(19) Deutsches Patent- und Markenamt

(10) **DE 11 2015 006 438 T5** 2018.01.04

(12) # Veröffentlichung

der internationalen Anmeldung mit der
(87) Veröffentlichungs-Nr.: **WO 2016/162720**
in deutscher Übersetzung (Art. III § 8 Abs. 2 IntPatÜG)
(21) Deutsches Aktenzeichen: **11 2015 006 438.9**
(86) PCT-Aktenzeichen: **PCT/IB2015/000883**
(86) PCT-Anmeldetag: **10.04.2015**
(87) PCT-Veröffentlichungstag: **13.10.2016**
(43) Veröffentlichungstag der PCT Anmeldung
in deutscher Übersetzung: **04.01.2018**

(51) Int Cl.: ***G06F 8/52*** (2018.01)
G06F 9/455 (2006.01)
G06F 9/44 (2018.01)

(71) Anmelder:
GOOGLE INC., Mountain View, Calif., US

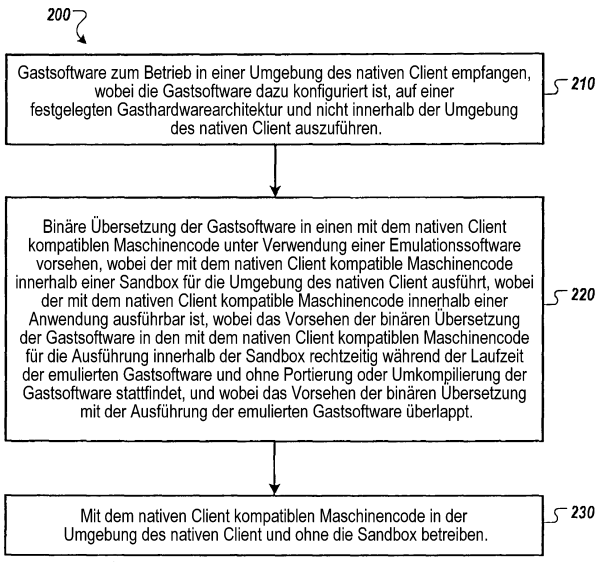
(74) Vertreter:
Betten & Resch Patent- und Rechtsanwälte PartGmbB, 80333 München, DE

(72) Erfinder:
Eltsin, Evgeny, Moscow, RU; Igotti, Nikolay, Moscow, RU; Polukhin, Dmitry, Moscow, RU; Khalyavin, Andrey, Moscow, RU

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen.

(54) Bezeichnung: **Binäre Übersetzung in nativen Client**

(57) Zusammenfassung: Es werden Systeme und Verfahren für die binäre Übersetzung offenbart. In einigen Implementierungen wird eine Gastsoftware zum Betrieb in einer Umgebung eines nativen Client empfangen. Die Gastsoftware ist dazu konfiguriert, auf einer festgelegten Gasthardwarearchitektur und nicht innerhalb der Umgebung des nativen Client auszuführen. Eine binäre Translation der Gastsoftware in einen mit dem nativen Client kompatiblen Maschinencode wird unter Verwendung einer Emulationssoftware geschaffen. Der mit dem nativen Client kompatible Maschinencode führt innerhalb einer Sandbox für die Umgebung des nativen Client aus. Der mit dem nativen Client kompatible Maschinencode ist innerhalb einer Anwendung ausführbar. Das Vorsehen der binären Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode für die Ausführung innerhalb der Sandbox geschieht rechtzeitig während einer Laufzeit der emulierten Gastsoftware und ohne Portierung oder Umkompilierung der Gastsoftware. Das Vorsehen der binären Übersetzung überlappt mit der Ausführung der emulierten Gastsoftware.



Beschreibung

QUERVERWEIS AUF VERWANDTE ANMELDUNG

[0001] Diese Anmeldung bezieht sich auf die internationale Patentanmeldung, die gleichzeitig hiermit eingereicht wird, mit der Anwaltsregisternr. 096553-0073 und mit dem Titel "BINARY TRANSLATION ON SHARED OBJECT LEVEL", deren gesamte Offenbarung hier durch Bezugnahme vollständig mit aufgenommen ist.

HINTERGRUND

[0002] Die vorliegende Technologie ist im Allgemeinen auf binäre Übersetzungstechniken gerichtet. Einige Software wird als binäre Programme für spezielle CPU-Architekturen wie z. B. ARM® oder x86® und ein spezielles Betriebssystem wie z. B. Android® oder Microsoft Windows® kompiliert. Die binären Programme können in einen Computer eines Benutzers über das Internet heruntergeladen werden. Der Benutzer kann jedoch dem Programm nicht trauen und kann das Programm in einem sicheren Modus laufen lassen wollen, wobei das Programm begrenzten Zugriff auf Daten hat, die auf dem Computer außerhalb des Programms gespeichert sind. Wie das Vorangehende darstellt, kann eine Methode zum sicheren Ausführen einer Software auf einem Computer erwünscht sein.

[0003] Ein nativer Client kann verwendet werden, um Software auf dem Computer sicher auszuführen. Einige Software ist jedoch nicht für einen nativen Client geschrieben und ist nicht mit dem nativen Client kompatibel. Eine solche Software kann für den nativen Client umkompiliert und portiert werden müssen, was eine nicht triviale Anstrengung für einige große, moderne Softwareprodukte erfordern kann. Wie das Vorangehende darstellt, kann das Übersetzen eines Codes, der mit einem nativen Client nicht kompatibel ist, in den nativen Client erwünscht sein.

ZUSAMMENFASSUNG

[0004] Gemäß einigen Aspekten bezieht sich die vorliegende Technologie auf ein Verfahren. Das Verfahren umfasst das Empfangen einer Gastsoftware zum Betrieb in einer Umgebung des nativen Client, wobei die Gastsoftware dazu konfiguriert ist, auf einer festgelegten Gasthardwarearchitektur und nicht innerhalb der Umgebung des nativen Client auszuführen. Das Verfahren umfasst das Vorsehen einer binären Übersetzung der Gastsoftware in einen mit dem nativen Client kompatiblen Maschinencode unter Verwendung einer Emulationssoftware, wobei der mit dem nativen Client kompatible Maschinencode innerhalb einer Sandbox für die Umgebung des nativen Client ausführt, wobei der mit dem nativen Client kompatible Maschinencode innerhalb einer Anwendung ausführbar ist, wobei das Vorsehen der binären Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode für die Ausführung innerhalb der Sandbox rechtzeitig während einer Laufzeit der emulierten Gastsoftware und ohne Portierung oder Umkompilierung der Gastsoftware stattfindet, und wobei das Vorsehen der binären Übersetzung mit der Ausführung der emulierten Gastsoftware überlappt. Das Verfahren umfasst das Betreiben des mit dem nativen Client kompatiblen Maschinencodes in der Umgebung des nativen Client und innerhalb der Sandbox.

[0005] Gemäß einigen Aspekten bezieht sich die vorliegende Technologie auf ein nicht transitorisches computerlesbares Medium, das Befehle speichert. Die Befehle umfassen einen Code zum Empfangen einer Gastsoftware zum Betrieb in einer Umgebung des nativen Client, wobei die Gastsoftware dazu konfiguriert ist, auf einer festgelegten Gasthardwarearchitektur und nicht innerhalb der Umgebung des nativen Client auszuführen. Die Befehle umfassen einen Code zum Vorsehen einer binären Übersetzung der Gastsoftware in einen mit dem nativen Client kompatiblen Maschinencode unter Verwendung einer Emulationssoftware, wobei der mit dem nativen Client kompatible Maschinencode innerhalb einer Sandbox für den mit dem nativen Client kompatiblen Maschinencode ausführt, wobei er mit dem nativen Client kompatible Maschinencode innerhalb einer Anwendung ausführbar ist, wobei das Vorsehen der binären Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode für die Ausführung innerhalb der Sandbox rechtzeitig während einer Laufzeit der emulierten Gastsoftware und ohne Portierung oder Umkompilierung der Gastsoftware stattfindet, und wobei das Vorsehen der binären Übersetzung mit der Ausführung der emulierten Gastsoftware überlappt. Das Vorsehen der binären Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode umfasst: Erzeugen eines Satzes von virtuellen Registern, die Register darstellen, die durch die Gastsoftware verwendet werden, wenn sie auf der festgelegten Gasthardwarearchitektur ausgeführt wird, wobei die Adressen der virtuellen Register durch einen Basiszeiger (RBP) plus einen vorbestimmten Versatz bezeichnet werden, und wobei jedes virtuelle Register in dem Satz von virtuellen Registern aus dem Inneren der Sandbox über einen einzelnen Befehl zugänglich ist.

[0006] Gemäß einigen Aspekten bezieht sich die vorliegende Technologie auf ein System. Das System umfasst einen oder mehrere Prozessoren und einen Arbeitsspeicher, der Befehle speichert. Die Befehle umfassen einen Code zum Empfangen einer Gastsoftware zum Betrieb in einer Umgebung des nativen Client, wobei die Gastsoftware dazu konfiguriert ist, auf einer festgelegten Gasthardwarearchitektur und nicht innerhalb der Umgebung des nativen Cli-

ent auszuführen. Die Befehle umfassen einen Code zum Vorsehen einer binären Übersetzung der Gastsoftware in einen mit dem nativen Client kompatiblen Maschinencode unter Verwendung einer Emulationssoftware, wobei der mit dem nativen Client kompatible Maschinencode innerhalb einer Sandbox für die Umgebung des nativen Client ausführt, wobei der mit dem nativen Client kompatible Maschinencode innerhalb einer Anwendung ausführbar ist, wobei das Vorsehen der binären Übersetzung der Gastsoftware in den nativen Client für die Ausführung innerhalb der Sandbox-Software rechtzeitig während einer Laufzeit der emulierten Gastsoftware und ohne Portierung oder Umkompilierung der Gastsoftware stattfindet, wobei die Sandbox für die Umgebung des nativen Client auf einen Satz von emulierten Gastregistern zugreift, die im Arbeitsspeicher oder in Registern gespeichert sind, die an einer Host-Maschine verfügbar sind, wobei Daten, die dem mit dem nativen Client kompatiblen Maschinencode zugeordnet sind, innerhalb der Sandbox für die Umgebung des nativen Client gespeichert werden, und wobei die emulierten Gastregister Registern der festgelegten Gasthardwarearchitektur entsprechen.

[0007] Selbstverständlich werden andere Konfigurationen der vorliegenden Technologie aus der folgenden ausführlichen Beschreibung leicht ersichtlich, wobei verschiedene Konfigurationen der vorliegenden Technologie zur Erläuterung gezeigt und beschrieben werden. Wie erkannt wird, ist die vorliegende Technologie zu anderen und unterschiedlichen Konfigurationen in der Lage und ihre verschiedenen Details sind zu einer Modifikation in verschiedenen anderen Hinsichten in der Lage, alles ohne vom Schutzbereich der vorliegenden Technologie abzuweichen. Folglich sollen die Zeichnungen und die ausführliche Beschreibung als im Wesen erläuternd und nicht als einschränkend betrachtet werden.

KURZBESCHREIBUNG DER ZEICHNUNGEN

[0008] Merkmale der vorliegenden Technologie werden in den beigefügten Ansprüchen dargelegt. Für den Zweck der Erläuterung werden jedoch mehrere Aspekte des offenbarten Gegenstands in den folgenden Figuren dargelegt.

[0009] Fig. 1A stellt eine Beispielgastmaschine dar, die an der binären Übersetzung in einen nativen Client beteiligt sein kann.

[0010] Fig. 1B stellt eine Beispiel-Host-Maschine dar, die an der binären Übersetzung in einen nativen Client beteiligt sein kann.

[0011] Fig. 2 stellt einen Beispielprozess dar, durch den eine binäre Übersetzung in einen nativen Client vollendet werden kann.

[0012] Fig. 3 stellt begrifflich ein elektronisches Beispielsystem dar, mit dem einige Implementierungen der vorliegenden Technologie implementiert werden.

AUSFÜHRLICHE BESCHREIBUNG

[0013] Die nachstehend dargelegte ausführliche Beschreibung ist als Beschreibung von verschiedenen Konfigurationen der vorliegenden Technologie bestimmt und soll nicht die einzigen Konfigurationen darstellen, in denen die vorliegende Technologie ausgeführt werden kann. Die beigefügten Zeichnungen sind hier integriert und bilden einen Teil der ausführlichen Beschreibung. Die ausführliche Beschreibung umfasst spezielle Details für den Zweck des Vorsehens eines gründlichen Verständnisses der vorliegenden Technologie. Es ist jedoch klar und ersichtlich, dass die vorliegende Technologie nicht auf die hier dargelegten speziellen Details begrenzt ist und ohne diese speziellen Details ausgeführt werden kann. In einigen Fällen sind bestimmte Strukturen und Komponenten in Blockdiagrammform gezeigt, um es zu vermeiden, die Konzepte der vorliegenden Technologie unklar zu machen.

[0014] Wie hier verwendet, kann sich der "native Client" auf eine Sandbox-Umgebung beziehen, die Software-Fehlerisolations- und Binärcode-Überprüfungsmethoden einsetzt. Der native Client implementiert einen begrenzten Satz von Anwendungsprogrammierschnittstellen (APIs) ähnlich zu einer portierbaren Betriebssystemschnittstelle (POSIX) und kann innerhalb eines Webbrowsers ausgeführt werden. Die "binäre Übersetzung" kann sich auf einen Mechanismus beziehen, um einen Maschinencode (Gastcode genannt) einer Anwendung oder ein Betriebssystem, das auf einer Plattform (dem Gast) ausführt, um einen für die Ausführung auf einer anderen Plattform (dem Host) geeigneten Code zu erzeugen, beziehen. Eine "Plattform" kann eine Hardware- und Software-Stapelkombination umfassen. Anwendungen werden typischerweise für eine spezielle Plattform wie z. B. ARM oder x86 entwickelt. Jeder der obigen Begriffe umfasst auch seine einfache und gewöhnliche Bedeutung.

[0015] Die vorliegende Technologie bezieht sich auf Techniken der binären Übersetzung zum Betrieb einer existierenden Software, die für eine Gastplattform (z. B. Android ARM) entwickelt ist, auf einer anderen Plattform, der Host-Plattform (z. B. nativer Client x86-64), die bessere API-Ebenen-Portierbarkeit-, Sicherheits- und Zuverlässigkeitseigenschaften besitzt. Eine Anwendung für die Gastplattform kann in einer portierbaren Programmiersprache (z. B. Java) geschrieben sein, die auf native Komponenten in Form von dynamisch geladenen Bibliotheken zugreifen kann. Die nativen Komponenten können als geteilte Objekte bezeichnet werden und können auf der Gastplattform unter Verwendung der Hardware-

architektur und des Betriebssystems der Gastplattform kompiliert werden. Um die Anwendung zu unterstützen, kann die Host-Plattform sowohl Java als auch den nativen Code betreiben müssen. Die vorliegende Technologie bezieht sich auf den Betrieb von nicht portierbaren, für die Gastplattform spezifischen Anwendungsbibliotheken innerhalb der Sandbox des nativen Client.

[0016] Gemäß einigen Implementierungen der vorliegenden Technologie wird, um den Gastcode innerhalb der Sandbox des nativen Client auszuführen, ein Host-Code, der den Sandbox-Regeln des nativen Client entspricht, erzeugt. Um eine vernünftige Leistung zu erzielen, können Optimierungstechniken angewendet werden. Die Optimierungstechniken können das Erzeugen eines rechtzeitigen mit der Softwarefehlerisolation (SFI) kompatiblen Host-Codes umfassen, der das Verhalten des Gastcodes emuliert. Die Optimierungstechniken können das Speichern von emulierten Gastregistern in einem schnellen Speicher wie z. B. dem Cache in den Hostregistern oder in einem relativen Basiszeigerarbeitsspeicher (RBP-Arbeitsspeicher) umfassen. Diese Erzeugung des Host-Codes ermöglicht einen schnellen Zugriff ohne explizite Sandbox-Zugriffe auf den emulierten Gastkontext. Die Optimierungstechniken können das Emulieren von Aufrufen von externen geteilten Plattformobjekten unter Verwendung von geteilten Objekten des Host, mit anderem Worten eine vorangehende binäre Übersetzung von Standardbibliotheken, wenn möglich, umfassen. Diese Emulationsstrategie ist im Einzelnen in der US-Patentanmeldung, die gleichzeitig hiermit eingereicht wird, mit der Anwaltsregisternr. 093054-0893 und mit dem Titel "BINARY TRANSLATION ON SHARED OBJECT LEVEL" beschrieben, deren gesamte Offenbarung durch den Hinweis hier aufgenommen wird.

[0017] In einigen Fällen kann die Host-Plattform keinen Zugriff auf einige APIs haben, die auf der Gastplattform verfügbar sind, beispielsweise aufgrund von Sicherheitsanforderungen. Zusätzliche Techniken, um eine nicht verfügbare Funktionalität zu emulieren, werden verwendet.

[0018] Einige Zentraleinheitsarchitekturen (CPU-Architekturen) unterstützen eine explizite Markierung eines ausführbaren Codes über ein geeignetes Bit (X-Bit) in der Seitentabelle. Ein Code von einem virtuellen Arbeitsspeicher kann nur ausgeführt werden, wenn das X-Bit gesetzt ist. Eine direkte Unterstützung der X-Bit-Bearbeitung aus dem Inneren der Sandbox des nativen Client könnte in einigen Fällen aus Sicherheitsgründen nicht zugelassen werden. Da die Gastcodeausführung durch den binären Übersetzer die Anwesenheit des Host-X-Bit im Gastcode nicht berücksichtigt, könnten auch solche Bearbeitungen in einigen Fällen nicht den gewünschten Effekt haben. Um die Funktionalität in Bezug auf die Bearbeitung des X-Bits zu emulieren (z. B. typischerweise unter Verwendung der mmap(PROT_EXEC)-API in POSIX-Systemen geschaffen), müssen zusätzliche Strukturen, die durch die Gast-mmap()-API bearbeitet werden, durch die binäre Übersetzungsmaschine berücksichtigt werden. Dies kann über ein Bitmap im Arbeitsspeicher durchgeführt werden, der speichert, ob eine spezielle Gastseite ausführbar ist oder nicht (z. B. ob das X-Bit der speziellen Gastseite gesetzt ist). Um eine Prüfung des Bitmap bei der Ausführung jedes Befehls zu vermeiden, können X-Bit-Prüfungen während der Übersetzungszeit stattfinden. Später kann das Ändern des X-Bits für ausführbare Seiten eine Code-Cache-Leerung verursachen.

[0019] Der native Client kann das POSIX-Merkmal von Signalen nicht unterstützen. Signale können verwendet werden, um Informationen über Ausnahmesituationen in einer Anwendung zu liefern, wie z. B. ein Zusammenbruch oder ein Ausführungsfehler. Signale können auch für Speicherbereinigungs- und gemanagte Sprachlaufzeiten verwendet werden. Software, die Signale verwendet, kann in einigen Fällen nicht in der Sandbox des nativen Client laufen können, wenn nicht die binäre Übersetzungsmaschine die Einstellung von Signalsteuerungsprogrammen und die Lieferung der Signale zu den für den Gast spezifischen Signalsteuerungsprogrammen emuliert. Um Signale zu emulieren, kann folglich die binäre Übersetzungsmaschine eine aktive Signalmaske aufrechterhalten. Wenn ein Signal geliefert werden soll, kann die Ausführung der binären übersetzten Software innerhalb des nativen Client unterbrochen werden und das Signalsteuerungsprogramm kann ausgeführt werden. Nachdem das Signalsteuerungsprogramm ausgeführt hat, kann die Ausführung der binären übersetzten Software fortfahren. Diese Verarbeitung von Signalen kann durch Hinzufügen von Prüfungen der Signalmaske innerhalb des Zuteilers der binären Übersetzung und explizites Aufrufen des Signalsteuerungsprogramms mit korrekten Argumenten unter Verwendung eines Funktionshüllmechanismus erreicht werden.

[0020] Der native Client ist eine Sandbox-Technologie zum sicheren Betrieb eines nativen kompilierten Codes (z. B. in einigen Fällen ein kompilierter C– oder C++-Code) in einem Browser. Wie hier verwendet, können sich die Ausdrücke "Sandbox" oder "Sandbox-Technologie" auf einen Computersicherheitsmechanismus zum Trennen von laufenden Programmen durch Begrenzen eines Raums im Arbeitsspeicher, auf den ein Programm zugreifen kann, beziehen. Es wird gesagt, dass ein Sandbox-Programm auf einen Arbeitsspeicherbereich "innerhalb der Sandbox" zugreifen kann und auf einen Arbeitsspeicherbereich "außerhalb der Sandbox" nicht zugreifen kann. Der Code des nativen Client ist vom Betriebssystem unabhängig und kann auf irgendeinem Betriebssystem betrieben werden, solange eine Unterstützung

für das Betriebssystem implementiert wird. Der native Client ermöglicht, dass ein Programmierer einen nativen Code von irgendeinem Client-System durch einen Webbrowser entwickelt, verteilt und ausführt, während die Sicherheitsmerkmale des Browsers aufrechterhalten werden. Der native Client schafft die Fähigkeit, einen nativen Code innerhalb einer Sandbox auszuführen und ermöglicht keinen direkten Zugriff auf den Browser oder auf das Host-Betriebssystem außerhalb der Sandbox. Der native Client kann einen begrenzten Zugriff auf den Browser oder das Host-Betriebssystem durch gesteuerte APIs schaffen. Die binäre Übersetzung in den mit dem nativen Client kompatiblen Maschinencode stellt sicher, dass nur ein legitimer Sandbox-Zugriff geschieht. Obwohl Aspekte der vorliegenden Technologie hier in Verbindung mit einem Browser beschrieben werden, kann in alternativen Implementierungen eine andere Anwendung anstelle des Browsers verwendet werden.

[0021] Die vorliegende Technologie übersetzt eine Software, die mit dem nativen Client nicht kompatibel ist, in einen mit dem nativen Client kompatiblen Maschinencode. Vorteilhafterweise kann ein Benutzer eine Software von einer Quelle betreiben, der er nicht traut, und sicher sein, dass die Software innerhalb einer Sandbox läuft und begrenzten Zugriff auf Informationen hat, die auf dem Computer des Benutzers außerhalb der Sandbox gespeichert sind. Die Übersetzung findet während der Laufzeit der emulierten Gastsoftware statt, die innerhalb des nativen Client läuft. Eine "rechtzeitige" Übersetzung findet während der Laufzeit des Codes zu der Zeit statt, zu der der Code ausgeführt wird. Folglich wird nur ein Code, der ausgeführt wird und der übersetzt werden muss, übersetzt. Abschnitte des Codes, die nicht ausgeführt werden, werden nicht übersetzt, wodurch Zeit und Verarbeitungsressourcen gespart werden. Die binäre Übersetzungstechnologie zusammen mit einer nicht trivialen Portierungsschicht kann eingesetzt werden. Der binäre Übersetzer kann den Code der Gastsoftware übersetzen. Standardbibliotheksaufrufe können jedoch unter Verwendung der Portierungsschicht ausgeführt werden, die in der Emulationssoftware bereitgestellt wird. Die Portierungsschicht kann die erwartete API der Gastplattform in den Satz von verfügbaren APIs auf dem nativen Client übersetzen. In einigen Fällen kann eine solche Übersetzung die Übersetzung von synchronen sperrenden Bibliotheksaufrufen in asynchrone nicht sperrende Aufrufe des nativen Client sowie das Emulieren von Merkmalen der Gasthardware, die im nativen Client fehlen, innerhalb des nativen Client umfassen.

[0022] Die binäre Übersetzung kann die Gastsoftware in ein Format übersetzen, das für eine weitere Überprüfung und Sandboxing geeignet ist. Software-Fehlerisolationstechniken (SFI-Techniken) können verwendet werden, um eine bessere Portierbarkeit und Sicherheit zu erreichen.

[0023] Die vorliegende Technologie kann als Spezialfall der binären Übersetzung gesehen werden, wobei der native Client das Betriebssystem des Host ist, was zu sicheren Programmen führt, die in einem Browser laufen können, der innerhalb irgendeines Betriebssystems ausführt. Infolge der Übersetzung in den nativen Client werden sowohl Sicherheit als auch Portabilität der Software erhöht.

[0024] Während der binären Übersetzung kann der Code in ein Format übersetzt werden, das mit den Regeln des nativen Client kompatibel ist. Der native Client erfordert beispielsweise, dass keine Befehle, die Adressen überqueren, die Vielfache von 32 sind, geliefert werden (mit anderen Worten, Befehle sind bündelausgerichtet). Wenn der binäre Übersetzer einen Maschinencode aussendet, stellt folglich der binäre Übersetzer sicher, dass die Bündelgrenze (Vielfaches von 32) niemals überquert wird.

[0025] In einigen Aspekten der vorliegenden Technologie empfängt ein Computer eine Gastsoftware zum Betrieb in einer Umgebung des nativen Client. Die Gastsoftware ist dazu konfiguriert, auf einer festgelegten Gasthardwarearchitektur (z. B. x86-Hardware oder ARM-Hardware) auszuführen. Die Gastsoftware kann nicht sicher sein oder kann nicht portierbar sein. Der Computer übersetzt die Gastsoftware binär in einen mit dem nativen Client kompatiblen Maschinencode. Der mit dem nativen Client kompatible Maschinencode führt innerhalb einer Sandbox aus, die den Zugriff des mit dem nativen Client kompatiblen Maschinencodes auf Daten begrenzt, die auf der Host-Maschine gespeichert sind. Der mit dem nativen Client kompatible Maschinencode ist sicher, portierbar und innerhalb eines Browsers in irgendeinem Betriebssystem ausführbar, das den Browser unterstützt.

[0026] Die Sandbox kann auf eine Teilmenge eines Satzes von Registern zugreifen können, die an der Host-Maschine verfügbar sind. Wichtige Gastdaten, wie z. B. Gastregister, können von der Gastsoftware zum mit dem nativen Client kompatiblen Maschinencode über eine Teilmenge des Satzes von Registern oder über Arbeitsspeicherpositionen an einem Versatz von Registern in der Teilmenge geliefert werden. Da der binäre übersetzte Code eine intensive Verwendung der Gastregister erfordern kann, kann diese Methode eine Emulation der Gasthardware mit hoher Leistung ermöglichen, während immer noch die Anforderungen der Sandbox des nativen Client erfüllt werden.

[0027] Das Vorsehen einer binären Übersetzung der Gastsoftware in einen mit dem nativen Client kompatiblen Maschinencode kann das Erzeugen eines Satzes von virtuellen Registern umfassen, die Register darstellen, die von der Gastsoftware verwendet werden, wenn sie auf der Host-Hardwarearchitektur

ausgeführt wird. Die Adressen der virtuellen Register können durch RBP plus einen vorbestimmten Versatz bezeichnet werden. Das Register R0 kann sich beispielsweise in der Position RBP befinden, das Register R1 kann sich in der Position RBP + 4 Versatzbytes befinden, das Register R2 kann sich in der Position RBP + 8 Versatzbytes befinden, usw. Folglich kann jedes virtuelle Register in dem Satz von virtuellen Registern aus dem Inneren der Sandbox über einen einzelnen Befehl zugänglich sein. Wenn beispielsweise der Schreibbefehl ein Format: SCHREIBEN (POSITION, WERT) aufweist, kann ein Befehl zum Schreiben des Werts "0" in das Register R2 geschrieben werden als: SCHREIBEN(RBP + 8, 0). Der einzelne Befehl kann vorteilhaft sein, da diese Technik zur maximalen Leistung des erzeugten Codes führt. Die Sandbox-Mechanismen des nativen Client ermöglichen spezielle Ausnahmen für die RBP-relative Adressierung. Insbesondere kann ein einzelner [RBP + 4·N]-Befehl im binären übersetzten Code verwendet werden, um auf das Register Nummer N zuzugreifen. Das RBP-Register wird als Basis verwendet. In einigen Fällen kann ein zusätzlicher Sandbox-Befehl erforderlich sein, um sicherzustellen, dass der Arbeitsspeicherzugriff sich innerhalb der Sandbox befindet.

[0028] Das Vorsehen einer binären Übersetzung der Gastsoftware in einen mit dem nativen Client kompatiblen Maschinencode kann das Emulieren von Merkmalen der festgelegten Gasthardwarearchitektur und Anwendungsprogrammierschnittstellenaufrufe (API-Aufrufe) der Gastsoftware innerhalb des nativen Client umfassen. Die Merkmale der festgelegten Gasthardwarearchitektur können Register, Stapel, Zeiger usw. umfassen. Die API-Aufrufe können Aufrufe an die System-APIs oder APIs, die der Gastsoftware zugeordnet sind, umfassen. Die Register, Stapel, Zeiger usw. werden emuliert, was bedeutet, dass der mit dem nativen Client kompatible Maschinencode Softwaredarstellungen von Hardwareelementen erzeugt, die sich ähnlich zu den Hardwareelementen verhalten, aber in einer Software existieren, die auf einem Prozessor und einem Arbeitsspeicher (ohne die zugehörigen Hardwarekomponenten) ausführt. Aufrufe, die auf die emulierten Komponenten zugreifen, können exakt dieselben wie oder ähnlich zu den Aufrufen sein, die auf die physikalischen Komponenten an der Gastmaschine zugreifen, was den Betrieb eines unmodifizierten Gastcodes in der emulierten Umgebung ermöglicht.

[0029] Um bestimmte Aspekte der Gastarchitektur zu emulieren, die innerhalb der Umgebung des nativen Client nicht verfügbar sind, wie z. B. ein ausführbares Bit (X-Bit) und asynchrone Signalverarbeitung, kann eine zusätzliche Softwareemulationslogik verwendet werden. Der native Client ermöglicht beispielsweise keine großzügige Steuerung über ausführbare Codebereiche oder Verfügbarkeit für POSIX-Systeme über die mmap(PROT_EXEC)-Schnittstelle. Folglich kann die Emulationssoftware eine Steuerung über ausführbare Bits über einen zusätzlichen Mechanismus implementieren. Die Emulationssoftware kann den ausführbaren Gastcode über separate Datenstrukturen verfolgen. Da der native Client keine Signallieferung unterstützt, kann ebenso die Emulationssoftware das Senden von asynchronen Signalen durch periodisches Prüfen einer Maske für ausstehende Signale senden und explizites Aufrufen eines Signalsteuerungsprogramms, sobald das Signal verursacht wird, unterstützen. Wie hier verwendet, kann sich der Ausdruck "Emulationssoftware" oder "Emulator" auf eine Software beziehen, die ermöglicht, dass die Gastsoftware innerhalb der Umgebung des nativen Client läuft.

[0030] Fig. 1A stellt eine Beispielgastmaschine **100A** dar, die an einer binären Übersetzung in einen nativen Client beteiligt sein kann. Die Gastmaschine **100A** kann irgendeine Rechenvorrichtung, beispielsweise ein Laptop-Computer, ein Desktop-Computer, ein Tablet-Computer, ein Mobiltelefon, ein persönlicher digitaler Assistent (PDA), ein elektronisches Musikabspielgerät, eine intelligente Uhr, ein Fernsehgerät, das mit einem oder mehreren Prozessoren und einem Arbeitsspeicher gekoppelt ist, usw. sein. In einigen Beispielen weist die Gastmaschine **100A** eine ARM-Hardware auf. Alternativ kann die Gastmaschine **100A** eine x86-Hardware aufweisen.

[0031] Wie gezeigt, umfasst die Gastmaschine **100A** eine Verarbeitungseinheit **102A**, eine Netzschnittstelle **104A** und einen Arbeitsspeicher **106A**. Die Verarbeitungseinheit **102A** umfasst einen oder mehrere Prozessoren. Die Verarbeitungseinheit **102A** kann eine Zentraleinheit (CPU), eine Graphikverarbeitungseinheit (GPU) oder irgendeine andere Verarbeitungseinheit umfassen. Die Verarbeitungseinheit **102A** führt Computerbefehle aus, die in einem computerlesbaren Medium, beispielsweise dem Arbeitsspeicher **106A**, gespeichert sind. Die Netzschnittstelle **104A** ermöglicht, dass die Gastmaschine **100A** Daten in einem Netz, beispielsweise dem Internet, einem Intranet, einem zellularen Netz, einem lokalen Netz, einem weiträumigen Netz, einem verdrahteten Netz, einem drahtlosen Netz, einem virtuellen privaten Netz (VPN) usw., sendet und empfängt. Der Arbeitsspeicher **106A** speichert Daten und/oder Befehle. Der Arbeitsspeicher **106A** kann einer oder mehrere einer Cache-Einheit, einer Speichereinheit, einer internen Arbeitsspeichereinheit oder einer externen Arbeitsspeichereinheit sein. Wie dargestellt, umfasst der Arbeitsspeicher **106A** Gastregister **108A**, ein Gastsoftwareprogramm **110A** und Gast-APIs **112A**.

[0032] Die Gastregister **108A** sind Register auf der Gastmaschine **100A**, die der Hardwarearchitektur der Gastmaschine **100A** zugeordnet sind (z. B. ARM oder x86). Zusätzlich zu den Registern **108A** kann

der Arbeitsspeicher **106A** andere Hardwarearchitekturen wie z. B. (einen) Stapel oder Zeiger umfassen. Die Gast-APIs **112A** sind APIs, die auf der Gastmaschine **100A** vorhanden sind, unter deren Verwendung die Software für die Gastmaschine **100A** und ihre zugehörige Hardwarearchitektur geschrieben werden kann. Die Gast-APIs **112A** können System-APIs, die der Hardware der Gastmaschine **100A** zugeordnet sind, oder Verkäufer-APIs, die durch den Verkäufer des Gastsoftwareprogramms **110A** zusammen mit dem Gastsoftwareprogramm **110A** bereitgestellt werden, umfassen.

[0033] Das Gastsoftwareprogramm **110A** ist ein Softwareprogramm, das dazu konfiguriert ist, auf der Hardwarearchitektur der Gastmaschine **100A** auszuführen und mit den Gastregistern **108A** und mit den Gast-APIs **112A** über eine Schnittstelle zu koppeln. Das Gastsoftwareprogramm **110A** kann außerstande sein, in einer Umgebung des nativen Client oder auf einer Hardwarearchitektur, die von jener der Gastmaschine **100A** verschieden ist, auszuführen. Wenn beispielsweise die Gastmaschine **100A** eine ARM-Hardware aufweist, kann das Gastsoftwareprogramm **110A** dazu konfiguriert sein, auf der ARM-Hardware, aber nicht auf der x86-Hardware auszuführen.

[0034] Fig. 1B stellt eine Beispiel-Host-Maschine **100B** dar, die an der binären Übersetzung auf einer Ebene von geteilten Objekten beteiligt sein kann. Die Host-Maschine **100B** kann irgendeine Rechenvorrichtung, beispielsweise ein Laptop-Computer, ein Desktop-Computer, ein Tablet-Computer, ein Mobiltelefon, ein persönlicher digitaler Assistent (PDA), ein elektronisches Musikabspielgerät, eine intelligente Uhr, ein Fernsehgerät, das mit einem oder mehreren Prozessoren und einem Arbeitsspeicher gekoppelt ist, usw. sein. Die Host-Maschine **100B** weist eine Hardwarearchitektur auf, die von der Hardwarearchitektur der Gastmaschine **100A** verschieden ist. Wenn beispielsweise die Hardwarearchitektur der Gastmaschine **100A** ARM-Hardware ist, ist die Hardwarearchitektur der Host-Maschine **100B** nicht ARM-Hardware und kann beispielsweise x86-Hardware sein. Alternativ kann die Host-Maschine **100B** dieselbe Hardwarearchitektur wie die Gastmaschine **100A** aufweisen.

[0035] Wie gezeigt, umfasst die Host-Maschine **100B** eine Verarbeitungseinheit **102B**, eine Netzschnittstelle **104B** und einen Arbeitsspeicher **106B**. Die Verarbeitungseinheit **102B** umfasst einen oder mehrere Prozessoren. Die Verarbeitungseinheit **102B** kann eine Zentraleinheit (CPU), eine Graphikverarbeitungseinheit (GPU) oder irgendeine andere Verarbeitungseinheit umfassen. Die Verarbeitungseinheit **102B** führt Computerbefehle aus, die in einem computerlesbaren Medium, beispielsweise dem Arbeitsspeicher **106B**, gespeichert sind. Die Netzschnittstelle **104B** ermöglicht, dass die Gastmaschine **100B** Daten in einem Netz, beispielsweise dem Internet, einem Intranet, einem zellularen Netz, einem lokalen Netz, einem weiträumigen Netz, einem verdrahteten Netz, einem drahtlosen Netz, einem virtuellen privaten Netz (VPN) usw., sendet und empfängt. Der Arbeitsspeicher **106B** speichert Daten und/oder Befehle. Der Arbeitsspeicher **106B** kann einer oder mehrere einer Cache-Einheit, einer Speichereinheit, einer internen Arbeitsspeichereinheit oder einer externen Arbeitsspeichereinheit sein. Wie dargestellt, umfasst der Arbeitsspeicher **106B** sichere Host-Register **114B**, einen sicheren Host-Inhalt **116B** und eine Sandbox **120B** des nativen Client.

[0036] Die sicheren Host-Register **114B** sind Register auf der Host-Maschine **100B**, die der Hardwarearchitektur der Gastmaschine **100B** zugeordnet sind (z. B. ARM oder x86). Zusätzlich zu den sicheren Host-Registern **114B** kann der Arbeitsspeicher **106B** andere Hardwarearchitekturen umfassen, wie z. B. (einen) Stapel oder Zeiger. Der sichere Host-Inhalt **116B** umfasst Inhalt, der auf der Host-Maschine **100B** gespeichert ist, wie z. B. Textverarbeitungsdokumente, Photographien, Videos, Audiodateien usw. Die sicheren Host-Register **114B** und der sichere Host-Inhalt **116B** befinden sich außerhalb der Sandbox **120B** des nativen Client und sind von der Sandbox **120B** des nativen Client aus nicht zugänglich.

[0037] Die Sandbox **120B** des nativen Client ist ein sicherer Bereich des Arbeitsspeichers **106B**, in dem ein Code des nativen Client ausführen kann. Der Code des nativen Client kann innerhalb der Sandbox **120B** des nativen Client ausführen und kann nicht auf Register oder Daten außerhalb der Sandbox **120B** des nativen Client zugreifen, wie z. B. die sicheren Host-Register **114B** oder den sicheren Host-Inhalt **116B**. In dieser Weise kann ein Benutzer der Host-Maschine **100B** einen potentiell nicht vertrauenswürdigen Code des nativen Client ausführen, während er sicher ist, dass seine persönlichen Daten und sicheren Register der Host-Maschine **100B** für den potentiell nicht vertrauenswürdigen Code nicht zugänglich sind.

[0038] Wie gezeigt, umfasst die Sandbox **120B** des nativen Client zugängliche Host-Register **122B**, emulierte Gastregister **108B**, einen mit dem nativen Client kompatiblen Maschinencode **110B** und emulierte Gast-APIs **112B**. Der mit dem nativen Client kompatible Maschinencode **110B** entspricht dem Gastsoftwareprogramm **110A**, das binär in den nativen Client übersetzt wird. Der mit dem nativen Client kompatible Maschinencode **110B** führt innerhalb der Sandbox **120B** des nativen Client aus. Ähnlich zum anderen Code des nativen Client kann der mit dem nativen Client kompatible Maschinencode **110B** nicht auf Register oder Daten außerhalb der Sandbox **120B** des nativen Client zugreifen, wie z. B. die sicheren

Host-Register **114B** oder den sicheren Host-Inhalt **116B**. Der mit dem nativen Client kompatible Maschinencode **110B** ist innerhalb eines Browsers auf im Wesentlichen irgendeiner Hardware oder irgendeinem Betriebssystem ausführbar. Insbesondere kann der mit dem nativen Client kompatible Maschinencode **110B** auf der Host-Maschine **100B** ausführen, die eine Hardwarearchitektur aufweisen kann, die von der Hardwarearchitektur der Gastmaschine **100A** verschieden ist. In einigen Fällen kann die binäre Übersetzung des Gastsoftwareprogramms **110A** in den mit dem nativen Client kompatiblen Maschinencode **110B** rechtzeitig während der Laufzeit des Programms des mit dem nativen Client kompatiblen Maschinencodes und ohne Portierung oder Umkompilierung des mit dem nativen Client kompatiblen Maschinencodes stattfinden. Folglich wird nur ein Code, der ausgeführt wird und der übersetzt werden muss, übersetzt. Abschnitte des Codes, die nicht ausgeführt werden, werden nicht übersetzt, wodurch Zeit und Verarbeitungsressourcen gespart werden.

[0039] Die emulierten Gastregister **108B** entsprechen den Gastregistern **108A** der Gastmaschine **100A**, die in Software emuliert wird, die innerhalb der Sandbox **120B** des nativen Client ausführt. Andere Hardware wie z. B. (ein) Stapel oder Zeiger, die vom Gastsoftwareprogramm **110A** verwendet werden, das auf der Gastmaschine **100A** ausführt, können auch in der Sandbox **120B** des nativen Client emuliert werden. Die Register **108B** werden emuliert, was bedeutet, dass der mit dem nativen Client kompatible Maschinencode Softwaredarstellungen von Hardwareelementen erzeugt, die sich ähnlich zu den Hardwareelementen verhalten, aber in einer Software existieren, die auf einem Prozessor und einem Arbeitsspeicher ausführt (ohne die zugehörigen Hardwarekomponenten). Aufrufe, die auf die emulierten Komponenten zugreifen, können exakt dieselben wie oder ähnlich zu den Aufrufen sein, die auf die physikalischen Komponenten auf der Gastmaschine zugreifen, was ermöglicht, dass der Code leicht von der Gastmaschine **100A** auf die Sandbox **120B** des nativen Client portiert wird. Gemäß einigen Fällen wird der Code des Gastsoftwareprogramms **110A** durch den Code ersetzt, der mit der Umgebung des nativen Client kompatibel ist.

[0040] Die emulierten Gast-APIs **112B** entsprechen den Gast-APIs **112A** der Gastmaschine **100A**, die binär in Software übersetzt oder emuliert werden, die innerhalb der Sandbox **120B** des nativen Client ausführt. Die emulierten Gast-APIs **112B** können Emulationen von System-APIs, die der Hardware der Gastmaschine **100A** zugeordnet sind, oder Verkäufer-APIs, die durch den Verkäufer des Gastsoftwareprogramms **110A** zusammen mit dem Gastsoftwareprogramm **110A** bereitgestellt werden, umfassen. Ähnlich dazu, wie das Gastsoftwareprogramm **110A** auf die Gastregister **108A** und die Gast-APIs **112A** an der Gastmaschine **100A** zugreift, kann der entsprechende mit dem nativen Client kompatible Maschinencode **110B** auf die emulierten Gastregister **108B** und die emulierten Gast-APIs **112B** an der Sandbox **120B** des nativen Client zugreifen.

[0041] Die Sandbox **120B** des nativen Client umfasst auch zugängliche Host-Register **122B**. Die zugänglichen Host-Register **122B** sind Register der Host-Maschine **100B**, die aus dem Inneren der Sandbox **120B** des nativen Client für Software, einschließlich des mit dem nativen Client kompatiblen Maschinencodes **110B**, der innerhalb der Sandbox **120B** des nativen Client ausführt, zugänglich sind. Die Register der Host-Maschine können entweder sichere Host-Register **114B**, die für den mit dem nativen Client kompatiblen Maschinencode unzugänglich sind, oder zugängliche Host-Register **122B**, auf die durch den mit dem nativen Client kompatiblen Maschinencode zugegriffen werden kann, sein.

[0042] Fig. 2 stellt einen Beispielprozess **200** dar, durch den eine binäre Übersetzung in einen nativen Client vollendet werden kann.

[0043] Der Prozess **200** beginnt in Schritt **210**, wobei eine Host-Maschine (z. B. Host-Maschine **100B**) eine Gastsoftware (z. B. Gastsoftwareprogramm **110A**) zum Betrieb in einer Umgebung des nativen Client (z. B. Sandbox **120B** des nativen Client) empfängt. Die Gastsoftware ist dazu konfiguriert, auf einer festgelegten Gasthardwarearchitektur (z. B. Gastmaschine **100A**) und nicht innerhalb der Umgebung des nativen Client auszuführen. Die Umgebung des nativen Client ist dazu konfiguriert, innerhalb irgendeiner von mehreren verschiedenen Hardwarearchitekturen auszuführen. Beispielsweise kann die Umgebung des nativen Client dazu konfiguriert sein, innerhalb einer Anwendung wie z. B. eines Browsers auszuführen, die in irgendeiner von mehreren verschiedenen Hardwarearchitekturen existieren kann.

[0044] In Schritt **220** sieht die Host-Maschine unter Verwendung einer Emulationssoftware eine binäre Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode (z. B. den mit dem nativen Client kompatiblen Maschinencode **110B**) vor. Der mit dem nativen Client kompatible Maschinencode führt innerhalb einer Sandbox (z. B. Sandbox **120B** des nativen Client) für die Umgebung des nativen Client aus. Der mit dem nativen Client kompatible Maschinencode ist innerhalb einer Anwendung wie z. B. eines Browsers ausführbar. Das Vorsehen der binären Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode für die Ausführung innerhalb der Sandbox kann rechtzeitig während der Laufzeit der emulierten Gastsoftware und ohne Portierung oder Umkompilierung der Gastsoftware stattfinden. Das Vorsehen der

binären Übersetzung überlappt mit der Ausführung der emulierten Gastsoftware.

[0045] Die Gastsoftware kann beispielsweise eine ARM-Software oder x86-Software umfassen, die dazu ausgelegt ist, auf einem ARM-Hardwaresystem oder einem x86-Hardwaresystem auszuführen. Irgendein Code in der Gastsoftware kann durch den Code des nativen Client oder den Code, der mit der Umgebung des nativen Client kompatibel ist, ersetzt werden. Die Gastsoftware kann in einigen Fällen nicht sicher oder nicht portierbar sein. Der mit dem nativen Client kompatible Maschinencode kann sicher und portierbar sein. Die Gastsoftware kann dazu konfiguriert sein, nur innerhalb eines festgelegten Betriebssystems auszuführen. Die Umgebung des nativen Client kann innerhalb irgendeines von mehreren Betriebssystemen ausführen können.

[0046] Das Vorsehen der binären Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode kann das Erzeugen eines Satzes von virtuellen Registern (z. B. emulierte Gastregister **108B**), die Register darstellen (z. B. Gastregister **108A**), die durch die Gastsoftware verwendet werden, wenn sie auf der festgelegten Gasthardwarearchitektur ausgeführt wird, umfassen. Adressen der virtuellen Register können durch einen Basiszeiger (RBP) plus einen vorbestimmten Versatz bezeichnet werden. Infolge der Technik von RBP plus Versatz kann jedes Register aus dem Inneren der Sandbox über einen einzelnen Befehl zugänglich sein. Das Register R0 kann sich beispielsweise an der Adresse RBP befinden, das Register R1 kann sich an der Adresse RBP + 4 befinden, das Register R2 kann sich an der Adresse RBP + 8 befinden, das Register Rn kann sich an der Adresse RBP + 4n befinden, usw. (wobei n eine ganze Zahl ist).

[0047] Das Vorsehen der binären Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode kann das Emulieren von Merkmalen der Gasthardwarearchitektur und API-Aufrufe der Gastsoftware innerhalb des nativen Client umfassen. System-APIs oder Verkäufer-APIs von der Gasthardwarearchitektur können binär übersetzt werden, um innerhalb der Sandbox des nativen Client auszuführen. Alternativ können einige der System-APIs der Gasthardwarearchitektur durch System-APIs der Sandbox des nativen Client ersetzt werden.

[0048] Während der binären Übersetzung kann die Gastsoftware in ein Format übersetzt werden, das mit den Regeln des nativen Client kompatibel ist. Der native Client erfordert beispielsweise, dass keine Befehle, die Adressen überqueren, die durch 32 dividierbar sind, geliefert werden. In diesen Fällen können Adressen, die durch 32 dividierbar sind, als "Bündelgrenze" bezeichnet werden. Alle Befehle, die durch den Emulator erzeugt werden, werden in einer solchen Weise ausgesendet, dass die Befehle niemals die Bündelgrenze überqueren, und mit Keine-Operation-Befehlen (NOP-Befehlen) gefüllt sind. Die Sandbox für die Umgebung des nativen Client greift auf einen Satz von emulierten Gastregistern zu, die im Arbeitsspeicher oder in Registern gespeichert sind, die auf der Host-Maschine zugänglich sind. Daten, die dem mit dem nativen Client kompatiblen Maschinencode zugeordnet sind, werden innerhalb der Sandbox für die Umgebung des nativen Client gespeichert. Die emulierten Gastregister entsprechen Registern der festgelegten Gasthardwarearchitektur.

[0049] In Schritt **230** betreibt die Host-Maschine den mit dem nativen Client kompatiblen Maschinencode in der Umgebung des nativen Client und innerhalb der Sandbox. Der mit dem nativen Client kompatible Maschinencode kann auf Register und Daten innerhalb der Sandbox des nativen Client zugreifen, kann jedoch nicht auf Register oder Daten außerhalb der Sandbox des nativen Client zugreifen. Die Sandbox-Software greift auf emulierte Gastregister zu, die vorübergehend in Registern des Host oder im Arbeitsspeicher gespeichert werden. Nach Schritt **230** endet der Prozess **200**.

[0050] Wie vorstehend beschrieben, werden die Schritte **210**–**230** des Prozesses **200** gemäß einer bestimmten Reihenfolge und der Reihe nach implementiert. Die Schritte **210**–**230** können jedoch in irgendeiner Reihenfolge implementiert werden. In einigen Beispielen können zwei oder mehr der Schritte **210**–**230** parallel implementiert werden.

[0051] **Fig.** 3 stellt begrifflich ein elektronisches System **300** dar, mit dem einige Implementierungen der vorliegenden Technologie implementiert werden. Eine oder mehrere der Gastmaschine **100A** oder der Host-Maschine **100B** können beispielsweise unter Verwendung der Anordnung des elektronischen Systems **300** implementiert werden. Das elektronische System **300** kann ein Computer (z. B. ein Mobiltelefon, PDA) oder irgendeine andere Art von elektronischer Vorrichtung sein. Ein solches elektronisches System umfasst verschiedene Typen von computerlesbaren Medien und Schnittstellen für verschiedene andere Typen von computerlesbaren Medien. Das elektronische System **300** umfasst einen Bus **305**, (einen) Prozessor(en) **310**, einen Systemarbeitsspeicher **315**, einen Festwertarbeitsspeicher **320**, eine dauerhafte Speichervorrichtung **325**, eine Eingabevorrichtungsschnittstelle **330**, eine Ausgabevorrichtungsschnittstelle **335** und eine Netzschnittstelle **340**.

[0052] Der Bus **305** stellt gemeinsam alle System-, Peripheriegerät- und Chipsatz-Busse dar, die kommunikativ die zahlreichen internen Vorrichtungen des elektronischen Systems **300** verbinden. Der Bus **305** verbindet beispielsweise kommunikativ den (die) Prozessor(en) **310** mit dem Festwertarbeitsspeicher **320**,

dem Systemarbeitsspeicher **315** und der dauerhaften Speichervorrichtung **325**.

[0053] Aus diesen verschiedenen Arbeitsspeichereinheiten ruft (rufen) der (die) Prozessor(en) **310** Befehle zur Ausführung und Daten zum Verarbeiten ab, um die Prozesse der vorliegenden Technologie auszuführen. Der (die) Prozessor(en) kann (können) einen einzelnen Prozessor oder einen Mehrkernprozessor in verschiedenen Implementierungen umfassen.

[0054] Der Festwertarbeitsspeicher (ROM) **320** speichert statische Daten und Befehle, die für den (die) Prozessor(en) **310** und andere Module des elektronischen Systems erforderlich sind. Die dauerhafte Speichervorrichtung **325** ist andererseits eine Lese- und Schreib-Arbeitsspeichervorrichtung. Die Vorrichtung ist eine nicht transitorische Arbeitsspeichereinheit, die Befehle und Daten speichert, selbst wenn das elektronische System **300** ausgeschaltet ist. Einige Implementierungen der vorliegenden Technologie verwenden eine Massenspeichervorrichtung (beispielsweise eine magnetische oder optische Platte und ihr entsprechendes Plattenlaufwerk) als dauerhafte Speichervorrichtung **325**.

[0055] Andere Implementierungen verwenden eine entnehmbare Speichervorrichtung (beispielsweise eine Diskette, ein Flash-Laufwerk oder ein Plattenlaufwerk) als dauerhafte Speichervorrichtung **325**. Wie die dauerhafte Speichervorrichtung **325** ist der Systemarbeitsspeicher **315** eine Lese- und Schreib-Arbeitsspeichervorrichtung. Im Gegensatz zur Speichervorrichtung **325** ist jedoch der Systemarbeitsspeicher **315** ein flüchtiger Lese- und Schreib-Arbeitsspeicher wie z. B. ein Direktzugriffsarbeitsspeicher. Der Systemarbeitsspeicher **315** speichert einige der Befehle und Daten, die der Prozess zur Laufzeit benötigt. In einigen Implementierungen werden die Prozesse der vorliegenden Technologie im Systemarbeitsspeicher **315**, in der dauerhaften Speichervorrichtung **325** oder im Festwertarbeitsspeicher **320** gespeichert. Die verschiedenen Arbeitsspeichereinheiten umfassen beispielsweise Befehle für die binäre Übersetzung in den nativen Client gemäß einigen Implementierungen. Aus diesen verschiedenen Arbeitsspeichereinheiten ruft (rufen) der (die) Prozessor(en) **310** Befehle zur Ausführung und Daten zur Verarbeitung ab, um die Prozesse von einigen Implementierungen auszuführen.

[0056] Der Bus **305** verbindet auch mit den Eingabe- und Ausgabevorrichtungsschnittstellen **330** und **335**. Die Eingabevorrichtungsschnittstelle **330** ermöglicht, dass der Benutzer Informationen und ausgewählte Befehle an das elektronische System übermittelt. Eingabevorrichtungen, die bei der Eingabevorrichtungsschnittstelle **330** verwendet werden, umfassen beispielsweise alphanumerische Tastaturen und Zeigevorrichtungen (auch "Cursor-Steuervorrichtungen" genannt). Ausgabevorrichtungsschnittstellen **335** ermöglichen beispielsweise die Anzeige von Bildern, die durch das elektronische System **300** erzeugt werden. Ausgabevorrichtungen, die bei der Ausgabevorrichtungsschnittstelle **335** verwendet werden, umfassen beispielsweise Drucker und Anzeigevorrichtungen, beispielsweise Kathodenstrahlröhren (CRT) oder Flüssigkristallanzeigen (LCD). Einige Implementierungen umfassen Vorrichtungen, beispielsweise einen Berührungsbildschirm, der sowohl als Eingabe- als auch Ausgabevorrichtung funktioniert.

[0057] Wie in **Fig.** 3 gezeigt, koppelt der Bus **305** schließlich auch das elektronische System **300** mit einem Netz (nicht dargestellt) durch eine Netzschnittstelle **340**. In dieser Weise kann das elektronische System **300** ein Teil eines Netzes von Computern (beispielsweise eines lokalen Netzes (LAN), eines weiträumigen Netzes (WAN) oder eines Intranets oder eines Netzes von Netzen, beispielsweise des Internets, sein. Irgendeine oder alle Komponenten des elektronischen Systems **300** können in Verbindung mit der vorliegenden Technologie verwendet werden.

[0058] Die vorstehend beschriebenen Merkmale und Anwendungen können als Softwareprozesse implementiert werden, die als Satz von Befehlen festgelegt sind, die auf einem computerlesbaren Speichermedium (auch als computerlesbares Medium bezeichnet) aufgezeichnet sind. Wenn diese Befehle durch einen oder mehrere Prozessoren ausgeführt werden (die beispielsweise einen oder mehrere Prozessoren, Kerne von Prozessoren oder andere Verarbeitungseinheiten umfassen können), bewirken sie, dass der (die) Prozessor(en) die Handlungen durchführt (durchführen), die in den Befehlen angegeben sind. Beispiele von computerlesbaren Medien umfassen, sind jedoch nicht begrenzt auf CD-ROMs, Flash-Laufwerke, RAM-Chips, Festplattenlaufwerke, EPROMs, usw. Die computerlesbaren Medien umfassen keine Trägerwellen und elektronischen Signale, die drahtlos oder über verdrahtete Verbindungen laufen.

[0059] In dieser Patentbeschreibung soll der Begriff "Software" Firmware, die sich im Festwertarbeitsspeicher befindet, oder Anwendungen, die in einem magnetischen Speicher oder Flash-Speicher, beispielsweise einem Halbleiterlaufwerk, gespeichert sind, die in den Speicher zur Verarbeitung durch einen Prozessor gelesen werden können, umfassen. In einigen Implementierungen können auch mehrere Softwaretechnologien als Unterteile eines größeren Programms implementiert werden, während sie unterschiedliche Softwaretechnologien bleiben. In einigen Implementierungen können mehrere Softwaretechnologien auch als separate Programme implementiert werden. Schließlich liegt irgendeine Kombination von separaten Programmen, die zusammen eine

hier beschriebene Softwaretechnologie implementieren, innerhalb des Umfangs der vorliegenden Technologie. In einigen Implementierungen definieren die Softwareprogramme, wenn sie installiert sind, um auf einem oder mehreren elektronischen Systemen zu arbeiten, eine oder mehrere spezielle Maschinenimplementierungen, die die Operationen der Softwareprogramme ausführen und durchführen.

[0060] Ein Computerprogramm (auch als Programm, Software, Softwareanwendung, Skript oder Code bekannt) kann in irgendeiner Form von Programmiersprache geschrieben sein, einschließlich kompilierter oder interpretierter Sprachen, deklarativer oder Prozedursprachen, und es kann in irgendeiner Form eingesetzt werden, einschließlich als eigenständiges Programm oder als Modul, Komponente, Subroutine, Objekt oder andere Einheit, die zur Verwendung in einer Rechenumgebung geeignet ist. Ein Computerprogramm kann, muss jedoch nicht einer Datei in einem Dateisystem entsprechen. Ein Programm kann in einem Abschnitt einer Datei, der andere Programme oder Daten hält (z. B. ein oder mehrere Skripts, die in einem Auszeichnungssprachdokument gespeichert sind), in einer einzelnen Datei, die für das fragliche Programm zweckgebunden ist, oder in mehreren koordinierten Dateien (z. B. Dateien, die ein oder mehrere Module, Unterprogramme oder Abschnitte eines Codes speichern), gespeichert sein. Ein Computerprogramm kann so eingesetzt werden, dass es auf einem Computer oder auf mehreren Computern ausgeführt wird, die an einem Ort angeordnet oder über mehrere Orte verteilt sind und durch ein Kommunikationsnetz miteinander verbunden sind.

[0061] Diese vorstehend beschriebenen Funktionen können in einer digitalen elektronischen Schaltungsanordnung, in Computer-Software, Computer-Firmware oder Computer-Hardware implementiert werden. Die Techniken können unter Verwendung von einem oder mehreren Computerprogrammprodukten implementiert werden. Programmierbare Prozessoren und Computer können in mobilen Vorrichtungen enthalten oder als diese gepackt sein. Die Prozesse und Logikabläufe können durch einen oder mehrere programmierbare Prozessoren und durch eine oder mehrere programmierbare Logikschaltungsanordnungen durchgeführt werden. Universal- und Spezial-Rechenvorrichtungen und Speichervorrichtungen können durch Kommunikationsnetze miteinander verbunden sein.

[0062] Einige Implementierungen umfassen elektronische Komponenten, beispielsweise Mikroprozessoren, einen Speicher und Arbeitsspeicher, die Computerprogrammbefehle in einem maschinenlesbaren oder computerlesbaren Medium (alternativ als computerlesbare Speichermedien, maschinenlesbare Medien oder maschinenlesbare Speichermedien bezeichnet) speichern. Einige Beispiele von solchen computerlesbaren Medien umfassen RAM, ROM, Nur-Lese-Kompaktdisks (CD-ROM), aufzeichnungsfähige Kompaktdisks (CD-R), wiederbeschreibbare Kompaktdisks (CD-RW), digitale vielseitige Nur-Lese-Platten (z. B. DVD-ROM, Doppelschicht-DVD-ROM), eine Vielfalt von aufzeichnungsfähigen/wiederbeschreibbaren DVDs (z. B. DVD-RAM, DVD-RW, DVD+RW usw.), einen Flash-Speicher (z. B. SD-Karten, Mini-SD-Karten, Mikro-SD-Karten usw.), magnetische oder Halbleiterfestplattenlaufwerke, Nur-Lese- und aufzeichnungsfähige Blu-Ray®-Platten, ultradichte optische Platten, beliebige andere optische oder magnetische Medien und Disketten. Die computerlesbaren Medien können ein Computerprogramm speichern, das durch mindestens einen Prozessor ausführbar ist und Sätze von Befehlen zum Durchführen von verschiedenen Operationen umfasst. Beispiele von Computerprogrammen oder eines Computercodes umfassen einen Maschinencode, der beispielsweise durch einen Kompilierer erzeugt wird, und Dateien mit einem Code höherer Ebene, der durch einen Computer, eine elektronische Komponente oder einen Mikroprozessor unter Verwendung eines Interpretierers ausgeführt wird.

[0063] Obwohl sich die obige Erörterung hauptsächlich auf einen Mikroprozessor oder Mehrkernprozessoren bezieht, die eine Software ausführen, werden einige Implementierungen durch eine oder mehrere integrierte Schaltungen, beispielsweise anwendungsspezifische integrierte Schaltungen (ASICs) oder anwenderprogrammierbare Verknüpfungsfelder (FPGAs), durchgeführt. In einigen Implementierungen führen solche integrierten Schaltungen Befehle aus, die in der Schaltung selbst gespeichert sind.

[0064] Wie in dieser Patentbeschreibung und in irgendwelchen Ansprüchen dieser Anmeldung verwendet, beziehen sich die Begriffe "Computer, "Server", "Prozessor" und "Arbeitsspeicher" alle auf elektronische oder andere technologische Vorrichtungen. Diese Begriffe schließen Leute oder Gruppen von Leuten aus. Für die Zwecke der Patentbeschreibung bedeuten die Begriffe Anzeige oder Anzeigen das Anzeigen auf einer elektronischen Vorrichtung. Wie in dieser Patentbeschreibung und in beliebigen Ansprüchen dieser Anmeldung verwendet, sind die Begriffe "computerlesbares Medium" und "computerlesbare Medien" vollständig auf konkrete, physikalische Objekte eingeschränkt, die Informationen in einer Form speichern, die für einen Computer lesbar ist. Diese Begriffe schließen beliebige drahtlose Signale, verdrahtete heruntergeladene Signale und beliebige andere kurzlebige Signale aus.

[0065] Um eine Zusammenwirkung mit einem Benutzer bereitzustellen, können Implementierungen des in dieser Patentbeschreibung beschriebenen Gegenstandes auf einem Computer mit einer Anzeige-

vorrichtung, z. B. einem Kathodenstrahlröhrenmonitor (CRT-Monitor) oder Flüssigkristallanzeigemonitor (LCD-Monitor), zum Anzeigen von Informationen für den Benutzer und einer Tastatur und einer Zeigevorrichtung, z. B. einer Maus oder einer Rollkugel, durch die der Benutzer eine Eingabe in den Computer liefern kann, implementiert werden. Andere Arten von Vorrichtungen können ebenso verwendet werden, um für eine Zusammenwirkung mit einem Benutzer zu sorgen; eine Rückkopplung, die zum Benutzer geliefert wird, kann beispielsweise irgendeine Form von sensorischer Rückkopplung, z. B. visueller Rückkopplung, akustischer Rückkopplung oder taktiler Rückkopplung, sein; und eine Eingabe vom Benutzer kann in irgendeiner Form empfangen werden, einschließlich akustischer, Sprach- oder taktiler Eingabe. Außerdem kann ein Computer mit einem Benutzer durch Senden von Dokumenten zu und Empfangen von Dokumenten von einer Vorrichtung zusammenwirken, die durch den Benutzer verwendet wird; beispielsweise durch Senden von Webseiten zu einem Webbrowser auf einer Client-Vorrichtung des Benutzers in Reaktion auf Anforderungen, die vom Webbrowser empfangen werden.

[0066] Der in dieser Patentbeschreibung beschriebene Gegenstand kann in einem Rechensystem implementiert werden, das eine Backend-Komponente, z. B. als Datenserver, umfasst oder das eine Middleware-Komponente, z. B. einen Anwendungsserver, umfasst oder das eine Frontend-Komponente, z. B. einen Client-Computer mit einer graphischen Benutzerschnittstelle oder einen Webbrowser, durch den ein Benutzer mit einer Implementierung des in dieser Patentbeschreibung beschriebenen Gegenstandes zusammenwirken kann, oder irgendeine Kombination von einer oder mehreren von solchen Backend-, Middleware- oder Frontend-Komponenten umfasst. Die Komponenten des Systems können durch irgendeine Form oder irgendein Medium der digitalen Datenkommunikation, z. B. ein Kommunikationsnetz, miteinander verbunden sein. Beispiele von Kommunikationsnetzen umfassen ein lokales Netz (LAN) und ein weiträumiges Netz (WAN), ein Internetz (z. B. das Internet) und Peer-to-Peer-Netze (z. B. ad-hoc-Peer-to-Peer-Netze).

[0067] Das Rechensystem kann Clients und Server umfassen. Ein Client und ein Server sind im Allgemeinen voneinander entfernt und wirken typischerweise durch ein Kommunikationsnetz zusammen. Die Beziehung von Client und Server entsteht durch Computerprogramme, die auf den jeweiligen Computern laufen und eine Client-Server-Beziehung zueinander aufweisen. In einigen Aspekten des offenbarten Gegenstandes überträgt ein Server Daten (z. B. eine HTML-Seite) zu einer Client-Vorrichtung (z. B. für Zwecke der Anzeige der Daten zu und Empfangen einer Benutzereingabe von einem Benutzer, der mit der Client-Vorrichtung zusammenwirkt). Daten, die an der Client-Vorrichtung erzeugt werden (z. B. infolge der Benutzerzusammenwirkung), können von der Client-Vorrichtung am Server empfangen werden.

[0068] Selbstverständlich ist irgendeine spezielle Reihenfolge oder Hierarchie von Schritten in den offenbarten Prozessen eine Darstellung von Beispielmethoden. Auf der Basis der Konstruktionsvorlieben kann selbstverständlich die spezielle Reihenfolge oder Hierarchie von Schritten in den Prozessen umgeordnet werden oder alle dargestellten Schritte werden durchgeführt. Einige der Schritte können gleichzeitig durchgeführt werden. Unter bestimmten Umständen können beispielsweise Multitasking und parallele Verarbeitung vorteilhaft sein. Überdies sollte die Trennung von verschiedenen Systemkomponenten, die vorstehend dargestellt sind, nicht als eine solche Trennung erfordernd verstanden werden und es sollte verstanden werden, dass die beschriebenen Programmkomponenten und Systeme im Allgemeinen zusammen in einem einzigen Softwareprodukt integriert oder in mehrere Softwareprodukte gepackt werden können.

[0069] Verschiedene Modifikationen an diesen Aspekten sind leicht ersichtlich und die hier definierten allgemeinen Prinzipien können auf andere Aspekte angewendet werden. Folglich sollen die Ansprüche nicht auf die hier gezeigten Aspekte begrenzt sein, sondern ihnen soll der vollständige Schutzbereich gewährt werden, der mit den Sprachansprüchen konsistent ist, wobei eine Bezugnahme auf ein Element im Singular nicht bedeuten soll "eines und nur eines", wenn nicht speziell so angegeben, sondern vielmehr "eines oder mehrere". Wenn nicht speziell anders angegeben, bezieht sich der Begriff "einige" auf ein oder mehrere. Pronomen im Maskulin (z. B. sein) umfassen das feminine und neutrale Geschlecht (z. B. ihr und sein) und umgekehrt. Titel und Untertitel, falls vorhanden, werden nur der Zweckmäßigkeit halber verwendet und begrenzen die vorliegende Technologie nicht.

[0070] Ein Ausdruck, beispielsweise ein "Aspekt" impliziert nicht, dass der Aspekt für die vorliegende Technologie wesentlich ist oder dass der Aspekt für alle Konfigurationen der vorliegenden Technologie gilt. Eine Offenbarung in Bezug auf einen Aspekt kann für alle Konfigurationen oder eine oder mehrere Konfigurationen gelten. Ein Ausdruck, beispielsweise ein Aspekt, kann sich auf einen oder mehrere Aspekte beziehen und umgekehrt. Ein Ausdruck, beispielsweise eine "Konfiguration", impliziert nicht, dass eine solche Konfiguration für die vorliegende Technologie wesentlich ist oder dass eine solche Konfiguration für alle Konfigurationen der vorliegenden Technologie gilt. Eine Offenbarung in Bezug auf eine Konfiguration kann für alle Konfigurationen oder eine oder mehrere Konfigurationen gelten. Ein Ausdruck, beispielsweise eine Konfiguration, kann sich auf eine

oder mehrere Konfigurationen beziehen und umgekehrt.

Patentansprüche

1. Verfahren, das umfasst:
Empfangen einer Gastsoftware zum Betrieb in einer Umgebung eines nativen Client, wobei die Gastsoftware dazu konfiguriert ist, auf einer festgelegten Gasthardwarearchitektur und nicht innerhalb der Umgebung des nativen Client auszuführen; und
Vorsehen einer binären Übersetzung der Gastsoftware in einen mit dem nativen Client kompatiblen Maschinencode unter Verwendung einer Emulationssoftware, wobei der mit dem nativen Client kompatible Maschinencode innerhalb einer Sandbox für die Umgebung des nativen Client ausführt, wobei der mit dem nativen Client kompatible Maschinencode innerhalb einer Anwendung ausführbar ist, wobei das Vorsehen der binären Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode für die Ausführung innerhalb der Sandbox rechtzeitig während einer Laufzeit der emulierten Gastsoftware und ohne Portierung oder Umkompilierung der Gastsoftware stattfindet, und wobei das Vorsehen der binären Übersetzung mit der Ausführung der emulierten Gastsoftware überlappt.

2. Verfahren nach Anspruch 1, wobei die Sandbox für die Umgebung des nativen Client auf einen Satz von emulierten Gastregistern zugreift, die im Arbeitsspeicher oder in Registern gespeichert sind, die an einer Host-Maschine verfügbar sind, wobei Daten, die dem mit dem nativen Client kompatiblen Maschinencode zugeordnet sind, innerhalb der Sandbox für die Umgebung des nativen Client gespeichert werden, und wobei die emulierten Gastregister Registern der festgelegten Gasthardwarearchitektur entsprechen.

3. Verfahren nach Anspruch 1, wobei das Vorsehen der binären Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode umfasst:
Erzeugen eines Satzes von virtuellen Registern, die Register darstellen, die von der Gastsoftware verwendet werden, wenn sie auf der festgelegten Gasthardwarearchitektur ausgeführt wird, wobei die Adressen der virtuellen Register durch einen Basiszeiger (RBP) plus einen vorbestimmten Versatz bezeichnet werden, und wobei jedes virtuelle Register in dem Satz von virtuellen Registern aus dem Inneren der Sandbox über einen einzelnen Befehl zugänglich ist.

4. Verfahren nach Anspruch 1, wobei das Vorsehen der binären Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode umfasst:
Emulieren von Merkmalen der festgelegten Gasthardwarearchitektur und Anwendungsprogrammierschnittstellenaufrufen (API-Aufrufen) der Gastsoftware innerhalb des nativen Client.

5. Verfahren nach Anspruch 1, wobei das Vorsehen der binären Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode umfasst:
Ersetzen des Codes in der Gastsoftware durch einen Code, der mit der Umgebung des nativen Client kompatibel ist.

6. Verfahren nach Anspruch 1, wobei die Gastsoftware eine ARM-Software oder x86-Software umfasst.

7. Verfahren nach Anspruch 1, wobei der mit dem nativen Client kompatible Maschinencode sicher und portierbar ist und wobei die Gastsoftware nicht sicher oder nicht portierbar ist.

8. Verfahren nach Anspruch 1, wobei die Gastsoftware dazu konfiguriert ist, nur innerhalb eines speziellen Gastbetriebssystems auszuführen, und wobei die Umgebung des nativen Client dazu konfiguriert ist, innerhalb irgendeines von mehreren verschiedenen Betriebssystemen auszuführen.

9. Verfahren nach Anspruch 1, wobei die Anwendung ein Browser ist.

10. Nicht transitorisches computerlesbares Medium mit Befehlen, die, wenn sie durch einen oder mehrere Computer ausgeführt werden, bewirken, dass der eine oder die mehreren Computer ein Verfahren implementieren, wobei das Verfahren umfasst:
Empfangen einer Gastsoftware zum Betrieb in einer Umgebung eines nativen Client, wobei die Gastsoftware dazu konfiguriert ist, auf einer festgelegten Gasthardwarearchitektur und nicht innerhalb der Umgebung des nativen Client auszuführen; und
Vorsehen einer binären Übersetzung der Gastsoftware in einen mit dem nativen Client kompatiblen Maschinencode unter Verwendung einer Emulationssoftware, wobei der mit dem nativen Client kompatible Maschinencode innerhalb einer Sandbox für die Umgebung des nativen Client ausführt, wobei der mit dem nativen Client kompatible Maschinencode innerhalb einer Anwendung ausführbar ist, wobei das Vorsehen der binären Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode für die Ausführung innerhalb der Sandbox rechtzeitig während einer Laufzeit der emulierten Gastsoftware und ohne Portierung oder Umkompilierung der Gastsoftware stattfindet, wobei das Vorsehen der binären Übersetzung mit der Ausführung der emulierten Gastsoftware überlappt, wobei das Vorsehen der binären Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode umfasst:
Erzeugen eines Satzes von virtuellen Registern, die Register darstellen, die von der Gastsoftware

verwendet werden, wenn sie auf der festgelegten Gasthardwarearchitektur ausgeführt wird, wobei die Adressen der virtuellen Register durch einen Basiszeiger (RBP) plus einen vorbestimmten Versatz bezeichnet werden, und wobei jedes virtuelle Register in dem Satz von virtuellen Registern aus dem Inneren der Sandbox über einen einzelnen Befehl zugänglich ist.

11. Nicht transitorisches computerlesbares Medium nach Anspruch 9, wobei die Sandbox für die Umgebung des nativen Client auf einen Satz von emulierten Gastregistern zugreift, die im Arbeitsspeicher oder in Registern gespeichert sind, die an einer Host-Maschine verfügbar sind, wobei Daten, die dem mit dem nativen Client kompatiblen Maschinencode zugeordnet sind, innerhalb der Sandbox für die Umgebung des nativen Client gespeichert werden, und wobei die emulierten Gastregister Registern der festgelegten Gasthardwarearchitektur entsprechen.

12. Nicht transitorisches computerlesbares Medium nach Anspruch 9, wobei das Vorsehen der binären Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode umfasst:
Emulieren von Merkmalen der festgelegten Gasthardwarearchitektur und Anwendungsprogrammierschnittstellenaufrufen (API-Aufrufen) der Gastsoftware innerhalb des nativen Client.

13. Nicht transitorisches computerlesbares Medium nach Anspruch 9, wobei das Vorsehen der binären Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode umfasst:
Ersetzen des Codes in der Gastsoftware durch einen mit der Umgebung des nativen Client kompatiblen Code.

14. Nicht transitorisches computerlesbares Medium nach Anspruch 9, wobei die Gastsoftware eine ARM-Software oder x86-Software umfasst.

15. Nicht transitorisches computerlesbares Medium nach Anspruch 9, wobei der mit dem nativen Client kompatible Maschinencode sicher und portierbar ist und wobei die Gastsoftware nicht sicher oder nicht portierbar ist.

16. System, das umfasst:
einen oder mehrere Prozessoren; und
einen Arbeitsspeicher mit Befehlen, die, wenn sie durch den einen oder die mehreren Prozessoren ausgeführt werden, bewirken, dass der eine oder die mehreren Prozessoren ein Verfahren implementieren, wobei das Verfahren umfasst:
Empfangen einer Gastsoftware zum Betrieb in einer Umgebung eines nativen Client, wobei die Gastsoftware dazu konfiguriert ist, auf einer festgelegten Gasthardwarearchitektur und nicht innerhalb der Umgebung des nativen Client auszuführen; und
Vorsehen einer binären Übersetzung der Gastsoftware in einen mit dem nativen Client kompatiblen Maschinencode unter Verwendung einer Emulationssoftware, wobei der mit dem nativen Client kompatible Maschinencode innerhalb einer Sandbox für die Umgebung des nativen Client ausführt, wobei der mit dem nativen Client kompatible Maschinencode innerhalb einer Anwendung ausführbar ist, wobei das Vorsehen der binären Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode für die Ausführung innerhalb der Sandbox rechtzeitig während einer Laufzeit der emulierten Gastsoftware und ohne Portierung oder Umkompilierung der Gastsoftware stattfindet, wobei das Vorsehen der binären Übersetzung mit der Ausführung der emulierten Gastsoftware überlappt, wobei die Sandbox für die Umgebung des nativen Client auf einen Satz von emulierten Gastregistern zugreift, die im Arbeitsspeicher oder in Registern gespeichert sind, die an einer Host-Maschine verfügbar sind, wobei Daten, die dem mit dem nativen Client kompatiblen Maschinencode zugeordnet sind, innerhalb der Sandbox für die Umgebung des nativen Client gespeichert werden, und wobei die emulierten Gastregister Registern der festgelegten Gasthardwarearchitektur entsprechen.

17. System nach Anspruch 16, wobei das Vorsehen der binären Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode umfasst:
Erzeugen eines Satzes von virtuellen Registern, die Register darstellen, die von der Gastsoftware verwendet werden, wenn sie auf der festgelegten Gasthardwarearchitektur ausgeführt wird, wobei die Adressen der virtuellen Register durch einen Basiszeiger (RBP) plus einen vorbestimmten Versatz bezeichnet werden, und wobei jedes virtuelle Register in dem Satz von virtuellen Registern aus dem Inneren der Sandbox über einen einzelnen Befehl zugänglich ist.

18. System nach Anspruch 16, wobei das Vorsehen der binären Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode umfasst:
Emulieren von Merkmalen der festgelegten Gasthardwarearchitektur und Anwendungsprogrammierschnittstellenaufrufen (API-Aufrufen) der Gastsoftware innerhalb des nativen Client.

19. System nach Anspruch 16, wobei das Vorsehen der binären Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode umfasst:
Ersetzen des Codes in der Gastsoftware durch einen Code, der mit der Umgebung des nativen Client kompatibel ist.

20. System nach Anspruch 16, wobei die Gastsoftware eine ARM-Software oder x86-Software umfasst.

Es folgen 4 Seiten Zeichnungen

Anhängende Zeichnungen

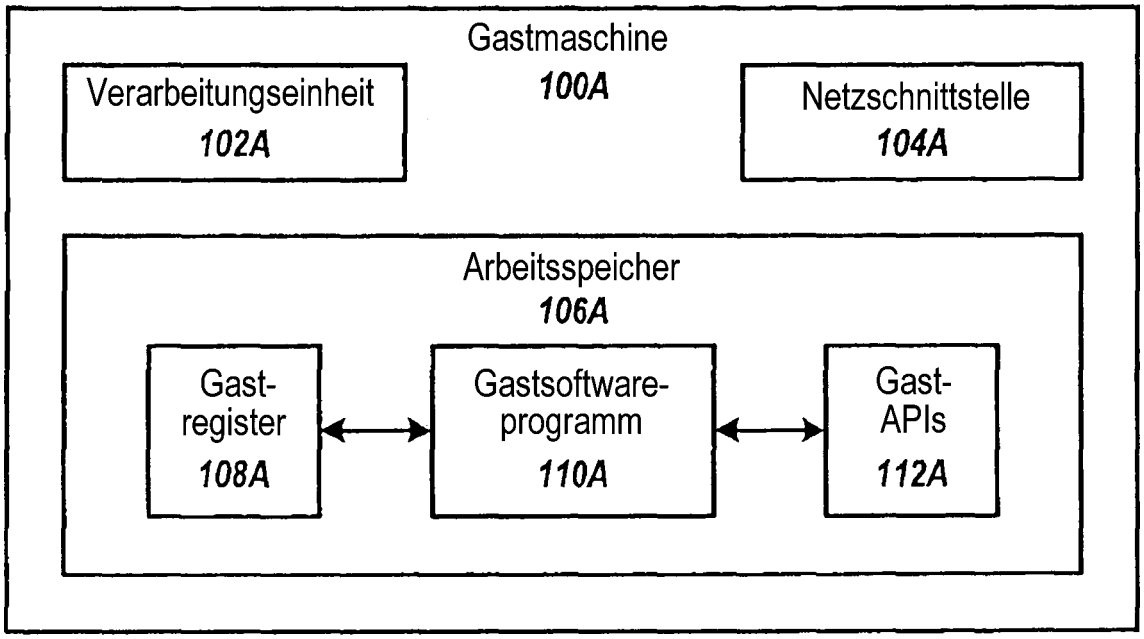


FIG. 1A

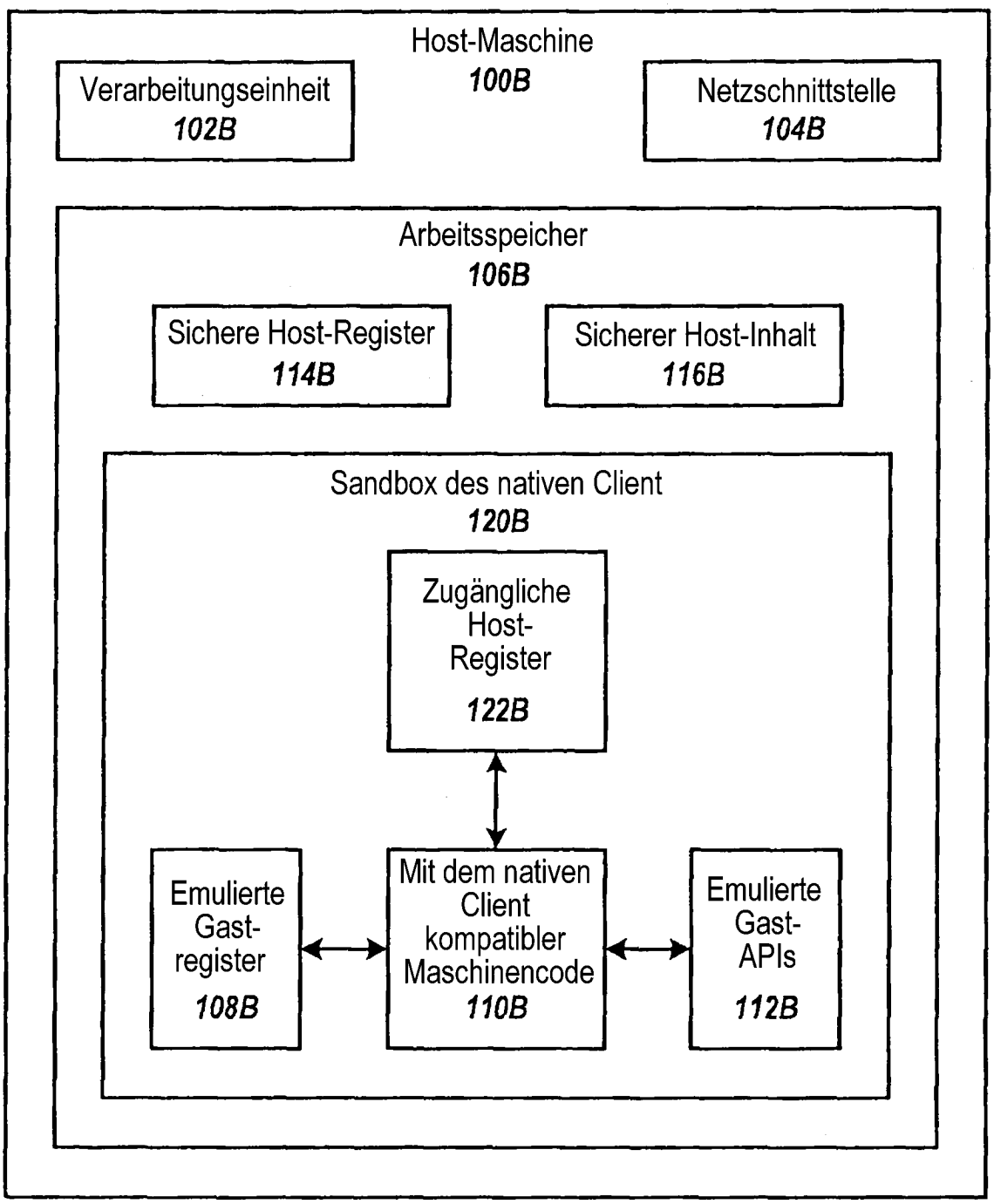

Host-Maschine
100B
Verarbeitungseinheit
102B
Netzschnittstelle
104B
Arbeitsspeicher
106B
Sichere Host-Register
114B
Sicherer Host-Inhalt
116B
Sandbox des nativen Client
120B
Zugängliche Host-Register
122B
Emulierte Gast-register
108B
Mit dem nativen Client kompatibler Maschinencode
110B
Emulierte Gast-APIs
112B

FIG. 1B

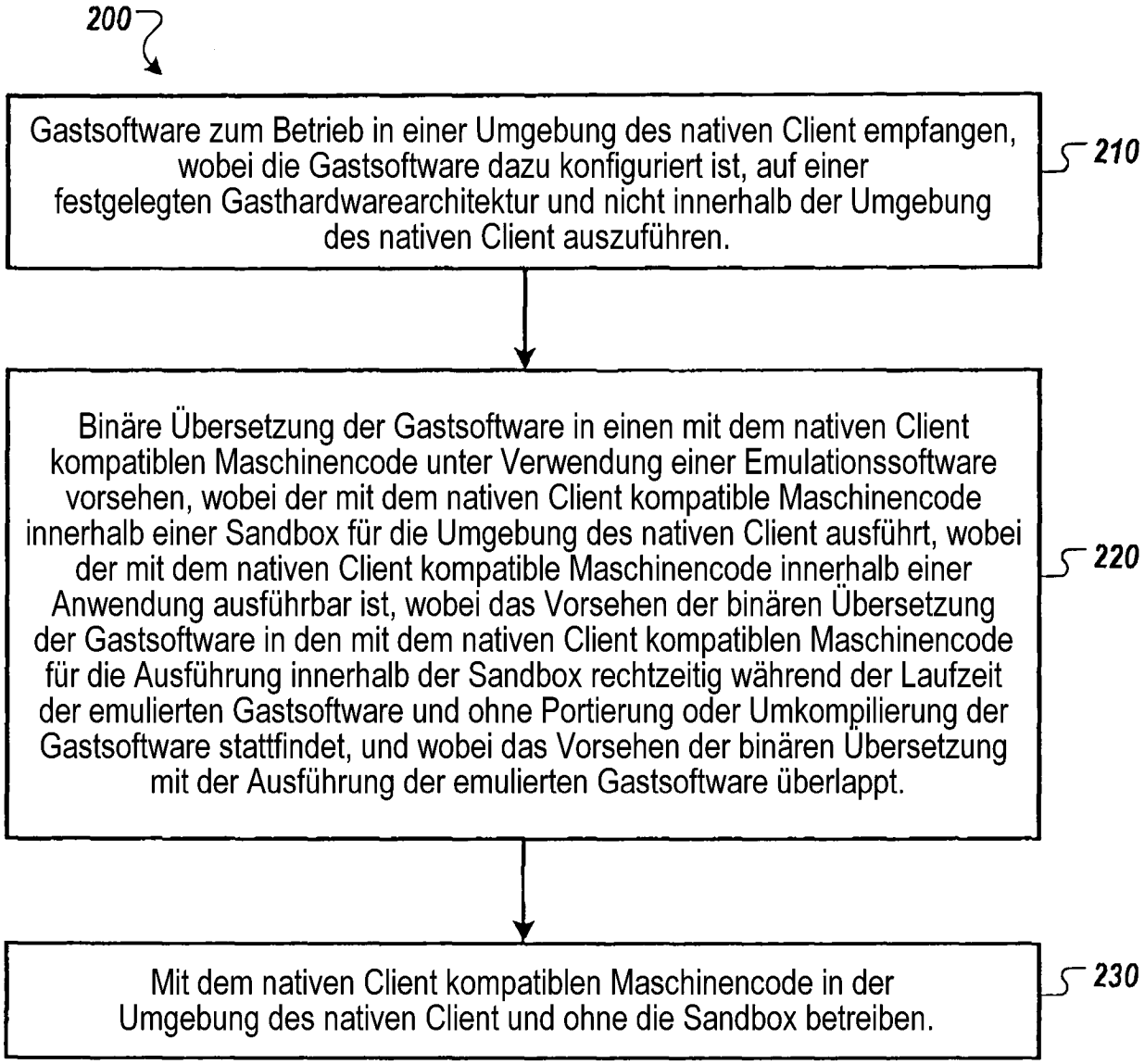

200
Gastsoftware zum Betrieb in einer Umgebung des nativen Client empfangen, wobei die Gastsoftware dazu konfiguriert ist, auf einer festgelegten Gasthardwarearchitektur und nicht innerhalb der Umgebung des nativen Client auszuführen.
210
Binäre Übersetzung der Gastsoftware in einen mit dem nativen Client kompatiblen Maschinencode unter Verwendung einer Emulationssoftware vorsehen, wobei der mit dem nativen Client kompatible Maschinencode innerhalb einer Sandbox für die Umgebung des nativen Client ausführt, wobei der mit dem nativen Client kompatible Maschinencode innerhalb einer Anwendung ausführbar ist, wobei das Vorsehen der binären Übersetzung der Gastsoftware in den mit dem nativen Client kompatiblen Maschinencode für die Ausführung innerhalb der Sandbox rechtzeitig während der Laufzeit der emulierten Gastsoftware und ohne Portierung oder Umkompilierung der Gastsoftware stattfindet, und wobei das Vorsehen der binären Übersetzung mit der Ausführung der emulierten Gastsoftware überlappt.
220
Mit dem nativen Client kompatiblen Maschinencode in der Umgebung des nativen Client und ohne die Sandbox betreiben.
230

FIG. 2

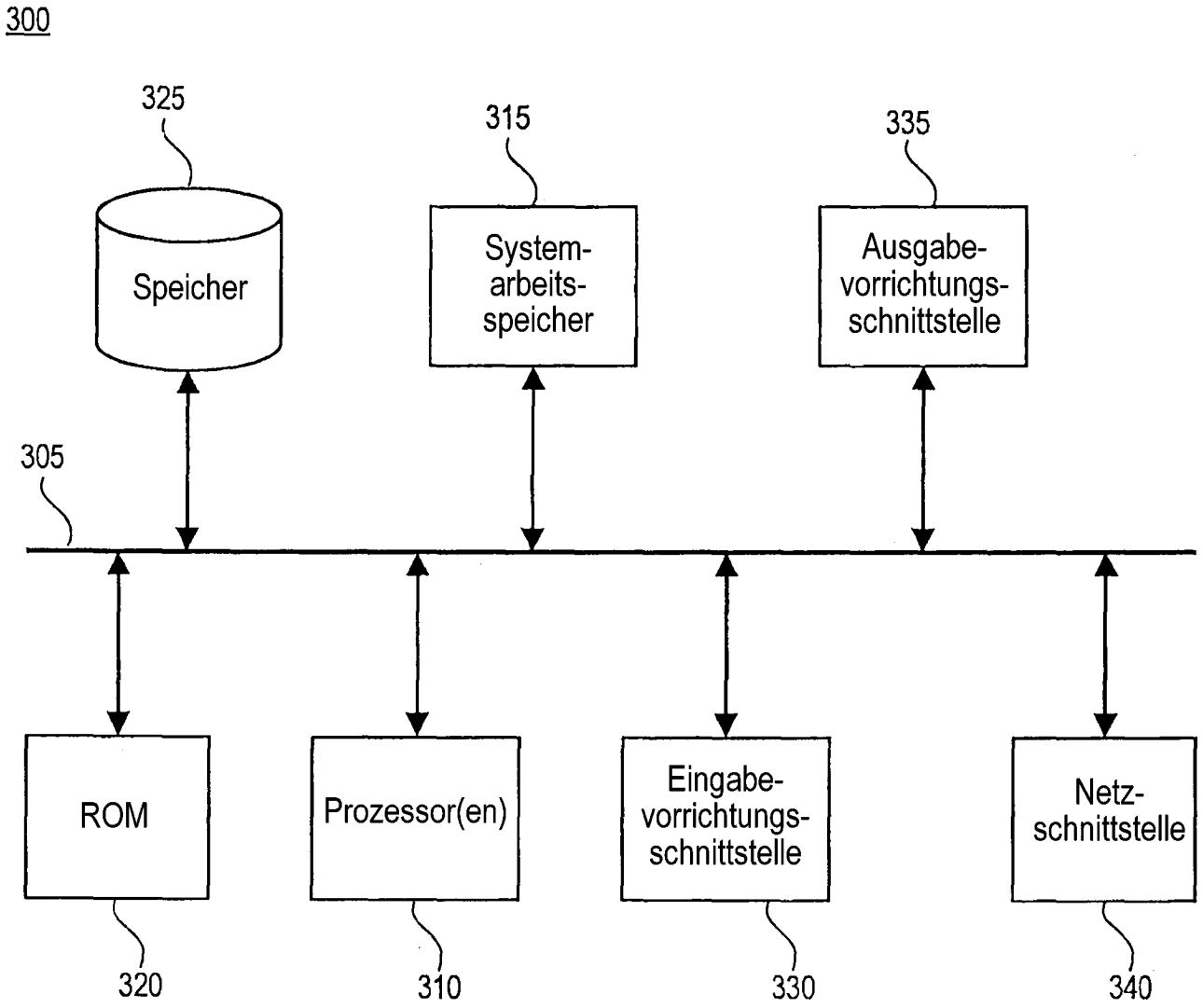

300
325
Speicher
315
System-
arbeits-
speicher
335
Ausgabe-
vorrichtungs-
schnittstelle
305
ROM
Prozessor(en)
Eingabe-
vorrichtungs-
schnittstelle
Netz-
schnittstelle
320
310
330
340

FIG. 3