriffe laufen über das VFS

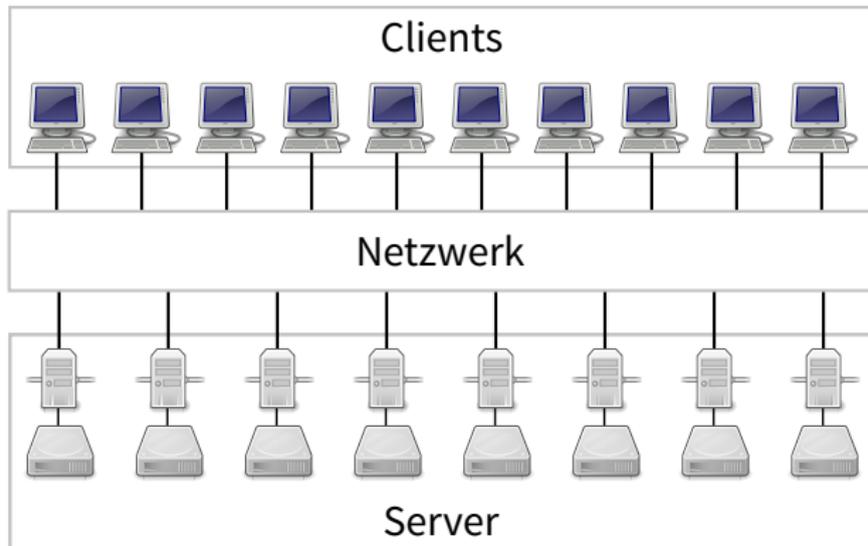


# Definition...

- Storage Area Network (SAN)
  - Stellt nur Blockgeräte über das Netzwerk bereit
  - Darauf kann beliebiges Dateisystem verwendet werden
  - Parallele verteilte Dateisysteme können SANs nutzen
  
- Network Attached Storage (NAS)
  - Abstrahiert von Speichergeräten
  - Stellt direkt ein Dateisystem bereit
    - Üblicherweise NFS oder SMB



# Architektur



**Abbildung:** Paralleles verteiltes Dateisystem









# Schnittstelle

- Zugriff auf das Dateisystem über E/A-Schnittstelle
  - Üblicherweise standardisiert
  - Proprietäre für mehr Funktionen und Leistung
- Schnittstelle besteht aus Syntax und Semantik
  - Syntax legt verfügbare Operation fest
  - Semantik legt Verhalten der Operationen fest
- Häufig POSIX-Schnittstelle
  - Standardisiert und portabel



# Semantik

- POSIX hat strenge Konsistenzanforderungen
  - Änderungen müssen nach `writes` global sichtbar sein
  - E/A soll atomar geschehen
- POSIX für lokale Dateisysteme
  - Anforderungen dort einfach zu unterstützen
  - Alles über das VFS abgewickelt
- Kleinigkeiten änderbar
  - `strictatime`, `relatime` und `noatime` für Verhalten bezüglich Zugriffszeitstempel
  - `posix_fadvise` für Ankündigung des Zugriffsmusters







# Beispiele

- GPFS (IBM)
- Lustre
- OrangeFS (früher PVFS)
- CephFS (Red Hat)
- BeeGFS (Fraunhofer, früher FhGFS)
- GlusterFS (Red Hat)



# Generelles

- E/A ist im Vergleich zu Berechnung teuer
  - Kontextwechsel, langsame Speichergeräte etc.
- Parallele verteilte Dateisystem müssen E/A über das Netzwerk abwickeln
  - Einschränkungen bezüglich Durchsatz und Latenz
- Neuartige Konzepte wie Burst-Buffer
  - Nehmen Daten temporär auf
  - Gleichförmiges Weiterreichen an Dateisystem
  - Z.B. knoten-lokale nichtflüchtige Speicher (NVRAM)

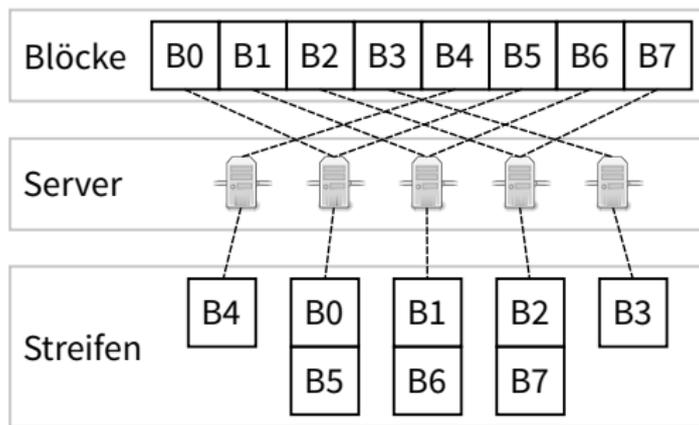
# Daten

- Daten werden potentiell von mehreren Clients zugegriffen
  - Überlappend vs. nicht überlappend
  - Lesezugriffe meist unproblematisch
- Überlappende Schreibzugriffe
  - Stark abhängig von der E/A-Semantik
  - Benötigt üblicherweise Sperren
  - Daher häufig verteilte Sperrenverwaltung

# Daten...

- Datenverteilung relevant für Leistung
  - Anzahl der zu kontaktierenden Datenserver
  - Realisiert über Verteilungsfunktionen
    - Üblicherweise simples Round-Robin
    - Manchmal durch Benutzer steuerbar
    - Z.B. zur Unterstützung heterogener Zugriffsmuster

## Daten...



**Abbildung:** Round-Robin-Datenverteilung

- Datei aus Blöcken, wird in Streifen auf Server verteilt
  - In diesem Fall entspricht Blockgröße der Streifengröße
- Verteilung muss nicht auf dem ersten Server starten



# Metadaten...

- Verteilte Ansätze zur Metadatenverwaltung
  - Veränderliche Metadaten nicht zentral speichern
    - Z.B. Größe und Zeitstempel
  - Berechnung zur Laufzeit
    - Client kontaktiert alle relevanten Datenserver
    - Aktualisierung auf Kosten der Abfrage beschleunigt
- Metadatenverteilung analog zu Datenverteilung
  - Bestimmung des Servers z.B. durch Hashing des Pfades
  - Muss üblicherweise deterministisch sein
    - Clients müssen autonom zuständige Server ermitteln können
  - Ein Objekt üblicherweise von einem Server verwaltet
    - Neuerdings Ausnahmen bei Verzeichnissen

# Metadaten...

- Viele Metadatenoperationen sind inhärent seriell
  - Z.B. Pfadauflösung

## 1 /foo/bar

- 1 Inode des Wurzelverzeichnis lesen
- 2 Zugriffsrechte überprüfen
- 3 Wurzelverzeichnis lesen und nach foo durchsuchen

## 2 /**foo**/bar

- 1 Inode des Verzeichnisses lesen
- 2 Zugriffsrechte überprüfen
- 3 Verzeichnis lesen und nach bar durchsuchen

## 3 /foo/**bar**

- 1 Inode der Datei lesen
- 2 Zugriffsrechte überprüfen
- 3 Auf Datei zugreifen

# Metadaten...

Technologie	Gerät	IOPS
HDD	7.200 RPM	75–100
	10.000 RPM	125–150
	15.000 RPM	175–210
SSD	Intel X25-M G2	8.600
	OCZ Vertex 4	85.000–90.000
	Fusion-io ioDrive Octal	1.000.000+

**Tabelle:** IOPS für ausgewählte HDDs und SSDs [3]

- Getrennte Server erlauben gezielte Optimierung
  - Z.B. Festplatten für Daten, Flashspeicher für Metadaten
  - Unterschiedliche Preise (Faktor  $\geq 10$ )
  - Metadaten machen ca. 5 % der Gesamtdaten aus

# Leistungsgrößen

- Parallele verteilte Dateisysteme erlauben riesige und hochperformante Speichersysteme
- Blizzard (DKRZ, GPFS)
  - Größe: 7 PB
  - Durchsatz: 30 GB/s
- Mistral (DKRZ, Lustre)
  - Größe: 50 PB
  - Durchsatz: 400 GB/s
  - IOPS: 80.000 Operationen/s
- Titan (ORNL, Lustre)
  - Größe: 40 PB
  - Durchsatz: 1,4 TB/s

# Übersicht

- Eines der bekanntesten parallelen verteilten Dateisystemen
- Open Source (GPLv2)
  - > 550.000 Zeilen Code
- Unterstützung für Linux
  - Name abgeleitet von Linux und Cluster
- Sehr weit verbreitet
  - Mehr als die Hälfte der TOP100
  - Mehr als ein Drittel der TOP500



# Geschichte

- 1999: Entwicklungsbeginn
  - Forschungsprojekt an der Carnegie Mellon University, geleitet von Peter Braam
- 2001: Gründung Cluster File Systems
- 2007: Kauf durch Sun
  - Integration in HPC-Hardware
  - Kombination mit ZFS
- 2010: Kauf durch Oracle
  - Einstellung der Entwicklung
- Weiterentwicklung durch Community
  - Intel (ehemals Whamcloud), Seagate (ehemals Xyratex), OpenSFS, EOFS etc.

# Geschichte... [1]

- Version 2.3 (Oktober 2012)
  - Experimentelle Unterstützung für ZFS
- Version 2.4 (Mai 2013)
  - Distributed Namespace (DNE)
  - ZFS für Daten und Metadaten
- Version 2.5 (Oktober 2013)
  - Hierarchical Storage Management (HSM)
- Version 2.6 (Juli 2014)
  - Experimentelle Unterstützung für verteilte Verzeichnisse
- Version 2.7 (März 2015)
  - Bessere Unterstützung für verteilte Verzeichnisse (experimentell)
  - Objektplatzierung via `lfs setstripe`
- Aktuell Version 2.8 (März 2016)

# Geschichte...

```
1 $ lfs mkdir --index 0 /lustre/home
2 $ lfs mkdir --index 1 /lustre/scratch
3 $ lfs mkdir --count 3 /lustre/striped
```

Listing 1: DNE in Lustre

- DNE erlaubt unterschiedliche Verzeichnisse auf unterschiedliche Metadatenserver zu verteilen
  - /scratch für große Dateien
  - /home üblicherweise mit vielen kleineren
  - Statischer Ansatz, manuelle Konfiguration
- Experimentelle Unterstützung für verteilte Verzeichnisse
  - /striped über drei Metadatenserver verteilt

# Architektur

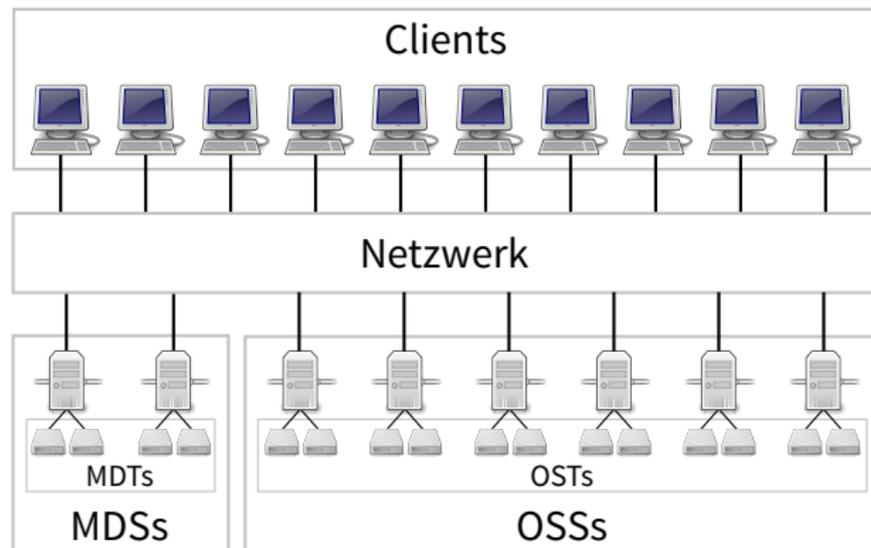


Abbildung: Lustre-Architektur

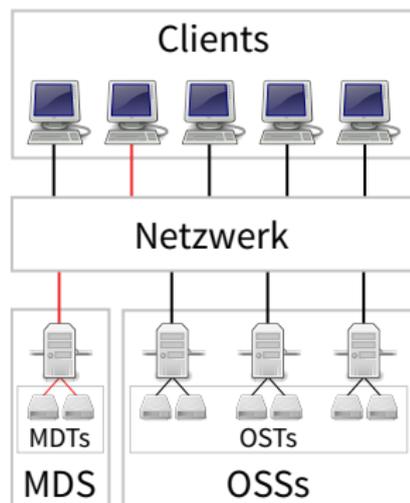
# Architektur...

- Object Storage Servers (OSSs)
  - Verwalten Daten
  - Objekt-basierter Zugriff auf Byte-Ebene
  - Ein oder mehrere Object Storage Targets (OSTs)
- Metadata Servers (MDSs)
  - Verwalten Metadaten
  - Nicht in eigentliche E/A involviert
  - Ein oder mehrere Metadata Targets (MDTs)
- Verteilung über Targets, nicht über Server

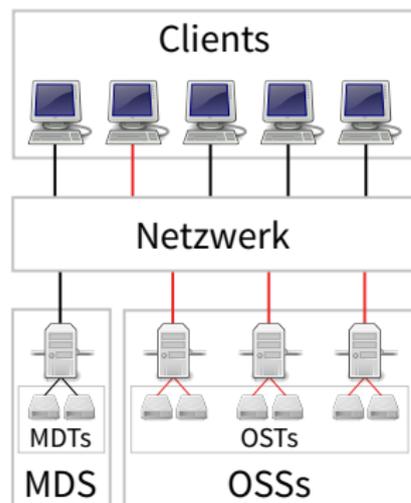
# Architektur...

- Daten- und Metadatenserver nutzen lokales Dateisystem
  - Üblicherweise `ldiskfs` (`ext4-Fork`)
  - Alternativ `ZFS` (`Data Management Unit`)
    - Dadurch kein `POSIX-Overhead`
- Kein direkter Zugriff auf Speichergeräte durch Clients
  - Clients senden Anfragen an Server
  - Server führen Operationen durch
  - Server schicken Ergebnisse an Clients

# Architektur...



(a) Metadatenzugriff



(b) Datenzugriff

- Metadatenserverzugriff nur für initiales Öffnen
- Danach direkter paralleler Zugriff auf Datenserver



# Unterstützung

- Lustre ist ein Kernel-Dateisystem
  - Sowohl Client als auch Server
- Client unterstützt (relativ) aktuelle Kernel
  - Seit 3.12 direkt in den Kernel integriert (aber veraltet)
  - Unterstützung neuer Kernel dauert manchmal eine Weile
- Server unterstützt nur ausgewählte Enterprise-Kernel
  - Z.B. Red Hat Enterprise Linux (oder CentOS)
  - Hauptsächlich verursacht durch ldiskfs
  - Mit ZFS auch Unterstützung des Standard-Kernels

# Funktionalität

- Verteilte Sperrenverwaltung
  - Sowohl für Daten als auch Metadaten
  - Überlappende Lesesperren und nicht-überlappende Schreibsperrern auf Byte-Ebene
  - Mountoptionen flock bzw. `localflock`
- POSIX-konform
  - POSIX-Schnittstelle durch VFS
  - Keine native Unterstützung für MPI-IO



# Funktionalität...

- Hierarchical Storage Management
  - Wichtige Anforderung für große Speichersysteme
  - Unterstützt mehrere „Tiers“
    - Festplatten, Tapes etc.
  - Metadaten werden weiterhin durch Lustre verwaltet
  - Daten werden transparent in andere Tiers verschoben
- Hochverfügbarkeit
  - Unterstützung von Failover-Mechanismen
  - Aktiv/Passiv- und Aktiv/Aktiv-Konfigurationen



# Überblick

- Open Source (LGPL)
  - > 250.000 Zeilen Code
- Entwicklung durch Clemson University, Argonne National Laboratory und Omnibond
- Weiterentwicklung von PVFS
  - 2007: Start als Entwicklungszweig
  - 2010: Ersetzt PVFS als Hauptversion

## Funktionalität [2]

- Verteilte Metadaten und Verzeichnisse
  - Verteilte Verzeichnisse seit Version 2.9
- Läuft komplett im User-Space
- Sehr gute MPI-IO-Unterstützung
  - Natives Backend in ROMIO
- Unterstützung für POSIX-Schnittstelle
  - FUSE-Dateisystem
  - Optionales Kernelmodul

# Funktionalität...

- OrangeFS ist nicht POSIX-konform
  - Garantiert atomare Ausführung nicht-zusammenhängender und nicht-überlappender Zugriffe
  - Bezieht sich auf Lese- und Schreiboperationen
  - Unterstützt somit (nicht-atomares) MPI-IO
- Ausreichend für viele Anwendungsfälle
  - Striktere Semantik allerdings nicht unterstützt
  - MPI-IO-Atomic-Modus mangels Sperren nicht verfügbar

## Lustre

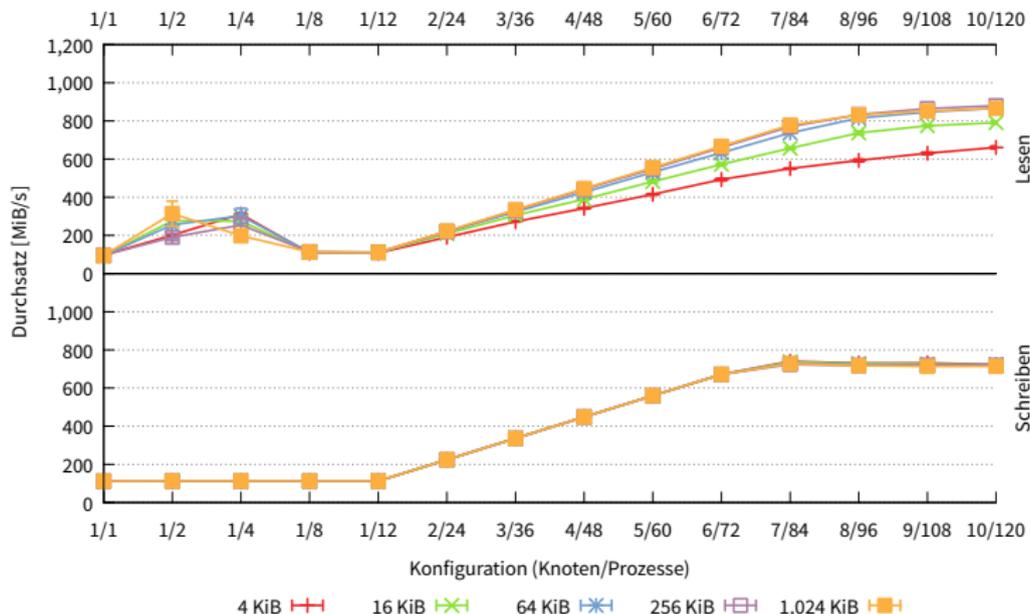


Abbildung: Lustre mit prozess-lokalen Dateien

## Lustre...

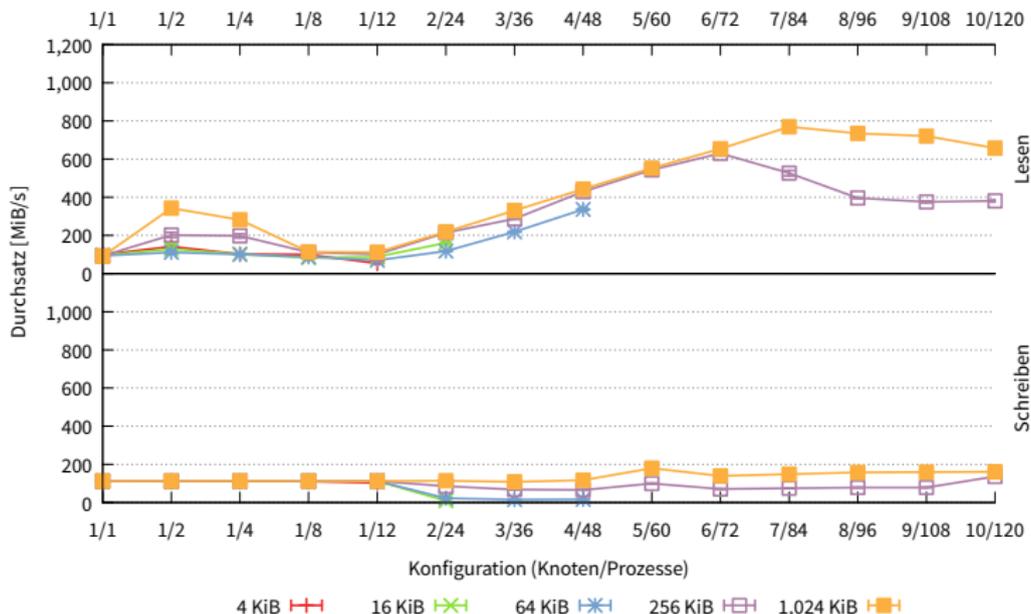


Abbildung: Lustre mit gemeinsamer Datei



# Lustre...

- Lustre hochoptimiert für viele parallele Clients
  - Kein Leistungseinbruch mit wachsender Clientzahl
- Allerdings nur für prozess-lokale Dateien
  - Gemeinsame Dateien skalieren nicht
  - Auswirkung der POSIX-Einschränkungen
  - Hohe Leistung möglich aber kompliziert
- Diverse Optimierungen
  - Aggregiert Schreibzugriffe im Hauptspeicher
  - Führt Readahead durch

- Für hohe Leistung mit gemeinsamen Dateien ist es notwendig die Zugriffe an den Streifengrenzen auszurichten
  - Außerdem möglichst 1:1-Zugriffe, d.h. ein Client kommuniziert mit nur einem Server

# OrangeFS

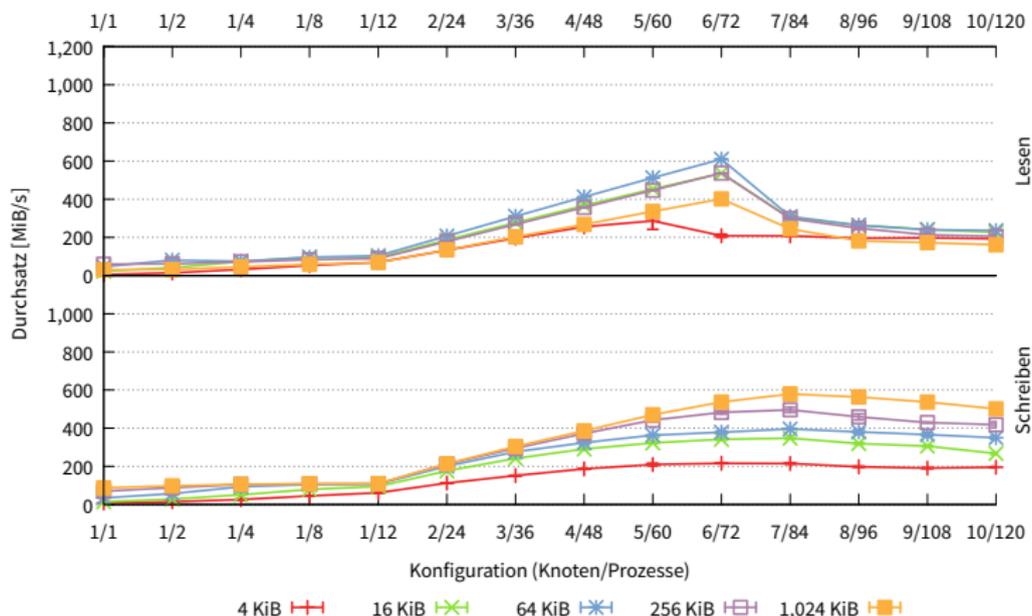


Abbildung: OrangeFS mit prozess-lokalen Dateien

# OrangeFS...

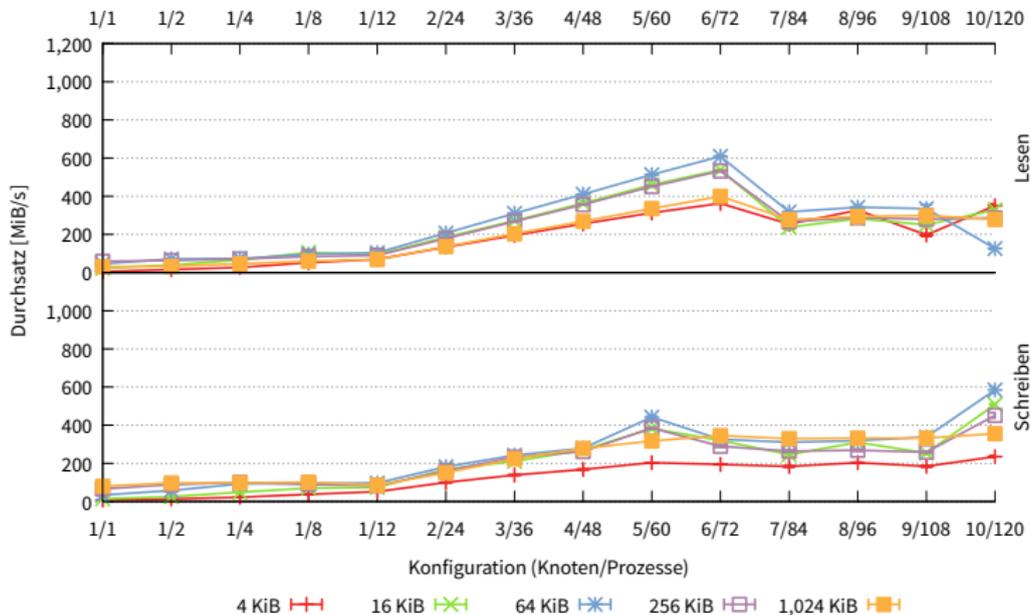


Abbildung: OrangeFS mit gemeinsamer Datei

# OrangeFS...

- OrangeFS ist optimiert für nicht-konfligierende Zugriffe
  - Dadurch keine Sperren notwendig
  - Hohe Leistung auch bei gemeinsamen Dateien
- Leistungsprobleme durch darunter liegendes Dateisystem
- Außerdem geringe Metadatenleistung

## Lustre

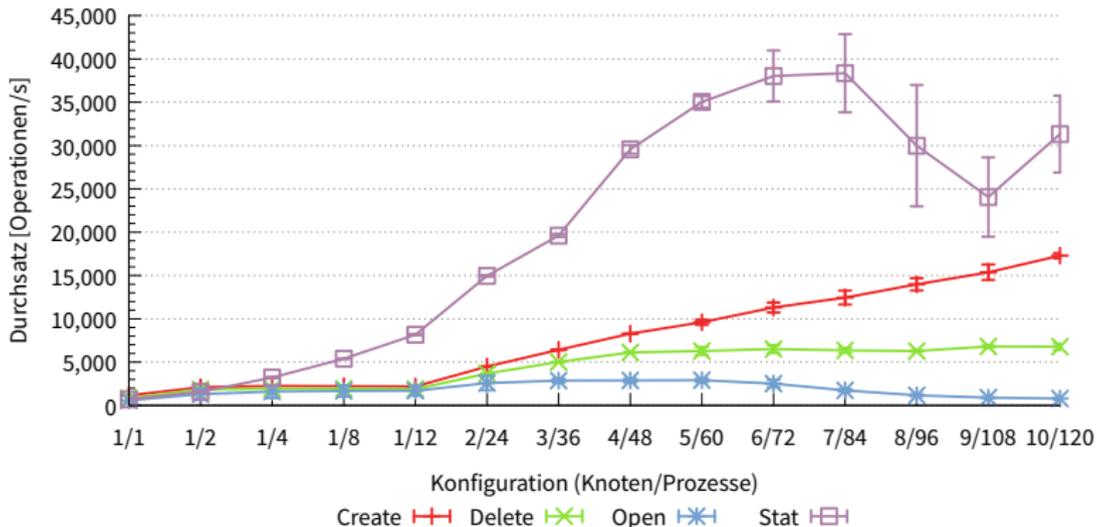
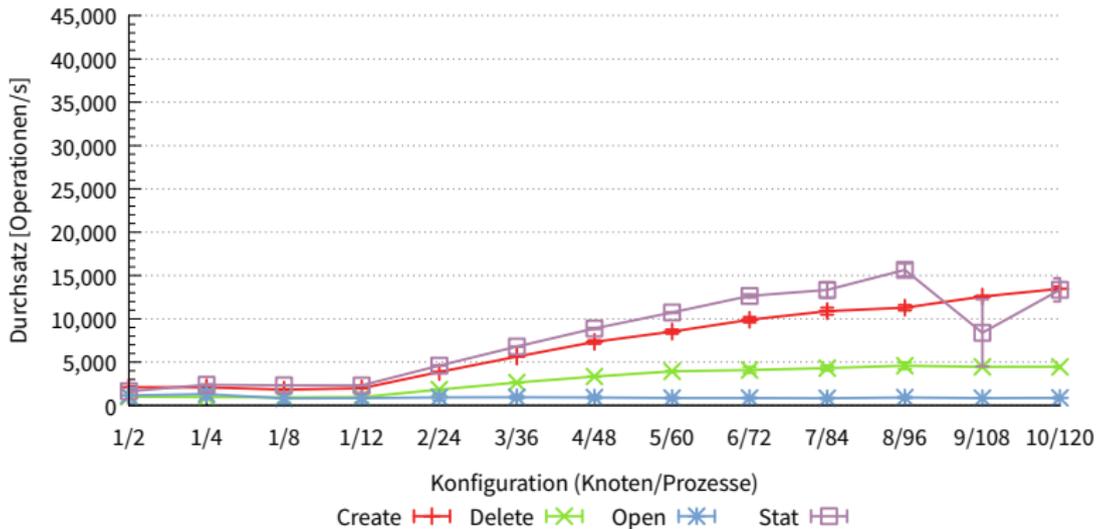


Abbildung: Lustre mit prozess-lokalen Verzeichnissen

## Lustre...

**Abbildung:** Lustre mit gemeinsamem Verzeichnis

# Lustre...

- Deutliche Unterschiede zwischen prozess-lokalen und gemeinsamen Verzeichnissen
- Passable Leistung für Dateierzeugung
- Leistungsprobleme beim Öffnen und Löschen
  - Interessanterweise ist Öffnen langsamer als Erzeugen

# Ausblick

- E/A-Bibliotheken bieten Komfortfunktionen auf Basis von parallelen verteilten Dateisystemen
  - Z.B. selbst-beschreibende Daten mit NetCDF und HDF
- Semantiken haben großen Einfluss auf erreichbare Leistung
  - Forschungsthema: Dynamisch anpassbare Semantiken

# Zusammenfassung

- Parallele verteilte Dateisysteme bieten gleichzeitigen Zugriff für viele Clients
  - Skalierbarer gleichzeitiger Zugriff schwierig
  - Verteilen außerdem Daten und Metadaten für erhöhten Durchsatz und Kapazität
- Üblicherweise Aufteilung in Daten- und Metadatenserver
- Zugriff über E/A-Schnittstelle
  - Häufig POSIX oder MPI-IO
- Wichtige Vertreter sind Lustre und GPFS
  - OrangeFS bietet alternativen Ansatz

# Abhängigkeiten

```
1 systemctl stop firewalld
2 systemctl disable firewalld
3 sed -i '/^SELINUX=/s/./SELINUX=disabled/'
   ↪ /etc/selinux/config
4 setenforce 0
```

Listing 2: System vorbereiten

```
1 yum -y upgrade
2 yum -y groupinstall 'Development Tools'
3 yum -y install xmlto asciidoc elfutils-libelf-devel
   ↪ zlib-devel binutils-devel newt-devel python-devel
   ↪ hmaccalc perl-ExtUtils-Embed bison elfutils-devel
   ↪ audit-libs-devel python-docutils sg3_utils expect
   ↪ attr lsof quilt libselinux-devel
```

Listing 3: Abhängigkeiten installieren

# ZFS

```
1 URL='http://archive.zfsonlinux.org'  
2  
3 yum -y install epel-release  
4 yum -y install --nogpgcheck  
    ↪ $URL/epel/zfs-release.el7.noarch.rpm  
5 yum -y install zfs zfs-dkms libzfs2-devel libuuid-devel  
6  
7 systemctl start zfs.target
```

Listing 4: ZFS installieren





- Um normale Speichergeräte zu nutzen, müssen statt Dateien in /tmp einfach Geräte in /dev angegeben werden
  - Außerdem sollte dann der `device-size`-Parameter weggelassen werden
- Bei mehreren MDTs bzw. OSTs muss der Index jeweils erhöht werden

# Lustre-Konfiguration...

```
1 echo "$(hostname) - mgs zfs:lustre-mgs/mgs" >>
  ↪ /etc/ldev.conf
2 echo "$(hostname) - mdt0 zfs:lustre-mdt0/mdt0" >>
  ↪ /etc/ldev.conf
3 echo "$(hostname) - ost0 zfs:lustre-ost0/ost0" >>
  ↪ /etc/ldev.conf
4
5 systemctl daemon-reload
6 systemctl start lustre
```

Listing 7: Dienste starten

```
1 mkdir -p /mnt/lustre/client
2 mount -t lustre $(hostname):/lustre /mnt/lustre/client
```

Listing 8: Lustre mounten

- Der beim Mounten angegebene Pfad `/lustre` entspricht dem Dateisystemnamen (`fsname`)

- 1 Parallele verteilte Dateisysteme
  - Orientierung
  - Konzepte
  - Leistungsüberlegungen
  - Lustre
  - OrangeFS
  - Leistungsanalyse
  - Ausblick und Zusammenfassung
  - Lustre-Tutorial

## 2 Quellen



# Quellen I

- [1] OpenSFS and EOFS. Lustre. <http://lustre.org/>.
- [2] OrangeFS Development Team. OrangeFS.  
<http://www.orangefs.org/>.
- [3] Wikipedia. IOPS.  
<http://en.wikipedia.org/wiki/IOPS>.