



(10) **DE 10 2023 201 143 A1** 2024.08.14

(12) **Offenlegungsschrift**

(21) Aktenzeichen: **10 2023 201 143.3**

(22) Anmeldetag: **13.02.2023**

(43) Offenlegungstag: **14.08.2024**

(51) Int Cl.: **G06F 9/50 (2006.01)**

G06F 9/455 (2006.01)

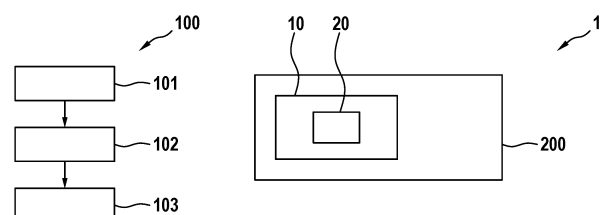
(71) Anmelder: Robert Bosch Gesellschaft mit beschränkter Haftung, 70469 Stuttgart, DE	Pressler, Michael, 76149 Karlsruhe, DE; Grosch, Franz-Josef, 64297 Darmstadt, DE
(72) Erfinder: Hamann, Arne, 71634 Ludwigsburg, DE; Ziegenbein, Dirk, 71691 Freiberg, DE; Dasari, Dakshina Narahari, 71034 Böblingen, DE;	

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen.

(54) Bezeichnung: **Verfahren zur Ressourcenzuweisung für Anwendungen und/oder Anwendungscontainern in einem verteilten System heterogener Rechenknoten**

(57) Zusammenfassung: Die Erfindung betrifft ein Verfahren zur Ressourcenzuweisung für Anwendungen (50) und/oder Anwendungscontainern (60) in einem verteilten System (1) heterogener Rechenknoten (200), welche zumindest hinsichtlich ihrer Hardwareplattformen und/oder Betriebssysteme heterogen ausgeführt sind, umfassend die nachfolgenden Schritte:

- Ausführen (101) einer der Anwendungen (50) und/oder der Anwendungscontainer (60) auf Basis einer Zwischenbefehlssatzarchitektur durch eine Laufzeitumgebung, wobei die Laufzeitumgebung auf zumindest einem der Rechenknoten (200) vorgesehen ist,
- Durchführen (102) einer Ressourcenüberwachung einer Ressourcennutzung bei dem Rechenknoten (200) durch die Laufzeitumgebung, wobei die Ressourcenüberwachung auf Basis der Zwischenbefehlssatzarchitektur durchgeführt wird,
- Durchführen (103) der Ressourcenzuweisung auf Basis der durchgeführten Ressourcenüberwachung.



Beschreibung

[0001] Die Erfindung betrifft ein Verfahren zur Ressourcenzuweisung für Anwendungen und/oder Anwendungscontainern in einem verteilten System heterogener Rechenknoten. Ferner betrifft die Erfindung ein Computerprogramm und eine Vorrichtung zu diesem Zweck.

Stand der Technik

[0002] Herkömmliche Cloud- und Edge-Computing-Lösungen ermöglichen Nutzern den Einsatz und ein Deployment von Anwendungen in der Cloud oder der Edge in verschiedenen virtuellen Ausführungsumgebungen, auch als Container bezeichnet. Die virtuellen Ausführungsumgebungen werden dabei auf Servern (Farmen) gehostet. Die Container können von einer darunter liegenden Schicht überwacht und verwaltet werden, die als Container-Runtime oder im Folgenden auch kurz als Laufzeit (engl. runtime) bezeichnet wird. Diese Laufzeitschicht liegt im Allgemeinen über dem Betriebssystem. Dies kann als ein Systemstapel angesehen werden, der aus der Hardware-schicht, dem Betriebssystem, der Laufzeit und schließlich den Containern besteht, in denen die Anwendungen untergebracht sind.

[0003] Eine wichtige Aufgabe der Laufzeitumgebung besteht darin, eine isolierte Umgebung bereitzustellen, die die einzelnen Container voneinander schützt und außerdem sicherstellt, dass keiner der ausgeführten Container die Systemressourcen monopolisiert oder unverhältnismäßig stark beansprucht. Entsprechend besteht ein Bedarf an Ressourcenmanagement zwischen diesen Laufzeiten, um die Nutzung von Ressourcen wie CPU, Speicher oder Netzwerknutzung zu kontrollieren.

[0004] Üblicherweise stützen sich die Laufzeiten auf das zugrunde liegende Betriebssystem, um Mechanismen zur Ressourcennutzung durchzusetzen. Die aus dem Stand der Technik bekannte Docker-Container-Laufzeitumgebung stützt sich beispielsweise auf Control Groups (cgroups) durch das Linux-Betriebssystem.

[0005] Je nachdem, ob ein bestimmter Rechenknoten ressourcenbeschränkt ist oder nicht, unterstützt er möglicherweise auch nur einige Funktionen des Betriebssystems. Mit anderen Worten: Die Container-Laufzeiten sind ebenfalls durch die Möglichkeiten des Betriebssystems begrenzt. Eine unerwünschte Auswirkung ist, dass ein Edge- bzw. Cloud-Orchestrator, der Richtlinien für die Ressourcenverwaltung von Containern durchsetzen muss, die Fähigkeiten des Betriebssystems kennen und über eine normalisierte (einheitliche) Methode verfügen muss, um die Ressourcenanforderungen für verschiedene Rechenknoten zu spezifizieren. In neueren Edge-Computing-Szenarien, in denen es notwendig sein kann, Anwendungen auf heterogenen Geräten mit jeweils eigenem Betriebssystem bereitzustellen, kann es jedoch eine Umsetzung der Ressourcenzuweisung ohne eine einheitliche Sicht auf die Ressourcenverwaltung technisch herausfordernd sein.

Offenbarung der Erfindung

[0006] Gegenstand der Erfindung ist ein Verfahren mit den Merkmalen des Anspruchs 1, ein Computerprogramm mit den Merkmalen des Anspruchs 9 sowie eine Vorrichtung mit den Merkmalen des Anspruchs 10. Weitere Merkmale und Details der Erfindung ergeben sich aus den jeweiligen Unteransprüchen, der Beschreibung und den Zeichnungen. Dabei gelten Merkmale und Details, die im Zusammenhang mit dem erfindungsgemäßen Verfahren beschrieben sind, selbstverständlich auch im Zusammenhang mit dem erfindungsgemäßen Computerprogramm sowie der erfindungsgemäßen Vorrichtung, und jeweils umgekehrt, so dass bezüglich der Offenbarung zu den einzelnen Erfindungsaspekten stets wechselseitig Bezug genommen wird bzw. werden kann.

[0007] Gegenstand der Erfindung ist insbesondere ein Verfahren zur Ressourcenzuweisung für Anwendungen und/oder Anwendungscontainern in einem verteilten System heterogener Rechenknoten. Dabei können die Rechenknoten zumindest hinsichtlich ihrer Hardwareplattformen und/oder Betriebssysteme heterogen ausgeführt sind, sich also vorzugsweise durch ihre Hardware und/oder der darauf ausgeführten Betriebssystem unterscheiden. Bei dem erfindungsgemäßen Verfahren können die nachfolgenden Schritte vorgesehen sein, welche vorzugsweise automatisiert und/oder nacheinander und/oder wiederholt ausgeführt werden:

- Ausführen wenigstens einer der Anwendungen und/oder der Anwendungscontainer auf Basis einer Zwischenbefehlssatzarchitektur durch eine Laufzeitumgebung, wobei die Laufzeitumgebung auf zumindest einem oder genau einem der Rechenknoten vorgesehen ist, wobei vorzugsweise auch die weiteren Rechenknoten jeweils eine Laufzeitumgebung zur Ausführung der Anwendungen und/oder der Anwendungscontainer aufweisen,

- Durchführen einer Ressourcenüberwachung einer Ressourcennutzung, vorzugsweise der ausgeführten Anwendung und/oder des Anwendungscontainers, bei dem Rechenknoten durch die Laufzeitumgebung, wobei bevorzugt die Ressourcenüberwachung, insbesondere unabhängig von den Hardwareplattformen und/oder der Betriebssysteme, auf Basis der Zwischenbefehlssatzarchitektur durchgeführt wird,
- Durchführen der Ressourcenzuweisung, vorzugsweise für die ausgeführte Anwendung und/oder den Anwendungscontainer, auf Basis der durchgeführten Ressourcenüberwachung.

[0008] Dies hat den Vorteil, dass eine zuverlässige Möglichkeit zur Ressourcenüberwachung und damit eine verbesserte Ressourcenzuweisung ermöglicht werden kann. Es ist dabei möglich, dass die Schritte ebenfalls für weitere Anwendungen und/oder Anwendungscontainer und/oder Rechenknoten des verteilten Systems ausgeführt werden.

[0009] Das Verfahren gemäß der Erfindung kann für eine, vorzugsweise adaptive, Ressourcenzuweisung für Anwendungen und/oder Anwendungscontainer in einem verteilten System heterogener Rechenknoten vorgesehen sein. Dabei können die Rechenknoten dadurch heterogen sein, dass sie unterschiedliche Rechenkapazitäten und/oder unterschiedliche Funktionalität und/oder unterschiedliche Typen von Hardware aufweisen.

[0010] Die heterogenen Rechenknoten können auch als mindestens eine der nachfolgenden Gerätekategorien ausgebildet sein:

- unterschiedliche kleinere IoT-Geräte (IoT steht hierbei für „Internet of Things“),
- unterschiedliche Telefone,
- unterschiedliche Cloud-Server, welche bspw. unterschiedliche Leistung bereitstellen,
- unterschiedliche Rechenknoten bei einem Edge-Computing, z. B. umfassend eine Datenverarbeitungsvorrichtung in einem Fahrzeug und eine Datenverarbeitungsvorrichtung außerhalb des Fahrzeuges,
- unterschiedliche Rechenkapazitäten bei einem SoC, d. h. System-on-Chip,
- unterschiedliche Prozessoren,
- unterschiedliche Rechenkapazitäten, wie Allzweck- oder Spezialfließkomma- oder Grafikverarbeitungs- oder Beschleunigungskapazitäten.

[0011] Damit ist es auch möglich, dass das verteilte System eine verteilte Middleware und/oder verschiedene Betriebssysteme oder Hypervisoren und/oder eine Edge-Computing-Umgebung und/oder eine IoT-Umgebung und/oder eine Cloud-Computing-Umgebung und/oder einen Fahrzeug-Controller umfasst. Ebenfalls ist es optional vorgesehen, dass die Rechenknoten zumindest teilweise mobil ausgeführt sind, sich also während der Ausführung des erfindungsgemäßen Verfahrens bewegen.

[0012] Neben den Rechenknoten können auch die Kommunikationsressourcen und/oder die Anwendungen bzw. Anwendungscontainer heterogen sein, also unterschiedliche Leistung bereitstellen bzw. benötigen. Somit ist das Verfahren vorzugsweise besonders geeignet, für eine heterogene Umgebung eingesetzt zu werden. Ferner können die Anwendungen bzw. Anwendungscontainer ebenfalls heterogen und/oder auch als Anwendungs-Container ausgeführt sein. Die Rechenkapazität kann ferner eine Rechenleistung sein, welche ein Rechenknoten zur Ausführung der Anwendung bereitstellt.

[0013] Als ein Anwendungscontainer oder auch kurz Container bezeichnet kann eine sandboxartige und/oder kontrollierte Umgebung für die Ausführung einer Anwendung (oder mehrerer Anwendungen) bezeichnet werden. Die Anwendung kann bei der Ausführung im Container eine begrenzte Sichtbarkeit und einen begrenzten Zugriff auf die zugrunde liegenden Systemressourcen haben. Die Begrenzung des Zugriffs und der Sichtbarkeit kann dabei von der zugrunde liegenden Container-Laufzeit vorgegeben werden.

[0014] Ein weiterer Vorteil kann im Rahmen der Erfindung erzielt werden, wenn die Laufzeitumgebung eine Verwaltung der ausgeführten Anwendung und/oder des Anwendungscontainers durchführt und die ausgeführte Anwendung und/oder den Anwendungscontainer von den weiteren Anwendungen und/oder Anwendungscontainern in dem verteilten System isoliert. Die weiteren Anwendungen und/oder Anwendungscontainer können dabei ebenfalls durch jeweilige Laufzeitumgebungen der jeweiligen Rechenknoten ausgeführt werden. Auch für diese kann eine entsprechende Ressourcenüberwachung gemäß dem erfindungsgemäßen Verfahren vorgesehen sein. Als eine Laufzeit, vorzugsweise auch Laufzeitumgebung und/oder Container-Laufzeit oder engl. auch als Container Runtime bezeichnet, kann im Rahmen der Erfindung eine Verwal-

tungsschicht zur Verwaltung des Lebenszyklus von Containern verstanden werden. Die Verwaltung kann bspw. das Starten und/oder Herunterfahren und/oder einen Neustart und/oder einen Abbruch der Container bzw. darin ausgeführten Anwendungen umfassen. Ferner kann die Container-Laufzeit zur Isolierung zwischen Containern und zur Überwachung der Ressourcennutzung, des Ressourcenzugriffs und der Ressourcensichtbarkeit der einzelnen Container dienen. Innerhalb des Systemstapels kann dabei die Container-Laufzeit in der Regel über dem Betriebssystem liegen und die Funktionen des Betriebssystems zur Durchsetzung der Ressourcenkontrolle nutzen.

[0015] Das verteilte System kann als ein Cloud-Computing System ausgeführt sein. Als Cloud Computing System werden insbesondere Systeme zur Datenverarbeitung bezeichnet, welche zumindest teilweise an einem vom Benutzer entfernten Standort vorgesehen sind. Die „Cloud“ erscheint dem Benutzer zwar als ein einziger logischer Computer, kann aber mehrere Rechenknoten, kurz als Knoten bezeichnet, umfassen, welche jeweils Anwendungen und/oder Container ausführen können. Jeder Rechenknoten kann somit eine Container-Laufzeit hosten. Die Benutzer können eine Verbindung mit einer Cloud-Schnittstelle herstellen und ihre Anwendung und/oder Container bereitstellen (deployment). Ein spezieller Cloud-Orchestrator kann die Anwendung und/oder den Anwendungscontainer einem geeigneten Knoten zuordnen. Typischerweise sind Clouds durch einen riesigen Cluster einheitlicher Knoten mit hohen Rechenkapazitäten gekennzeichnet.

[0016] Das verteilte System kann als ein Edge-Computing System ausgeführt sein. Eine weitere Möglichkeit zur Bereitstellung von Anwendungen und/oder Containern bietet das Edge-Computing. Die Edge bezieht sich in der Regel auf einen Standort, der sich näher am Nutzer oder an der Anwendung befindet und beherbergt ebenfalls mehrere Rechenknoten, die Anwendungen und/oder Container ausführen können. Die Benutzer können eine Verbindung zur Edge-Schnittstelle herstellen und ihre Anwendung und/oder Container auf der Edge bereitstellen. Ein spezieller Edge-Orchestrator kann dann die eingehende Anwendung und/oder Container einem geeigneten Knoten zuordnen. Typischerweise ist Edge Computing durch einen riesigen Cluster verschiedener heterogener Knoten (von Mikrocontrollern über Einplatinencomputer bis hin zu großen Servern) mit unterschiedlichen Rechenleistungen gekennzeichnet. Edge Computing bietet den Vorteil, dass die Menge der abfließenden Daten reduziert werden können, um bspw. die Latenzzeit aufgrund langer Round-Trip-Zeiten zu einem entfernten Standort zu verringern.

[0017] Ferner kann von der Laufzeit eine Befehlssatzarchitektur und insbesondere Zwischenbefehlssatzarchitektur verwendet werden. Im Allgemeinen definiert eine Befehlssatzarchitektur (auch engl. als Instruction Set Architecture oder kurz ISA bezeichnet) für eine bestimmte physische Architektur wie x86, ARM usw. die unterstützten Befehle und/oder Datentypen und/oder Register und/oder die Hardwareunterstützung für die Verwaltung des Hauptspeichers und/oder grundlegende Merkmale wie Speicherkonsistenz, Adressierungsmodi und virtueller Speicher und das Eingabe-/Ausgabemodell einer Familie von Implementierungen der ISA.

[0018] Die Ressourcenverwaltungsmechanismen in herkömmlichen Virtualisierungsmechanismen sind stark von den Funktionen abhängig, die das zugrunde liegende Betriebssystem und die Architektur bieten. Zum Beispiel unterstützen Linux-Betriebssysteme cgroups, aber QNX oder andere Betriebssysteme haben kein entsprechendes Konzept. Ebenso verfügen ARM-Architekturen im Vergleich zu Intel-Architekturen über einen anderen Satz von Leistungsüberwachungszählern, so dass eine genaue Zuordnung nicht möglich ist. In einem Edge-Computing-System, das aus verschiedenen Betriebssystemen und Architekturen besteht, kann dies zu einer uneinheitlichen Art und Weise führen, mit welcher Ressourcenanforderungen spezifiziert und Ressourcenkontrollen durchgesetzt werden. Außerdem kann es schwierig sein, das Betriebssystem zu ändern, um neuere und angepasste Ressourcenverwaltungsmechanismen hinzuzufügen.

[0019] Es kann ferner möglich sein, dass die heterogenen Rechenknoten verschiedene Betriebssysteme und Architekturen aufweisen, sodass sich die heterogenen Rechenknoten insbesondere hinsichtlich ihrer Ressourcenverwaltungsmechanismen unterscheiden, wobei auf Basis der Zwischenbefehlssatzarchitektur eine einheitliche Ressourcenverwaltung über die verschiedenen Hardwareplattformen der Rechenknoten hinweg bereitgestellt werden kann. Die Erfindung kann dabei den Vorteil bieten, dass eine einheitliche Ressourcenverwaltung über verschiedene Plattformen hinweg möglich ist, insbesondere unabhängig vom Betriebssystem. Hierbei kann eine Voraussetzung lediglich sein, dass die Laufzeitumgebung auf dem jeweiligen Rechenknoten installiert sein muss. Ferner kann die Laufzeitumgebung erweitert werden, um neuere Ressourcenverwaltungsmechanismen bereitzustellen. Die Laufzeit kann mit einem eigenen Scheduler und/oder prioritätsvererbungsähnlichen Mechanismen usw. angereichert werden, unabhängig davon, was das Betriebssystem bietet. Dadurch kann die Betriebssystemschicht verschlankt werden.

[0020] Des Weiteren kann vorgesehen sein, dass die Laufzeitumgebung wenigstens einen Mechanismus zur Durchsetzung von Ressourcen auf Basis der durchgeführten Ressourcenüberwachung aufweist, vorzugsweise, um bei einem Überschreiten eines Grenzwertes durch die Ressourcennutzung der ausgeführten Anwendung diese zu beenden. Dies stellt vorzugsweise eine Möglichkeit dar, um die Ressourcenzuweisung auf Basis der durchgeführten Ressourcenüberwachung durchzuführen. Dabei können vorteilhafterweise auf der Grundlage von Ressourcenüberwachungsfunktionen (insbesondere plattform- und betriebssystemunabhängige) Mechanismen zur Durchsetzung von Ressourcen implementiert werden. Diese Mechanismen können wenigstens eine der folgenden umfassen:

- Durchsetzung der zeitlichen Isolierung zwischen ausgeführten Anwendungen in der Laufzeit, insbesondere unabhängig von den Fähigkeiten des Host-Betriebssystems,
- Ausführung von Anwendungen mit gemischter Kritikalität (z. B. Best-Effort- und Echtzeitanwendungen) durch Durchsetzung von Ressourcenkontrollen,
- Schutz wichtiger Anwendungen vor fehlverhaltenden Anwendungen,
- Handhabung gemeinsam genutzter Ressourcen über Anwendungen hinweg unter Verwendung von Protokollen wie Prioritätsvererbung und Prioritätsobergrenzen.

[0021] Ferner kann bei dem erfindungsgemäßen Verfahren die Instrumentierung ggf. auf der nachstehend beschriebenen Byte-Code-Ebene (IISA-Ebene) unter Bezugnahme auf ein zugrunde liegendes Kostenmodell vorgesehen sein. Es besteht insbesondere keine Abhängigkeit von den zugrunde liegenden Hardware-Leistungszählern für die Instrumentierung.

[0022] Der Ausdruck Instrumentierung kann im Rahmen der Erfindung die Analyse des Codes an verschiedenen Punkten (Instrumentierungspunkten) bezeichnen, um Low-Level-Metriken wie die Anzahl der Anweisungen, die Speichernutzung usw. abzuleiten.

[0023] Ein weiterer Vorteil im Rahmen der Erfindung ist erzielbar, wenn die wenigstens eine ausgeführte Anwendung und/oder der wenigstens eine Anwendungscontainer in einem Bytecode, vorzugsweise gemäß einer Zwischenbefehlssatzarchitektur, vorgesehen ist, welcher durch die Laufzeitumgebung zur Ausführung der Anwendung interpretiert wird. Auf diese Weise kann die Anwendung bzw. der Anwendungscontainer auf Basis der Zwischenbefehlssatzarchitektur durch die Laufzeitumgebung ausgeführt werden. Als Bytecode kann ein Zwischencode bezeichnet werden. Dabei kann der Quellcode zunächst in ein Zwischenformat, den Bytecode, kompiliert und/oder interpretiert und erst dann in die zugrunde liegende Maschinenbefehlsarchitektur übersetzt werden. Der Bytecode kann dabei eine Sammlung von Befehlen für eine virtuelle Maschine umfassen. Eine Beispielformatdefinition für einen Bytecode ist WebAssembly (oder Wasm).

[0024] Es ist insbesondere ein Erfindungsgedanke, dass Laufzeiterweiterungen für ein einheitliches Ressourcenmanagement über verschiedene Plattformen hinweg bereitgestellt werden. Dies ist vorzugsweise möglich durch die Einführung von in die Laufzeit integrierten Ressourcenüberwachungs- und Durchsetzungsfunktionen (engl. resource monitoring bzw. enforcement capabilities) sowie eines Mechanismus zur Normalisierung der Anwendungsleistung über die verschiedenen Plattformen hinweg unter Verwendung des Konzepts der virtuellen Recheneinheiten (engl. virtual computing units).

[0025] Hierbei kann eine Zwischenbefehlssatzarchitektur (IISA) verwendet werden, die von der Laufzeit verstanden wird. Bei der Ausführung von Anwendungen interpretiert die Laufzeitumgebung die generierte IISA und führt sie dann auf dem zugrunde liegenden Rechner, insbesondere Rechenknoten, aus. Jede Anwendung, die auf der Laufzeitumgebung ausgeführt wird, kann bestimmten Ressourcenbeschränkungen unterliegen, die vordefiniert sein können (z. B. von einem Systemdesigner oder einem Benutzer festgelegt werden). Bei der Ausführung von Anwendungen kann die Laufzeitumgebung die Ressourcennutzung überwachen und bei Überschreitung der Ressourcennutzung einen entsprechenden Exception-Handler aufrufen, der die Anwendung entweder beendet oder aussetzt oder andere Maßnahmen gemäß konfigurierbaren Richtlinien ergreift. Die Container-Laufzeitumgebung kann auch einen benutzerdefinierten Scheduler bieten, der die Ausführungsreihenfolge der verschiedenen von der Laufzeitumgebung gehosteten Anwendungen festlegt und jeder dieser Anwendungen eine bestimmte CPU-Reservierung garantiert.

[0026] Nach einer weiteren Möglichkeit kann vorgesehen sein, dass die Ressourcenüberwachung auf Basis eines Kostenmodells durchgeführt wird, welches für die jeweiligen Hardwareplattformen der Rechenknoten spezifisch ist und bei welchem vorzugsweise die Kosten auf Basis einzelner Zwischenbefehle gemäß der Zwischenbefehlssatzarchitektur definiert werden. Auf diese Weise kann die Ressourcenüberwachung auf

Basis der Zwischenbefehlssatzarchitektur durchgeführt werden. Die Leistungsnormalisierung über verschiedene Plattformen hinweg kann damit über ein plattformspezifisches Kostenmodell realisiert werden, das im Wesentlichen eine Übersetzung zwischen der IISA-Ressourcenkontrolle und der zugrunde liegenden ISA darstellt. Wenn sich ein Benutzer mit einer Cloud-/Edge-Schnittstelle verbindet, kann der Orchestrator mit dem vorgeschlagenen Aufbau eine einheitliche Möglichkeit haben, die Ressourcenanforderungen für verschiedene Laufzeiten zu spezifizieren.

[0027] Es ist ferner denkbar, dass die Laufzeitumgebung als virtuelle Maschine mit der Zwischenbefehlssatzarchitektur ausgeführt ist.

[0028] Weiter ist im Rahmen der Erfindung denkbar, dass die Anwendung durch die Laufzeitumgebung auf Basis einer Auswertung von Zwischenbefehlen der Anwendung, vorzugsweise einem Zählen einer Anzahl der Zwischenbefehle, instrumentiert und überwacht wird, wobei hierzu vorzugsweise ein Zwischencode, insbesondere Bytecode, der Anwendung durch einen Instrumentierungscode ergänzt wird.

[0029] Die jeweilige Anwendung kann z. B. eine Funktion zum automatisierten Fahren bereitstellen. Es ist daher möglich, dass das erfindungsgemäße Verfahren bei einem Fahrzeug zum Einsatz kommt. Das Fahrzeug kann bspw. als ein Kraftfahrzeug und/oder Personenkraftfahrzeug und/oder autonomes Fahrzeug ausgebildet sein. Das Fahrzeug kann eine Fahrzeugeinrichtung, bspw. zur Bereitstellung der Funktion zum automatisierten Fahren und vorzugsweise einer autonomen Fahrfunktion, aufweisen. Die Funktion und/oder die Fahrzeugeinrichtung kann dazu ausgeführt sein, das Fahrzeug zumindest teilweise automatisch zu steuern und/oder zu beschleunigen und/oder abzubremesen und/oder zu lenken.

[0030] Ebenfalls Gegenstand der Erfindung ist ein Computerprogramm, insbesondere Computerprogrammprodukt, umfassend Befehle, die bei der Ausführung des Computerprogrammes durch einen Computer diesen veranlassen, das erfindungsgemäße Verfahren auszuführen. Damit bringt das erfindungsgemäße Computerprogramm die gleichen Vorteile mit sich, wie sie ausführlich mit Bezug auf ein erfindungsgemäßes Verfahren beschrieben worden sind.

[0031] Ebenfalls Gegenstand der Erfindung ist eine Vorrichtung zur Datenverarbeitung, die eingerichtet ist, das erfindungsgemäße Verfahren auszuführen. Als die Vorrichtung kann bspw. ein Computer vorgesehen sein, welcher das erfindungsgemäße Computerprogramm ausführt. Der Computer kann wenigstens einen Prozessor zur Ausführung des Computerprogramms aufweisen. Auch kann ein nicht-flüchtiger Datenspeicher vorgesehen sein, in welchem das Computerprogramm hinterlegt und von welchem das Computerprogramm durch den Prozessor zur Ausführung ausgelesen werden kann.

[0032] Ebenfalls Gegenstand der Erfindung kann ein computerlesbares Speichermedium sein, welches das erfindungsgemäße Computerprogramm umfasst. Das Speichermedium ist bspw. als ein Datenspeicher wie eine Festplatte und/oder ein nicht-flüchtiger Speicher und/oder eine Speicherkarte ausgebildet. Das Speichermedium kann z. B. in den Computer integriert sein.

[0033] Darüber hinaus kann das erfindungsgemäße Verfahren auch als ein computerimplementiertes Verfahren ausgeführt sein.

[0034] Weitere Vorteile, Merkmale und Einzelheiten der Erfindung ergeben sich aus der nachfolgenden Beschreibung, in der unter Bezugnahme auf die Zeichnungen Ausführungsbeispiele der Erfindung im Einzelnen beschrieben sind. Dabei können die in den Ansprüchen und in der Beschreibung erwähnten Merkmale jeweils einzeln für sich oder in beliebiger Kombination erfindungswesentlich sein. Es zeigen:

Fig. 1 eine schematische Visualisierung eines Verfahrens, einer Vorrichtung sowie eines Computerprogramms gemäß Ausführungsbeispielen der Erfindung.

Fig. 2 eine schematische Darstellung der vorgeschlagenen Betriebssystem-agnostischen Container-Laufzeitumgebung gemäß Ausführungsbeispielen der Erfindung.

Fig. 3 eine schematische Darstellung eines Edge-Orchestrierungsmechanismus mit der vorgeschlagenen Container-Laufzeit gemäß Ausführungsbeispielen der Erfindung.

[0035] In den nachfolgenden Figuren werden für die gleichen technischen Merkmale auch von unterschiedlichen Ausführungsbeispielen die identischen Bezugszeichen verwendet.

[0036] Fig. 1 veranschaulicht gemäß Ausführungsbeispielen der Erfindung ein Verfahren 100 zur Ressourcenzuweisung für Anwendungen 50 und/oder Anwendungscontainern 60 in einem verteilten System 1 heterogener Rechenknoten 200. Die Rechenknoten 200 können sich dabei hinsichtlich ihrer Hardwareplattformen und/oder Betriebssysteme unterscheiden. Gemäß einem ersten Verfahrensschritt 101 kann eine Ausführung wenigstens oder genau einer der Anwendungen 50 und/oder der Anwendungscontainer 60 auf Basis einer Zwischenbefehlssatzarchitektur durch eine Laufzeitumgebung vorgesehen sein. Dabei kann die Laufzeitumgebung auf zumindest einem der Rechenknoten 200 vorhanden sein. Anschließend kann gemäß einem zweiten Verfahrensschritt 102 eine Ressourcenüberwachung einer Ressourcennutzung bei dem Rechenknoten 200 durch die Laufzeitumgebung durchgeführt werden. Hierbei kann die Ressourcenüberwachung unabhängig von den Hardwareplattformen und/oder der Betriebssysteme auf Basis der Zwischenbefehlssatzarchitektur vorgesehen sein. Gemäß einem dritten Verfahrensschritt 103 ist sodann die Ressourcenzuweisung auf Basis der durchgeführten Ressourcenüberwachung möglich.

[0037] Ferner zeigt Fig. 1 ein Computerprogramm 20 und eine Vorrichtung 10 gemäß Ausführungsbeispielen der Erfindung. Diese können dabei Teil des verteilten Systems 1 und insbesondere eines der Rechenknoten 200 sein.

[0038] In Fig. 2 ist gemäß Ausführungsbeispielen der Erfindung die Laufzeitumgebung in der Form einer Betriebssystem-agnostischen Container-Laufzeit 201 gezeigt. Diese fungiert als Verwaltungsschicht, die den Lebenszyklus von Containern verwaltet, insbesondere das Starten, Herunterfahren, Neustart, Abbruch. Ferner kann die Laufzeit die Isolierung zwischen Containern gewährleisten und die Ressourcennutzung, den Ressourcenzugriff und die Ressourcensichtbarkeit jedes einzelnen Containers überwachen. Vorteilhafterweise werden in verschiedenen Sprachen entwickelte Anwendungen nicht direkt in das Format der Maschinenebene kompiliert, sondern in einen Zwischenbefehlssatz (IIS), der von der Laufzeitumgebung verstanden wird. Das heißt, die Laufzeitumgebung dient als virtuelle Maschine mit einer eigenen Zwischenbefehlssatzarchitektur (IISA). Die Laufzeitumgebung wiederum interpretiert den generierten IIS und führt ihn auf dem zugrunde liegenden physischen Rechner, insbesondere Rechenknoten, aus.

[0039] Jede Anwendung, die auf der Laufzeitumgebung ausgeführt wird, unterliegt bestimmten Ressourcenbeschränkungen, die z. B. von einem Systemdesigner (oder einem Benutzer) festgelegt werden. Bei der Ausführung von Anwendungen überwacht die Laufzeitumgebung die Ressourcennutzung und ruft bei Überschreitung der Ressourcennutzung einen geeigneten Exception-Handler auf, der die Anwendung entweder beendet oder aussetzt oder andere Maßnahmen gemäß konfigurierbarer Richtlinien ergreift. Die Übersetzung zwischen der IISA-Ressourcensteuerung und der zugrunde liegenden ISA erfolgt über ein plattformspezifisches Kostenmodell.

[0040] Da die Anwendungen in die IISA kompiliert werden, ist die generierte IIS für eine bestimmte Anwendung unabhängig vom Betriebssystem oder der zugrunde liegenden Hardwareplattform dieselbe. Dies bedeutet auch, dass Anwendungen auf der IIS-Ebene agnostisch instrumentiert und überwacht werden können - durch Zählen der Anzahl der Zwischenbefehle.

[0041] Zusätzliche Überlegungen zur Hardware der Plattform können mit Hilfe eines Kostenmodells angestellt werden. Das Kostenmodell ist daher ein Mittel zur Normalisierung der Ausführungszeit über verschiedene Plattformen hinweg. Es verwendet den Begriff der virtuellen CPU-Zeiteinheiten (vcpu), d.h. virtuelle CPU-Zeiteinheiten können abgeleitet werden aus: der für die Laufzeit generierten IISA, der tatsächlichen ISA der zugrunde liegenden Maschine und der Frequenz der Maschine.

[0042] Eine Möglichkeit, diese Zuordnung zu realisieren, ist die folgende: Jede Anweisung im IISA wird auf eine interne architekturbezogene Maschinenanweisung abgebildet. Für eine bestimmte Architektur wird die Anzahl der Zyklen zur Ausführung des Befehls bestimmt. Zum Beispiel kann die IISA einen Befehl namens IAdd haben, der auf den internen Add-Befehl abgebildet wird, der für die gegebene Architektur 3 Zyklen verbraucht.

[0043] Für jede Plattform können ferner die Vcpu-Zeiteinheiten definiert werden, die eine Abstraktion der Berechnungszeit einer Anwendung auf dieser bestimmten Plattform sind. Wenn z. B. 1 Sekunde der CPU 10^6 vcpus entspricht oder 1 vcpu einer Mikrosekunde entspricht, kann eine solche Annahme der Normalisierung über verschiedene Maschinen mit unterschiedlichen Frequenzen hinweg dienen. Dann muss insbesondere für jede Zielmaschine im verteilten System ein solches Mapping und die Kosten für die Ausführung eines IISA-Befehls in vcpu-Zeiteinheiten gefunden werden. Für jede Architektur kann ein solches Kostenmodell abgeleitet werden, z. B. wie folgt:

Maschine	IISA Opcode O1	ISA Opcode O2 = M1(o1, X)	Latency in Cycles L = M2(O2, X)	f= Frequency of machine	vcpu time units
X86_ 1Ghz	Add	Add	3	1 GHz	.0003 vcpus
X86_ 1Ghz	Mov	Mov	1	1 GHz	.0001 vcpus

[0044] Es kann ferner ein benutzerdefinierter Scheduler (engl. custom scheduler) vorgesehen sein. Die vorgeschlagene Container-Laufzeitumgebung wird dann durch einen benutzerdefinierten Scheduler ergänzt, der über die Reihenfolge der Ausführung der fertigen Anwendungen auf der Laufzeitumgebung entscheidet. Jeder Anwendung kann eine Priorität, ein Budget und ein Wiederauffüllungszeitraum zugewiesen werden. Die Container-Laufzeitumgebung unterhält vorzugsweise eine Warteschlange von Anwendungen, die zur Ausführung bereit sind. Bei jedem Scheduling-Ereignis (Ticken eines Timers, Ankunft einer neuen Anwendung, Beendigung/Suspendierung einer Anwendung, Ankunft eines Interrupts oder einer Nachricht, oder eine Anwendung wird blockiert/freigegeben) kann der Scheduler die Ready Queue durchsuchen, um die nächste auszuführende Anwendung auszuwählen. Ein benutzerdefinierter Zeitplanungsmechanismus kann wie folgt ausgeführt sein:

- Aus der Menge der bereitstehenden Anwendungen filtert der Scheduler alle Anwendungen heraus, die das Budget noch nicht ausgeschöpft haben
- Aus der Menge der ausgewählten Anwendungen wählt er die Anwendung mit der höchsten Priorität aus und startet sie zur Ausführung.

[0045] Die Grenzen der CPU-Nutzung können in der Form eines Budgets, vorzugsweise CPU-Budgets in vcpu-Zeiteinheiten und eines Wiederauffüllungsintervalls, angegeben werden. Das Budget kann den Umfang der Verarbeitung (auf IISA-Ebene) angeben, den eine Anwendung ausführen kann, bevor sie angehalten wird. Während die Anwendung ausgeführt wird, wird ihr Budget aufgebraucht. Wenn es Null wird, wird die Anwendung ausgesetzt. Nach jedem Auffüllungsintervall kann das Budget der Anwendung wieder voll aufgeladen und die Anwendung wieder in die Warteschlange gestellt werden. Mit dem oben beschriebenen Mechanismus kann den Anwendungen eine bestimmte Verarbeitungszeit durch die zugrunde liegende Laufzeit garantiert werden. Ferner ermöglicht dieser Mechanismus die Koexistenz von sicherheitskritischen und Best-Effort-Anwendungen auf derselben Plattform.

[0046] Allgemeiner formuliert können entsprechend dem Beispiel für reservierungsbasiertes Scheduling auch andere benutzerdefinierte Scheduling-Mechanismen von der Container-Laufzeit definiert werden, um die Ausführung verschiedener Anwendungen zu planen. Abhängig von der Anzahl der der Container-Laufzeit zur Verfügung stehenden Kerne kann der Scheduler bspw. Anwendungen auf jedem dieser Kerne planen. Ebenso können Budgets für die Bandbreiten- und Netznutzung festgelegt werden, um den Verbrauch dieser Ressourcen zu begrenzen bzw. zu erzwingen.

[0047] Die Anwendungen können in eine Zwischendarstellung IISA und damit in Zwischencode und/oder Bytecode kompiliert werden. Wenn die Überwachung aktiviert ist, kann der generierte IISA-Code durch Instrumentierungscode ergänzt werden, der die Anzahl der ausgeführten Anweisungen und die verbrauchte CPU-Zeit zählt. Das Programm kann in Basisblöcke (lineare Ausführungssequenzen) unterteilt werden, und um verschiedene Basisblöcke herum können Instrumentierungspunkte hinzugefügt werden, um die den einzelnen Basisblöcken entsprechenden CPU-Zeiteinheiten zu addieren. Während die Anwendung ausgeführt wird, steigt ihr CPU-Verbrauch.

[0048] Wenn das Budget überschritten wird, kann die Laufzeitumgebung auf verschiedene Weise damit umgehen. Die Laufzeitumgebung kann die Anwendung z. B. unterbrechen, die Ausführung drosseln oder beenden oder einen anderen vom Benutzer konfigurierten Mechanismus verwenden. Eine ähnliche Instrumentierung kann für die Nutzung des Speichers und des Netzes vorgenommen werden.

[0049] Ein weiterer Vorteil ergibt sich dadurch, dass die vcpu-Zeiteinheiten in die Laufzeitumgebung integriert sind und nicht von externen Hardware-Leistungszählern oder der Unterstützung spezieller Ressourcen-Kontingenzierungsmechanismen wie cgroups im Betriebssystem abhängen. Damit kann eine einheitliche Ressourcenkontrolle auf der Anwendungsebene selbst mit Hilfe eines zusätzlichen Kostenmodells spezifiziert und durchgesetzt werden. Ein beispielhaftes Cloud- oder Edge-Orchestrierungs-Setup mit der vorgeschlagenen Lösung ist in **Fig. 3** dargestellt. Dabei bezeichnet 301 das Betriebssystem, 302 die Hardware,

303 eine Edge- und/oder Cloud Schnittstelle, 304 einen Orchestrator, 305 das Kostenmodell und 306 ein Cluster aus Rechenknoten 200.

[0050] Ferner kann die vorgeschlagene Lösung für ein Ressourcenmanagement über Betriebssysteme/Hypervisoren und/oder in Fahrzeug-Computern eingesetzt werden.

[0051] Die voranstehende Erläuterung der Ausführungsformen beschreibt die vorliegende Erfindung ausschließlich im Rahmen von Beispielen. Selbstverständlich können einzelne Merkmale der Ausführungsformen, sofern technisch sinnvoll, frei miteinander kombiniert werden, ohne den Rahmen der vorliegenden Erfindung zu verlassen.

Patentansprüche

1. Verfahren (100) zur Ressourcenzuweisung für Anwendungen (50) und/oder Anwendungscontainern (60) in einem verteilten System (1) heterogener Rechenknoten (200), welche zumindest hinsichtlich ihrer Hardwareplattformen und/oder Betriebssysteme heterogen ausgeführt sind, umfassend die nachfolgenden Schritte:

- Ausführen (101) wenigstens einer der Anwendungen (50) und/oder der Anwendungscontainer (60) auf Basis einer Zwischenbefehlssatzarchitektur durch eine Laufzeitumgebung, wobei die Laufzeitumgebung auf zumindest einem der Rechenknoten (200) vorgesehen ist,
- Durchführen (102) einer Ressourcenüberwachung einer Ressourcennutzung bei dem zumindest einen Rechenknoten (200) durch die Laufzeitumgebung, wobei die Ressourcenüberwachung auf Basis der Zwischenbefehlssatzarchitektur durchgeführt wird,
- Durchführen (103) der Ressourcenzuweisung auf Basis der durchgeführten Ressourcenüberwachung.

2. Verfahren (100) nach Anspruch 1, **dadurch gekennzeichnet**, dass die Laufzeitumgebung eine Verwaltung der ausgeführten Anwendung (50) und/oder des Anwendungscontainers (60) durchführt und die ausgeführte Anwendung (50) und/oder den Anwendungscontainer (60) von den weiteren Anwendungen (50) und/oder Anwendungscontainern (60) in dem verteilten System (1) isoliert.

3. Verfahren (100) nach einem der vorhergehenden Ansprüche, **dadurch gekennzeichnet**, dass die heterogenen Rechenknoten (200) verschiedene Betriebssysteme und Architekturen aufweisen, so dass sich die heterogenen Rechenknoten (200) hinsichtlich ihrer Ressourcenverwaltungsmechanismen unterscheiden, wobei auf Basis der Zwischenbefehlssatzarchitektur eine einheitliche Ressourcenverwaltung über die verschiedenen Hardwareplattformen der Rechenknoten (200) hinweg bereitgestellt wird.

4. Verfahren (100) nach einem der vorhergehenden Ansprüche, **dadurch gekennzeichnet**, dass die Laufzeitumgebung wenigstens einen Mechanismus zur Durchsetzung von Ressourcen auf Basis der durchgeführten Ressourcenüberwachung aufweist, vorzugsweise, um bei einem Überschreiten eines Grenzwertes durch die Ressourcennutzung der ausgeführten Anwendung (50) diese zu beenden.

5. Verfahren (100) nach einem der vorhergehenden Ansprüche, **dadurch gekennzeichnet**, dass die ausgeführte Anwendung (50) in einem Bytecode vorgesehen ist, welcher durch die Laufzeitumgebung zur Ausführung der Anwendung (50) interpretiert wird.

6. Verfahren (100) nach einem der vorhergehenden Ansprüche, **dadurch gekennzeichnet**, dass die Ressourcenüberwachung auf Basis eines Kostenmodells durchgeführt wird, welches für die jeweiligen Hardwareplattformen der Rechenknoten (200) spezifisch ist und bei welchem die Kosten auf Basis einzelner Zwischenbefehle gemäß der Zwischenbefehlssatzarchitektur definiert werden.

7. Verfahren (100) nach einem der vorhergehenden Ansprüche, **dadurch gekennzeichnet**, dass die Laufzeitumgebung als virtuelle Maschine mit der Zwischenbefehlssatzarchitektur ausgeführt ist.

8. Verfahren (100) nach einem der vorhergehenden Ansprüche, **dadurch gekennzeichnet**, dass die Anwendung (50) durch die Laufzeitumgebung auf Basis einer Auswertung von Zwischenbefehlen der Anwendung (50), vorzugsweise einem Zählen einer Anzahl der Zwischenbefehle, instrumentiert und überwacht wird, wobei hierzu vorzugsweise ein Zwischencode der Anwendung (50) durch einen Instrumentierungscode ergänzt wird.

9. Computerprogramm (20), umfassend Befehle, die bei der Ausführung des Computerprogramms (20) durch einen Computer (10) diesen veranlassen, das Verfahren (100) nach einem der vorhergehenden Ansprüche auszuführen.

10. Vorrichtung (10) zur Datenverarbeitung, die eingerichtet ist, das Verfahren (100) nach einem der Ansprüche 1 bis 8 auszuführen.

Es folgen 3 Seiten Zeichnungen

Anhängende Zeichnungen

Fig. 1

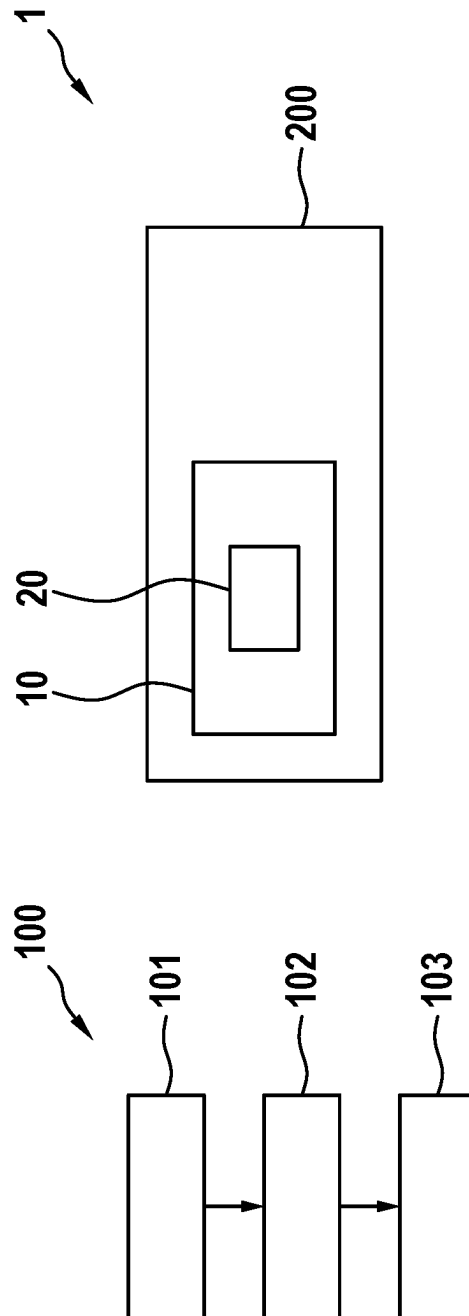


Fig. 2

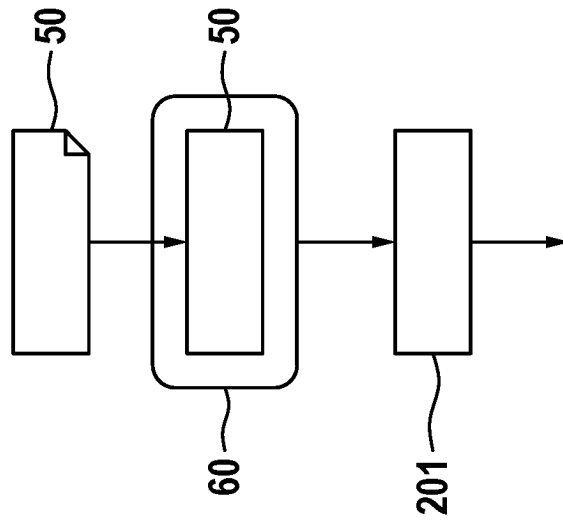


Fig. 3

