A Performance Study of Lustre File System Checker (LFSCK)

Seminar *Supercomputer: Forschung und Innovation*

Lisa Jiménez

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

28. Juni 2022

Agenda

Lustre – Was ist das?

- ist ein open–source, paralleles, verteiltes Dateisystem
- verwendet Striping, um Datenblöcke über Speicherorte zu verteilen (RAID–0)
- entworfen für Skalierbarkeit, hohe Leistung und hohe Verfügbarkeit

Typisches Lustre Cluster

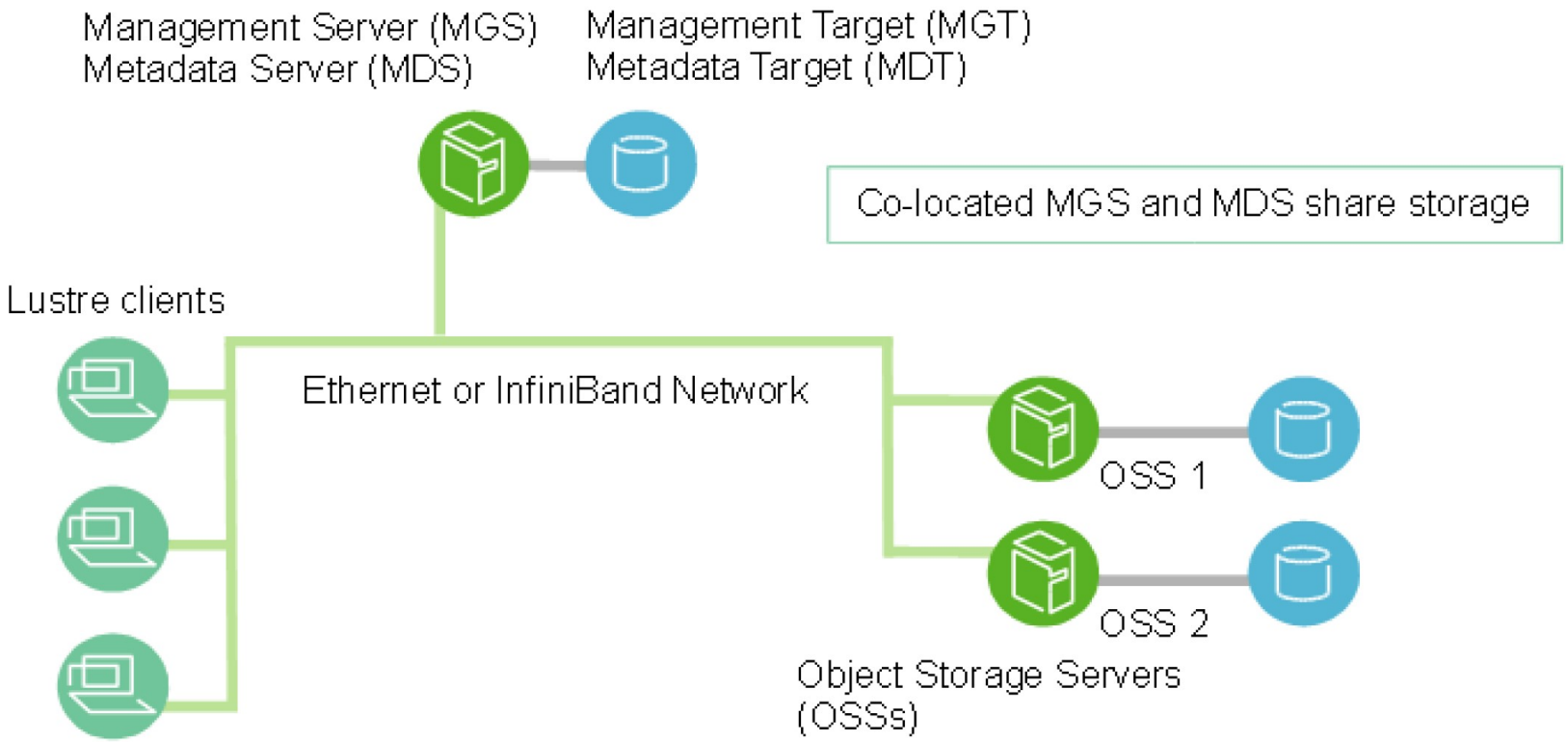

Abbildung: *Lustre file system components in a basic cluster* [3]

Lustre – drei Typen von Metadaten (1/2)

1. Abbildungsmetadaten
   - globale Objekte im System werden durch lokale inodes auf MDS repräsentiert
   - auch Objekte auf den OSSen
   - Abbildungsinformationen von Objekten auf inodes bilden Abbildungsmetadaten
   - werden im *Object Index* gespeichert

inode: spezielle Datei–Datenstruktur, die Informationen zu einer Linux Datei speichert außer Name und Daten [1]

Lustre – drei Typen von Metadaten (2/2)

2. Namensraummetadaten, wie z.B.
   - Dateinamen
   - Verzeichnisse/Ordner
   - Zugriffsrechte
3. Daten Layout Metadaten – Wo liegt der Inhalt (m)einer Datei?
   - Daten werden gestriped und auf OST Objekte geschrieben
   - um Daten einer Datei zu finden, werden im inode gespeichert:
     - Anzahl der Stripes
     - Größe der Stripes
     - Object Identifiers
     - weitere Layout Metadaten

Lustre Architektur im Detail

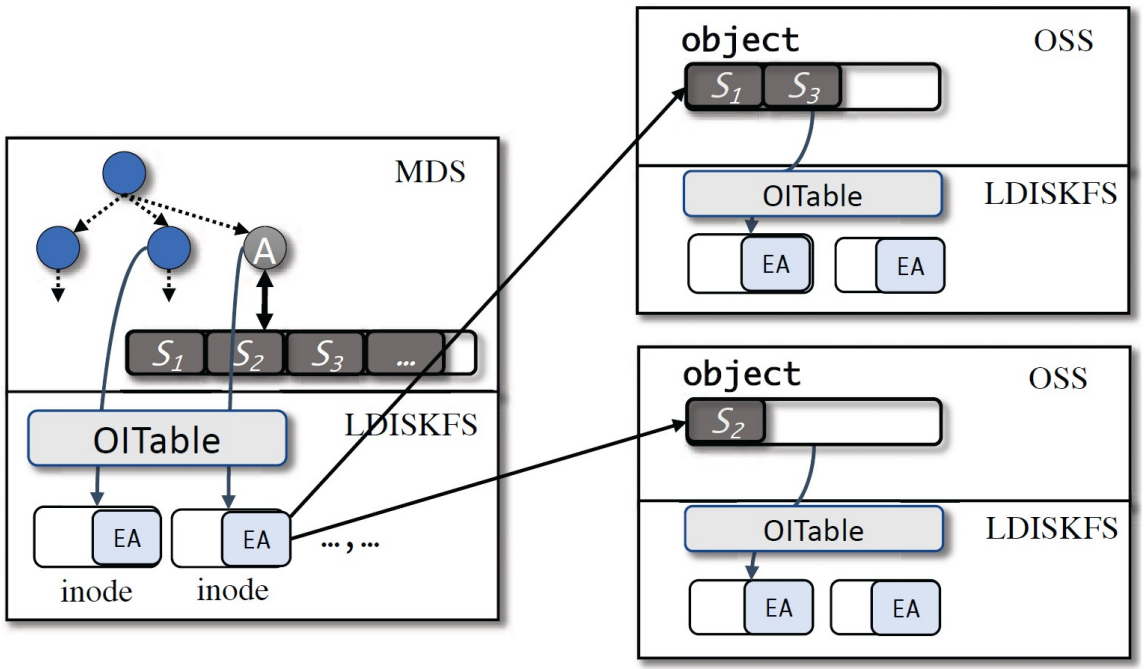

Fig. 1. Lustre Metadata Management Architecture. *Due to space limitation, we only plot one master metadata server and two object storage servers.*

Lustre File System Checker – Was ist das?

- LFSCK ist ein Tool von Lustre
- es scannt das System, identifiziert und fixt Metadaten Inkonsistenzen
- diese treten z.B. auf durch:
  - Software Fehler
  - Hardware Fehler
  - Konfigurationsfehler
- können zu Datenverlust führen

Problem: in der Praxis benötigt LFSCK zu lange, um es als Routine–Wartungstool einzusetzen

LFSCK – Was wird gecheckt?

1. Abbildungsmetadaten: FID der zugehörigen Datei
2. Namensraum Metadaten: FID vom Eltern–Verzeichnis; im Extended Attribute des inodes gespeichert
3. Datenlayout Metadaten:
   - auf MDS: Speicherorte aller Stripes
   - auf OSS: FID der zugehörigen Datei

FID: Lustre File System Identifier

LFSCK – Checking Workflow

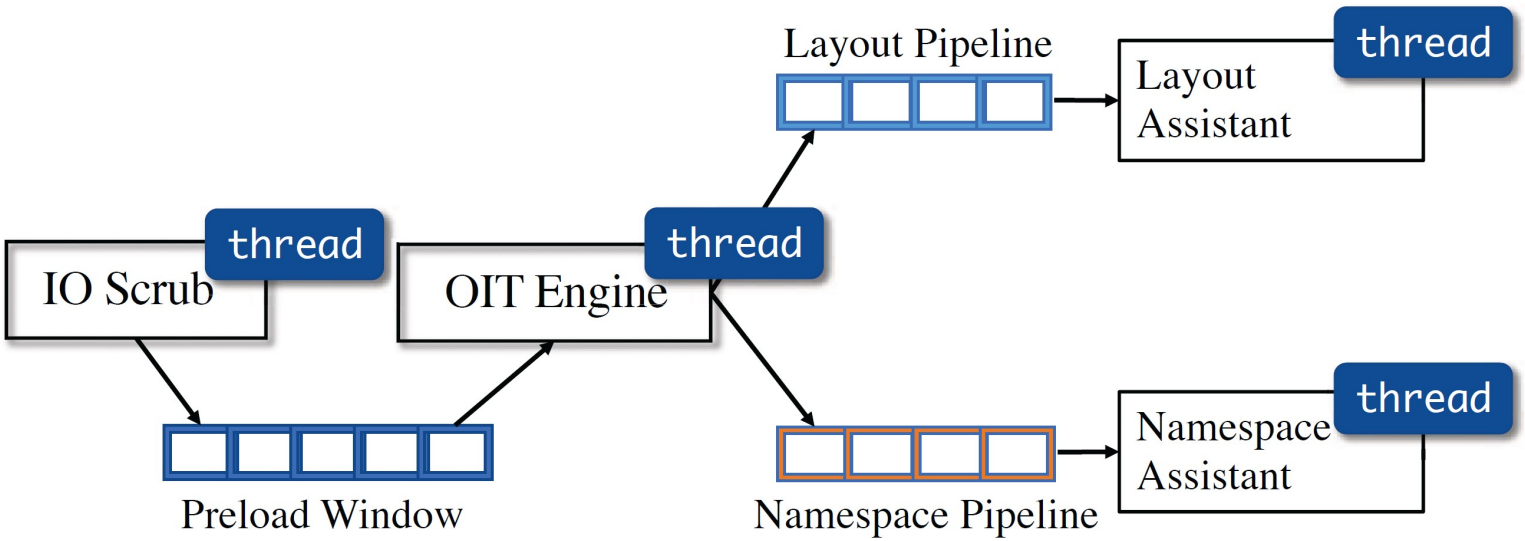

Abbildung: *Lustre file system checker (checking phase) on MDS.*

Performance Study – Aufbau der Experiemnte

Basis–Aufbau:

- 1 MDS und 8 OSSe
- algorithmisch erzeugte Dateien und Verzeichnisstrukturen
- jede Datei auf alle OSSe gestriped
- Monitoring Framework um Disk– und Netzwerknutzung zu überwachen

Scalability Bottlenecks – Experiment 1

- Anzahl OSSe bleibt gleich
- Anzahl Dateien im System wird erhöht
- dadurch erhöht sich die Anzahl der inodes

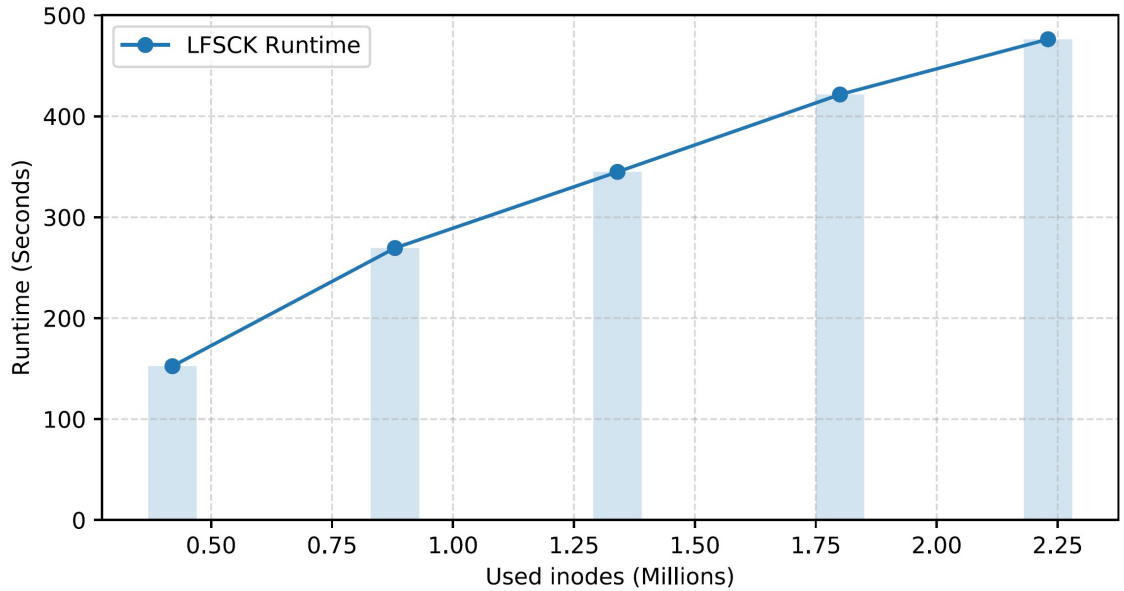

Abbildung: Performance von LFSCK bei steigender Anzahl inodes

Scalability Bottlenecks – Experiment 2

- Anzahl Dateien im System bleibt gleich
- variieren die Anzahl der OSSe – von 2 auf 4 auf 8
- behalten Anzahl an inodes bei
- dadurch erhöht sich die Anzahl der Stripes je Datei

TABLE I
EXECUTION TIME OF LFSCK (IN SECONDS) ON LUSTRE WITH DIFFERENT NUMBER OF OSS NODES

# of OSS Nodes	2-OSS	4-OSS	8-OSS
Execution Time	144.8	170.5 (**1.18X**)	218.4 (**1.50X**)

Internal Bottlenecks

- Experiment mit Basis–Aufbau
- Beobachtung eines LFSCK Durchlaufs

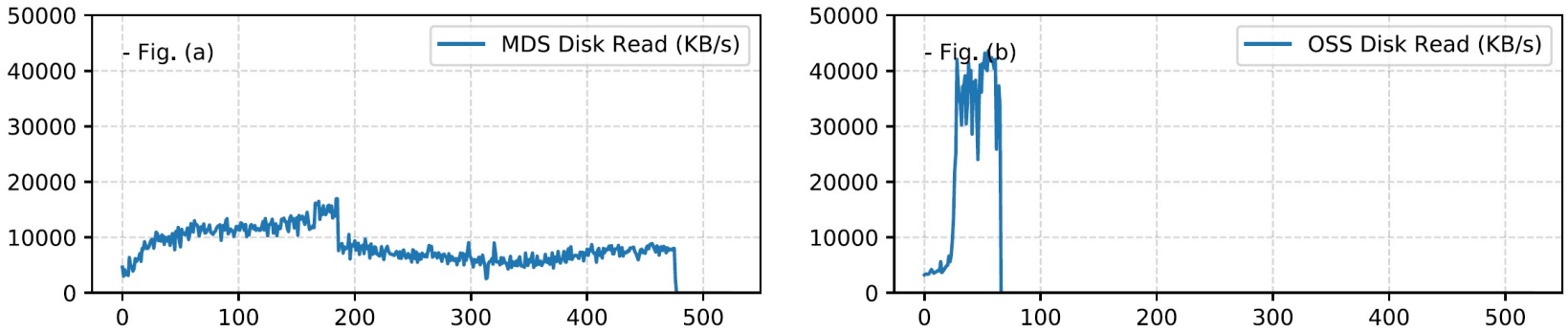

Abbildung: Lesende Zugriffe in Kilobyte pro Sekunde auf MDS bzw. OSS

Layout Checking Bottlenecks – Experiment 1

- Experiment mit Basis–Aufbau
- Beobachtung eines LFSCK Durchlaufs

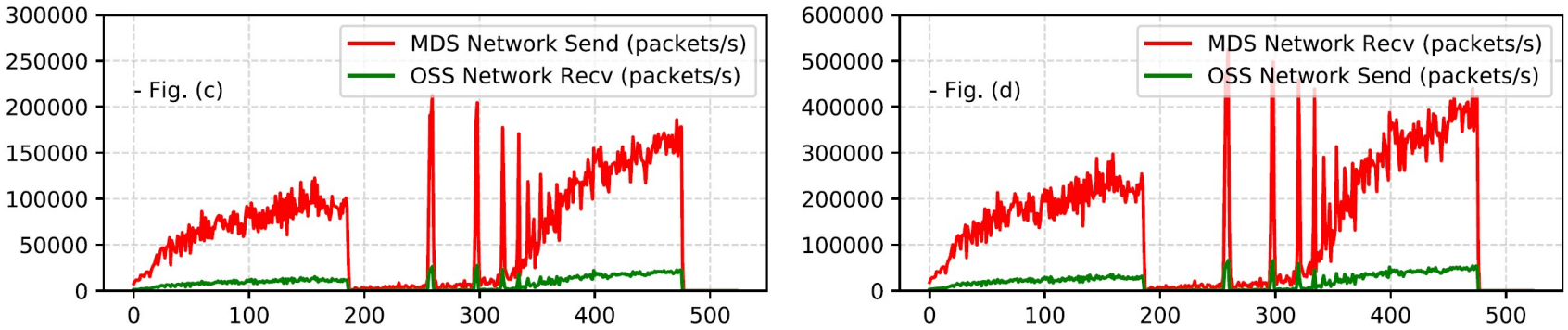

Abbildung: Anzahl der Pakete pro Sekunde, die MDS und ein OSS senden und empfangen (links) bzw. empfangen und senden (rechts)

Layout Checking Bottlenecks – Experiment 2

- Layout Checking unter idealen Bedingungen
- d.h. alle Metadaten gecached auf MDS, OSS

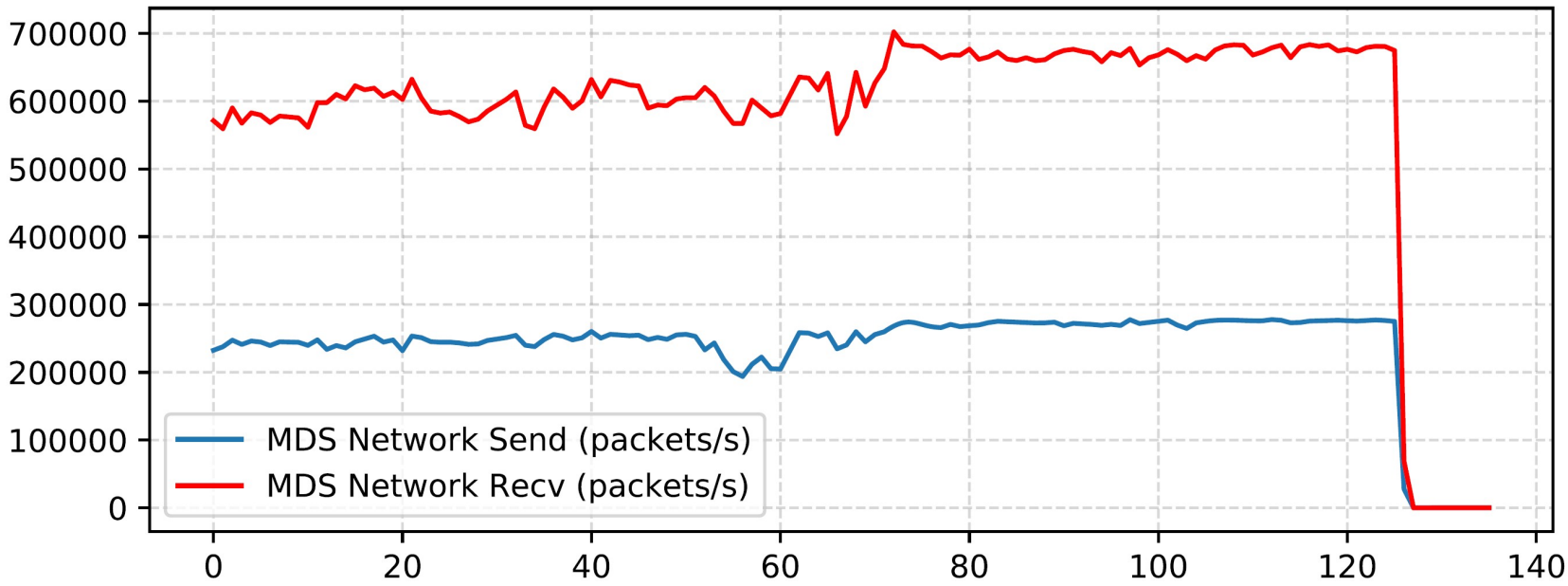

Abbildung: Anzahl der Pakete pro Sekunde, die MDS sendet und empfängt

Namespace Checking Bottlenecks – Experiment

- Namespace Checking nur auf MDS
- MDS bearbeitet lesende Zugriffe sehr langsam (vgl. Folie 14)
  - zum Teil weniger als 7,5MB pro Sekunde

Experiment: LFSCK nur mit Namespace Checking laufen lassen

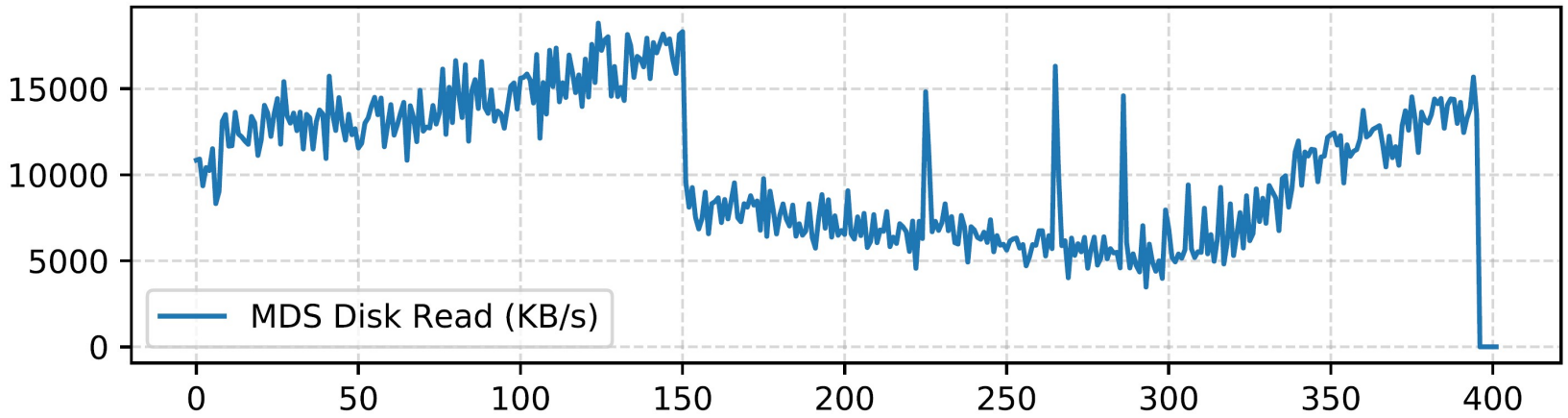

Abbildung: Lesende Zugriffe auf MDS in Kilobyte pro Sekunde

Zusammenfassung

LFSCK Problemzonen:

- es ist sub–optimal designed und implementiert
- hat ein Scalability Bottleneck auf dem MDS
- verwendet weder Disk Bandbreite noch Netzwerk Bandbreite vollständig

Verbesserungsmöglichkeiten:

- Layout Checking Antworten der OSSe komprimieren
- Layout Checking und Namespace Checking entkoppeln

Literatur

[1] bluematador. *What is an inode and what are they used for?* URL: https://www.bluematador.com/blog/what-is-an-inode-and-what-are-they-used-for (besucht am 27. 05. 2022).

[2] Dong Dai, Om Rameshwar Gatla und Mai Zheng. „A Performance Study of Lustre File System Checker: Bottlenecks and Potentials“. In: *2019 35th Symposium on Mass Storage Systems and Technologies (MSST)*. 2019, S. 7–13. DOI: 10.1109/MSST.2019.00-20.

[3] Lustre Wiki und Handbuch. URL: https://wiki.lustre.org/Main_Page (besucht am 01. 06. 2022).