Konzeptionierung und Kostenanalyse einer Softwarelösung zur Verwaltung von Rechenressourcen

Manuel Habbel, 3275250

Erstbetreuer: Prof. Dr. rer. nat. Phillip Rohde

Zweitbetreuer: Amarin Vincent Klöker M. Sc.

15.12.2022

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die Seminararbeit mit dem Thema

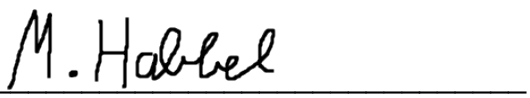
Konzeptionierung und Kostenanalyse einer Softwarelösung zur Verwaltung von Rechenressourcen

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind und die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Ich verpflichte mich, ein Exemplar der Seminararbeit fünf Jahre aufzubewahren und auf Verlangen dem Prüfungsamt des Fachbereiches Medizintechnik und Technomathematik auszuhändigen.

Name: Manuel Habbel

Aachen, den 15.12.2022



Unterschrift der Studentin / des Studenten

Abstrakt

In dieser Arbeit wird für neue Rechenressourcen im Institut für Kraftfahrzeuge der RWTH Aachen ein Monitoring- und Reservierungstool konzeptioniert. Um dies umzusetzen, wurden sowohl Nutzer als auch Experten aus dem Institut befragt und dadurch Anforderungen und mögliche Umsetzungen ausgearbeitet. Für diese Arbeit ergab sich daraus ein Fokus auf Monitoring-Tools. Um herauszufinden, welche Tools genutzt werden sollten, wurden verschiedene Monitoring- und Dashboard-Tools bezüglich Anpassbarkeit an Anforderungen und Kosten- sowie Zeitaufwand der Implementierung verglichen. Dabei hat sich das Monitoring-Tool Icinga durchgesetzt, wobei ein entscheidender Punkt dieser Entscheidung war, dass Icinga bereits im Institut genutzt wird und erweitert werden kann. Die gesammelten Daten sollen dabei auf einer Website mit dem Dashboard-Tool Graphite dargestellt werden. Gleichzeitig wird auf dieser Website ein Buchungstool eingebunden, damit beide Tools an einem Ort für die Nutzer zu finden sind. Welches Buchungstool genau genutzt werden soll, muss noch erarbeitet werden.

Inhaltsverzeichnis

1 Einleitung

Im Institut für Kraftfahrzeuge der RWTH Aachen werden zukünftig 24 Rechner sowie drei Server für verschiedene Berechnungen und Simulationen genutzt. Da es momentan kein gutes System gibt, welches die Nutzung der Rechenressourcen regelt, soll eine Lösung gefunden werden, um die Nutzung der Rechenressourcen zu analysieren und anschließend zu optimieren. Dafür wird sowohl ein Monitoring-System, als auch ein Reservierungs-/ Buchungstool für die Rechenressourcen gebraucht. Um dies zu erreichen, muss zunächst eine Nutzer- und Expertenbefragung abgehalten werden, um danach eine Make-or-Buy-Analyse für die Monitoring- und Reservierungstools durchzuführen.

Monitoring

Mit Monitoring werden Systeme bezeichnet, die den Zustand von Geräten und Programmen überwachen und beim Abweichen vom gewünschten Zustand Alarmmeldungen verschicken.

Hätte man kein Monitoring-System, könnten die Betreuer des IT-Systems nur dann reagieren, wenn ein Nutzer ein Problem entdeckt oder die Betreuer selbst das System auf Probleme kontrollieren. Dies ist allerdings sehr zeitaufwendig und die meisten Probleme werden erst entdeckt, wenn es zu spät ist. Deshalb werden Monitoring-Systeme benötigt, um eine möglichst gute Nutzung der Dienste zu sichern.

Diese Monitoring-Systeme können z.B. Ergebnisse von Statusprüfungen und Logs ermitteln, das Aufrufen einer Website simulieren und Daten eines Systems, wie die Festplattenauslastung, prüfen. [B2]

2 Nutzerbefragung

Im Rahmen der Anforderungsanalyse wurde eine Nutzerbefragung durchgeführt, um alle Anforderungen herauszufinden. Hierbei wurden sieben Mitarbeiter des Forschungsbereichs Fahrzeugintelligenz & Automatisiertes Fahren zu diesem Thema befragt. Folgendes war das Ergebnis dieser Befragung.

2.1 Fragenkatalog

<table>
<tr><th colspan="3"><u>User-Experience</u></th></tr>
<tr><td rowspan="4">Monitoring</td><td colspan="2">Zeigt die Auslastung der Rechner, und somit die Auslastung derer CPU, GPU, RAM, genutzten und freien Festplattenspeicher sowie alle aktuell eingeloggten Nutzer an.</td></tr>
<tr><td colspan="2">Der Server empfängt dauerhaft Daten von den Rechnern und speichert diese. Die gespeicherten Daten/Auslastungen werden kalendarisch angezeigt, um historische Daten zu analysieren.</td></tr>
<tr><td colspan="2">Die größten Ordner auf den Servern/Rechnern sollen angezeigt werden, um mögliche Festplattenprobleme zu verhindern. Dabei sollen einige spezielle Ordner ignoriert werden.</td></tr>
<tr><td colspan="2">Statistiken über alle Docker-Container sollen angezeigt werden. Dazu gehört vor allem der genutzte Speicherplatz.</td></tr>
<tr><td rowspan="3">Nutzerfreundlichkeit</td><td colspan="2">Lageplan: Eine Karte des Büros, auf welcher alle Rechenressourcen eingezeichnet werden, und welche anklickbar sind, um diese zu Buchen.</td></tr>
<tr><td colspan="2">Das Webtool sollte möglichst leicht anpassbar und wartbar sein. Deshalb sollte der Softwarestack wenige, unkomplizierte Komponenten enthalten.</td></tr>
<tr><td>Schnelles Reservieren von</td><td>Über einen Mattermost-Bot sollen Rechner per Befehl reserviert werden und Informationen über diesen</td></tr>
</table>

	Rechnern	ausgegeben werden können.
		Es soll ein öffentliches Tool sein, welches von jedem online erreichbar und für Mobile-Geräte optimiert sein soll. Dieses Tool sollte nicht von dem Instituts eigenem Intranet (ifaic) abhängig sein.
		URLs sollen klar erkennbar sein, um jeden Rechner über eine eindeutige URL erreichen zu können. Bsp.: „…monitoring/ressourcen/r001“ URL, die zur Übersicht des Rechners r001 führt.
	Es soll möglich, sein sich im Web-Tool über eine SSH-Verbindung auf dem Rechner einzuloggen.	
Sprache	Englisch	
Reservierung von Ressourcen	Bei der Reservierung eines Rechners muss angegeben werden, ob das GUI, die CPU, die GPU, ein Desktop gebraucht wird und was der Verwendungszweck ist.	
	Es muss beachtet werden, ob ein Remote Zugriff wie Anydesk oder Teamviewer genutzt wird, welcher den Rechner blockiert.	
	Es soll möglich sein, eine bestehende Reservierung zu verlängern.	
	Es soll möglich sein, einen Serientermin zu reservieren.	
	Reservierungen sollen gelöscht werden können.	
	Besteht bereits eine Reservierung zu einem Zeitpunkt, an dem man reservieren möchte, so kann man mit einem Klick zu einem Mattermost-Chat oder einem Email-	

	Fenster weitergeleitet werden, mit der Person, welche diese Reservierung blockiert.
Datenschutz	
Historie-Mitarbeiter	Die unter Monitoring angesprochene kalendarische Historie zeigt an, wie sehr ein Rechner in der Vergangenheit ausgelastet war. Diese Auslastung darf nicht auf einen Mitarbeiter zurückgeführt werden können.
Große Ordner	Wie unter Monitoring beschrieben, sollen die größten Ordner, welche sich auf den Rechnern befinden, angezeigt werden. Diese dürfen nicht von jedem Mitarbeiter eingesehen werden.
Optimierung	
Reservierungsvorschläge	Aufgrund der angegebenen, benötigten Daten soll automatisch berechnet werden, welcher Rechner für die Reservierung benutzt werden sollte.
Überflüssige Ordner erkennen	Es soll automatisch Ordner, von Mitarbeiter welche nicht mehr in der Firma arbeiten, erkennen und diese auflisten, um überflüssige Daten löschen zu können.
Rechtemanagement	
LDAP-Anbindung	Es soll eine LDAP-Anbindung geben, um die LDAP-Gruppen für das Rechtemanagement zu nutzen.
Buchungseinschränkung	Studierende dürfen nur x Tage in die Zukunft reservieren. Dieser Wert x soll als Variable angelegt werden und leicht geändert werden können.
	Studierende dürfen keine Serientermine löschen. Sie dürfen nur Einzeltermine löschen.
SSH-Verbindung	Nur Admins dürfen sich über eine SSH-Verbindung auf einem Rechner einloggen.

2.2 Auswertung

Aus der Nutzerbefragung ergab sich, dass die Monitoring-Funktion für die Nutzer am wichtigsten ist, denn mit dieser kann die Nutzung der Ressourcen überblickt und ausgewertet werden. Die Reservierung der Rechenressourcen sowie die Optimierung stehen somit an zweiter Stelle. Im Folgenden wird erläutert, wie die Nutzer sich eine Website mit Monitoring-System und Reservierungstool vorstellen, unter Berücksichtigung der Anforderungen. Die reale Umsetzbarkeit im Unternehmen wird dabei zunächst nicht beachtet.

Die Website sowie die genutzten Tools sollten in Python geschrieben und sollten in Englisch verfügbar sein, dadurch wird die Nutzbarkeit und Wartbarkeit für die Nutzer des Monitoring-Tools gesichert. Da ein einfacher Softwarestack gewünscht wurde, ist die Nutzung von Django als Framework eine gute Idee. Mit Django können viele Sicherheitsprobleme schnell vermieden und vor allem mit einfachem Code viel erreicht werden.

Das Frontend-Framework „Bootstrap“ könnte genutzt werden, um das Webtool geräteübergreifend, optimiert dazustellen und somit den mobilen Online-Zugang zu gewährleisten.

Weiter kann eine SQLite-Datenbank genutzt werden, denn die Datenmenge besteht aus nur wenigen Rechenressourcen sowie deren Kapazitätsdaten (CPU Cores, GPU, Festplattenspeicher, Vorhandensein eines Desktops), den Reservierungsdaten und der Auslastungshistorie.

Monitoring:

Zunächst sollte es eine Übersicht geben, welche alle Rechensysteme und deren Daten listet. Dazu gehören Daten zur CPU, GPU, RAM, zum Festplattenspeicher und ob ein Desktop vorhanden ist. Über diese Übersicht sollte dann auf die einzelnen Rechner weitergeleitet werden, wo sowohl die vollständige Auslastung angezeigt wird, als auch die Möglichkeit besteht, diesen Rechner zu reservieren. Die bisherige Auslastung als auch die Reservierungsübersicht sollte in einem Kalender dargestellt werden. Aus Datenschutzgründen werden alle Mitarbeiterinformationen bezüglich vergangener Reservierungen anonymisiert, damit Auslastungsinformationen nicht auf einzelne Mitarbeiter zurückgeführt werden können. Weiter soll in der Übersicht eine Auflistung der größten Ordner angezeigt werden. Informationen über diese Ordner

sollen wiederum nicht von jedem einsehbar sein. Weiter soll in der Übersicht ein Lageplan als Karte der Büroräume realisiert werden, auf welcher alle Rechenressourcen angezeigt werden und anklickbar sind, um zur Übersichtsseite dieser Ressource zu kommen. Als letztes sollen auch Informationen zu den genutzten Docker-Containern angezeigt werden, wobei vor allem der genutzte Speicherplatz wichtig ist.

Reservierung:

Um eine Optimierung der Ressourcennutzung zu schaffen, wird nach dem Monitoring Feature ein Reservierungssystem gefordert. Bei diesem soll angegeben werden, ob die CPU/GPU/GUI und zu welchem Zweck die Ressource gebraucht wird. Mit diesen Angaben kann die zukünftige Auslastung der Rechner besser geplant werden. Weiter sollte auch die Möglichkeit bestehen, wiederholende Termine zu reservieren und bestehende Reservierungen zu verlängern. Darüber hinaus sollten die Einträge auch löschbar sein, allerdings dürfen Studierende nur einzelne Termine löschen. Sollte ein Termin bereits belegt sein, soll es die Möglichkeit geben, mit nur einem Klick den Mitarbeiter zu kontaktieren, welcher diesen Termin bereits gebucht hat, indem sofort ein Email-Fenster oder der Mattermost-Chat mit diesem Mitarbeiter angezeigt wird.

Optimierung:

Um die Ressourcennutzung weiter zu optimieren, soll durch die angegebenen Daten bei der Reservierung der bestmögliche Rechner für die geplante Aufgabe berechnet werden. Um das System noch weiter zu verbessern, sollen Ordner von Personen, welche nicht länger in der Firma arbeiten, automatisch erkannt werden, damit diese gelöscht werden können.

Verbesserte Nutzung:

Eine wichtige Anforderung war es, die Rechner schnell reservieren zu können. Dazu ist eine Idee, einen Bot für den Instant-Messaging-Dienst Mattermost einzubauen, mit welchem man über einen Befehl eine Rechenressource reservieren oder Informationen über diese abfragen kann. Weiter ist eine Mobile-Version verlangt worden. Wichtig ist auch, dass die URL zu den Infos und zur Reservierung der einzelnen Ressourcen klar ersichtlich und für jede Ressource eindeutig ist (z.B.: „…monitoring/ressourcen/r001“ URL die zur Übersicht des Rechners r001 führt).

Rechtemanagement:

Da es bei der Reservierung Einschränkungen geben soll, wird eine LDAP-Anbindung gebraucht. Damit kann sichergestellt werden, dass Studierende nicht alle Termine löschen und nur eine begrenzte Zeit in die Zukunft planen dürfen. Außerdem sollen Admins eine SSH-Verbindung zu den Rechnern aufbauen können, um in Notfällen schnell eingreifen zu können.

3 Expertenbefragung

Um die Umsetzung und technischen Anforderungen eines Monitoring- und Buchungstools für Rechenressourcen zu prüfen, wurden fünf Experten aus der IT-Abteilung des Instituts für Kraftfahrzeuge befragt. Dabei wurde auf die Anforderungen aus der Nutzerbefragung eingegangen, auf die Umsetzbarkeit sowie ob die Anforderungen sinnvoll sind. Die daraus entstandenen Ergebnisse werden im Folgenden dargestellt.

Zunächst ergab sich aus der Expertenbefragung eine klare Empfehlung, das Monitoring-Tool sowie auch das Buchungstool mit schon existierenden, kostenlosen, Open Source-Tools zu realisieren. Gründe dafür sind, dass es viel Arbeitszeit und somit auch Geld kosten würde, diese Tools selber zu bauen. Da es viele kostenlose Tools gibt, mit denen ein Teil der Anforderungen schon erfüllt werden können, ist es mit Blick auf die Kosten besser, eines dieser Tools zu benutzen und ein paar Anforderungen auszulassen.

Es gibt einige Probleme, die sich aus den Anforderungen der Nutzer ergeben haben. Zuerst ergab sich das Speicherplatzproblem, denn durch die geforderte Historie würden sehr viele Daten generiert und gespeichert werden. Deshalb wäre es nicht sinnvoll, diese Daten über mehrere Jahre zu speichern, sondern höchstens über einen kurzen Zeitraum. Weiter wurde die Nutzung der Historie in Frage gestellt. Denn für die meisten und wichtigsten Auswertungen, wie z.B. Speicherplatzprobleme oder eine dauerhafte volle oder geringe Auslastung werden keine jahrelangen Daten gebraucht, sondern nur die Daten von ein paar Monaten.

Ein weiteres großes Problem war die Anforderung, das Tool weltweit online erreichen zu können, ohne in dem VPN der Firma zu sein. Dadurch ergeben sich viele Datenschutzprobleme, denn es handelt sich auch um sensible Daten, auf welche man nur aus dem internen Netz zugreifen dürfen sollte. Außerdem sorgt ein VPN schon früher für Schutz vor unbefugtem Zugriff auf die Daten. Weiter dürfen nicht alle Mitarbeiter auf die gesammelten Daten zugreifen. Darüber hinaus werden bei einigen

Tools auch personenbezogene Daten in den Auslastungsdaten gespeichert, welche nicht abgespeichert werden dürfen.

Für die Umsetzung eines Reservierungssystems mit den Anforderungen der Nutzer gibt es mehrere Möglichkeiten, welche mit unterschiedlich viel Aufwand verbunden sind. Ein Reservierungstool von Grund auf neu zu bauen wäre dabei keine sinnvolle Lösung. Stattdessen wäre die Erweiterung von bestehenden Tools ein sinnvoller Weg. Mit wenig Arbeit können schon die meisten Anforderungen abgedeckt werden, allerdings könnte die Perfektionierung etwas mehr Zeit in Anspruch nehmen.

Wie oben schon erläutert wäre es sinnvoll, kostenlose, Open Source-Software zu nutzen, um das geplante Tool zu realisieren. Weiter gibt es in der Firma schon ähnliche Tools, welche man möglicherweise erweitern könnte, um das gewünschte Ergebnis zu erhalten. So gibt es zum Beispiel schon einen Buchungsplan, in welchem Ressourcen wie Räume und Beamer gebucht werden können. Dieser könnte um die Rechenressourcen erweitert werden und bei der Buchung könnten alle wichtigen Informationen angegeben werden. Genauso gibt es für das Monitoring bereits ein ähnliches Tool in der Firma. Hierbei wird mit dem Tool Icinga die Auslastung von zahlreichen Ressourcen überwacht. Die Nutzung dieses Tools könnte ebenfalls auf die Rechenressourcen erweitert und möglicherweise eine andere Darstellung der Auslastungen erarbeitet werden.

Sollte dennoch entschieden werden, ein neues Tool zu entwickeln, sind einige Anforderungen bei der Implementierung zu beachten. Zunächst ist wichtig, dass nur Bibliotheken benutzt werden, welche auch in Zukunft ohne Probleme genutzt werden können und nicht durch nachfolgende oder keine Updates zu Problemen führen. Weiter sollte sich an Standards gehalten und möglichst neue Updates sofort eingebracht werden, damit der Abstand von der genutzten Version zur neuesten nicht zu groß wird.

4 Monitoring-Tools

Warum sollte man Monitoring-Tools benutzen? Vor allem Systeme in der Produktion sollten dauerhaft überwacht werden, sodass diese im besten Fall nie ausfallen. Mit einem gut aufgestellten Monitoring-System können kritische Änderungen erkannt und diese behoben werden, bevor das betroffene System in ein Problem läuft. Sollte es trotzdem passieren, dass ein System in einen Fehler-Zustand läuft, wird dieses sowie der Fehler schneller erkennbar und somit schneller behebbar sein. Es ist unvermeidbar, ein solches System in einem laufenden Betrieb einzuführen, um eine maximale Produktivität zu erreichen und längere Ausfälle zu vermeiden.

Im Folgenden werden mögliche Monitoring-Tools untersucht und deren Vor- und Nachteile tabellarisch dargestellt. Dazu wird das Tool Icinga, welches in einem anderen Monitoring-System der Firma verwendet wird, untersucht, sowie einige Alternativen. Daraufhin werden die Vor- und Nachteile mit den Anforderungen verglichen und so ein sinnvolles Tool für den Anwendungsfall dieser Arbeit gefunden.

4.1 Marktübersicht

Da in der Firma bereits das Tool Icinga genutzt wird, wird darauf zunächst eingegangen und die Vor- und Nachteile des Erweiterns dieses Tools erläutert. Danach werden weitere Tools untersucht und daraufhin eine Entscheidung ausgearbeitet, welches Tool genutzt werden sollte.

Icinga	
Pro	Alle Monitoring Anforderungen, welche aus der Nutzerbefragung hervorgingen, können mit Icinga abgedeckt werden. Weiter gibt es für fast alle Monitoring Anforderungen bereits ein kostenlos verfügbares Plugin. Diese werden meist von der Nagios-Community angeboten. Dazu zählt auch das Überprüfen der Auslastung von CPU, GPU, RAM, sowie genutztem und freiem Festplattenspeicher von Rechenressourcen.
	Icinga ist vielseitig anpassbar und konfigurierbar für alle Anforderungen.
	Es besitzt eine Rechteverwaltung, mit welcher Benutzer in verschiedene Gruppen eingeteilt werden können. Diese können an die LDAP Gruppen angepasst werden und somit ist sichergestellt,

	dass nicht jeder alles sehen kann.
	Es können Warnungen gesendet werden, sollte ein System nicht mehr laufen oder wenn zum Beispiel der Speicher eines Systems zu einem bestimmten Teil gefüllt ist.
	Es können einzelne Services überwacht werden. Außerdem werden für die einzelnen Services die Daten für verschiedene Zeitintervalle angegeben. So kann festgestellt werden, ob ein Service zum Beispiel nur für die letzte Minute oder schon für 15 Minuten zu stark ausgelastet ist.
	Die Abfragen für das Monitoring bestehen aus einfachen Skripten (z.B. Python oder Bash). Somit kann fast alles abgefragt werden, was mit Skripten überprüft werden kann.
	Es ist möglich, z.B. Grafana oder Graphite einzubinden, um die Darstellung zu verbessern oder anzupassen. Mit diesen Tools könnte zum Beispiel die geforderte Historie dargestellt werden.
	Man kann verschiedene Ansichten (Views) erstellen um unterschiedliche Checks zu gruppieren und leichter zu überblicken.
	Jeder Service/ jedes System/ jede Komponente die überwacht wird, hat einen klaren Link, um zu diesem zu gelangen. Im Link sind zusätzlich Informationen enthalten was bei Icinga angezeigt werden soll. Zum Beispiel können zwei verschiedene Checks getrennt auf einer Seite angezeigt werden. Es kann also dynamisch der Link angepasst werden um die Ausgabe anzupassen.
Contra	Icinga darf aus Datenschutzgründen nicht ohne VPN erreichbar sein.
	Die Nutzung von Icinga könnte für die Nutzer komplizierter sein als gewünscht. Das liegt daran, dass dieses Tool bei der ersten Nutzung unübersichtlich/kompliziert sein könnte. Außerdem könnte der Link für die Nutzer kompliziert und unverständlich aussehen.
	Es ist schwer, die Möglichkeit einzubauen, aus dem Web-Tool eine SSH-Verbindung zum Zielsystem aufzubauen.

	Es ist kompliziert für die Nutzer das Tool zu warten/bearbeiten.

Icinga [B9] wurde aus dem Code des Open Source-Tools Nagios entwickelt. Deshalb sind sich die beiden Monitoring-Tools in vielen Punkten ähnlich. Der wichtigste Punkt ist, dass Icinga mit den Plugins von Nagios kompatibel ist. Da Nagios eine sehr große Community besitzt, welche für fast alle Monitoring-Anforderungen bereits ein frei zugängliches Plug-In veröffentlicht hat, können diese praktischerweise auch in Icinga genutzt werden. Um alle Punkte zusammenzufassen könnte man also sagen, dass Icinga die meisten Anforderungen der Nutzer erfüllt und das bereits bestehende System an diese angepasst werden könnte. Zwar gibt es ein paar Gegenargumente, diese unterliegen allerdings dem Punkt das Icinga kostenlos ist. Außerdem könnte mit einer kurzen Einweisung das Tool von fast allen sehr gut genutzt werden.

Prometheus

Prometheus [B12] ist ein Open Source-Monitoring- und Alerting-Tool veröffentlicht von SoundCloud von ex-Google Mitarbeitern im Januar 2015 und ist Teil der Cloud Native Computing Foundation seit März 2016. Geschrieben wurde es vor allem in der Sprache Go und als Inspiration galt das von Google genutzte Monitoring-System „Borgmon“. [B11] Prometheus besitzt eine Time Series Database (TSDB), für welche die dafür entwickelte Query Language PromQL genutzt wird, um flexible Query Abfragen zu ermöglichen. [B16] Das Wurzelelement dieser Datenbank ist ein Zeitstrahl, wobei sämtliche Informationen mit einem spezifischen Punkt auf dem Zeitstrahl verknüpft sind. Dadurch entsprechen die Datenbankabfragen bereits dem nativen Datenschema der Datenbank. Mit der Nutzung einer TSDB sowie der dafür entwickelten Query Language PromQL werden Performance-Probleme beim Speichern und Auslesen von Metrik-Daten behoben. Andere Monitoring-Systeme mit relationalen Datenbanken werden beim Auslesen deutlich mehr oder komplexere Anfragen stellen müssen, was zu Performance-Problemen führt. [B1] Prometheus nutzt ein Pull-Model über http, um Metriken in einem bestimmten Intervall von dem Monitoring Ziel abzufragen und danach in der TSDB zu speichern. [B16]

Weitere Monitoring-Tools		
Name	Pro	Contra
Nagios	Kompatibel mit fast allen Geräten (z.B. Laptop, PCs und Smartphones). Braucht nur einen modernen Browser.	Nagios-Version mit Support und leichterer Implementierung kostet viel. (Kosten starten bei ca. 1.995$)
	Robustes API-Backend, somit ist es leichter, eigene Anwendungen einzubauen	Die Installation der kostenlosen Variante kann sehr komplex werden, da kein Support verfügbar ist.
	Open Source	
	Kompatibel mit Grafana. Verbessert die Darstellung der gesammelten Daten	
	Benachrichtigungsoptionen über SMS und Email, falls Probleme durch das Monitoring-Tool erkannt werden	
	Professioneller technischer Support	
	Blackbox-Monitoring	
Prometheus	Besitzt eine TSDB (Time Series Database), welche Daten über einen Zeitraum besser speichern kann.	Da PromQL als Query Language genutzt wird, muss man diese lernen
	Open Source	
	Whitebox-Monitoring	
	Kompatibel mit Grafana. Verbessert die Darstellung der gesammelten Daten	

	Prometheus hat eine eigene Query Language namens PromQL. Dadurch können Daten besser abgespeichert und abgerufen werden.	

Monitoring-Tools im Vergleich bezogen auf die Monitoring-Anforderungen der Nutzer				
Tool\Anforderungen	Auslastung verschiedener Komponenten	Datenausgabe als Historie	Die größten Ordner anzeigen	Statistiken über Docker-Container Anzeigen
Icinga	✓	✓	✓	✓
Nagios	✓	✓	✓	✓
Prometheus	✓	✓	?	✓

4.2 Blackbox- vs. Whitebox-Monitoring

Monitoring-Tools können in verschiedene Kategorien eingeordnet werden. Dazu gehören zum Beispiel das Blackbox-Monitoring und Whitebox-Monitoring. Diese werden nun kurz erläutert und die Monitoring-Tools diesen zugeordnet.

Beim Whitebox-Monitoring werden Anwendungen oder Services, welche auf einem Server laufen, überwacht. Dabei kann vieles überwacht werden, wie zum Beispiel die Anzahl der HTTP-Anfragen an einen Web-Server oder den von der Anwendung generierten Response-Code. Diesem Ansatz kann unter anderem das Monitoring-Tool Prometheus mit seinen Grundfunktionen zugeordnet werden.

Blackbox-Monitoring dagegen fokussiert sich auf Bereiche wie Festplattenspeicher, CPU Auslastung, RAM Auslastung, Netzwerk Switches und andere Netzwerk Geräte. Weiter ist auch das Alarmieren bei möglichen Problemen ein Teil von Blackbox-Monitoring. Man könnte also sagen, dass der aktuelle Zustand eines Service oder

Host überprüft werden und bei Problemen auch ein Alert gesendet werden kann. Diesem Ansatz können die Monitoring-Tools Nagios und Icinga mit ihren Grundfunktionen zugeordnet werden.

Beide Monitoring Ansätze können wichtige Infos über mögliche Probleme geben. So kann das Blackbox-Monitoring zum Beispiel eine hohe Auslastung der Festplatte oder hohen Networktraffic erkennen und dazu einen Alarm auslösen. Allerdings wüsste man noch nicht, woher diese Auslastung kommt und wie viel Zeit noch bleibt, bis diese Auslastung ein Problem wird. Hätte man nun zusätzlich Whitebox-Monitoring eingesetzt, könnte man z.B. die Zeit, welche der überwachte Job schon braucht, die Anzahl eingegangener Anfragen oder die Rate bzw. Geschwindigkeit, mit welcher sich eine Festplatte füllt, sehen. Somit könnte weiter eingegrenzt werden woher die hohe Auslastung kommt und ob man sofort etwas gegen das erkannte Problem unternehmen muss. Daraus könnte man schlussfolgern, dass für ein optimales Monitoring sowohl Whitebox-Monitoring als auch Blackbox-Monitoring genutzt werden sollte. [B10] [B14]

Für den hier behandelten Anwendungsfall sollte ein Blackbox-Monitoring ausreichen, um die Anforderungen der Nutzer zu erfüllen, da es sich auch nicht um ein groß skaliertes System handelt, vor allem die aktuelle Auslastung für die Nutzer wichtig ist und die Rate, in welcher die Auslastung steigt, nicht zwingend benötigt wird. Allerdings sollte für die zukünftige Nutzung des Monitoring-Systems auch das Hinzufügen eines oder Umstellen auf ein Whitebox-Monitoring-Tool in Betracht gezogen werden. Denn wie oben schon beschrieben, kann dadurch abgeschätzt oder sogar vorhergesagt werden, wann Probleme auftreten können. Sowohl Prometheus als auch Nagios und Icinga können zwar diesen Monitoring Ansätzen zugeordnet werden, doch sind sie vielseitig anpassbar. So gibt es sowohl für Prometheus als auch für Icinga Plugins, welche Funktionen des anderen Monitoring-Ansatzes hinzufügen. Icinga hat dabei allerdings den Vorteil, dass man die meisten Checks leicht selber schreiben kann, so auch welche, die dem Whitebox Ansatz zugeordnet werden können. Außerdem hat die Icinga Community schon sehr viele Plugins entwickelt, welche dem Whitebox-Ansatz zuzuordnen sind.

Schlussendlich kann man sagen, dass mit den passenden Plugins sowohl Icinga als auch Prometheus für White- und Blackbox-Monitoring angepasst werden können. Dies scheint durch die Arbeit der Community bei Icinga allerdings leichter.

4.3 Fazit zu den Monitoring-Tools

Aus den erläuterten Punkten stellt sich nun die Frage, welches der Monitoring-Tools für diesen Anwendungsfall gewählt werden sollte oder ob, wie unter der Software Konzeption beschrieben, ein solches Monitoring-Tool selbst programmiert werden sollte.

Durch die Auflistung der Pro und Contra Argumente der verschiedenen Monitoring-Tools lässt sich eines dieser Tools schnell aus der Wahl entfernen. Nagios ist zwar in den Grundfunktionen kostenlos, kann aber bei der Installation kompliziert werden. Darüber hinaus wurde Icinga aus dem Open Source-Code von Nagios weiterentwickelt und ist mit dessen Plugins kompatibel. Dadurch hat Icinga eine große Ähnlichkeit zu Nagios aber übertrifft es zusätzlich durch den Aspekt, dass es eine größere bzw. aktivere Community besitzt.

Somit stellt sich die Frage, ob Icinga oder Prometheus als Monitoring-Tool gewählt werden sollte. Beide sind sehr weit entwickelte Tools mit einer großen Community, welche diese weiterentwickelt und Plugins bereitstellt. Auch ist es möglich, bei beiden ein Dashboard-Tool einzubinden, um die gesammelten Daten besser darstellen zu können. Wie schon unter dem Punkt Blackbox- und Whitebox-Monitoring ausgeführt, wäre die Nutzung eines Monitoring-Tools mit Blackbox- und Whitebox-Monitoring Ansatz sinnvoll. Da Icinga dies durch Plugins der Community leichter erreichen kann und dazu auch noch kostenfrei ist, wäre dies eine gute Wahl. Dazu kommt außerdem der sehr ausschlaggebende Punkt, dass Icinga bereits als Monitoring-Tool für andere Systeme im Unternehmen der Nutzer verwendet wird und um gewünschte Punkte erweitert werden kann. Dies würde viel Zeit und Kosten sparen, die bei der Implementierung eines Monitoring-Tools mit Prometheus anfallen würden. Deshalb sollte Icinga als Monitoring-Tool gewählt werden.

Ein eigenes Monitoring-Tool zu entwickeln würde viel Zeitaufwand und somit auch zu hohe Kosten verursachen. Ein oder mehrere Programmierer müssten daran arbeiten und viel Zeit aufwenden, welche aufgrund von vielen anderen Aufgaben und wenigen Programmierern im Institut keine Zeit haben, um ein neues Monitoring-System zu implementieren. Der große Vorteil darin, ein eigenes Monitoring-Tool zu bauen, liegt darin, dass man es beliebig erweitern und anpassen kann. Da aber bereits das gewählte Monitoring-Tool in Nutzung ist und einfach erweitert werden könnte, sollte von der Implementierung eines eigenen Tools abgesehen werden.

4.4 Dashboard-Tools

Definition:

An application (usually web-based) that provides a summary view of a service's core metrics. A dashboard may have filters, selectors, and so on, but is prebuilt to expose the metrics most important to its users. The dashboard might also display team information such as ticket queue length, a list of high-priority bugs, the current on-call engineer for a given area of responsibility, or recent pushes. [B4]

Wofür braucht man ein Dashboard-Tool?

Da die Darstellung der Monitoring Daten für die Nutzung ausschlaggebend ist, sollte ein eigenes Dashboard-Tool in Betracht gezogen werden, da das Dashboard der meisten Monitoring-Tools zwar funktional, aber nicht anpassbar an die Anforderungen und Nutzer ist. Deshalb sollten Dashboard-Tools genutzt werden, um alle Daten möglichst gut darzustellen. Im Folgenden werden kurz einige Tools untersucht und danach ermittelt, welches am sinnvollsten genutzt werden sollte.

Grafana

Grafana [B5] ist eins der größten Dashboard-Tools, welche es zurzeit gibt. Es ist zum Großteil geschrieben in Go und Typescript. Der Hauptverwendungszweck ist das Monitoring von Servern und Infrastruktur. Grafana bringt einige Vorteile mit sich, darunter eine offizielle Bibliothek mit vielen Dashboard Templates und Plugins. Außerdem unterstützt es eine große Auswahl an Datenquellen, darunter z.B. Prometheus.

Allerdings ist es ausschließlich ein Visualisierungstool und unterstützt keinen Datenspeicher.

Graphite

Graphite [B6] wurde als Nebenprojekt entworfen und geschrieben von Chris Davis bei Orbitz in 2006. Letztendlich entwickelte es sich zu einem grundlegenden Monitoring-System. 2008 wurde es unter der Open Source-Lizenz Apache 2.0 veröffentlicht. Graphite besteht aus drei Software Komponenten:

Carbon - Wartet auf Zeitreihen-Daten

Whisper – Eine einfache Datenbank-Bibliothek um Time-Series Data zu speichern

Graphite Webapp – Eine Django Webapp, welche auf Verlangen Graphen rendert. Dafür wird Cairo genutzt, eine 2D Grafiken Bibliothek.

Graphite wird genutzt, um numerische Zeitreihen-Daten zu speichern und diese bei Bedarf grafisch darzustellen, kann aber diese Daten nicht selber sammeln, sondern muss dabei auf andere Tools zurückgreifen. Geschrieben ist Graphite in der Programmiersprache Python bzw. Django. [B13]

Kibana

Kibana [B3] ist ein Open Source Daten Navigations- und Visualisierungstool, mit welchem die Nutzer ElasticStack Daten überwachen und managen können. ElasticStack wird genutzt, um Daten aus beliebigen Quellen und verschiedensten Formaten zu erfassen und danach in Echtzeit zu durchsuchen und zu analysieren. Kibana bietet ein Anomalieerkennungsfeature, welches Probleme in den Daten erkennt.

Kibana ist limitiert auf ElasticStack-Daten und dadurch für diesen Anwendungsfall uninteressant. [B7]

Welches der genannten Dashboard-Tools sollte genutzt werden?

Da Kibana begrenzt ist auf ElasticStack Daten, sollte dieses nicht genutzt werden. Deshalb ist nun die Frage, ob Grafana oder Graphit genutzt werden sollte. Graphit hat den Vorteil, dass es in Python bzw. Django geschrieben ist, mit welchem die Nutzer gut arbeiten können. Allerdings gibt es für Grafana eine bessere Unterstützung durch die Nagios/Icinga Community, welche sowohl die Einbindung weiterentwickelt, als auch Templates zur Verfügung stellt. Außerdem ist eine gute Anpassbarkeit bei Grafana sichergestellt. Deshalb sollte Grafana zumindest bei einem puren Monitoring-Tool als Dashboard-Tool gewählt werden.

Speicherplatznutzung beim Monitoring von Rechenressourcen

Im Unternehmen wurde Icinga über einen Zeitraum von einem Jahr dauerhaft zum Überwachen von Ressourcen genutzt. Dabei wurden auf ca. 100 Rechenressourcen jeweils ca. 10 Checks ausgeführt. Dabei kam es zu einer Datenmenge von ungefähr 100GB. Die Datenmenge hängt dabei auch von den ausgeführten Checks ab und kann demnach variieren. Da im Anwendungsfall dieser Arbeit nur 24 Rechner und 3 Server genutzt werden und in naher Zukunft nicht deutlich mehr Rechenressourcen

hinzugefügt werden, kann man aus diesen Daten schließen, dass die Datenmenge, welche durch das Monitoring erzeugt wird, nicht zu hoch wird.

5 Konkrete Umsetzungsmöglichkeit eines Monitoring-Tools

In diesem Abschnitt wird nun beschrieben, wie ein Monitoring-System mit den Tools Icinga und Grafana für den Anwendungsfall dieser Arbeit umgesetzt werden könnte. Warum diese Tools benutzt werden sollten, wurde bereits in den vorherigen Abschnitten beschrieben. Um herauszufinden, welche Schritte dafür nötig sind, wurde erneut ein Experte aus dem Unternehmen befragt. Icinga muss in diesem Fall nicht neu implementiert werden, da es bereits im Unternehmen als Monitoring-System genutzt wird, weshalb nur die Erweiterung dieses Systems beschrieben wird.

5.1 Erweiterung des Monitoring-Systems

Um nun die Rechenressourcen überwachen zu können, muss zunächst ein User auf jeder Rechenressource angelegt und diese als Host im Icinga System hinterlegt werden. Weiter müssen die genutzten Nagios-Plugins auf den Rechenressourcen installiert und eine SSH-Verbindung zwischen dem Icinga-Server und den neuen Rechenressourcen erstellt werden. Diese Schritte benötigen keinen großen Zeitaufwand.

Als nächstes müssen die benötigten Checks in Icinga erstellt werden, indem man diese entweder selber schreibt oder Plugins von der Nagios Community herunterlädt. Da Nagios eine sehr große und aktive Community hat und Icinga kompatibel mit allen Nagios Plugins ist, können zu fast allen Themen Plugins online gefunden werden, somit auch zu unseren Monitoring Anforderungen. Insgesamt sind es ca. 10 Checks pro Rechenressource, welche jede Minute ausgeführt werden sollen. Dazu zählen neben den Auslastungschecks der CPU, GPU, RAM und Festplatte auch Checks, welche die Ressource anpingen und überprüfen, ob diese überhaupt aktiv ist, sowie Checks, welche überprüfen, ob einzelne Services laufen. Durch diese zusätzlichen Checks kann ein logischer Aufbau erreicht werden, um unnötigen Networktraffic bei inaktiven Systemen zu vermeiden. Denn sollte ein System inaktiv sein, wäre das Ausführen der Checks auf der CPU, GPU usw. überflüssig. So kann eingestellt werden, dass diese Checks nur ausgeführt werden, wenn das Anpingen des Systems ein positives Ergebnis liefert.

Ist dieser Schritt erledigt, können bereits Daten gesammelt werden. Nun sollte die Rollenverwaltung eingebaut werden, um festzulegen, wer welche Daten sehen darf. Dafür kann in Icinga eine neue Gruppe für die Nutzer des Bereichs, aus welchem die

zu überwachenden Rechenressourcen stammen, erstellt werden. Dieser Gruppe oder diesen Gruppen können dann die einzelnen Checks zugeordnet und auch geordnet angezeigt werden.

Da nun die Übersicht in Icinga für neue Nutzer etwas schwierig sein könnte, sollte Grafana als Dashboard-Tool in Icinga eingebunden werden. Das Icinga-Module für Grafana wurde von der Icinga Community erstellt und wird von dieser weiter gepflegt. Links zu den passenden Modulen sowie zu von der Community erstellten Templates für Icinga mit Grafana gibt es auf einer Seite von Icinga [B8]. Grafana ist aber auch einfach an eigene Anforderungen anpassbar. Um bei der Nutzung von Grafana nicht unnötig viele Daten speichern zu müssen, bietet Grafana auch eine Möglichkeit, Daten zu komprimieren. Zwar ist die Datenmenge bei wenigen Rechenressourcen, wie es hier der Fall ist, noch übersehbar, doch sollte man die Möglichkeit, Speicherplatz zu sparen, auf jeden Fall nutzen. Das Einrichten der Checks sowie das Einbauen von Grafana sind dabei mit dem größten Arbeits- und Zeitaufwand verbunden, da zunächst passende Checks gesucht oder geschrieben werden müssen und der Programmierer sich mit Grafana vertraut machen muss. Sobald Grafana fertig konfiguriert wurde, ist ein vollwertiges Monitoring-System fertiggestellt.

Icinga ist aus Sicherheitsgründen nur aus dem Netz des Unternehmens zu erreichen. Somit kann es von außerhalb nur mit einem VPN erreicht werden. Zwar wurde in der Nutzerbefragung gefordert, dass es auch von außerhalb erreichbar sein soll, aber da es sich um viele sensible Daten handelt, würde der öffentliche Zugang diese gefährden, auch wenn zum Beispiel eine Zwei-Faktor-Authentifizierung hinzugefügt wird, da diese den Zugang erst bei dem Zugriff auf den Dienst prüft, wohingegen ein VPN schon vor dem Zutritt zum Netzwerk den Zutritt prüft. Außerdem würde ein freier Zugang den Sicherheitsrichtlinien des Unternehmens widersprechen und ist deshalb nicht zulässig.

5.2 Implementierung eines Reservierungstools

Um nun die Umsetzung eines Reservierungstools zu planen, wurde sich erneut mit einem Experten aus dem Unternehmen zusammengesetzt, um den Aufwand und die Kosten für ein solches Tool zu besprechen. Dabei kam es zu verschiedenen Möglichkeiten.

Der schnellste, einfachste und damit billigste Weg, das Reservierungstool umzusetzen, wäre den bereits vorhandenen Buchungsplan zu erweitern und dort einen Link zur entsprechenden Monitoring Seite einzubauen. Allerdings wäre dies für den Nutzer keine gute Lösung, da die Übersicht und Nutzbarkeit nicht so schnell und

einfach ist wie gewünscht. Eine etwas zeitaufwendigere und damit teurere Lösung wäre es, ein neues Reservierungstool zu bauen mit schon existierenden kostenlosen Tools. Der Vorteil dabei liegt darin, dass man das Reservierungstool sofort an die neuen Anforderungen anpassen kann. Weiter könnte man es mit dem Monitoring-Tool verbinden, um beide Tools an einem Ort zu haben. Da die Nutzer gerne ein Python-Tool benutzen würden, sollte man dann auf die Nutzung von Grafana verzichten und stattdessen Graphite als Dashboard-Tool wählen, da dieses bereits mit Python/ Django läuft. Offen bleibt nur die Frage, welches Buchungstool dazu eingebunden wird. Im Unternehmen wird momentan MRBS [B15] als Buchungstool verwendet, dieses wurde allerdings in PHP geschrieben. Am besten wäre also ein Tool welches bereits in Python geschrieben ist. Die Suche nach einem solchen Tool würde den Rahmen dieser Arbeit sprengen und müsste demnach in weiteren Recherchen herausgefunden werden.

6 Zusammenfassung

Nachdem nun einige Monitoring- und Dashboard-Tools sowie Möglichkeiten zur Umsetzung eines Reservierungstools untersucht wurden, ergab sich folgendes Ergebnis. Um die Anforderungen der Nutzer am besten erfüllen zu können, sollte das Monitoring-Tool Icinga, welches bereits im Institut benutzt wird, erweitert werden um die zu überwachenden Rechenressourcen. Außerdem sollte das Dashboard-Tool Graphite genutzt werden, da es in Python geschrieben ist. Nun sollte eine Website entwickelt werden, welche nur aus dem Netzwerk des Unternehmens, am besten aus dem ifaic, erreichbar ist. Auf dieser Website können dann die Daten aus Icinga mit Hilfe von Graphite dargestellt sowie ein Buchungstool für die Rechenressourcen eingebaut werden. Welches Buchungstool genutzt werden sollte, ist noch offen.

7 Quellenverzeichnis

[B1] Antje Landschutz, *Monitoring mit Prometheus und Grafana bei Systemen unter Last*, 2022. https://www.informatik-aktuell.de/entwicklung/methoden/monitoring-mit-prometheus-und-grafana-bei-systemen-unter-last.html#c32286 (accessed December 11, 2022).

[B2] Betz, L. and Widhalm, T., *Icinga 2: Ein praktischer Einstieg ins Monitoring,* 1. Auflage. iX Edition. Heidelberg: dpunkt.verlag, 2016.

[B3] Elastic, *Kibana: Visualisieren, Analysieren und Erkunden von Daten | Elastic*, 2022. https://www.elastic.co/de/kibana/ (accessed December 12, 2022).

[B4] Ewachuk, R. and Beyer, B., *Google - Site Reliability Engineering: Monitoring Distributed System*, 2021. https://sre.google/sre-book/monitoring-distributed-systems/ (accessed December 12, 2022).

[B5] Grafana Labs, *Grafana*, 2022. https://grafana.com/ (accessed December 12, 2022).

[B6] Graphite, *Graphite*, 2022. https://graphiteapp.org/ (accessed December 12, 2022).

[B7] Gupta, V. and Maan, N., *Top 8 Open Source Dashboards | MetricFire Blog:* MetricFire, 2022. https://www.metricfire.com/blog/top-8-open-source-dashboards/#Kibana (accessed December 12, 2022).

[B8] Icinga, *Icinga » Grafana*, 2021. https://icinga.com/products/integrations/grafana/ (accessed December 12, 2022).

[B9] Icinga, *Icinga*, 2022. https://icinga.com/ (accessed December 11, 2022).

[B10] Kumar, R., *White Box Monitoring and Black Box Monitoring explained!!! - DevOpsSchool.com*, 2020. https://www.devopsschool.com/blog/white-box-monitoring-and-black-box-monitoring-explained/ (accessed December 12, 2022).

[B11] Prometheus, *Blog | Prometheus*, 2021. https://prometheus.io/blog/ (accessed December 11, 2022).

[B12] Prometheus, *Prometheus*, 2022. https://prometheus.io/ (accessed December 11, 2022).

[B13] Read the Docs, *Graphite 1.2.0 documentation*, 2022. https://graphite.readthedocs.io/en/latest/overview.html (accessed December 12, 2022).

[B14] Rogers, K., “Black Box vs. White Box Monitoring: What You Need To Know,” *DevOps.com*, 2018. https://devops.com/black-box-vs-white-box-monitoring-what-you-need-to-know/ (accessed December 12, 2022).

[B15] SourceForge, *MRBS*, 2022. https://mrbs.sourceforge.io/ (accessed December 11, 2022).

[B16] Sukhija, N. and Bautista, E., “Towards a Framework for Monitoring and Analyzing High Performance Computing Environments Using Kubernetes and Prometheus,” 2019. In *2019*

IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), 257–62: IEEE.