Modellierung und Simulation von dynamischen container-basierten Software-Architekturen in Palladio

Nathan Hagel
Karlsruher Institut für Technologie
Karlsruhe, Deutschland
nathan@hagel.dev

Zusammenfassung

Moderne, verteilte Software-Systeme werden heutzutage nicht mehr nur statisch auf Maschinen deployed. Stattdessen werden die gewünschten Komponenten oder Container und deren Skalierungen deklarativ definiert. Eine Kontrollschleife versucht dann, den vorgegebenen Zustand des Systems dynamisch durch Starten und Stoppen von Containern und Pods zu erreichen. Auch Skalierungen von Diensten und Rollouts von neuen Produktversionen lassen sich auf diese Art realisieren.
Die Auswirkungen auf die Performance und Skalierbarkeit der Anwendung beim Einsatz dieser Techniken sind bisher nur schwer vorhersagbar.
Die in dieser Arbeit entwickelte Erweiterung für Palladio-Modelle verbessert die Modellierung und Simulation von dynamischen, container-basierten Software-Architekturen. Sie ermöglicht, container-basierte Anwendungen, sowie das Deployen mithilfe von Containerorchestrierungswerkzeugen wie Kubernetes im Palladio-Component-Model (PCM) abzubilden und zu analysieren. Für die Abbildung wurden die gängigsten Konzepte am Beispiel von Kubernetes ausgewählt und Anforderungen an dessen Abbildung definiert. Anschließend erfolgte eine Erweiterung des PCM durch diese Konzepte. Um die dynamische Allokation von Pods in Kubernetes abzubilden, wurde ein Pod-Allokations-Scheduler für PCM-Modelle entwickelt und prototypisch implementiert. Abschließend wurde ein dynamisches Simulationskonzept für Palladio erarbeitet, wobei Kontrollschleifen abgebildet wurden, die den Zustand des Clusters überwachen. Zur Evaluation wurde ein Referenz-Cluster definiert, welches mithilfe der PCM-Erweiterung und eines im Rahmen dieser Arbeit definierten Workflows abgebildet wurde. Zusammenfassend wurde festgestellt, dass sich die gängigsten Containerorchestrierungskonzepte mithilfe von Palladio abbilden und simulieren lassen. Dieses Paper basiert auf der Bachelorarbeit des Autors [1].

Motivation In den letzten Jahren wurden Containertechnologien für das Deployen von Software immer relevanter [5], [2]. Hierbei spielt neben der reinen Verwendung von Containern der Einsatz von Containerorchestrierungswerkzeugen wie Kubernetes eine große Rolle. Sie ermöglichen es, einen gewünschten Zustand des Systems deklarativ zu beschreiben. Dieser Zustand wird dann vom Containerorchestrierungswerkzeug automatisiert hergestellt und überwacht. Auch Skalierungen von Diensten und Rollouts neuer Versionen lassen sich damit vergleichsweise einfach umsetzen. Dabei stellt sich die Frage, wie sich die Verwendung dieser Technologien auf die dynamische Performance der Software auswirkt und welche Konfigurationen der Skalierungs- oder Allokationsstrategien sich für spezifische Anwendungsfälle am besten eignen. Auch für bestehende, traditionell deployte Systeme spielen diese Auswirkungen eine wichtige Rolle. Durch sich ändernde Anforderungen im Verlauf der Benutzung einer Anwendung kann es sinnvoll sein, die Art des Deployments hin zu einer container-basierten Anwendung zu evolvieren. Dabei kann ein Re-Engineering in eine container-basierte Anwendung, beispielsweise die Skalierungsmöglichkeiten erheblich verbessern.
Frühzeitige Informationen über die Auswirkungen einer Containerisierung einer bestehenden Software zu erhalten, kann bei der Entscheidung, ob ein Re-Engineering zu einer container-basierten Anwendung durchgeführt werden soll, hilfreich sein. Diese Informationen über reale Experimente, beispielsweise im Produktionssystem zu erlangen, kann schnell sehr kostspielig werden oder die Nutzer in der Verwendung der Anwendung einschränken. Mithilfe des Software-Architektursimulators Palladio [3] können bereits komponentenbasierte Anwendungen modelliert und analysiert werden. Eine Erweiterung um Container und Konzepte der Containerorchestrierung kann dabei die Entwicklung sowie den Entscheidungsprozess deutlich vereinfachen, insbesondere wenn für eine Anwendung bereits ein PCM-Modell existiert.

Zielsetzung Um dynamische, container-basierte Softwarearchitekturen im PCM modellieren und simulieren zu können, müssen neben Containern als Kapselung von Softwareartefakten auch die grundlegenden Konzepte aus der Conatinerorchestrierung abbildbar sein. Für die Auswahl der Konzepte wurde sich an Kubernetes als de-facto Standard in diesem Bereich orientiert. Zu diesen Konzepten gehören neben Containern: das Cluster, Nodes und Pods, Ressourcen Requests und Limits, Services und Ingress, Deployments und Replica Sets als deklarative Spezifikation des gewünschten Zustands, ein Scheduler für Pods, sowie eine Möglichkeit zur Skalierung des Systems. Neben der Abbildung der reinen Konzepte sollte eine Lösung gefunden werden, um nicht-statische Deployments mit Palladio zu analysieren. Hierbei spie-

len insbesondere die Pod-Allokation und die Skalierung eine wichtige Rolle. Durch eine Abbildung der genannten Konzepte in Kombination mit der Definition eines Workflows zur Analyse von dynamischen, container-basierten Anwendungen können Entwickler und DevOps-Spezialisten frühzeitig Informationen über die Auswirkung der Verwendung von Containerorchestration erhalten.

Durchführung Zuerst wurde analysiert, welche Konzepte aus der Containerorchestrierung für eine realistische Abbildung ins PCM notwendig sind. Anschließend erfolgte eine Anforderungsanalyse, um festzustellen, welche relevanten Eigenschaften der gewählten Konzepte abgebildet werden mussten. In einem ersten Schritt wurde sich auf die statischen Elemente wie Pods, Nodes und das Cluster konzentriert. Ausgehend davon wurden die bestehenden PCM-Elemente analysiert, um herauszufinden, wo diese wiederverwendet werden konnten, bzw. welche Elemente eine Erweiterung benötigten, um die gewählten Konzepte abzubilden. Nachdem Abbildungen für die grundlegenden Konstrukte gefunden wurden, wurden Analysen angestellt, um Abbildungen für Skalierung und Pod-Allokation zu definieren. Dabei wurden Möglichkeiten zur Modellrekonfiguration betrachtet und Konzepte erarbeitet, um jeweils die Pod-Allokation als auch die Autoskalierung, bspw. in Form eines Horizontal-Pod-Autoscaler (HPA) umzusetzen. Prototypisch wurde ein Pod-Allokations-Scheduler für Palladio-Modelle entwickelt und implementiert. Zur Evaluation wurde der Anteil der abgebildeten, für im Rahmen dieser Arbeit als relevant erachteten Konzepte bestimmt. Zusätzlich wurde ein Referenzcluster auf Basis einer bestehenden Anwendung definiert. Dieses wurde mithilfe des definierten Workflows in das PCM abgebildet und dabei Grenzen der Erweiterung untersucht. Abschließend wurden für ein Beispielszenario die Entscheidungen des Pod-Allokationsschedulers für Palladio-Modelle mit denen des Standard-Kubernetes-Schedulers verglichen.

Ergebnisse Auf Basis der definierten Anforderungen für die Kubernetes-Konzepte konnten für fast alle Elemente Abbildungen in das PCM gefunden oder definiert werden (z.B. Pods, Nodes, Deployments). Die einzige Ausnahme bildete das Replica Set, welches sich jedoch mit dem Deployment sehr stark überschneidet und deshalb nicht extra abgebildet wurde. Für den implementierten Pod-Allocator für Palladio-Modelle konnte anhand von Tests festgestellt werden, dass sich dieser grundsätzlich äquivalent zum Standard-Kubernetes-Scheduler verhält. Das Referenz-Cluster konnte im Rahmen der Evaluation mithilfe des definierten Workflows zur Verwendung der PCM-Erweiterung vollständig in Palladio abgebildet werden. Für die Abbildung der Kontrollschleifen wurde mit Modellrekonfigurationen basierend auf Query Views Transformation operational (QVTo) und dem Actions-Modell von Stier [4], welches zusätzlich die Möglichkeit bietet, transiente Effekte, bspw. den Overhead beim Starten eines neuen Containers zur Simulationszeit abzubilden, eine Möglichkeit gefunden, um auch dynamische Szenarien abzudecken.

Fazit & Ausblick In dieser Arbeit wurde eine Abbildung für container-basierte Anwendungen auf Basis von Containerorchestrierungswerkzeugen, wie Kubernetes entwickelt. Zusätzlich wurden Vorbereitungen, bspw. mit der Implementierung eines Pod-Allokations-Schedulers für PCM-Modelle getroffen, um dynamische Simulationen dieser Modelle durchzuführen. Darauf basierend wurde ein dynamisches Simulationskonzept entwickelt. Es wurde ein Workflow zur Verwendung der in dieser Arbeit entwickelten PCM-Erweiterung definiert. Die Abbildungen wurden anhand einer Referenzanwendung und eines Referenz-Cluster-Zustands evaluiert, sowie die Scheduling-Entscheidungen des implementierten Pod-Allokations-Schedulers mit denen des originalen Kubernetes-Schedulers verglichen. Als Fortsetzung dieses Projekts bietet sich die Umsetzung des dynamischen Simulationskonzepts an. Dies würde eine ausführlichere Evaluation der entwickelten Konzepte im Vergleich mit realen Systemen erleichtern. Damit können verschiedene Skalierungs- und Allokationsstrategien implementiert und evaluiert werden. Die genannten Schritte könnten die Performancevorhersage für Containeranwendungen erheblich erleichtern. Im Rahmen eines Folgeprojektes wird bereits an der Umsetzung dieser Überlegungen gearbeitet.

Literatur

[1] Nathan Hagel. "Modellierung und Simulation von dynamischen Container-basierten Software-Architekturen in Palladio". Bachelor's Thesis. Karlsruher Institut für Technologie, 2022.

[2] Grand view research. "Application Container Market Analysis Report By Deployment, By Platform (Kubernetes, Docker), By Organization Size (SMEs, Large Enterprise), By Service, By Application, By Region, And Segment Forecasts, 2019 - 2025". In: (1.01.2023). URL: https://www.grandviewresearch.com/industry-analysis/application-container-market#.

[3] Ralf Reussner u. a. *Modeling and Simulating Software Architectures - The Palladio Approach*. Okt. 2016, S. 1–349. ISBN: 9780262034760.

[4] Christian Stier. "Adaptation-aware architecture modeling and analysis of energy efficiency for software systems". Dissertation. 2019. URL: https://doi.org/10.5445/KSP/1000086089.

[5] Qi Zhang u. a. "A Comparative Study of Containers and Virtual Machines in Big Data Environment". In: *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. 2018, S. 178–185. DOI: 10.1109/CLOUD.2018.00030.