Master Thesis - Exposé

Robin Knapp

20. Januar 2023

1 Einleitung

Die Analyse großer Datenmengen ist ein zentraler Bestandteil wissenschaftlicher Forschung. Hierbei werden regelmäßig unterschiedliche Softwarekomponenten miteinander verknüpft. Die verknüpften Softwarekomponenten bilden einen sogenannten Data Analysis Workflow (DAW). Da bei der Ausführung eines DAW große Datenmengen verarbeitet werden, ist eine entsprechende Computer Infrastruktur notwendig. Um die Ausführung eines DAW möglichst unabhängig von der verwendeten Infrastruktur zu machen, werden sie auf Computer Clustern ausgeführt. Der Vorteil an der Ausführung des DAW auf einem Computer Cluster ist, dass Standardhardware verwendet werden kann. Dem zugrunde liegt die Annahme, dass ein DAW sonst nicht unabhängig von der Computer Infrastruktur wiederholt werden kann. Forschung, die auf einem DAW beruht, könnte somit ohne Zugriff auf eine bestimmte Computer Infrastruktur nicht wiederholt oder weiterentwickelt werden [10].

Damit der Workflow jedoch in einem Computer Clustern d.h. einem verteilten System ausgeführt werden kann, ist eine Koordination der einzelnen Verarbeitungsschritte (Jobs) notwendig. Hierzu gehören der Datenaustausch zwischen den einzelnen Verarbeitungsschritten sowie die Anwendung von Parallelisierungsstrategien. Diese sind erforderlich, um die verfügbaren Ressourcen im Computer Cluster sinnvoll zu nutzen. Durch die Anwendung entsprechender Parallelisierungsstrategien zerfallen die

Jobs in kleinere Verarbeitungsschritte (Task). Diese Tasks können dann parallel auf den Knoten des verteilten Systems ausgeführt werden. Hierzu werden sogenannte Scientific Workflow Management Systeme (SWMS) wie Pegasus [4] oder Nextflow [5] eingesetzt. SWMS reduzieren den Aufwand für die Ausführung eines DAW in einer verteilten Umgebung, indem sie Funktionalitäten, wie etwa für die Umsetzung von Parallelisierungsstrategien, implementieren. In der Folge tragen SWMS zur Reproduzierbarkeit von DAWs bei. Die SWMS nutzen Ressourcenmanager wie Kubernetes [3] oder Slurm [13], um den einzelnen Tasks den Ressourcen im Computer Cluster zuzuweisen [1].

In einem vorangegangenen Studienprojekt "FORCE on Argo" wurde der in [9] untersuchte DAW auf das SWMS Argo Workflow[1] (Argo) portiert, um die Portierbarkeit des DAWs zu untersuchen. Im Rahmen des Projekts wurden Probleme mit einem Verarbeitungsschritt des DAW festgestellt. So nutzt der Argo Workflow ein verteiltes Dateisystem auf dem die einzelnen Tasks auf Eingabedaten zugreifen sowie ihre Ergebnisse speichern. Es ließ sich feststellen, dass bei der Ausführung eines I/O-intensiven Jobs gehäuft Fehler auftreten. Die Fehler stehen im Zusammenhang mit der Synchronisation zwischen den Tasks, die zeitgleich auf Daten zugreifen. Darüber hinaus ließ sich auch feststellen, dass Knoten im Kubernetes Cluster ausfallen. Dieses wird auf eine Häufung der I/O-Operationen zurückgeführt, die bei der Verwendung des verteilten Dateisystems zu einer hohen Auslastung des Netzwerks führen. Dieser Umstand führt dazu, dass der DAW keine korrekten Ergebnisse liefert.

2 Zielsetzung

Die avisierte Master-Thesis untersucht ein Szenario, in dem I/O-intensive Tasks um die Ressourcen eines Kubernetes Cluster konkurrieren. Hierbei wird speziell untersucht, wie Tasks bei der Verwendung eines verteilten Dateisystems die verfügbare Netzwerkkapazität auslasten. Im Rahmen der Master-Thesis wird am Beispiel des "FORCE on Argo" Workflows untersucht, wie das SWMS Argo Ressourcenkonflikte im Sinne einer besseren

[1] `https://argoproj.github.io/argo-workflows/`

Portierbarkeit von DAWs handhaben kann. So soll zur Forschung bzgl. der Reproduzierbarkeit von DAWs beigetragen werden. Zusätzlich können gewonnene Erkenntnisse über auftretende Ressourcenkonflikte in die Entwicklung von Modellen zum Predictive Performance Modeling (PPM) einfließen.

3 Forschungsstand

Witt et al. [12] untersuchen in ihrer Erhebung zum PPM Verfahren, die eine Aussage über die benötigten Ressourcen eines Workflows ermöglichen. Hierbei werden statistische Modelle und maschinelles Lernen eingesetzt. Auf Basis verschiedener Eingabedaten werden Parameter wie etwa die Ausführungszeit einzelner Tasks bestimmt. Mit entsprechenden Informationen über den Ressourcenbedarf und die Ausführungszeit der einzelnen Tasks können Scheduler entwickelt werden, die Workflows etwa in kürzerer Zeit ausführen oder die verfügbaren Ressourcen optimal ausnutzen. Ohne die Anwendung von PPM wird der Ressourcenbedarf bspw. auf Basis vorheriger Ausführungen des Workflows geschätzt. Ferner geben die Autoren eine Übersicht über Szenarien, die zu Ressourcenkonflikten in einem Cluster führen. Etwa indem mehrere Workflows auf einem Knoten um die verfügbaren Ressourcen konkurrieren. Oder nicht alle Knoten in einem Cluster über eine identische Ausstattung mit Ressourcen verfügen. Dieses bezieht sich nicht nur auf die Dimensionierung der jeweiligen Ressourcen bspw. die Anzahl der verfügbaren Prozessoren. Vielmehr werden auch die Eigenschaften der verfügbaren Prozessoren betrachtet, etwa indem Knoten über eine neuere Prozessorgeneration verfügen als andere Knoten im Cluster [12].

Die Verwaltung von Ressourcen durch den Ressourcenmanager Kubernetes ist Gegenstand aktueller Forschung [7, 8]. Hierbei wird unter anderem untersucht, wie gut Kubernetes die Anwendungen von einander isoliert. Kubernetes verfügt über Mechanismen um die Ressourcen zu limitieren, welche einer Anwendung zur Verfügung stehen[2]. Die Li-

[2] `https://kubernetes.io/docs/concepts/configuration/manage-resources-containers`

mitierung kann zur Zeit für CPU und RAM konfiguriert werden. Eine Begrenzung der verfügbaren Netzwerkbandbreite ist als experimentelles Feature nutzbar[3]. Eine Begrenzung von I/O-Operationen befindet sich in der Entwicklung[4].

Kim et al. [8] untersuchen wie gut die Kubernetes Ressourcenverwaltung unterschiedliche Anwendungen von einander isoliert. Hierbei betrachten die Autoren Container mit einem Netzwerk-intensiven Anwendungsfall sowie einen Container mit CPU-intensiven Anwendungsfall. Die Autoren stellen fest, dass die Begrenzung der verfügbaren Netzwerkbandbreite zu einem Anstieg der CPU-Last auf den Kubernetes Systemen führt. Werden gleichzeitig ein CPU-intensiver und ein Netzwerk-intensiver Anwendungsfall auf dem Kubernetes Cluster ausgeführt, stehen durch die Begrenzung der Netzwerkbandbreite weniger CPU-Ressourcen für den CPU-intensiven Anwendungsfall zur Verfügung [8].

Kappes und Anastasiadis [7] untersuchen die Isolation von Anwendungen auf Ebene von I/O-Operationen in Kubernetes Clustern. Die Autoren beschreiben ein Szenario, in welchem verschiedene I/O-intensive Anwendungen auf einem Kubernetes Cluster ausgeführt werden. In dem untersuchten Szenario wird der Cluster von mehreren Mandanten genutzt. Die Mandanten sollen strickt voneinander isoliert sein. Die Trennung der Mandanten ist in Hinblick auf I/O-Operationen jedoch nicht vollständig möglich. Stattdessen beeinflussen sich I/O-intensive Anwendungen gegenseitig und beeinträchtigen die Leistungsfähigkeit der Anwendungen anderer Mandanten [7].

Der Einsatz eines verteiltes Dateisystem im Kontext von DAW wirkt sich auf die Leistungsfähigkeit der Anwendungen aus und muss bei der Bestimmung der benötigen Ressourcen berücksichtigt werden [11]. Matsunaga und Fortes [11] untersuchen wie maschinelles Lernen genutzt werden kann, um die Ausführungszeit sowie den Ressourcenbedarf von Anwendungen zu bestimmen. Hierzu untersuchen die Autoren zwei Anwendungen aus dem Bereich der Bioinformatik. Die Autoren analysieren in ihren

[3] `https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/network-plugins/`

[4] `https://github.com/kubernetes/kubernetes/issues/92287`

Versuchen auch den Speicherort der Eingabedaten. So unterscheiden sie Szenarien in denen die Daten lokal oder in einem verteilten Dateisystem verfügbar sind. Die Autoren stellen fest, dass sich bei der Verwendung eines verteilten Dateisystems die Leistungsfähigkeit der Anwendung nicht mehr linear zu den Ressourcen verhält. Entsprechende Situationen treten auf, wenn mehrere Clients gleichzeitig auf das verteilte Dateisystem zugreifen [11].

Um die Ausführung eines DAW zu überwachen, müssen Informationen aus verschiedenen Systemen ausgewertet werden [2]. Bader al. [2] konzipieren eine Monitoring Architektur für DAW. Die Autoren beschrieben mehrere Ebenen, in denen Informationen und Metriken über den Workflow erhoben werden. Die Ebenen der vorgestellten Monitoring Architektur umfassen die verschiedenen Komponenten des Workflow und verbinden unterschiedliche Informationsquellen etwa dem Ressourcenmanager und das SWMS [2].

4 Vorgehen

Im Rahmen der avisierten Master-Thesis werden Ressourcenkonflikte auf Computer Clustern untersucht. Das Vorhaben betrachtet exemplarisch einen Workflow, der mit dem SWMS Argo auf einem Computer Cluster ausgeführt wird. Hierbei werden speziell die Ressourcenkonflikte untersucht, die bei der Ausführung eines DAW durch den Einsatz eines verteilten Dateisystems entstehen. Im Fokus stehen hierbei die I/O-intensiven Tasks des betrachteten Workflows, welche um die begrenzte Netzwerkkapazität konkurrieren. Um dieses Szenario zu analysieren, wird der betrachtete Workflow auf einem Computer Cluster ausgeführt. Hierbei sollen Ressourcenkonflikte mit weiteren Prozessen im Cluster vermieden werden, indem neben dem betrachteten Workflow möglichst keine weiteren Prozesse im Cluster ausgeführt werden. Hierdurch soll untersucht werden, wie sich der Ressourcenbedarf mit verschiedenen Parametern des Workflows verändert. Etwa indem Tasks auf einem Cluster mit mehr oder weniger Knoten ausgeführt werden oder die Anzahl der parallelen Tasks variiert.

Für die Untersuchung soll zunächst der "FORCE on Argo" Workflow weiter untersucht werden. Für die Identifikation von Ressourcenkonflikten soll ein Monitoring etabliert werden, das geeignete Metriken aufzeichnet.

Darüber hinaus sollen die Parameter des Clusters erhoben werden. Da zunächst angenommen wurde, dass die verfügbare Netzwerkbandbreite einen Flaschenhals in der Ausführung des DAW darstellt, soll diese durch Messungen ermittelt werden. Hierbei gilt es, zunächst die verfügbare Bandbreite zwischen den Knoten des Kubernetes Clusters und den Knoten des verteilten Dateisystems zu bestimmen. Für entsprechende Messungen soll das Micro-Benchmark `iperf` verwendet werden. Das Benchmark-Werkzeug besteht aus einer Server und aus einer Client Komponente und kann so die Netzwerkbandbreite zwischen zwei Knoten im Netzwerk bestimmen [6, S. 533].

Eine Messung der empfangenen und gesendeten Daten auf Ebene der einzelnen Tasks kann Aufschluss darüber geben, ob die verfügbare Bandbreite etwa nur die Ausführung einer bestimmten Anzahl paralleler Tasks erlaubt. Entsprechende Informationen können etwa mittels BPF-Trace ermittelt werden [6, S. 550 ff.]. Alternativ kann ein Kuberentes Monitoring verwendet werden, das Metriken auf Basis eines Prometheus Exporters[5] erhebt. Die Metriken werden während der Ausführung des Workflows erhoben und sind für verschiedene Ebenen des Kubernetes Clusters verfügbar. Informationen über die gesendeten und empfangenen Pakete werden etwa auf der Ebene von Kubernetes Pods aber auch auf Ebene der Knoten im Cluster erhoben[6].

Das Monitoring auf Task Ebene hat zunächst keinen Zugriff auf Informationen höherer Monitoring Schichten [2]. Somit müssen auf höheren Ebenen der Monitoring Architektur Informationen genutzt werden, um den Zustand des Clusters während der einzelnen Verarbeitungsschritte bewerten zu können. Etwa indem bestimmt wird, welche Tasks parallel ausgeführt werden und zu welchem Verarbeitungsschritt diese gehören. Ein weiterer Zusammenhang bildet sich etwa zwischen der Auslastung

[5] `https://prometheus.io/docs/guides/node-exporter/`
[6] `https://grafana.com/solutions/kubernetes/kubernetes-monitoring-introduction/`

eines Kubernetes Knotens und den Metriken der einzelnen Tasks, die parallel auf dem Knoten ausgeführt wurden.

Mit dem Ziel die auftretenden Ressourcenkonflikte zu analysieren, soll zunächst ein Histogramm mit den gesendeten und empfangenen Daten pro Task entstehen. Durch die Verknüpfung mit weiteren Metriken des Kubernetes Clusters und der Struktur des Workflows soll eine Datengrundlage entstehen, die es erlaubt entstehende Ressourcenkonflikte zu identifizieren.

Literatur

[1] J. Bader, F. Lehmann, A. Groth, L. Thamsen, D. Scheinert, J. Will, U. Leser, and O. Kao. Reshi: Recommending resources for scientific workflow tasks on heterogeneous infrastructures. In *2022 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 269–274. IEEE, 2022.

[2] J. Bader, J. Witzke, S. Becker, A. Lößer, F. Lehmann, L. Doehler, A. D. Vu, and O. Kao. Towards advanced monitoring for scientific workflows. *arXiv preprint arXiv:2211.12744*, 2022.

[3] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes. Borg, omega, and kubernetes. *Communications of the ACM*, 59(5):50–57, 2016.

[4] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. F. Da Silva, M. Livny, et al. Pegasus, a workflow management system for science automation. *Future Generation Computer Systems*, 46:17–35, 2015.

[5] P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame. Nextflow enables reproducible computational workflows. *Nature biotechnology*, 35(4):316–319, 2017.

[6] B. Gregg. *Systems performance: enterprise and the cloud*. Pearson Education, 2021.

[7] G. Kappes and S. V. Anastasiadis. Libservices: dynamic storage provisioning for multitenant i/o isolation. In *Proceedings of the 11th ACM SIGOPS Asia-Pacific Workshop on Systems*, pages 33–41, 2020.

[8] E. Kim, K. Lee, and C. Yoo. On the resource management of kubernetes. In *2021 International Conference on Information Networking (ICOIN)*, pages 154–158. IEEE, 2021.

[9] F. Lehmann, D. Frantz, S. Becker, U. Leser, and P. Hostert. Force on nextflow: Scalable analysis of earth observation data on commodity clusters. In *CIKM Workshops*, 2021.

[10] U. Leser, M. Hilbrich, C. Draxl, P. Eisert, L. Grunske, P. Hostert, D. Kainmüller, O. Kao, B. Kehr, T. Kehrer, et al. The collaborative research center fonda, 2021.

[11] A. Matsunaga and J. A. Fortes. On the use of machine learning to predict the time and resources consumed by applications. In *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 495–504. IEEE, 2010.

[12] C. Witt, M. Bux, W. Gusew, and U. Leser. Predictive performance modeling for distributed batch processing using black box monitoring and machine learning. *Information Systems*, 82:33–52, 2019.

[13] A. B. Yoo, M. A. Jette, and M. Grondona. Slurm: Simple linux utility for resource management. In *Workshop on job scheduling strategies for parallel processing*, pages 44–60. Springer, 2003.