Datenreduktion

Hochleistungs-Ein-/Ausgabe

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Michael Kuhn

2019-05-28

Wissenschaftliches Rechnen
Fachbereich Informatik
Universität Hamburg

Datenreduktion

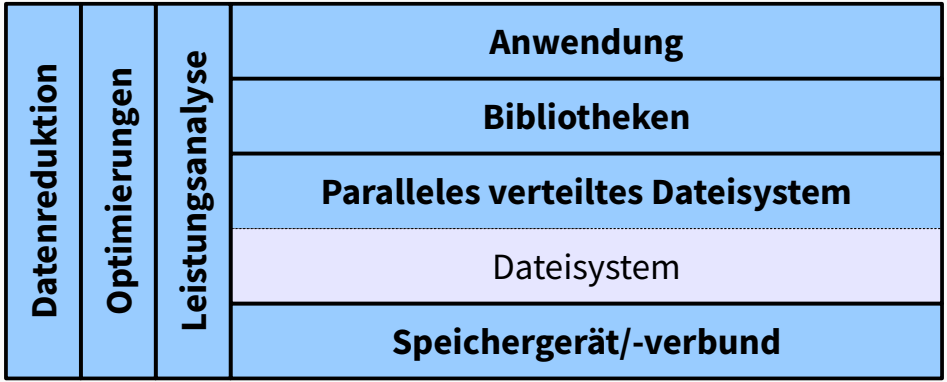
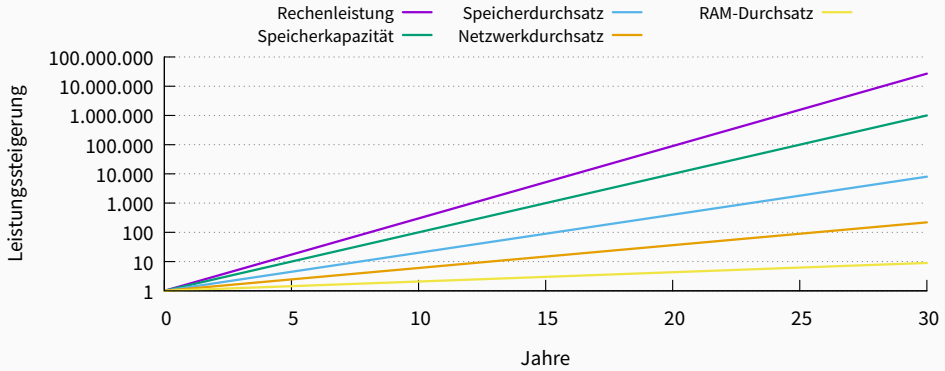


Abbildung 1: E/A-Schichten und orthogonale Themen

Rechen- und Speicherleistung

- Kapazität und Leistung erhöhen sich weiterhin exponentiell
  - Komponenten verbessern sich unterschiedlich schnell
- E/A wird zunehmend zu einem Problem
  - Daten können immer schneller produziert werden
  - Speicherung ist nicht immer problemlos möglich
- Konsequenz: Höhere Ausgaben für Speicherhardware
  - In der Folge weniger verfügbare Mittel für Rechenleistung
  - Alternativ insgesamt teurere Systeme
- Speicherhardware ist ein maßgeblicher Teil der TCO
  - DKRZ: Ca. 20 % der Gesamtkosten
    - Entspricht 6.000.000 € Anschaffungskosten

Rechen- und Speicherleistung...



- Rechenleistung: 300x alle zehn Jahre (basierend auf TOP500)
- Speicherkapazität: 100x alle zehn Jahre
- Speicherdurchsatz: 20x alle zehn Jahre

Beispiel: DKRZ [5]

	2009	2015	Faktor
Leistung	150 TF/s	3 PF/s	20x
Knotenzahl	264	2.500	9,5x
Knotenleistung	0,6 TF/s	1,2 TF/s	2x
Arbeitsspeicher	20 TB	170 TB	8,5x
Speicherkapazität	5,6 PB	45 PB	8x
Speicherdurchsatz	30 GB/s	400 GB/s	13,3x
Festplatten	7.200	8.500	1,2x
Archivkapazität	53 PB	335 PB	6,3x
Archivdurchsatz	9,6 GB/s	21 GB/s	2,2x
Stromverbrauch	1,6 MW	1,4 MW	0,9x
Beschaffungskosten	30 M€	30 M€	1x

Beispiel: DKRZ... [5]

	2020	2025	Exascale (2020)
Leistung	60 PF/s	1,2 EF/s	1 EF/s
Knotenzahl	12.500	31.250	100k–1M
Knotenleistung	4,8 TF/s	38,4 TF/s	1–15 TF/s
Arbeitsspeicher	1,5 PB	12,8 PB	3,6–300 PB
Speicherkapazität	270 PB	1,6 EB	0,15–18 EB
Speicherdurchsatz	2,5 TB/s	15 TB/s	20–300 TB/s
Festplatten	10.000	12.000	100k–1M
Archivkapazität	1,3 EB	5,4 EB	7,2–600 EB
Archivdurchsatz	57 GB/s	128 GB/s	—
Stromverbrauch	1,4 MW	1,4 MW	20–70 MW
Beschaffungskosten	30 M€	30 M€	200 M$

Ansätze

- Speicherkosten sollen stabil gehalten werden
  - Datenvolumen muss reduziert werden
- Erster Schritt: Bestimmung der Speicherkosten
  - Es ist wichtig, unterschiedliche Kostentypen aufzuschlüsseln
  - Kostenmodell für Berechnung, Speicherung und Archivierung [5]
- Analyse mehrerer Datenreduktionstechniken
  - Wiederberechnung, Deduplikation, Kompression

CMIP5 [5]

- Teil des AR5 des IPCC
  - Coupled Model Intercomparison Project Phase 5
  - Vergleich von Klimamodellen für gemeinsame Experimente
- Mehr als 10,8 Millionen Stunden Rechenzeit am DKRZ
  - 482 Läufe, Simulation von insgesamt 15.280 Jahren
  - Mehr als 640 TB Daten wurden berechnet, durch Post-Processing auf 55 TB verringert
- Typische Konfiguration mit niedriger Auflösung:
  - Ein Jahr benötigt ca. 1,5 Stunden auf dem 2009-System
    - Ein Monat Simulation entspricht 4 GB Daten
  - Schreibt am Ende einen Checkpoint (4 GB), weiterer Job startet auf Basis davon
    - Jeder zehnte Checkpoint wird archiviert
- Daten waren fast drei Jahre im Dateisystem gespeichert
  - Archivierung für 10 Jahre

		CMIP5			
System		2009	2015	2020	2025
Berechnung		10,50	0,55	0,03	0,001
Speicherung	Beschaffung	45,02	5,60	0,93	0,16
	Zugriff	0,09	0,01	0	0
	Metadaten	0,04	0	0	0
Checkpoint		0	0	0	0
Archivierung		10,35	1,66	0,41	0,10
Gesamt		66,01	7,82	1,38	0,26

- 2009: Berechnungskosten $\approx$ Archivierungskosten
  - Speicherkosten sind viel höher
- Speicherung und Archivierung werden (relativ) immer teurer

- High Definition Clouds and Precipitation for Climate Prediction
  - Besseres Verständnis der Wolken- und Niederschlagsprozesse
  - Einfluss auf Klimavorhersagen
- Simulation Deutschlands mit einer Gitterauflösung von 416 m
- Ein Lauf auf dem 2009-System benötigt 5.260 GB RAM
  - Simuliert zwei Stunden in 86 Minuten
  - Ergebnisse werden alle 30 Modellminuten geschrieben
- Am Programmende wird ein Checkpoint erstellt
- Ausgabe muss nur eine Woche im Dateisystem gehalten werden

		HD(CP)^2			
System		2009	2015	2020	2025
Berechnung		165,07	8,72	0,44	0,02
Speicherung	Beschaffung	2,37	0,30	0,05	0,01
	Zugriff	0,94	0,07	0,01	0
	Metadaten	0	0	0	0
Checkpoint		0,33	0,02	0	0
Archivierung		86,91	13,91	3,48	0,87
Gesamt		255,29	22,99	3,97	0,90

- Höhere Berechnungskosten als CMIP5
  - 2009: Berechnungskosten $\approx 2 \times$ Archivierungskosten
- Niedrige Speicherungskosten, da schneller archiviert wird

Datenreduktionstechniken

- Es gibt mehrere Konzepte, um das Datenvolumen zu reduzieren
- Wiederberechnung der Ergebnisse
  - Nicht alle Ergebnisse werden explizit gespeichert, sondern bei Bedarf neu berechnet
- Deduplikation
  - Identische Daten werden nur einmal gespeichert
- Kompression
  - Daten können in der Anwendung oder dem Dateisystem komprimiert werden

Überblick [5]

- Speichere nicht alle produzierten Daten
  - Daten werden in situ analysiert
- Benötigt eine genaue Definition der Analysen
  - Post-mortem-Analysen sind unmöglich
  - Neue Analysen benötigen neue Berechnungen
- Wiederberechnung kann unter Umständen sinnvoll sein
  - Wenn Speicher- und Archivierungskosten deutlich höher als Berechnungskosten sind
- Berechnungskosten sind mit dem 2009-System höher als die Archivierungskosten
  - Berechnungsleistung verbessert sich schneller als Speicher

Analyse [5]

- 2015
  - Wiederberechnung lohnt sich, wenn auf die Daten nur ein (HD(CP)^2) oder 13 (CMIP5) Mal zugegriffen wird
- 2020
  - HD(CP)^2: Wiederberechnung lohnt, wenn auf die Daten seltener als acht Mal zugegriffen wird
  - CMIP5: Archivierung ist kosteneffizienter, wenn auf die Daten mehr als 44 Mal zugegriffen wird
- 2025
  - Wiederberechnung ist sinnvoll, wenn auf die Daten nicht öfter als 44 (HD(CP)^2) oder 260 (CMIP5) Mal zugegriffen wird

Probleme: Erhaltung von Binärdateien [5]

- Erhalte Binärdateien von Anwendungen und aller ihrer Abhängigkeiten
  - Durch Container und virtuelle Maschinen viel einfacher
- Es ist schwierig Anwendungen auf abweichenden Architekturen auszuführen
  - x86-64 vs. POWER, Big-Endian vs. Little-Endian
  - Emulation verursacht üblicherweise starke Leistungseinbußen
- Wiederberechnung auf dem selben Supercomputer ist machbar
  - Behalte Abhängigkeiten (versionierte Module), statisches Linken

Probleme: Erhaltung von Quelltext [5]

- Alle Komponenten können auch auf abweichenden Hardwarearchitekturen kompiliert werden
  - Benötigt unter Umständen zusätzlichen Aufwand
  - Unterschiedliche Betriebssysteme, Compiler etc.
  - Alternativ können alle Abhängigkeiten erhalten werden
- Minimale Änderungen könnten Ergebnisse ändern
  - Unterschiedliche Prozessoren, Netzwerktechnologien etc.
  - Möglicherweise irrelevant, solange Ergebnisse „statistisch gleich“ sind

Zusammenfassung

- Wiederberechnung kann sich bei aktueller Entwicklung lohnen
- Nicht nur relevant zum Einsparen von Speicherplatz
  - Reproduzierbarkeit wird immer wichtiger
  - Ergebnisse sollen unabhängig und zeitlich versetzt nachvollzogen werden können
- Genaue Definition der Experimente notwendig
- Außerdem müssen alle Eingabedaten verfügbar gehalten werden

Überblick [5]

- Daten werden in Blöcke aufgeteilt (4–16 KB)
  - Unterschiedliche Aufteilungsmethoden (statisch oder content-defined)
- Jeder einzigartige Datenblock wird nur einmal gespeichert
  - Mehrfach vorkommende Blöcke werden referenziert
- Bisherige Studien zeigen 20–30 % Einsparung für HPC-Daten
  - Insgesamt mehr als 1 PB untersucht
  - Deduplikation ganzer Dateien: 5–10 %
- Deduplikation hat auch Nachteile
  - Deduplikationstabellen müssen im RAM gehalten werden
    - Je 1 TB Daten ca. 5–20 GB für Tabellen

Overhead [5]

- Deduplikationstabellen speichern Referenzen zwischen Hashes und Daten
  - SHA256-Hash (256 Bit = 32 Byte)
  - 8 KB große Dateisystemblöcke (mit Offsets der Größe 8 Byte)
  - Zusätzlicher Overhead von 8 Byte pro Hash
- Für effiziente Online-Deduplikation müssen sie im Arbeitsspeicher gehalten werden
  - Duplikate müssen bei jeder Schreiboperation gesucht werden
  - Schnelle Speichergeräte (SSDs) sind immer noch um Größenordnungen langsamer
    - NVRAM unter Umständen geeignet, um Tabellen zu speichern

$$1\,\mathrm{TB} \div 8\,\mathrm{KB} = 125.000.000$$

$$125.000.000 \cdot (32\,\mathrm{B} + 8\,\mathrm{B} + 8\,\mathrm{B}) = 6\,\mathrm{GB} \quad (0,6\,\%)$$

Analyse [5]

	2009	2015	2020	2025
Speicher	5,6+**1,68** PB	45+**13,5** PB	270+**81** PB	1,6+**0,48** EB
RAM	20+**33,6** TB	170+**270** TB	1,5+**1,62** PB	12,8+**9,6** PB
Strom	1,6+**0,24** MW	1,4+**0,20** MW	1,4+**0,14** MW	1,4+**0,09** MW
Kosten	30+**2,52** M€	30+**2,38** M€	30+**1,62** M€	30+**1,13** M€

- Optimistische Annahme von 30 % Einsparung
- Benötigt mehr zusätzlichen RAM als für Berechnung vorhanden ist (außer 2025)
- Benötigt deutlich mehr Strom (5–15 %)
- Erhöht Gesamtkosten (3–8 %)

Analyse... [5]

2009	2015	2020	2025
4,3+**1,3** PB	34,6+**10,4** PB	207,7+**62,3** PB	1,2+**0,4** EB
20+**25,8** TB	170+**207,7** TB	1,5+**1,2** PB	12,8+**7,4** PB
1,54+**0,19** MW	1,34+**0,15** MW	1,34+**0,1** MW	1,34+**0,07** MW
28,27+**1,94** M€	28,27+**1,83** M€	28,27+**1,25** M€	28,27+**0,87** M€

- Deduplikation wird genutzt, um selbe Kapazität zu erreichen
- Benötigt immer noch deutlich mehr Arbeitsspeicher
- Stromverbrauch erhöht sich (bis zu 8 %)
- Gesamtkosten sinken ab 2020

Zusammenfassung

- Größere Blöcke senken den Overhead
  - 8 KB → 0,6 %, 16 KB → 0,3 %, 32 KB → 0,15 %
  - Einfluss auf Deduplikationsrate muss beachtet werden
- Deduplikation ganzer Dateien
  - E/A-Durchsatz bleibt gleich
  - Dateien müssen immer erst komplett geschrieben werden
- Offline-Deduplikation
  - Einfacher möglich mit modernen Copy-on-Write-Dateisystemen
  - Nützlich für Deduplikation ganzer Dateien
  - Nicht ganz so leistungskritisch
    - Tabellen müssen nicht immer im RAM gehalten werden

Überblick [1]

- Ziel: Erfassung der wichtigsten Leistungsmetriken unterschiedlicher Kompressionsalgorithmen
  - Kompressionsrate, Prozessorlast, Stromverbrauch und Laufzeit
- $\approx$ 500 GB Klimadaten (MPI-OM)
  - Vorabtests mit sich wiederholenden und zufälligen Daten
  - Serielle Tests für Grundleistung
  - Parallele Tests für echte Anwendungen

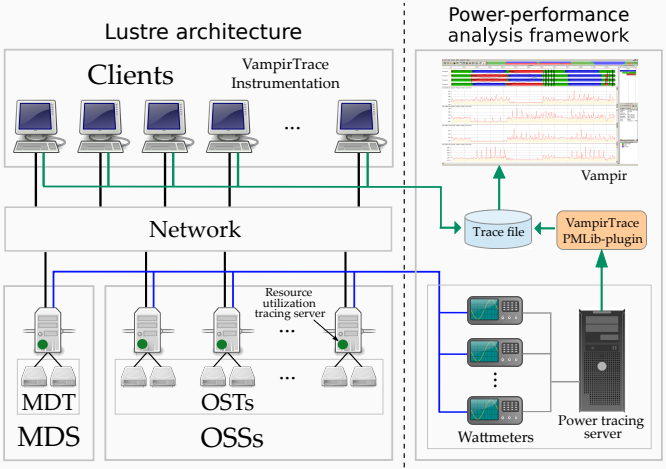


Abbildung 2: Framework zur Messung der Leistung und des Stromverbrauchs

Tracing... [1]

- Normale Lustre-Installation
  - Clients und Server auf unterschiedlichen Maschinen
- Zusätzliche Instrumentierung
  - VampirTrace für Client-Anwendungen
  - pmserver für Dateisystemserver
  - Server zur Erfassung des Stromverbrauchs
    - Verbunden mit Strommessgeräten
- pmlib-Plugin erlaubt Korrelation von Client- und Serveraktivität

Algorithmen [1]

Kompressions-algorithmus	Kompressions-rate	Prozessor-auslastung	Laufzeit-verhältnis
`none`	1,00	23,7	1,00
`zle`	1,13	23,8	1,04
`lzjb`	1,57	24,8	1,09
`lz4`	1,52	22,8	1,09
`gzip-1`	2,04	56,6	1,06
`gzip-9`	2,08	83,1	13,66

Tabelle 1: Leistungsdaten für Klimadaten

- Laufzeit erhöht sich nur leicht (außer für höhere `gzip`-Level)
- `gzip` erhöht Prozessorauslastung deutlich
- `lz4` (und `gzip-1`) am interessantesten

Algorithmen... [1]

Kompressions-algorithmus	Kompressions-rate	Prozessor-auslastung	Laufzeit-verhältnis
none	1,00	23,7	1,00
lz4	126,96	15,8	1,28
gzip-1	126,96	23,3	1,24

Tabelle 2: Leistungsdaten für sich wiederholende Daten

- Mit Hilfe des yes-Kommandos erzeugt
- lz4 erzeugt niedrigere Prozessorlast als keine Kompression
- Beide Algorithmen erhöhen die Laufzeit um ca. 25 %

Algorithmen... [1]

Kompressions-algorithmus	Kompressions-rate	Prozessor-auslastung	Laufzeit-verhältnis
none	1,00	23,5	1,00
lz4	1,00	24,1	0,97
gzip-1	1,00	66,1	1,03

Tabelle 3: Leistungsdaten für zufällige Daten

- Mit Hilfe des `frandom`-Kernelmoduls erzeugt
- `gzip-1` benötigt deutlich mehr Prozessorzeit
- Beide Algorithmen haben kaum Einfluss auf die Laufzeit
  - Erinnerung: Serieller Test mit einer Festplatte

Parallele Anwendung [1]

Kompressions-algorithmus	Laufzeit-verhältnis	Strom-verhältnis	Energie-verhältnis
none	1,00	1,00	1,00
lz4	0,92	1,01	0,93
gzip-1	0,92	1,10	1,01

- IOR-Benchmark, angepasst für realistische Schreibaktivität
- Anwendungsleistung wird nicht beeinträchtigt
  - Höherer E/A-Durchsatz auf den Servern
- Energieverbrauch bei `lz4` niedriger
  - Niedrigere Laufzeit und nur leicht erhöhter Stromverbrauch
- Sogar `gzip-1` erhöht den Energieverbrauch nur um 1 %

Analyse [5]

	2009	2015	2020	2025
Speicher	5,6+**2,8** PB	45+**22,5** PB	270+**135** PB	1,6+**0,8** EB
Strom	1,6+**0,025** MW	1,4+**0,025** MW	1,4+**0,025** MW	1,4+**0,025** MW

Tabelle 4: Vor- und Nachteile von Kompression

- Angenommene Kompressionsrate von 1,5 für `lz4`
- Pessimistische Annahme von 10 % höherem Stromverbrauch
- Laufzeitverhältnis von 1,0
  - Nicht notwendig zusätzliche Prozessoren zu beschaffen

Zusammenfassung

- Kompression kann Speicherkapazität deutlich erhöhen
  - Passende Algorithmen haben vernachlässigbaren Overhead
- Oft keine zusätzliche Hardware notwendig
- Geringer zusätzlicher Stromverbrauch
  - Insgesamt trotzdem lohnenswert
- Anwendungsspezifische Kompression kann Kompressionsraten deutlich erhöhen
  - Dadurch verlustbehaftete Kompression möglich
  - Kompressionsraten $\geq$ 10 sind möglich

Überblick

- Kompression im Dateisystem kann bereits genutzt werden
  - Lustre unterstützt ein ZFS-Backend
  - Kompression kann einfach in ZFS aktiviert werden
- Momentan allerdings nur statische Ansätze für Kompression
  - Ein Kompressionsalgorithmus pro Dateisystem
  - Dynamischere Ansätze könnten effizienter komprimieren
- Semantische Informationen zur Verbesserung der Kompression
  - Adaptive Kompression muss auch raten
  - Effizientere anwendungsspezifische Kompression

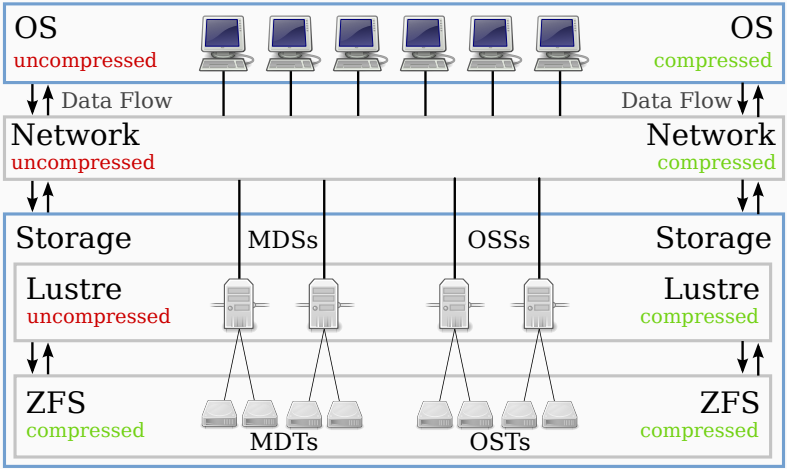


Abbildung 3: Lustre-Architektur mit Kompression

Unterstützung

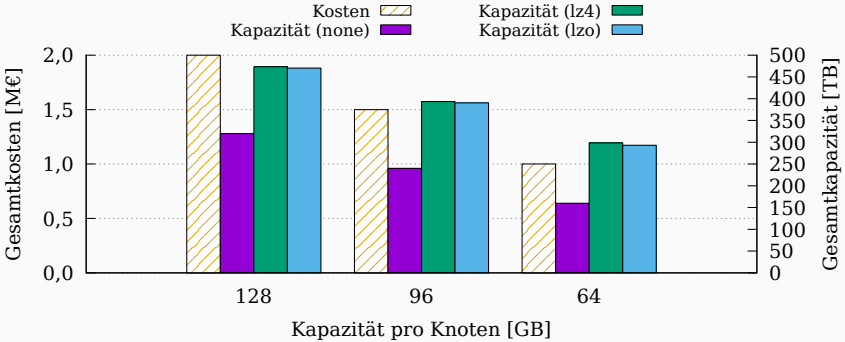
- Unterstützung auf mehreren Ebenen denkbar
  - Jeder Ansatz hat Vor- und Nachteile
  - Kompression auf dem Client beeinflusst die Berechnung, kann aber auch Netzwerkdurchsatz erhöhen
- Bisher keine Unterstützung für clientseitige Kompression
  - Komplett transparent für Awendungen
  - Konfigurierbar mittels `ladvise`
- Kompression ist statisch
  - Nutzung von Informationen über die Daten, die aktuelle Last etc.
  - Sowohl auf Clients als auch Servern nützlich

Kompression: Übersicht [4]

Algorithmus	Kompression	Dekompression	Rate
lz4fast	2.945 MB/s	6.460 MB/s	1.825
lz4	1.796 MB/s	5.178 MB/s	1.923
lz4hc	258 MB/s	4.333 MB/s	2.000
lzo	380 MB/s	1.938 MB/s	1.887
xz	26 MB/s	97 MB/s	2.632
zlib	95 MB/s	610 MB/s	2.326
zstd	658 MB/s	2.019 MB/s	2.326

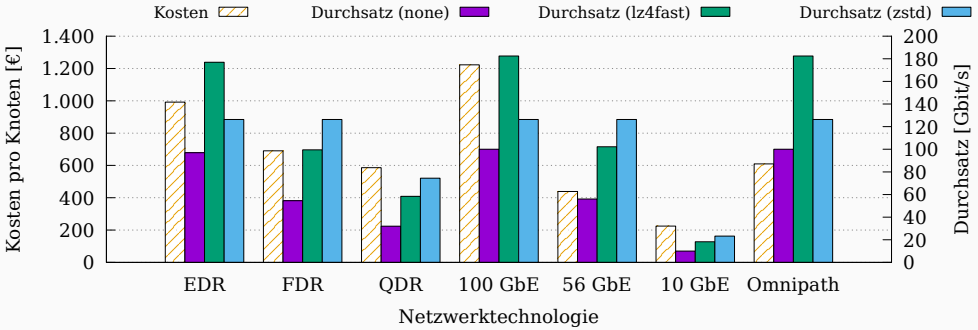
- lz4 und lz4fast allgemein sehr gut, zstd auch interessant
  - lz4fast und zstd können über Parameter angepasst werden
- Mehrere gute Kandidaten für Archivierung

Kompression: Arbeitsspeicher [4]



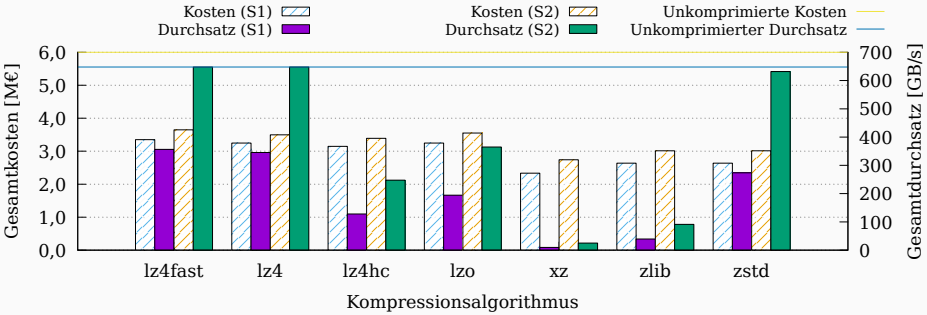
- Schätzungen mit zram
- Ziel: Kapazität pro Knoten von 128 GB
  - Nicht möglich mit 64 GB
  - 60 GB komprimiert, 4 GB unkomprimiert

Kompression: Netzwerk [4]



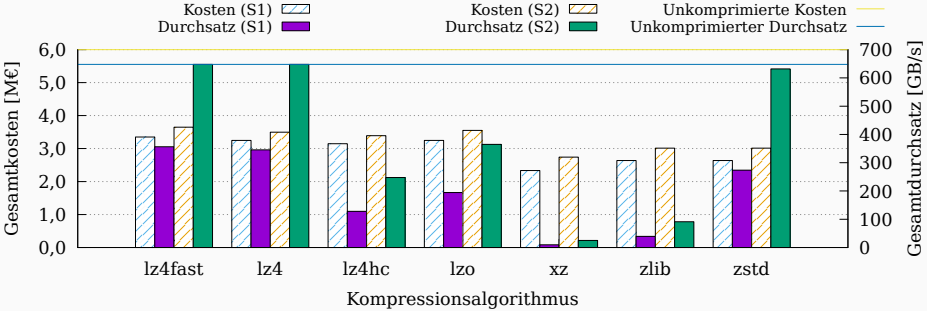
- zstd reduziert den Durchsatz bei Netzwerken mit hohem Durchsatz ($>$ 54 Gbit/s)
- FDR-InfiniBand kann mit QDR-InfiniBand ersetzt werden, wenn lz4fast genutzt wird (Kostenreduktion 15 %)
- Durchsatz kann mit lz4fast bzw. zstd auf 100 Gbit/s bzw. 125 Gbit/s erhöht werden

Kompression: Speicher [4]



- S1: So viele Server wie für 50 PB notwendig (geringere Kosten/Durchsatz)
- S2: 50 Server und so viele HDDs wie für 50 PB notwendig (höhere Kosten/Durchsatz)

Kompression: Speicher... [4]



- lz4 und lz4fast beeinflussen Leistung nicht
  - Kosten sinken auf 3.500.000 €
- zstd senkt Durchsatz um 20 GB/s
  - Kosten sinken um 50 % auf 3.000.000 €

Feature-Wunschliste

- Unterstützung für Kompression im parallelen Dateisystem
  - Interaktion mit anwendungsspezifischer Kompression
- Entwickler sollen nützliche Informationen spezifizieren können
  - Zusätzliches Wissen über die Daten (Varianz, Muster etc.)
  - Semantische Informationen im ganzen Stack nutzen
- Datenreduktion in einer zentralen Schicht
  - Momentan implementieren alle Schichten eigene Lösungen
  - Redundante Operationen, falsche Reihenfolge etc.

Adaptive Kompression [2]

- Adaptive Kompression in ZFS
  - Direkt durch Lustre nutzbar
- Unterstützung für unterschiedliche Modi
  - Leistung, Archivierung, Energieverbrauch
- Unterschiedliche Heuristiken zur Bestimmung der Kompression
  - Basierend auf Dateitypen und Kostenfunktionen
- Für Kostenfunktion werden alle Algorithmen getestet
  - Der beste wird für die nächsten Operationen genutzt

Adaptive Kompression... [2]

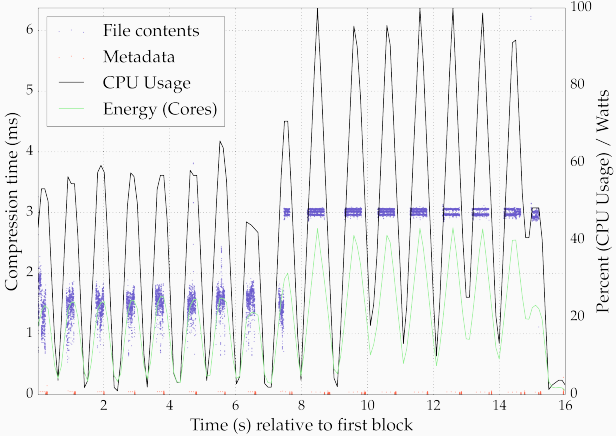


Abbildung 4: Komprimieren einer gemischten Datei mit `gzip-1`

Adaptive Kompression... [2]

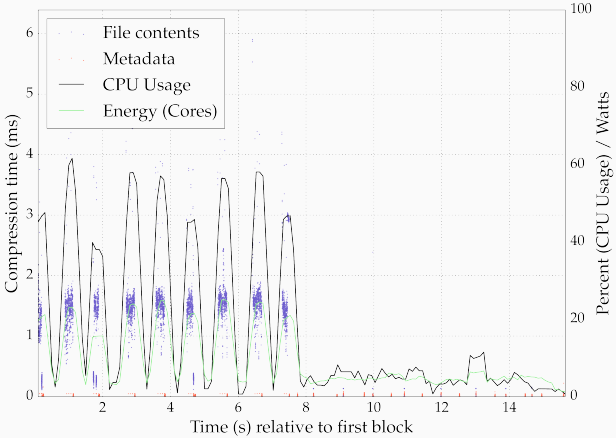
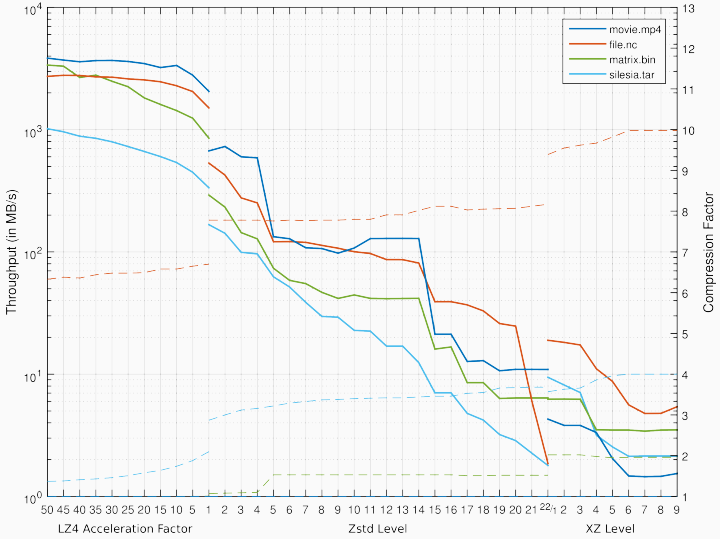


Abbildung 5: Komprimieren einer gemischten Datei mit dem Archivmodus

Adaptive Kompression... [3]



Zusammenfassung

- Wiederberechnung
  - Nicht alle Ergebnisse werden gespeichert
  - Negative Bilanz bei häufiger Wiederberechnung
- Deduplikation
  - Duplikate werden nicht gespeichert, sondern Referenzen auf existierende Blöcke
  - Overhead durch zusätzliche Checks
- Kompression kann die TCO deutlich senken
  - Positiver Effekt auf Arbeitsspeicher und Netzwerkdurchsatz
  - Nützlich für Daten, die nicht explizit durch Anwendungen komprimiert werden

Zusammenfassung...

- Genaue Analyse der Kostenfaktoren und Nutzung notwendig
  - Rechenleistung stieg bisher mit jeder Generation um Faktor 20
  - Speicherkapazität stieg nur um Faktor 8
- Schätzung der Kosten für Berechnung, Speicherung und Archivierung
  - Kostenmodelle für Langzeitarchivierung
  - Speicherung von Daten verursacht die größten Kosten
  - Das Problem wird sich weiter verschlimmern

Quellen

[1] Konstantinos Chasapis, Manuel Dolz, Michael Kuhn, and Thomas Ludwig. **Evaluating Power-Performace Benefits of Data Compression in HPC Storage Servers.** In Steffen Fries and Petre Dini, editors, *IARIA Conference*, pages 29–34. IARIA XPS Press, 04 2014.

[2] Florian Ehmke. **Adaptive Compression for the Zettabyte File System.** Master's thesis, Universität Hamburg, 02 2015.

[3] Janosch Hirsch. **Dynamic decision-making for efficient compression in parallel distributed file systems.** Master's thesis, Universität Hamburg, 08 2017.

[4] Michael Kuhn, Julian Kunkel, and Thomas Ludwig. **Data Compression for Climate Data.** *Supercomputing Frontiers and Innovations*, pages 75–94, 06 2016.

Quellen ...

[5] Julian Kunkel, Michael Kuhn, and Thomas Ludwig. **Exascale Storage Systems – An Analytical Study of Expenses.** *Supercomputing Frontiers and Innovations*, pages 116–134, 06 2014.