Programmierabstraktionen für die Verwaltung von Workflows auf mehrstufigen Speichersystemen

MAJA LINGL

SEMINAR SUPERCOMPUTER: FORSCHUNG UND INNOVATION

UNIVERSITÄT HAMBURG

20. DEZEMBER 2022

Ablauf

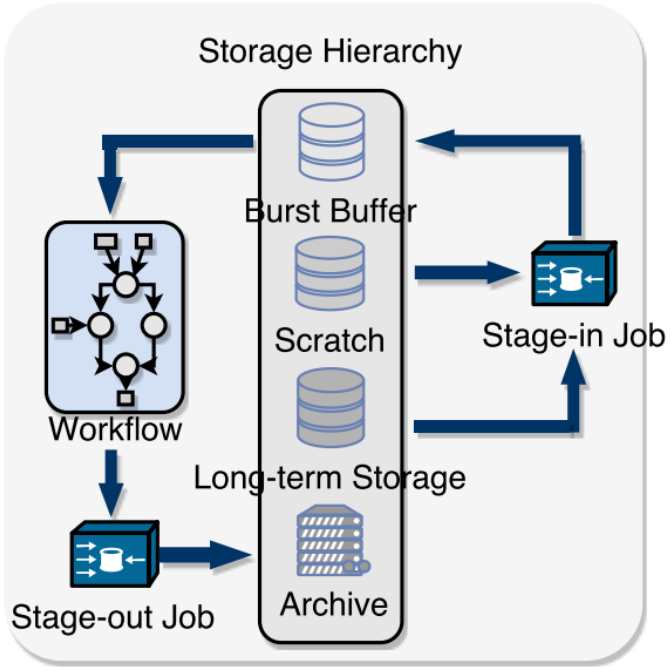
- Vorstellung des Papers
- Mehrstufige Speicherhierarchien
- Workflows
- Managing Data on Tiered Storage
- Programmierabstraktionen in MaDaTS
  - Virtual Data Space
  - Funktion und Nutzen
- Evaluation
- Zusammenfassung

Das Paper

- Veröffentlicht von Devarshi Ghoshal und Lavanya Ramakrishnan, Oktober 2021
- Hauptaussagen:
  - Mithilfe von Middleware Datenverwaltung abstrahieren, um Komplexität zu reduzieren
  - Datenzentrierte Workflows verwenden, um Spalt zwischen Datenverwaltung und Ausführung zu schließen

Mehrstufiger Speicher

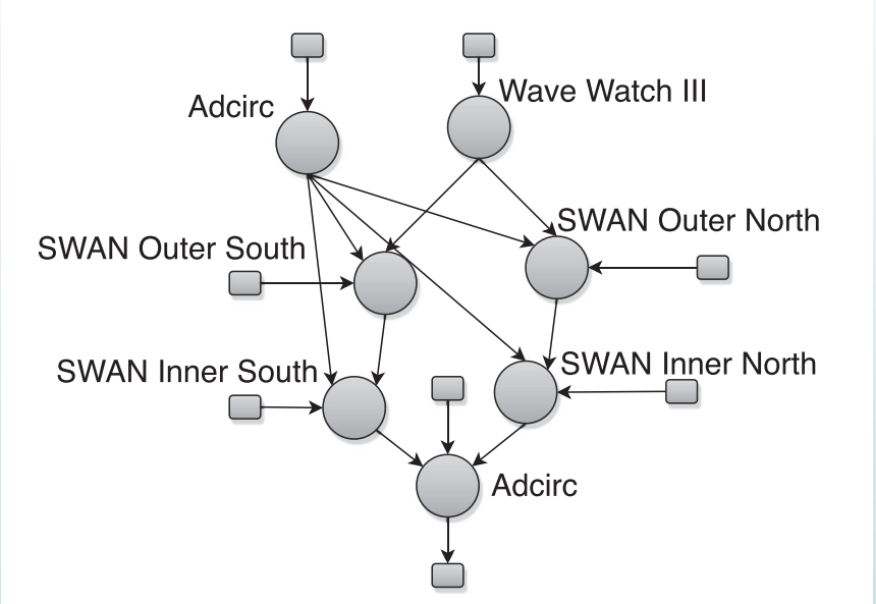
- Speicherebenen sind nach Zugriffshäufigkeit und Geschwindigkeit absteigend angeordnet
  - Verschiedene Ebenen haben verschiedene Namensräume
- Staging von Daten auf andere Ebenen soll minimiert werden



Traditionelle Workflow-Ausführung. Quelle 2

Workflows

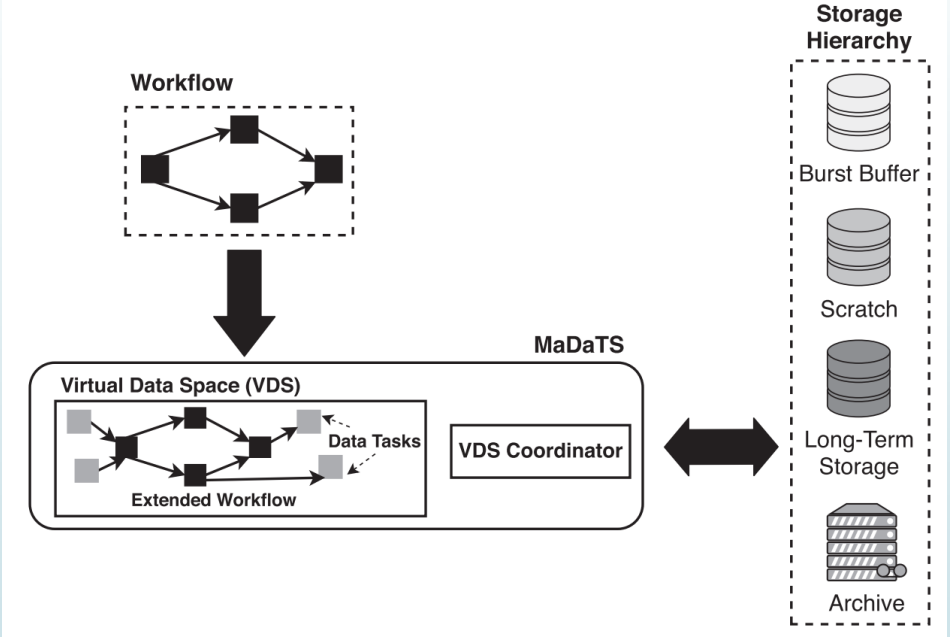
- Wissenschaftliche Arbeitsabläufe werden als prozesszentrierte gerichtete azyklische Graphen dargestellt
  - Datenbewegungen sind kein Teil der Workflow-Aufgaben
- Stattdessen: datenzentrierte Workflows nutzen



Workflow „Floodplain Mapping“. Quelle 1

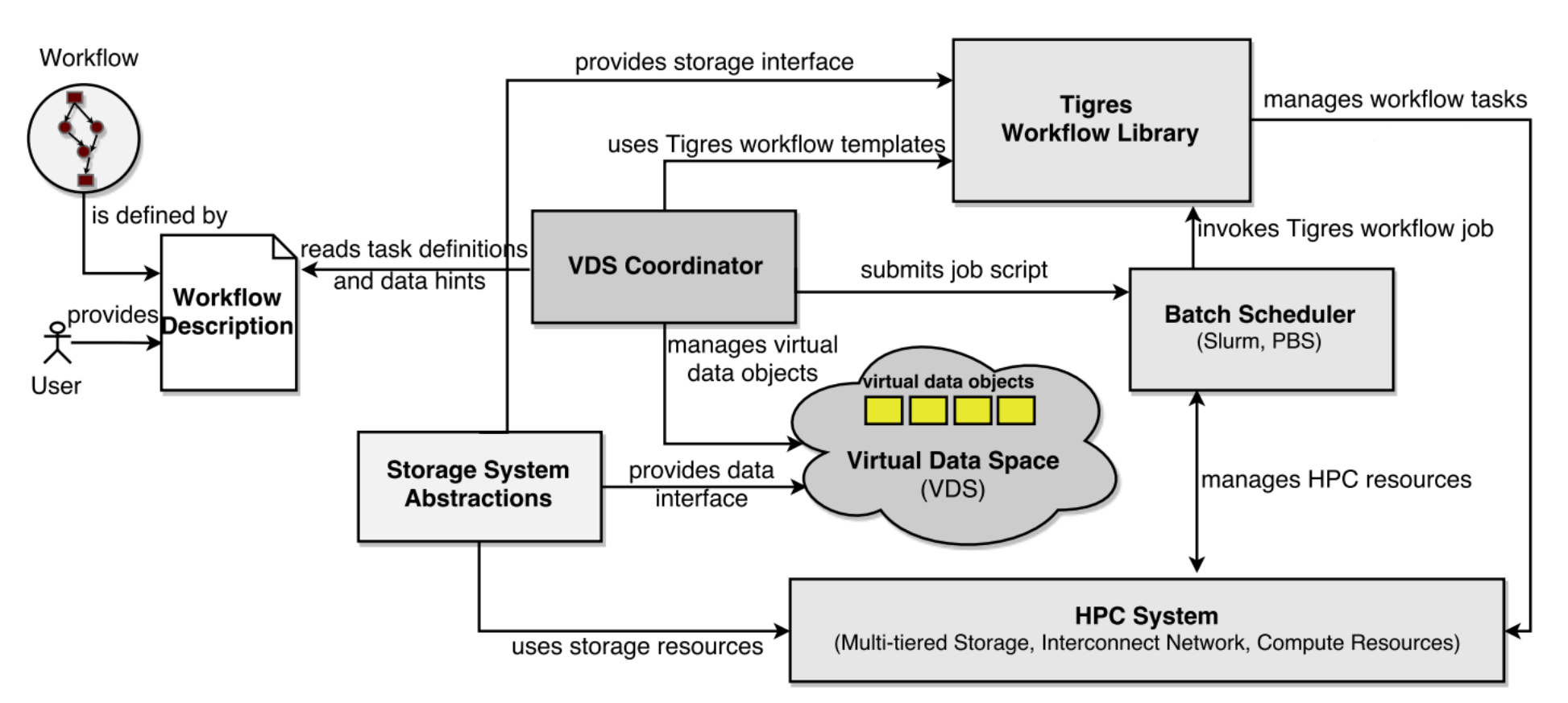
MaDaTS – Überblick

- Softwarearchitektur von Goshal und Ramakrishnan
- Erstellt nach festgelegter Strategie einen erweiterten datenzentrierten Daten-verwaltungsplan
  - Mithilfe eines Virtual Data Space VDS die Komplexitäten des Speichers abstrahieren



MaDaTS Aufbau. Quelle 1

MaDaTS – Funktion



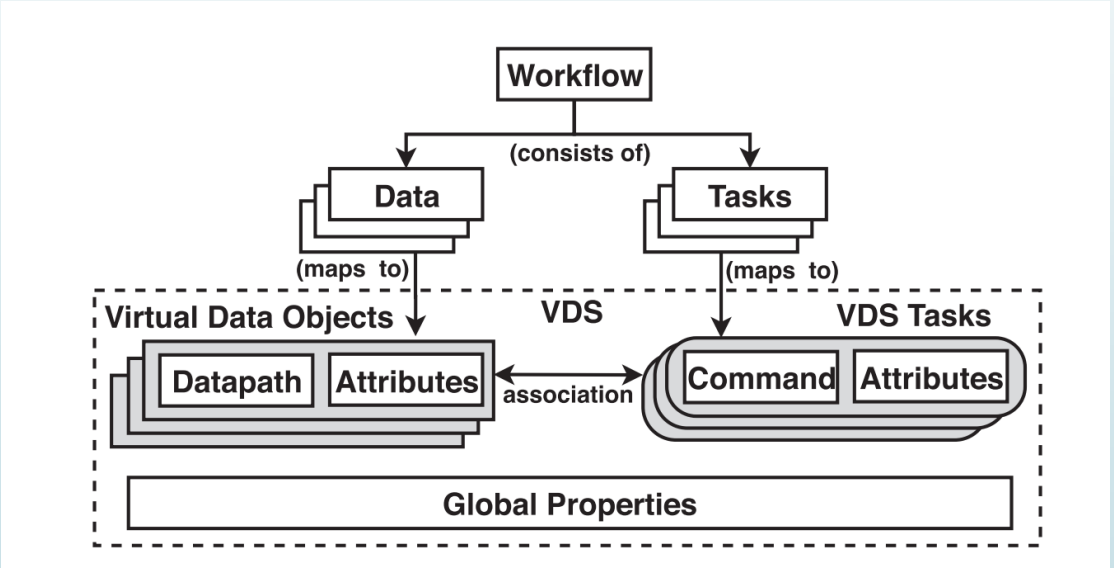
MaDaTS Architektur. Quelle 2

MaDaTS – Funktion

- Vordefinierte Datenverwaltungsstrategien
  - Passiv oder benutzerdefiniert
  - Workflow-aware
  - Storage-aware
- Übergebene Dateninformationen (z.B. Persistenz, Größe) dienen zur Zuordnung der Daten
- Abstraktion auf zwei Ebenen: Speichersystemabstraktion und durch VDS

Virtual Data Space

- Vereinfachte Darstellung der genutzten Daten aus dem Speicher bietet bessere Verwaltungsmöglichkeiten
- Daten aus dem Workflow werden auf Virtuelle Datenobjekte (VDO) abgebildet
  - Für jeden Workflow wird ein VDS erstellt
  - wird nach Abschluss gelöscht



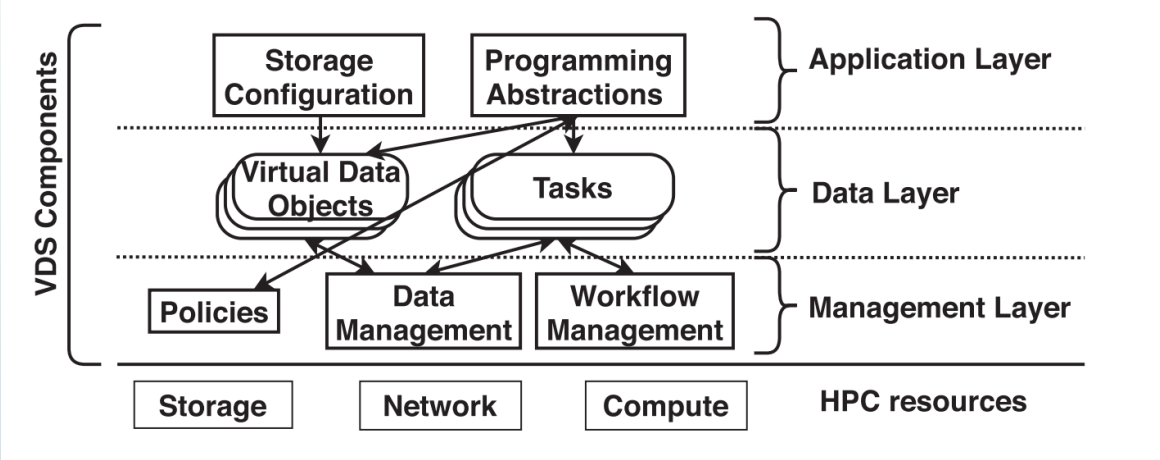
VDS-Mapping in MaDaTS. Quelle 1

Virtual Data Object

- Eindeutiger Bezeichner wird auf Basis des Speicherpfads des Datenelements festgelegt
- Ein VDO hat Produzenten und Konsumenten
- Verschieben von Daten im Speicher wird durch VDOs dargestellt

VDS-Aufbau

- VDS hat drei Ebenen
  - Anwendungsebene
  - Datenebene
  - Verwaltungsebene
    - Verwaltet Lebenszyklus eines VDO



VDS Komponente. Quelle 1

VDS

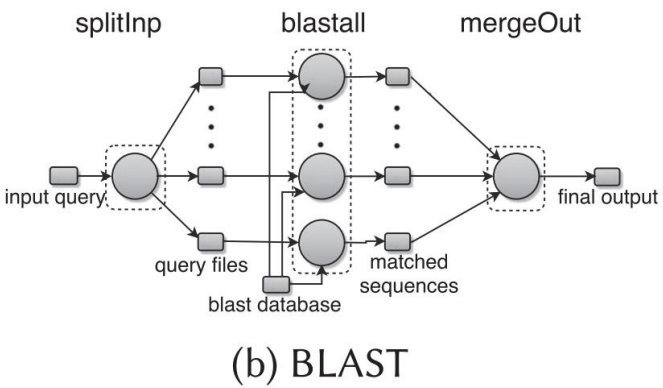
- Bereitet Datenausführungsplan vor, mit gegebenen Abstraktionen
- Hat verschiedene Konfigurationen, die für verschiedene Prozesse geeignet sind
  - Default
  - Clean up: löscht Zwischendaten
  - Persist: speichert Zwischendaten
  - Clean up + Persist

Abstraction
$task$ = **Task**($command$)
vdo = **VirtualDataObject**($dataset$) vdo.**add_producer**($task$) vdo.**add_consumer**($task$)
vds = **VirtualDataSpace**() vdo = vds.**map**($dataset$) vds.**add**(vdo)
dag = **get_workflow_dag**(vds)
manage(vds)
validate(vds)

Programmierabstraktionen MaDaTS. Quelle 1

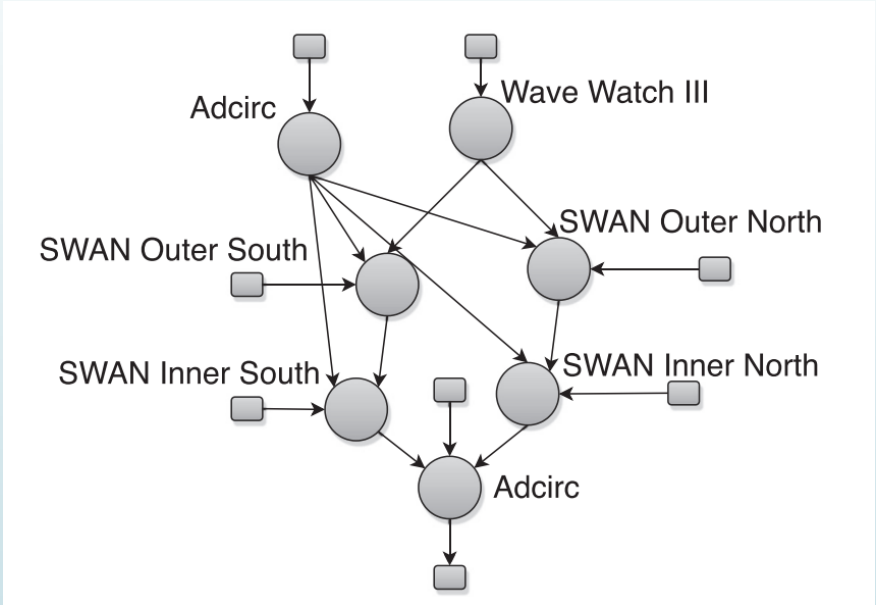
Evaluation

- Wurde auf NERSC‘s Supercomputer Cori vorgenommen
- Jeweils drei verschiedene Workflows verglichen
  - Montage: datenintensiv
  - Blast: rechenintensiv
  - Floodplain Mapping: I/o-intensiv
- Vergleich mit handoptimierem Skript, Datenbewegung am Anfang und Ende des Workflows

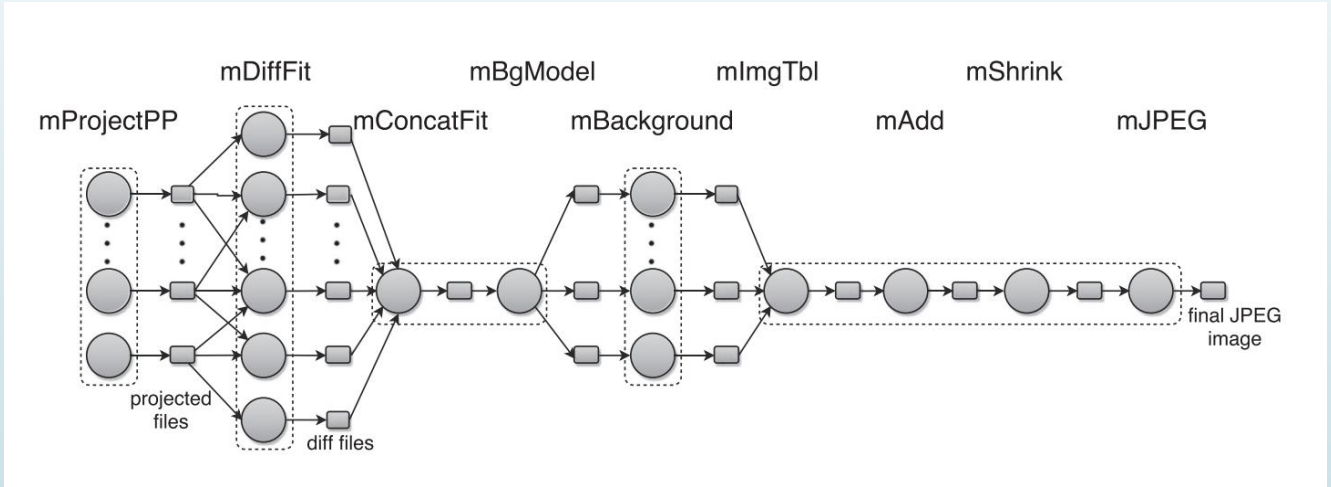


Workflow „Blast“. Quelle 1

Evaluation



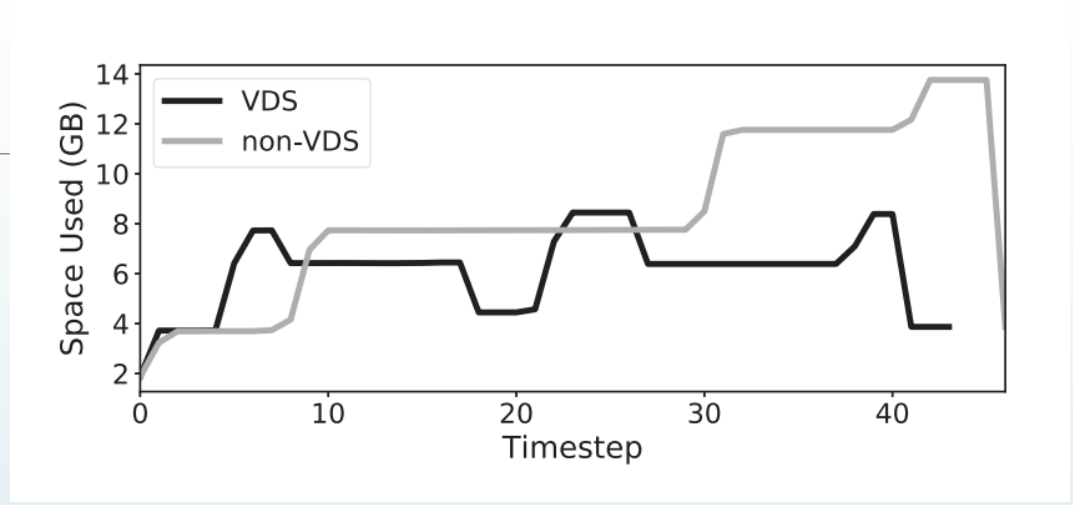
Workflow „Floodplain Mapping“. Quelle 1



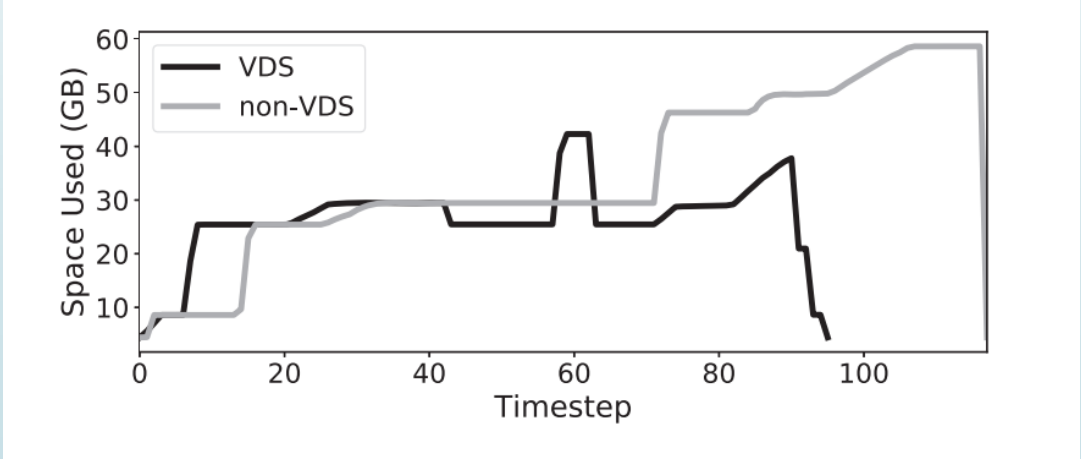
Workflow „Montage“. Quelle 1

Evaluation – Speicherkapazität

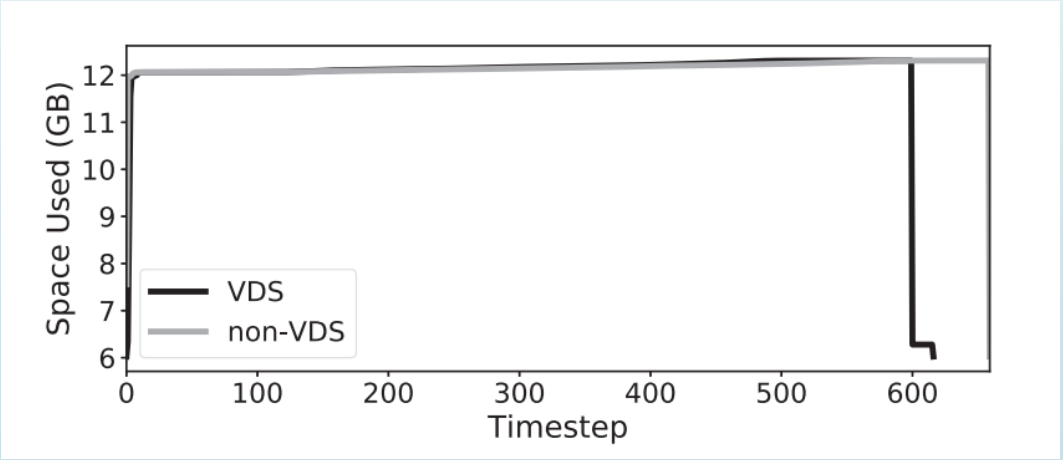
- Zwischendaten werden immer wieder gelöscht sofern möglich
  - Reduziert Speichernutzung bis zu 40%



Speichernutzung „Floodplain Mapping“. Quelle 1



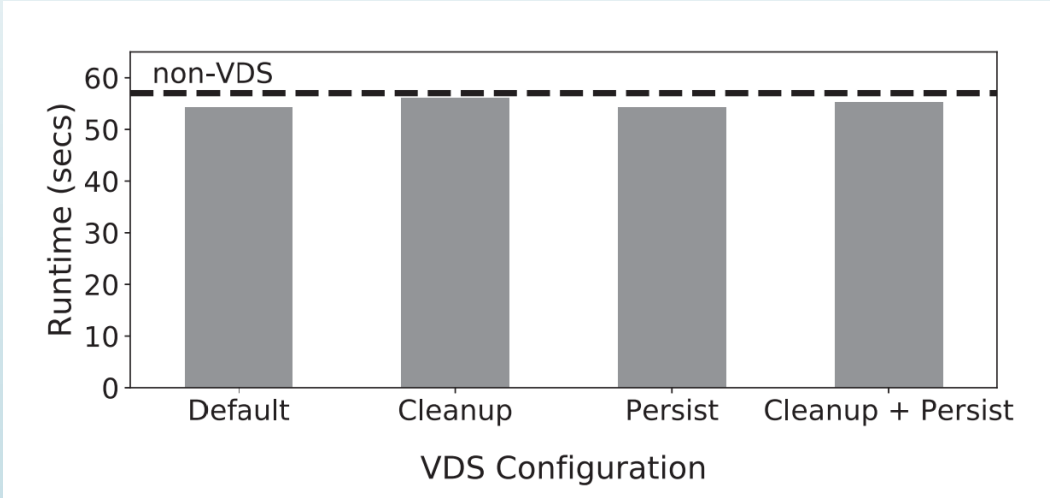
Speichernutzung „Montage“. Quelle 1



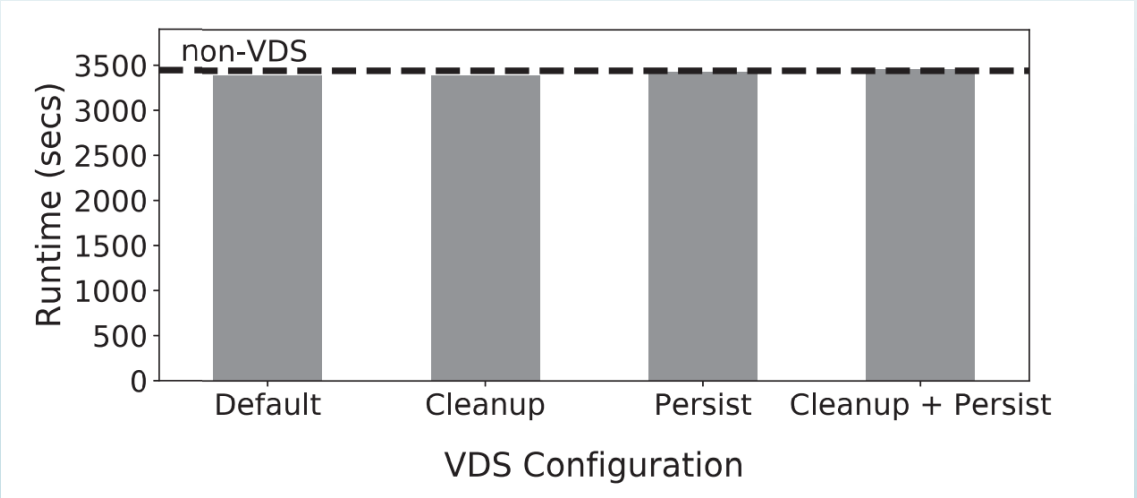
Speichernutzung „Blast“. Quelle 1

Evaluation – Laufzeit

- Laufzeit bei einigen Einstellungen schneller (Floodplain Mapping und Montage)
- Bei rechenintensivem Blast wenig Änderungen
- Clean Up hat keine verkürzte Laufzeit, da es auf Speicheroptimierung ausgelegt ist



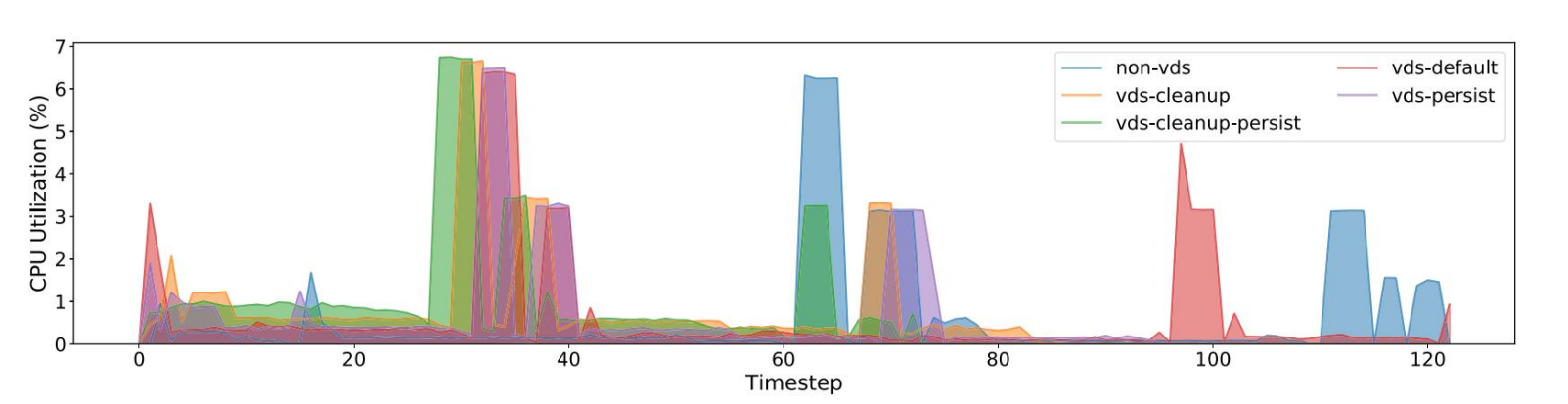
Laufzeit „Floodplain Mapping“. Quelle 1



Laufzeit „Blast“. Quelle 1

Evaluation – CPU-Nutzung

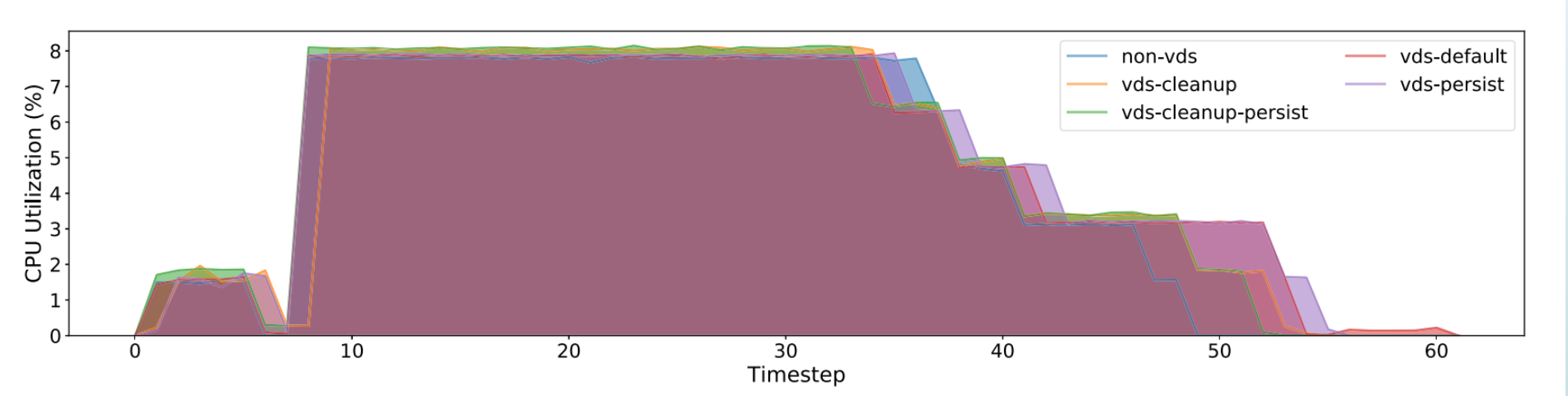
- Leichter Anstieg von bis zu 1,5%
  - Durch zusätzlich kreierte Datenaufgaben, insbesondere bei Clean Up + Persist
  - Wenig Belastung, da es hauptsächlich I/O- Aufgaben sind.



CPU-Nutzung „Floodplain Mapping“. Quelle 1

Evaluation – CPU-Nutzung

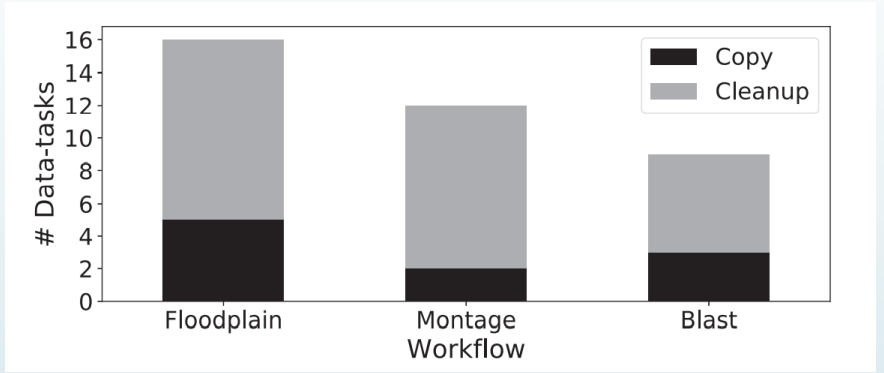
- Blast ist rechenintensiver, die Belastung ist durchgehend höher
  - Auswirkungen der VDS Konfigurationen bleiben jedoch ähnlich



CPU-Nutzung „Blast“. Quelle 1

Evaluation – Datenaufgaben

- Worst-case der nicht-VDS Version schlechter als alle VDS-Konfigurationen
- Viele Datenaufgaben bei Clean Up, haben wenig Auswirkungen auf Laufzeit



Anzahl Datenaufgaben. Quelle 1

Workflow	Data-tasks/Data-movement-operations					
	VDS				non-VDS	
	Default	Cleanup (C)	Persist (P)	C+P	Best-case	Worst-case
Floodplain	6/5	17/5	12/11	23/11	-/5	-/16
Montage	3/2	13/2	10/9	20/9	-/2	-/10
Blast	4/3	10/3	6/5	12/5	-/3	-/7

Datenaufgaben und –bewegungen mit und ohne VDS. Quelle 1

Evaluation – Programmierkomplexität

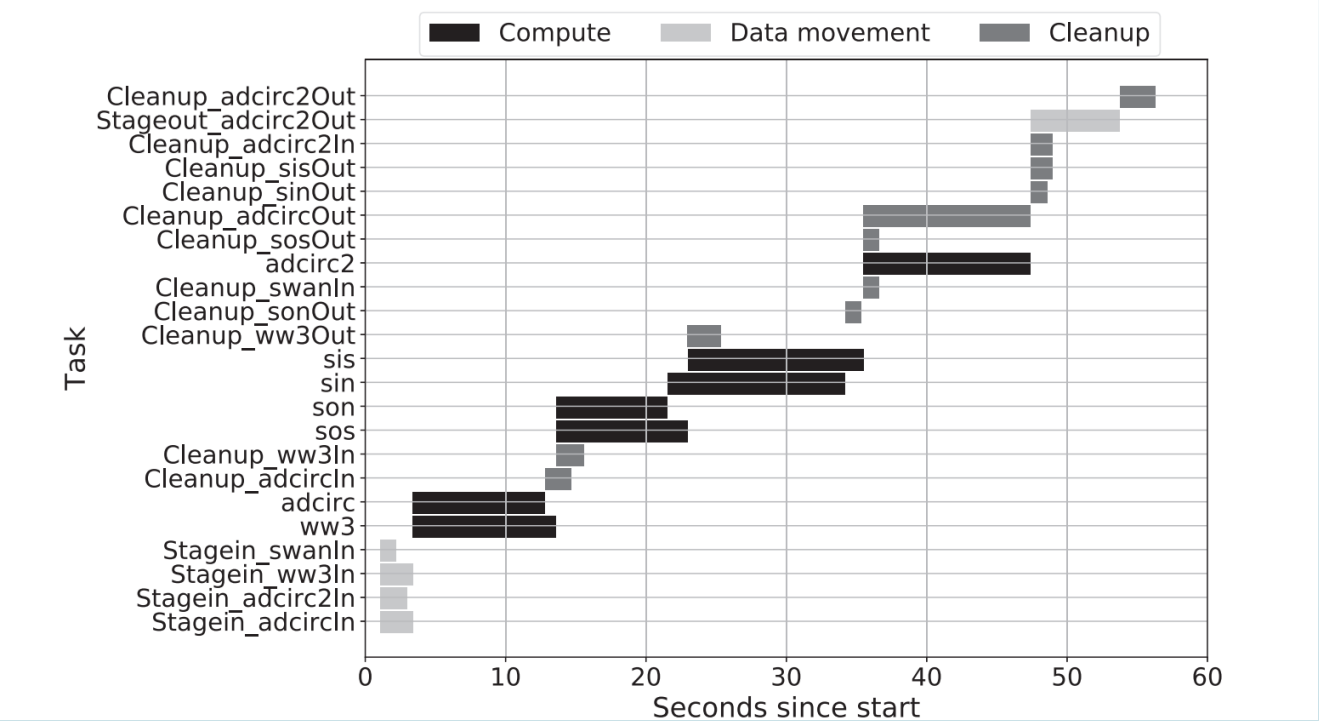
- Dargestellt anhand der Menge an Workflow Skripte und der Lines of Code
- Eingebundene Tigres Programme sind nicht in LoC enthalten
- MaDaTS benötigt nur ein Job Skript zur Verwaltung

Workflow	Workflow scripts		Total LoC	
	MaDaTS	Original	MaDaTS	Original
Floodplain	1	8	86	97
Montage	1	10	78	102
Blast	1	4	48	37

Programmierkomplexität mit und ohne MaDaTS. Quelle 1

Evaluation – Datenverarbeitung

- Gantt Diagramm stellt die Überlagerung von Aufgaben dar
  - Steht im Gegensatz zu Stage In und Out vor- und nachher



Aufgabenausführung „Floodplain Mapping“ Diagramm. Quelle 1

Zusammenfassung

- MaDaTS abstrahiert die Verwaltung von Daten, um die Effizienz zu erhöhen
- Je nach Art des Workflows sind verschiedene VDS-Konfigurationen hilfreich
- Generell:
  - Abstraktion hilft Komplexität zu verbergen und teilweise Automatisierung der Datenverwaltung zu ermöglichen

Quellen

- [Quelle 1, Programmierabstraktionen] Devarshi Ghoshal and Lavanya Ramakrishnan. 2021. Programming Abstractions for Managing Workflows on Tiered Storage Systems. ACM Trans. Storage 17, 4, Article 29 (October 2021), 21 pages. https://doi.org/10.1145/3457119
- [Quelle 2, MaDaTS] Devarshi Ghoshal and Lavanya Ramakrishnan. 2017. MaDaTS: Managing data on tiered storage for scientific workflows. In ACM Symposium on High Performance Parallel and Distributed Computing (HPDC'17). ACM Press, 12 pages
- Ciprian Docan, Manish Parashar, and Scott Klasky. 2012. DataSpaces: An interaction and coordination framework for coupled simulation workflows. Cluster Computing 15, 2 (2012).
- https://docs.nersc.gov/systems/cori/