(12) NACH DEM VERTRAG ÜBER DIE INTERNATIONALE ZUSAMMENARBEIT AUF DEM GEBIET DES PATENTWESENS (PCT) VERÖFFENTLICHTE INTERNATIONALE ANMELDUNG

(19) Weltorganisation für geistiges Eigentum
Internationales Büro

(43) Internationales Veröffentlichungsdatum
15. November 2012 (15.11.2012)

WIPO | PCT

(10) Internationale Veröffentlichungsnummer
WO 2012/152326 A1

(51) Internationale Patentklassifikation:
G06F 9/455 (2006.01) *G06F 9/50* (2006.01)

(21) Internationales Aktenzeichen: PCT/EP2011/057631

(22) Internationales Anmeldedatum:
11. Mai 2011 (11.05.2011)

(25) Einreichungssprache: Deutsch

(26) Veröffentlichungssprache: Deutsch

(71) Anmelder *(für alle Bestimmungsstaaten mit Ausnahme von US)*: **SIEMENS AKTIENGESELLSCHAFT** [DE/DE]; Wittelsbacherplatz 2, 80333 München (DE).

(72) Erfinder; und

(75) Erfinder/Anmelder *(nur für US)*: **GRAF, Rene** [DE/DE]; Gibitzenhofstr. 73, 90443 Nürnberg (DE). **HARTMANN, Wolfgang** [DE/DE]; Röthenäcker Str. 11, 91086 Aurachtal (DE).

(74) Gemeinsamer Vertreter: **SIEMENS AKTIENGESELLSCHAFT**; Postfach 22 16 34, 80506 München (DE).

(81) Bestimmungsstaaten *(soweit nicht anders angegeben, für jede verfügbare nationale Schutzrechtsart)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

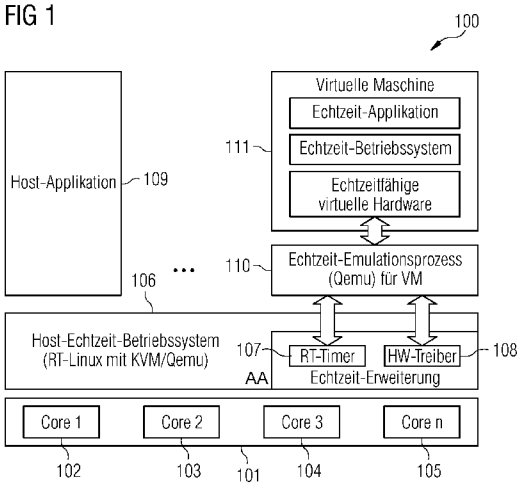
(84) Bestimmungsstaaten *(soweit nicht anders angegeben, für jede verfügbare regionale Schutzrechtsart)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), eurasisches (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), europäisches (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Fortsetzung auf der nächsten Seite]

(54) **Title**: SYSTEM AND METHOD FOR PROVIDING AND RUNNING ONE OR MORE REAL-TIME VIRTUAL MACHINES ON A MULTICORE COMPUTER

(54) **Bezeichnung** : SYSTEM UND VERFAHREN ZUM BEREITSTELLEN UND BETREIBEN VON EINER ODER MEHREREN ECHZEITFÄHIGEN VIRTUELLEN MASCHINEN AUF EINEM MEHRKERN HOCH-RECHNER

FIG 1



106... Host real-time operating system (RT-Linux with KVM/Qemu)
107... RT timer
108... HW driver
109... Host application
110... Real-time emulation process (Qemu) for VM
111... Virtual machine
Real-time application
Real-time operating system
Real-time capable virtual hardware
AA... Real-time expansion

(57) **Abstract**: According to one examplary aspect of the invention, a host system for providing a real-time capable virtual machine is provided, wherein the host system comprises a host that has a plurality of physical cores, wherein a real-time operating system is installed on the host.

(57) **Zusammenfassung**: Gemäß einem exemplarischen Aspekt wird ein Host-System zum Bereitstellen einer echtzeitfähigen virtuellen Maschine geschaffen, wobei das Host-System einen Host aufweist, welcher eine Mehrzahl von physikalischen Kernen aufweist, wobei auf dem Host ein Echtzeit-Betriebssystem installiert ist.

WO 2012/152326 A1

Veröffentlicht:

— *mit internationalem Recherchenbericht (Artikel 21 Absatz 3)*

Beschreibung

SYSTEM UND VERFAHREN ZUM BEREITSTELLEN UND BETREIBEN VON EINER ODER MEHREREN ECHZEITFÄHIGEN VIRTUELLEN MASCHINEN AUF EINEM MEHRKERN HOCH-RECHNER

Die Erfindung betrifft ein Host-System, insbesondere ein Host-System für eine virtuelle Maschine. Ferner betrifft die Erfindung ein Verfahren zum Betreiben eines Host-Systems für eine virtuelle Maschine. Darüber hinaus betrifft die Erfindung ein Programmelement und ein computerlesbares Medium.

Stand der Technik

Im Stand der Technik ist die sogenannte Virtualisierungstechnik (VT) bekannt, welche im Wesentlichen der Aufteilung einer Hardware in kleinere Teile dient. Populärstes Beispiel sind Mehrkern-Prozessoren oder Multi-Core Prozessoren, die mittels der VT in virtuelle Prozessoren mit nur einem oder wenigen Kernen zerteilt werden. Ebenso kann der Arbeitsspeicher, der nur einmal physikalisch vorhanden ist, entsprechend geteilt werden, so dass jeder virtuelle Prozessor einen festen Anteil an dem Arbeitsspeicher zugeordnet bekommt. Diese Aufteilung kann statisch sein, aber auch dynamisch durchgeführt werden, d.h. eine Veränderung der Zuweisung während der Laufzeit kann zugelassen sein. Ähnliches gilt für die im gesamten Rechnersystem vorhandene Peripherie wie Festplatten, Netzwerkanschlüsse und ähnliches.

Das System oder Konglomerat aus virtuellen Prozessoren, anteiligen Speicher und zugeordneter Peripherie wird als virtuelle Maschine (VM) bezeichnet. Ein physikalischer Rechner mit entsprechenden Ressourcen kann somit quasi eine beliebige Anzahl an VMs betreiben oder zur Verfügung stellen. Der physikalische Rechner selber wird dabei als Host-Rechner bezeichnet. Während des Betreibens der virtuellen

Maschinen muss der Host-Rechner oder kurz Host selbst aktiv werden, wenn nämlich eine Hardware in einer VM kein physikalisches Gegenstück hat, sondern diese nur emuliert wird.

Speziell bei Peripherie-Geräten, die physikalisch nur einmal vorhanden sind, muss der Host-Rechner diese ggf. mehrfach emulieren, um jeder VM ein Exemplar zur Verfügung zu stellen. Ein wichtiges Beispiel hierfür ist der Systemtimer, der nicht im Prozessor, sondern in der Regel im Chipsatz des Multi-Core-Systems verbaut oder implementiert ist und damit nur einmal vorhanden ist. Der Systemtimer bietet zwar mehrere einzelne Timer, die von den virtuellen Maschinen individuell genutzt werden können, aber der Zugriff auf die Hardware, d.h. den physikalischen, Systemtimer, ist nur für eine Instanz, nämlich dem Host-Rechner oder dem Host-System, gestattet. Die jeweiligen Timer in den virtuellen Maschinen bzw. der virtuellen Hardware müssen somit emuliert werden und arbeitet daher prinzipbedingt nicht mehr deterministisch, d.h. es kann nicht mehr zu allen Zeiten sichergestellt werden, dass eine maximale Zugriffszeit garantiert wird.

Eine andere Variante eines Timers bei einer x86-Architektur ist der Timer des "Local APIC", der individuell den einzelnen Kernen zugewiesen ist. Dennoch kann auch dieser Timer nur über eine Emulation angesprochen werden, was zu deutlich messbaren Sprüngen im Ablauf der Uhr führt, die eigentlich äquidistant ablaufenden sollte. Somit ist auch mittels dieses Timers keine deterministische Zeit innerhalb einer virtuellen Maschine sicherzustellen.

Beim Einsatz der Virtualisierung im Server-Umfeld ist dies unkritisch, da hier nur Zeitanforderungen im oberen Millisekunden oder gar Sekundenbereich zu erfüllen sind. Des Weiteren spielen erhöhte Latenzen im Server-Umfeld keine Rolle, weil die Applikationen typischer Weise keine beson-

deren Zeitanforderungen stellen. Zudem kann die Zeit mittels Uhrzeit-Synchronisation in regelmäßigen Abständen korrigiert werden, die aber viel zu grobgranular für eine Echtzeitanwendung sind.

Aus dem Stand der Technik ist beispielsweise bekannt, die virtuelle Maschine mittels einer Kombination des Kernel-Treibers und eines Emulationsprogrammes zu realisieren. Beispielsweise ist bei der Verwendung von Linux als Host-Betriebssystem bekannt, eine sogenannte KVM (Kernel-Virtual-Machine) als Kernel-Treiber und das Programms Qemu als Emulator zu verwenden. Bei einer solchen Kombination sorgt der Kernel-Treiber für die Konfiguration und Nutzung der Virtualisierungstechnologie des Prozessors und des Chipsatzes, während der Emulator für die Emulation der notwendigen Peripherie sorgt. Damit ist der Emulator ein Prozess des Host-Betriebssystem, der dessen Scheduling-Verhalten unterliegt und somit nicht deterministisch ist.

Fig. 3 zeigt ein Standard Host-System 300 mit virtuellen Maschinen, insbesondere ein Host-System, welches Linux als Host-Betriebssystem verwendet. Es mag zwar prinzipiell möglich sein, ein Echtzeit-Betriebssystem in eine virtuelle Maschine (VM) zu installieren, aber dieses wird nur so gut laufen können, wie es die Qualität des emulierten Timers und der emulierten Peripherie zulässt.

Das Host-System 300 weist physikalische Host-Hardware 301 auf, welche schematisch als Kerne oder Cores 302, 303, 304 und 305 dargestellt ist. Auf der Host-Hardware läuft ein Host-Betriebssystem 306, welches beispielsweise ein Linux-Betriebssystem ist, welches Hardwaretreiber 307 und 308 zur Verfügung stellt. Ferner laufen auf dem Host Applikationen, welche schematisch mit dem Block 309 in Fig. 3 dargestellt sind.

Aufsetzend auf den Hardware-Treibern 307 und 308 werden Emulationsprozesse (Qemu) 310 bzw. 311 aufgesetzt, welche dem Emulieren virtueller Maschinen 312 und 313 dienen. Auf den virtuellen Maschinen 312 und 313 kann dann nun wieder ein Betriebssystem und Applikationen laufen, welche auch virtuelle Hardware nutzen können.

Somit zeigt Fig. 3 eine Konfiguration in einem Standard-Host-System mit mehreren virtuellen Maschinen. Jede Maschine enthält in diesem Beispiel einen Core oder Kern exklusiv, muss aber für bestimmte Peripherie inkl. Systemtimer auf die Emulation zurückgreifen, die als normaler Prozess des Host-Betriebssystems neben anderen Applikationen läuft. Qemu startet für jeden Core, den die virtuelle Maschine haben soll, einen Thread (vCPU-Thread) sowie ggf. weitere Threads für die Emulation verschiedener Peripherien des virtuellen Systems.

Die Zuordnung der Qemu-Threads zu den physikalischen Cores des Host-Systems kann nur manuell mit Hilfe von Management-Tools (z.B. taskset, virsh) erfolgen. Um im obigen Beispiel eine VM einem Core zuzuordnen, wird Qemu mit dem taskset-Kommando gestartet, wobei der entsprechende Core als CPU-Affinity Maske angegeben wird. Damit weiß der Betriebssystem-Scheduler, auf welche physikalischen Cores er Threads verteilen darf.

Soll die virtuelle Maschine zwei oder mehr Cores enthalten, können dem Programm über die Management-Tools zwar auch entsprechend viele physikalische Cores zugeteilt werden, aber die detaillierte Zuordnung der Threads auf die Cores wird vom Linux-Scheduler vorgenommen, so dass nicht zwingend eine direkte Zuordnung der vCPU-Threads auf die physikalischen Cores entstehen muss. Dies hat aber zur Konsequenz, dass die Rechenleistung eines virtuellen Cores nicht

vorhersagbar ist und somit kein Echtzeit-Betriebssystem deterministisch in der virtuellen Maschine laufen kann.

Eine solche Implementierung auf einem Host-System führt somit dazu, dass die virtuelle Maschine keine harten Echtzeitanforderungen erfüllt.

Zusammenfassung der Erfindung

Es ist Aufgabe der Erfindung, ein Host-System und ein Verfahren zum Betreiben eines Host-Systems zu schaffen, welches hinsichtlich Echtzeitanforderungen eine verbesserte Performance erzielt.

Dieser Bedarf wird durch ein Host-System, ein Verfahren zum Betreiben eines Host-Systems, ein Computerprogrammelement und durch ein computerlesbares Medium gemäß den unabhängigen Patentansprüchen erfüllt. Weitere Ausgestaltungen sind in den abhängigen Ansprüchen angegeben.

Gemäß einem exemplarischen Aspekt wird ein Host-System zum Bereitstellen einer echtzeitfähigen virtuellen Maschine geschaffen, wobei das Host-System einen Host aufweist, welcher eine Mehrzahl von physikalischen Kernen aufweist, wobei auf dem Host ein Echtzeit-Betriebssystem installiert ist.

Insbesondere wird mittels des Vorsehens eines Echtzeit-Betriebssystem auf dem Host ermöglicht, dass virtuellen Maschinen, die auf dem Host eingerichtet werden, ein Echtzeit-Systemtimer zur Verfügung gestellt werden kann. Somit ist es erreichbar, dass in der virtuellen Maschine ein Echtzeit-System zufriedenstellend laufen kann.

Mit einer solchen Konfiguration, bei dem ein Echtzeit-Betriebssystem anstelle eines General-Purpose-

Betriebssystems zum Einsatz kommt, ist es möglich, dass ein Emulationsprozess als Echtzeit-Prozess gefahren wird, so dass der Emulationsprozess selbst deterministisch von dem Host-Betriebssystem aufgerufen wird, und diesen Determinismus in die virtuelle Maschine weiterleiten kann.

Gemäß einem anderen exemplarischen Aspekt wird ein Verfahren zum Betreiben eines Host-Systems für echtzeitfähige virtuelle Maschinen geschaffen, wobei das Host-System eine Mehrzahl von physikalischen Kernen aufweist, wobei das Verfahren ein Betreiben eines Echtzeit-Betriebssystems auf dem Host-System aufweist.

Gemäß einem anderen exemplarischen Aspekt der Erfindung wird ein Programmelement geschaffen, welches derart eingerichtet ist, dass es, wenn es auf einem Prozessor ausgeführt wird, ein Verfahren gemäß einem exemplarischen Aspekt der Erfindung steuert.

Gemäß einem anderen exemplarischen Aspekt der Erfindung wird ein computerlesbares Medium geschaffen, auf welchem ein Computerprogramm gespeichert ist, wobei das Computerprogramm derart eingerichtet ist, dass es, wenn es auf einem Prozessor ausgeführt wird, ein Verfahren gemäß einem exemplarischen Aspekt der Erfindung steuert.

Eine Grundidee eines beispielhaften Aspekts ist es, dass ein Host-System geschaffen wird, welches die Realisierung von echtzeitfähigen virtuellen Maschinen ermöglicht. Hierzu weist das Host-System eine Mehrzahl von physikalischen Kernen oder physikalischen Cores auf. Auf den Host-System bzw. dem dazugehörigen Host ist hierbei ein Echtzeit-Betriebssystem implementiert. Dies mag im Gegensatz zu dem bekannten Host-Systemen stehen, bei denen auf dem Host ein sogenanntes General-Purpose-Betriebssystem (GPOS) installiert ist. Mittels des Vorsehens eines Echtzeit-

Betriebssystem (RTOS) mag es möglich sein, auf dem Host-System eine Mehrzahl von virtuellen Maschinen zu starten, welche jeweils echtzeitfähig sind, d.h. ein deterministisches Zeitverhalten aufweisen.

Neben den echtzeitfähigen VMs sind optional auch weitere, nicht-echtzeitfähige VMs auf dem Host-System installiert, sofern dieses über eine ausreichende Anzahl von Rechencores verfügt. Damit wird eine Simulation verschiedener Geräte in einem System möglich, wobei die virtuellen Geräte sowohl untereinander als auch nach außen kommunizieren können. Dieses Szenario wird auch virtuelle Inbetriebnahme genannt.

Nachfolgend werden exemplarische Ausführungsbeispiele des Host-System beschrieben. Jedoch gelten die entsprechenden Ausgestaltungen und Merkmale auch für das Verfahren zum Betreiben eines Host-Systems, das Computerprogrammelement und das computerlesbare Medium.

Gemäß einem anderen Ausführungsbeispiel ist das Host-System derart eingerichtet, dass ein Emulationsprozess für eine echtzeitfähige virtuelle Maschine als Echtzeit-Prozess implementiert ist. Insbesondere kann der Host derart eingerichtet sein, dass der Emulationsprozess auf dem Host als Echtzeit-Prozess ausgeführt wird.

Im Falle dass der Emulationsprozess als Echtzeit-Prozess implementiert ist, mag es möglich sein, dass er selber deterministisch von dem Host-Betriebssystem aufgerufen wird. Hierdurch wird es möglich, dass der Determinismus in die virtuelle Maschine, welche mittels des Emulationsprozesses emuliert wird, weiterleitbar ist.

Gemäß einem anderen beispielhaften Ausführungsbeispiel des Host-Systems ist das Host-System derart eingerichtet, dass

eine Zuordnung eines virtuellen Kerns zu einem der Mehrzahl von physikalischen Kernen statisch ist.

Insbesondere kann jedem virtuellen Kern und/oder jeder virtuellen Maschine ein oder mehrere physikalische Kerne statisch zugewiesen sein. Alternativ mag die Zuordnung auch dynamisch sein oder es mag eine hybride Zuordnung verwendet werden.

Sowohl im statischen, dynamischen als auch in der hybriden Zuordnung mag es bevorzugt sein, dass jedem physikalischen Kern nur ein virtueller Kern oder eine virtuelle Maschine zugeordnet ist. D.h. kein physikalischer Kern ist mehreren virtuellen Maschinen zugeordnet oder anders ausgedrückt, jeder physikalische Kern ist exklusiv einer virtuellen Maschine zugeordnet. Eine solche Zuordnung kann insbesondere eine eins-zu-eins Beziehung sein, d.h. jedem virtuellen Kern ist genau ein physikalischer Kern zugeordnet und jeder physikalische Kern ist genau einem virtuellen Kern oder einer virtuellen Maschine zugeordnet. Jedoch mag auch einer virtuellen Maschine mehr als ein physikalischer Kern zugeordnet sein. Bevorzugt sollte jedoch kein physikalischer Kern mehreren virtuellen Maschinen zugeordnet sein. Auf diese Weise mag verhinderbar sein, dass ein dynamischer Wechsel von virtuellen CPU-Threads von einem physikalischen Kern auf einen anderen physikalischen Kern stattfinden muss.

Alternativ ist das Host-System auch derart eingerichtet, dass neben einer oder einer Mehrzahl von echtzeitfähigen virtuellen Maschinen auch eine oder eine zweite Mehrzahl an nicht echtzeitfähigen virtuellen Maschinen gestartet werden kann. Hierbei sei es dann auch möglich, dass sich mehrere nicht echtzeitfähige virtuelle Maschinen einen gemeinsamen physikalischen Kern teilen, während jedoch jeder echtzeitfähigen virtuellen Maschine zumindest ein physikalischer

Kern exklusiv zugewiesen ist. Beispielsweise können auf einem Host-System mit vier physikalischen Kernen somit zwei echtzeitfähige virtuelle Maschinen, welchen jeweils ein physikalischer Kern exklusiv zugewiesen wird, und eine beliebige Anzahl von nicht echtzeitfähigen Systemen, welche sich dann die zwei verbleibenden physikalischen Kerne teilen, gestartet werden.

Gemäß einem anderen Ausführungsbeispiel ist das Host-System eingerichtet, eine Mehrzahl von virtuellen Maschinen zu emulieren, wobei die Mehrzahl von physikalischen Kernen größer oder gleich der Mehrzahl von virtuellen Maschinen ist. Anders ausgedrückt mag die Anzahl von physikalischen Kernen größer als die Anzahl von virtuellen Maschinen sein.

Insbesondere bedeutet dies, dass einem oder jedem Emulationsprogramm für virtuelle Maschinen mehrere physikalische Kerne zuweisbar sind. Auf diese Art ist es möglich, dass eine Peripherie wie in einem realen oder physikalischen Rechner autark agiert, bevor die Peripherie einem Kern per Interrupt auffordert, die zur Peripherie gehörende Software zu starten. Hierdurch mag es möglich sein, exakt genaue Timer in der Emulation zu realisieren, so dass auch ein Echtzeit-Betriebssystem in der virtuellen Maschine deterministisch laufen kann und damit implizit auch alle darauf aufgesetzten Applikationen.

Gemäß einem anderen Ausführungsbeispiel ist das Host-System derart eingerichtet, dass ein Treiber für eine virtuelle Maschine als Echtzeit-Treiber auf dem Host implementiert ist.

Insbesondere mag der Treiber ein Hardware-Treiber sein, welcher für eine Hardware, die von der virtuellen Maschine benötigt wird, verwendbar ist. Hierdurch ist es möglich, dass zusätzliche Peripherie, die Interrupts produzieren

mag, deterministisch einbindbar ist, indem der Treiber im Host-System, d.h. mittels des Host-Betriebssystems, ebenfalls als Echtzeit-Treiber behandelt wird, so dass es möglich ist, dass der korrespondierende Emulationsprozess entsprechend schnell benachrichtigt werden kann. Dieser wiederum löst in der virtuellen Hardware den passenden Interrupt aus, so dass das ganze System deterministisch arbeitet.

Gemäß einem anderen Ausführungsbeispiel ist das Host-System derart eingerichtet, dass mittels ihm eine Mehrzahl von virtuellen Maschinen bereitstellbar ist.

Gemäß einem anderen Ausführungsbeispiel ist das Host-System derart eingerichtet, dass ein Echtzeit-Prozess pro Kern des Hosts unterstützbar ist.

Nachfolgend werden exemplarische Ausführungsbeispiele des Verfahrens zum Betreiben eines Host-Systems beschrieben. Jedoch gelten die entsprechenden Ausgestaltungen und Merkmale auch für das Host-System, das Computerprogrammelement und das computerlesbare Medium.

Gemäß einem anderen Ausführungsbeispiel des Verfahrens, welches ferner ein Starten eines Emulationsprozesses für eine virtuelle Maschine aufweist, wobei in dem Emulationsprozess eine Zuordnung eines virtuellen Kernes zu einem der Mehrzahl von physikalischen Kernen festgeschrieben wird.

Insbesondere kann der Emulationsprozess als Echtzeit-Prozess auf dem Host-System gestartet werden. Neben einem Emulationsprozess wird ein Kernel-Treiber gestartet. Die Kombination aus einem Emulationsprozess, z.B. dem sogenannten Qemu Emulator, und einem Kernel-Treiber, z.B. dem Kernel-Virtual-Machine, beim Betriebssystem Linux mag eine Möglichkeit sein, eine virtuelle Maschine zu verwirklichen.

Gemäß einem anderen Ausführungsbeispiel weist das Host-System ferner ein Starten einer virtuellen Maschine auf, wobei das Starten der virtuellen Maschine das Starten eines Echtzeit-Prozesses auf dem Host-System aufweist, welcher Echtzeit-Prozess der virtuellen Maschine zugeordnet ist. Insbesondere mögen eine Mehrzahl von virtuellen Maschinen auf dem Host-System gestartet werden, wobei jedem der Mehrzahl von virtuellen Maschinen ein Echtzeit-Prozess auf dem Host-System zugeordnet ist.

Zusammenfassend wird ein Host-System geschaffen, welches zulässt, echtzeitfähige virtuelle Maschinen zu verwirklichen. Hierzu kann das Host-System oder der Host-Rechner eine Mehrzahl von Kernen aufweisen und mit einem Echtzeit-Betriebssystem betrieben werden. Bevorzugt wird auf dem Host-System eine Mehrzahl von virtuellen Maschinen implementiert, wobei die Mehrzahl oder Anzahl von virtuellen Maschinen geringer ist als die Mehrzahl oder Anzahl von physikalischen Kernen des Host-Systems. Somit wird sichergestellt, dass kein physikalischer Kern mehreren virtuellen Maschinen zugeordnet wird. Vorteilhafterweise werden Treiber für die von der virtuellen Maschine benötigte Hardware sowie die Timer als Teil des Echtzeit-Betriebssystems auf dem Host betrieben. Entsprechend werden Emulationsprozesse für die Implementierung der virtuellen Maschinen nur mit dem Host kommunizieren und selber auf einer, oder bei der Verwendung von Threads zur Entkoppelung der einzelnen Hardware-Teile auf mehreren, Echtzeit-Prioritäten laufen. Das Echtzeit-Betriebssystem des Host-Systems ist insbesondere ein Linux Betriebssystem. Jedoch ist jedes andere Echtzeit-Betriebssystem, welches das Bereitstellen von virtuellen Maschinen ermöglich, geeignet. Dabei bleibt das oben beschriebene Grundprinzip erhalten.

Im Falle des Verwendens eines Linux Betriebssystems für den Host oder Server, wird eine Kombination aus einem Kernel-Treiber, beispielsweise einer Kernel-Virtual-Machine (KVM) und eines Emulationsprogrammes oder Emulators, beispielsweise das sogenannte "Qemu" verwendet, um die virtuellen Maschinen zu realisieren. Hierbei mag sich die KVM um die Konfiguration und Nutzung der Virtualisierungstechnologie des Prozessors und des Chipsatzes sorgen, während Qemu für die Emulation der notwendigen Peripherie sorgt. Auf einem Server bietet die virtuelle Maschine den weiteren Vorteil, dass der virtuelle Rechner auch neu gestartet werden kann, wenn er durch einen Fehler unbenutzbar geworden ist. Die anderen VMs auf diesem Host bleiben davon unberührt.

Gerade bei der Emulation von Hardware kann die Verwendung eines RTOS im Host-Rechner einen Vorteil bringen. Der Emulationsprozess (Qemu) wird als Echtzeit-Prozess gefahren, so dass er selbst deterministisch von dem Host-Betriebssystem aufgerufen wird und diesen Determinismus in die virtuelle Maschine weiterleiten bzw. weitergeben kann.

Das beschriebene Prinzip mag sich auch auf den Bereich der Automatisierungswelt ausdehnen lassen, in der heute eine mehr oder wenige verteilte Landschaft aus speicherprogrammierbaren Steuerungen (SPS) existiert, die in einer großen Anlage ihren Dienst verrichten mögen. Mittels der Einrichtung mehrere echtzeitfähiger, virtueller Maschinen vereinen ein oder mehrere Server die vielen SPS in sich. Die an die jeweiligen SPS angeschlossenen Feldgeräte sind hierbei an den bzw. die Server angeschlossen, und es wird ein Netzwerk-Interface der entsprechenden VM zugewiesen, in der die dazugehörigen SPS als Software laufen.

Die oben erläuterten und weiteren Aspekte und Ausführungsbeispiele werden dem Fachmann durch die nachfolgend erläuterten exemplarischen Ausführungsbeispiele klarer verständlich werden. Ferner sollte bemerkt werden, dass Merkmale, welche oben im Zusammenhang mit einem bestimmten Aspekt oder Ausführungsbeispiel beschrieben wurden, auch mit anderen Aspekten und Ausführungsbeispielen kombiniert werden können.

Kurzbeschreibung der Figuren

Fig. 1 zeigt eine schematische Darstellung einer echtzeitfähigen virtuellen Maschine gemäß einem Ausführungsbeispiel.

Fig. 2 zeigt eine schematische Darstellung einer Konfiguration mit zwei echtzeitfähigen virtuellen Maschinen.

Fig. 3 zeigt eine schematische Darstellung eines Host-Systems mit virtuellen Maschinen.

Die Darstellungen in den Figuren sind schematisch. Gleiche oder ähnliche Bauteile oder Elemente in den verschiedenen Figuren sind mit gleichen oder ähnlichen Bezugszeichen versehen.

Fig. 1 zeigt eine schematische Darstellung einer echtzeitfähigen virtuellen Maschine gemäß einem Ausführungsbeispiel. Insbesondere zeigt die Fig. 1 ein Host-System 100, welches Linux als Host-Betriebssystem verwendet und bei welchem ein Echtzeit-Betriebssystem (RTOS) anstelle eines General-Purpose-Betriebssystem (GPOS) verwendet wird.

Das Host-System 100 weist physikalische Host-Hardware 101 auf, welche schematisch als physikalische Kerne oder Cores 102, 103, 104 und 105 dargestellt ist. Auf der Host-

Hardware läuft ein Host-Echtzeit-Betriebssystem 106, welches beispielsweise ein RT-Linux-Betriebssystem mit einem Kernel-Treiber, z.B. Kernel-Virtual-Machine (KVM) und einem Emulator (Qemu) ist. Das RT-Host-Betriebssystem stellt einen Echtzeit-Timer (RT-Timer) 107 und einen Hardware-Treiber (HW-Treiber) 108 je virtueller Maschine zur Verfügung. Zum Zwecke der Übersichtlichkeit ist in Fig. 1 nur eine virtuelle Maschine dargestellt. Ferner laufen auf dem Host Applikationen, welche schematisch mit dem Block 109 in Fig. 1 dargestellt sind.

Aufsetzend auf den Hardware-Treiber 108 und den RT-Timer 107 wird je virtuelle Maschine ein Emulationsprozess (Qemu) 110 gestartet, welcher dem Emulieren einer virtuellen Maschine 111 dient. Auf der virtuellen Maschine 111 kann dann wieder ein Betriebssystem und Applikationen laufen, welche auch virtuelle Hardware nutzen können. Somit kann das Host-System 100 echtzeitfähige virtuelle Maschinen mittels Echtzeit-Hardware-Emulation verwirklichen.

In Fig. 1 ist ein System skizziert, in welchem es mittels einer transparenten Echtzeit-Erweiterung erlaubt sein mag, einen beliebigen Linux-Prozess zum Echtzeit-Prozess hochzustufen, indem zum einen seine Priorität auf einer der höchsten statischen Prioritäten von Linux gehoben wird und zum anderen, indem der Echtzeit-Prozess sich bei der Echtzeit-Erweiterung, z.B. dem Echtzeit-Linux AuDis der I IA&DT ATS 11), registriert. Dies mag dafür sorgen, dass der Prozess nicht weiter von Linux geschedult wird, sondern von dem Echtzeit-Kernel. Damit wird der Prozess sofort deterministisch aktiv, sobald der Timer-Tick oder ein Interrupt einer der Emulation zugeordneten Hardware auftritt. Die Echtzeit-Erweiterung mag einen Echtzeit-Prozess pro Core des Host-Prozessors unterstützen.

Im Unterschied zum System, welches in Fig. 3 dargestellt ist, laufen die Treiber gemäß dem exemplarischen Ausführungsbeispiel der Fig. 1 für die in der virtuellen Maschine (VM) benötigte Hardware sowie die Timer Teil des Echtzeit-Betriebssystems auf dem Host. Entsprechend kommuniziert der Emulationsprozess nur mit diesen und läuft selbst auf einer oder bei der Verwendung von Threads zur Entkopplung der einzelnen Hardware-Teile auf mehreren Echtzeit-Prioritäten.

Die oben im Zusammenhang mit der Fig. 1 angesprochene Kombination Linux mit KVM und Qemu wird nur als ein beispielhaftes Ausführungsbeispiel genannt. Das Grundprinzip zur Realisierung echtzeitfähiger virtueller Maschinen ist bei der Verwendung anderer Programme und Betriebssystem identisch.

Eine solche Virtualisierung mag insbesondere im Server-Umfeld vorteilhaft sein, weil diese zu einer Konsolidierung der physikalischen Rechner führen mag. Obwohl logisch eine Vielzahl von Servern zur Verfügung gestellt werden können, sind jedoch nur wenige Rechner sichtbar. Die Vielzahl an Servern existiert nur virtuell.

Das gezeigte Prinzip mag sich auch auf den Bereich der Automatisierungswelt ausdehnen lassen, in der heute eine mehr oder wenige verteilte Landschaft aus speicherprogrammierbaren Steuerungen (SPS) existiert, die in einer großen Anlage ihren Dienst verrichten mögen. Mittels der Einrichtung mehrere echtzeitfähiger, virtueller Maschinen mögen ein oder mehrere Server die vielen SPS in sich vereinen. Die an die jeweiligen SPS angeschlossenen Feldgeräte mögen hierbei an den bzw. die Server angeschlossen und ein Netzwerk-Interface der entsprechenden VM zugewiesen werden, in der die dazugehörige SPS als Software läuft.

Ein Vorteil bei der Verwendung innerhalb eines Servers mag sich implizit aus der möglichen Nutzung eines physikalischen Systemstimers zur Triggerung von emulierten Timern für einzelne virtuelle Maschinen ergeben. Deren Uhrzeiten mögen hierdurch implizit immer synchronisiert sein und können prinzipbedingt nicht auseinanderlaufen, da die Cores aller VM ebenfalls an einen zentralen Takt angeschlossen sind. Somit mag im Vergleich zur dezentralen SPS-Lösung sogar eine höhere Qualität bei weniger Kommunikationsaufwand erreichbar sein. Über die Synchronisation der Host-Systeme können somit auch virtuelle Maschinen mehrerer Server synchronisiert werden, ohne dass die Systeme in den VMs miteinander kommunizieren müssen.

Dieser Vorteil mag sich ebenfalls bei einem anderen Anwendungsfall ergeben, nämlich der Integration mehrerer bislang getrennter Geräte in einem System. Hier sind vorwiegend sogenannte Legacy-Betriebssysteme (auch mit Echtzeit) im Einsatz, die nicht für den Betrieb in einer virtuellen Maschine entwickelt wurden, sondern eine deterministische Hardware für den Betrieb voraussetzen. Da die echtzeitfähigen VMs der beschriebenen Bauart einen deterministischen Betrieb ermöglichen, mag es möglich sein, dass Legacy-Betriebssysteme verwendbar sind. Damit mag es sogar möglich sein, ein und dasselbe Diskimage, in dem Betriebssystem und Applikationen gespeichert sind, sowohl für eine reale Hardware als auch innerhalb der VM zu nutzen, was einen Deploy-Prozess vereinfachen und den Testaufwand reduzieren mag.

Die einzelnen Emulationsprozesse mögen zwar getrennt auf verschiedenen Cores laufen, sind aber alle Prozesse des gleichen Host-Betriebssystems, so dass sie mit dessen Mitteln, z.B. Semaphoren, Message-Queues, Shared-Memory) untereinander kommunizieren können. Damit mögen auch

virtuelle Kommunikationskanäle zwischen VMs geschaffen werden, über welche sogar eine deterministische Kommunikation möglich sein mag.

Zusätzlich mag jeder Thread der Emulation, der einen Core oder eine Peripherie darstellt, fest an einen physikalischen Core des Host-Systems gebunden sein, so dass er deterministisch arbeiten kann, auch in Hinblick auf das Cache-Verhalten. Die Peripherie-Threads mögen entweder auf dem gleichen Core oder falls verfügbar auf einem weiteren laufen, so dass sich die Qualität des Echtzeitverhaltens weiter verbessern mag.

Neben den echtzeitfähigen VMs mögen optional auch weitere, nicht-echtzeitfähige VMs auf einem Host-System installiert werden, sofern dieses über eine ausreichende Anzahl von Rechencores verfügt.

Fig. 2 zeigt eine schematische Darstellung einer Konfiguration eines Host-Rechners 200 mit zwei echtzeitfähigen virtuellen Maschinen 201 und 202. Eine solche Konfiguration mag sowohl für einen SPS-Server als auch für ein Integrationsszenario gültig sein. Basierend auf einem Echtzeit-Betriebssystem des Host-Rechners werden die zwei virtuellen Maschinen 201 und 202 gestartet. Als Hardware sind in dem Host-Rechner 200 zwei Feldbus-Karten 203 und 204 eingebaut, die durch Treiber auf der Echtzeitseite des Host-Rechners 200 gesteuert oder kontrolliert werden. Darauf aufbauend stellen zwei Emulationsprozesse in ihrer jeweiligen virtuellen Maschine diese Geräte zur Verfügung und leiten die Interrupts bei eingehenden Telegrammen entsprechend deterministisch an das Echtzeit-Betriebssystem in der VM weiter.

Timer in den virtuellen Maschinen werden von ein und demselben Echtzeit-Trigger in dem Host-Rechner getriggert, so dass die Zeiten in den VMs immer streng synchron sind.

Eine solche Lösung mittels echtzeitfähiger virtueller Maschinen unterscheidet sich grundlegend von Lösungen, die einen sogenannten Hypervisor (HV) verwenden. Dieser teilt zwar auch das System in mehrere virtuelle Maschinen auf, lässt diese aber getrennt von einander laufen, so dass sich keine implizite Synchronisation ergibt. Ebenso kann der HV nicht das Problem der sogenannten Shared-Ressources lösen, da er nicht über eine Hardware-Emulation verfügt, insbesondere über keine echtzeitfähige. Der Hypervisor des Standes der Technik partitioniert stattdessen das System und ordnet jede Peripherie eindeutig einer Partition zu, so dass diese von den Betriebssystemen der anderen Partitionen nicht mehr gesehen werden kann.

Die Ausführung der Erfindung ist nicht auf diese Anwendungsfälle und die weiter oben erwähnten Systemkonfigurationen beschränkt, sondern ebenso in einer Vielzahl von Abwandlungen möglich, die im Rahmen fachgemäßen Handelns liegen. Ferner sollte darauf hingewiesen werden, dass Bezugszeichen in den Ansprüchen nicht als beschränkend aufzufassen sind und dass die Begriffe "aufweisen" bzw. "aufweisend" und ähnliche Begriffe nicht das Vorhandensein von weiteren Elementen oder Schritten ausschließt. Auch schließt ein Aufzählen als mehrere Mittel oder Elemente nicht aus, dass diese Mittel oder Elemente als ein einziges Mittel oder Element ausgebildet werden können.

Bezugszeichenliste

100 Host-System
101 Host Hardware
102-105 Cores
106 Host-Betriebssystem
107 Echtzeit-Timer
108 Hardware-Treiber
109 Host-Applikationen
110 Emulationsprozess
111 Virtuelle Maschine
200 Host-System
201 Virtuelle Maschine
202 Virtuelle Maschine
203 Feldbus Karte
204 Feldbus Karte
300 Host-System
301 Host Hardware
302-305 Cores
306 Host-Betriebssystem
307 Hardware-Treiber
308 Hardware-Treiber
309 Host-Applikationen
310 Emulationsprozess
311 Emulationsprozess
312 Virtuelle Maschine
313 Virtuelle Maschine

Patentansprüche

1. Host-System (100, 200) zum Bereitstellen einer echtzeitfähigen virtuellen Maschine (111, 201, 202), wobei das Host-System (100, 200) aufweist:

einen Host (101, 201), welcher eine Mehrzahl von physikalischen Kernen (102, 103, 104, 105, 202, 203, 204, 205) aufweist;

wobei auf dem Host ein Echtzeit-Betriebssystem (106) installiert ist.

2. Host-System (100, 200) gemäß Anspruch 1,

wobei das Host-System (100, 200) derart eingerichtet ist, dass ein Emulationsprozess (110) für eine echtzeitfähige virtuelle Maschine als Echtzeit-Prozess implementiert ist.

3. Host-System (100, 200) gemäß Anspruch 2,

wobei das Host-System (100, 200) derart eingerichtet ist, dass eine Zuordnung eines virtuellen Kerns zu einem der Mehrzahl von physikalischen Kernen (102-105, 202-205) statisch ist.

4. Host-System (100, 200) gemäß einem der Ansprüche 1 bis 3,

wobei das Host-System (100, 200) eingerichtet ist eine Mehrzahl von virtuellen Maschinen zu emulieren,

wobei die Mehrzahl von physikalischen Kernen (102-105, 202-205) größer oder gleich der Mehrzahl von virtuellen Maschinen ist.

5. Host-System (100, 200) gemäß einem der Ansprüche 1 bis 4,

wobei das Host-System derart eingerichtet ist, dass ein Treiber für eine virtuelle Maschine als Echtzeit-Treiber auf dem Host implementiert ist.

6. Host-System (100, 200) gemäß einem der Ansprüche 1 bis 5,

wobei das Host-System derart eingerichtet ist, dass mittels ihm eine Mehrzahl von virtuellen Maschinen (111, 201, 202) bereitstellbar ist.

7. Host-System (100, 200) gemäß Anspruch 6,

wobei das Host-System derart eingerichtet ist, dass ein Echtzeit-Prozess pro Kern des Hosts unterstützbar ist.

8. Verfahren zum Betreiben eines Host-Systems (100, 200) für echtzeitfähige virtuelle Maschinen (111, 201, 202), wobei das Host-System eine Mehrzahl von physikalischen Kernen (102-105, 202-205) aufweist, wobei das Verfahren aufweist:

Betreiben eines Echtzeit-Betriebssystems (110) auf dem Host-System.

9. Verfahren gemäß Anspruch 8, welches ferner aufweist:

Starten eines Emulationsprozesses (110) für eine virtuelle Maschine (111, 201, 202), wobei in dem Emulationsprozess (110) eine Zuordnung eines virtuellen Kernes zu einem der Mehrzahl von physikalischen Kernen (102-105, 202-205) festgeschrieben wird.

10. Verfahren gemäß Anspruch 8 oder 9, wobei das Verfahren ferner aufweist:

Starten einer virtuellen Maschine,

wobei das Starten der virtuellen Maschine das Starten eines Echtzeit-Prozesses (107) auf dem Host-System aufweist, welcher Echtzeit-Prozess der virtuellen Maschine zugeordnet ist.

11. Programmelement, welches derart eingerichtet ist, dass es, wenn es auf einem Prozessor ausgeführt wird, ein Verfahren gemäß einem der Ansprüche 8 bis 10 steuert.

12. Computerlesbares Medium, auf welchem ein Computerprogramm gespeichert ist, wobei das Computerprogramm derart eingerichtet ist, dass es, wenn es auf einem Prozessor ausgeführt wird, ein Verfahren gemäß einem der Ansprüche 8 bis 10 steuert.

FIG 1

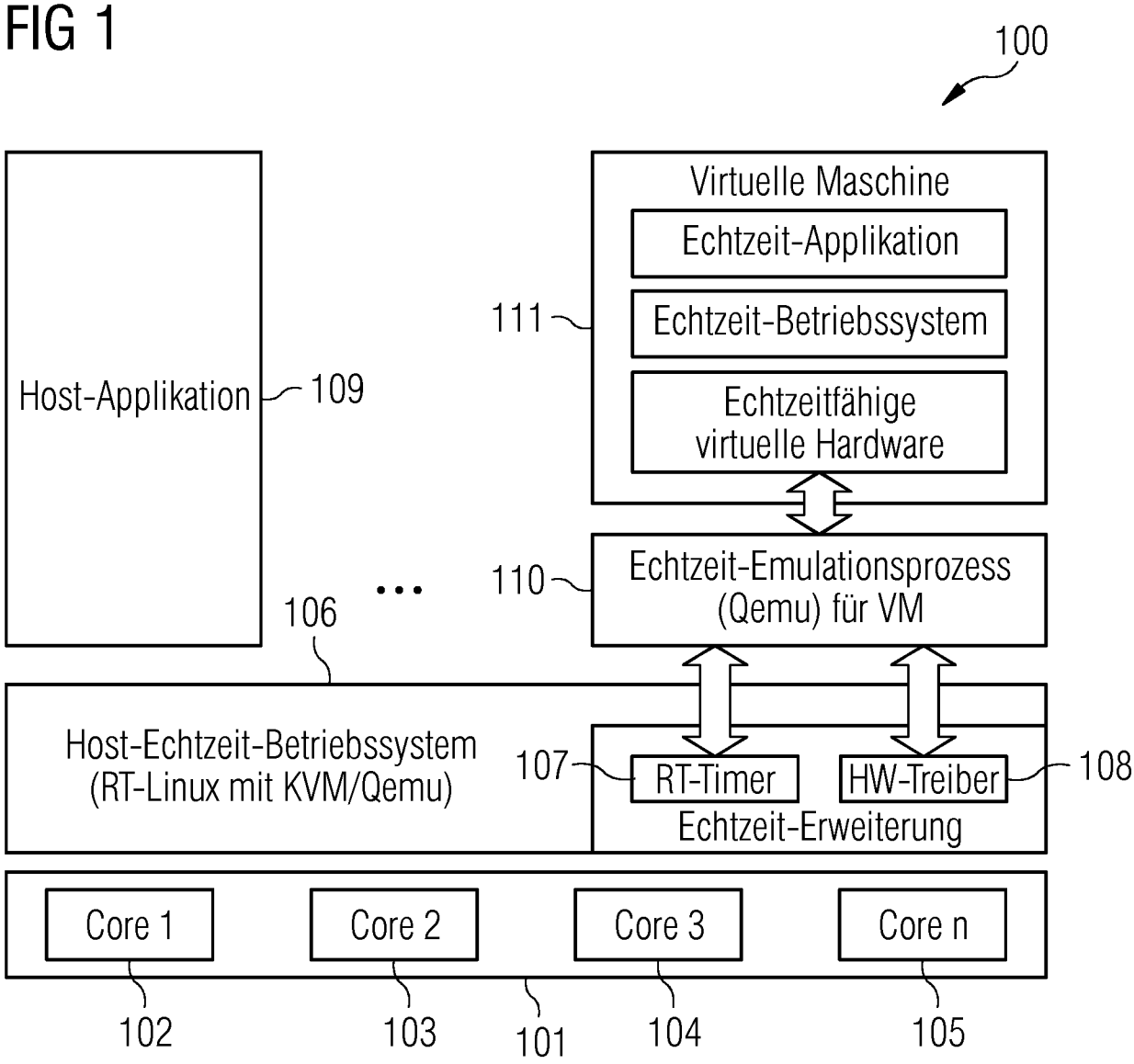

100
Virtuelle Maschine
Echtzeit-Applikation
Echtzeit-Betriebssystem
Echtzeitfähige virtuelle Hardware
111
Host-Applikation
109
110
Echtzeit-Emulationsprozess (Qemu) für VM
106
Host-Echtzeit-Betriebssystem (RT-Linux mit KVM/Qemu)
107
RT-Timer
HW-Treiber
108
Echtzeit-Erweiterung
Core 1
Core 2
Core 3
Core n
102
103
101
104
105

FIG 2

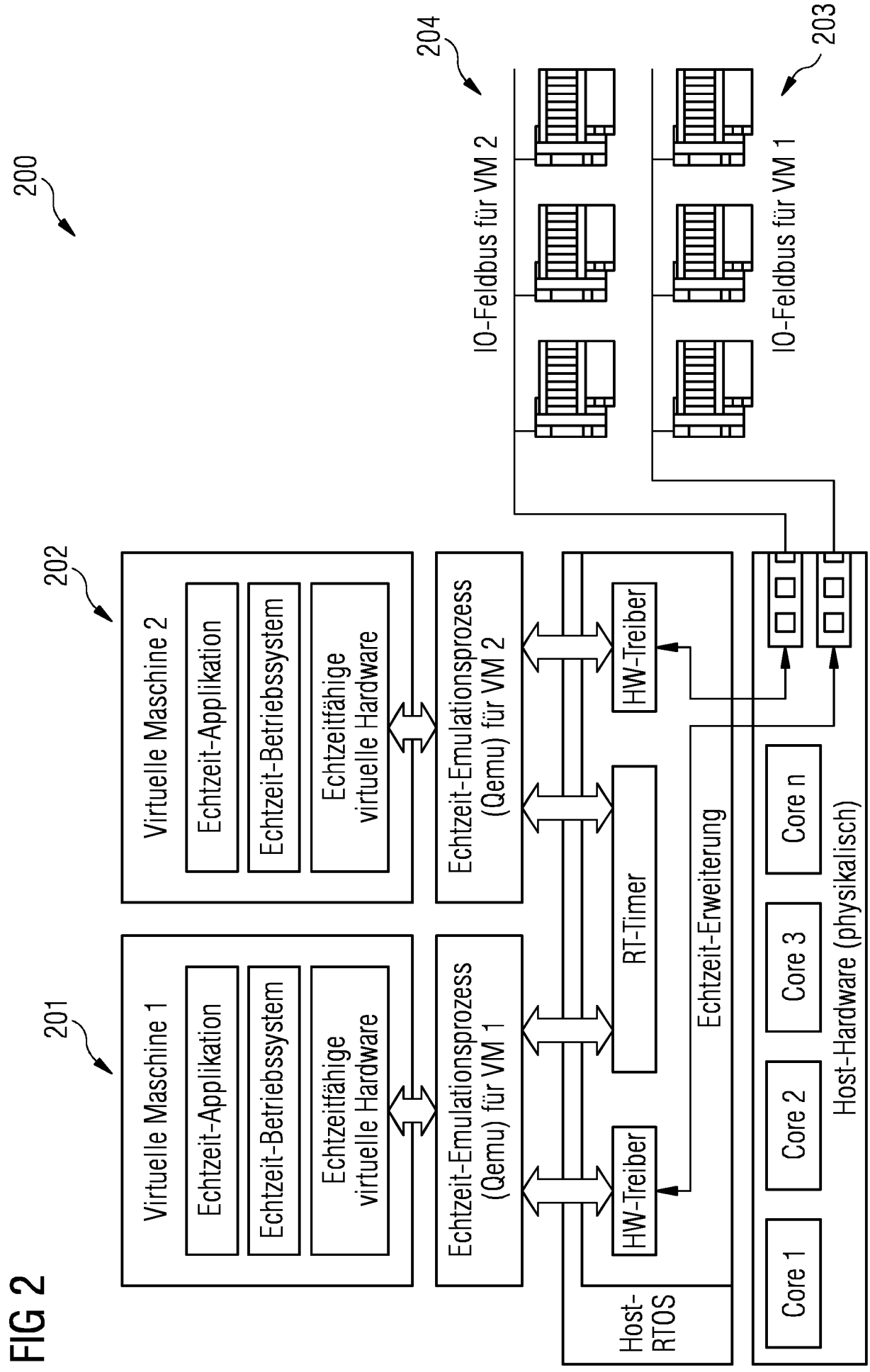
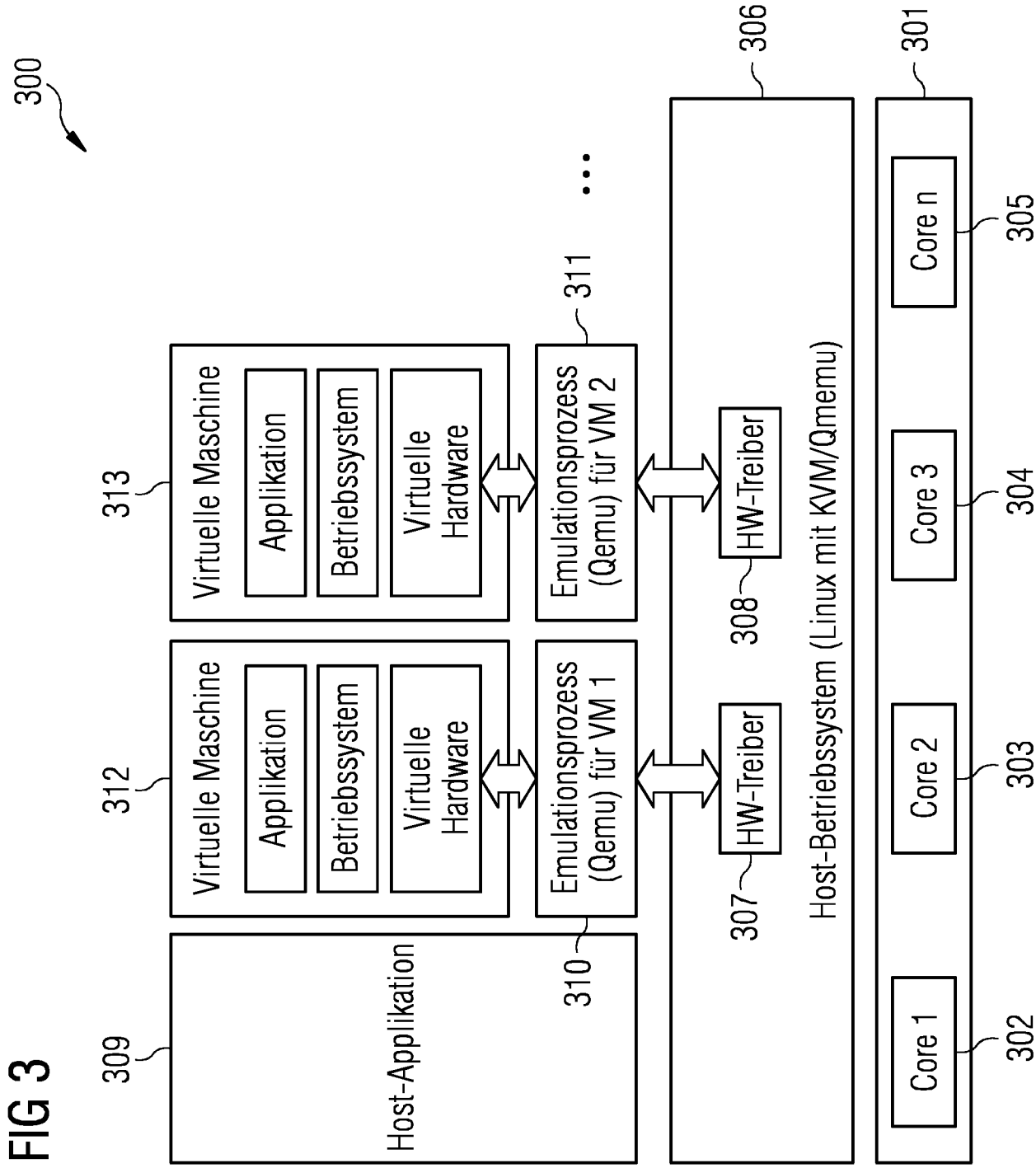

200
201
202
203
204
Virtuelle Maschine 1
Echtzeit-Applikation
Echtzeit-Betriebssystem
Echtzeitfähige virtuelle Hardware
Virtuelle Maschine 2
Echtzeit-Applikation
Echtzeit-Betriebssystem
Echtzeitfähige virtuelle Hardware
Echtzeit-Emulationsprozess (Qemu) für VM 1
Echtzeit-Emulationsprozess (Qemu) für VM 2
HW-Treiber
HW-Treiber
RT-Timer
Echtzeit-Erweiterung
Host-RTOS
IO-Feldbus für VM 1
IO-Feldbus für VM 2
Core 1
Core 2
Core 3
Core n
Host-Hardware (physikalisch)

FIG 3


300
309
Host-Applikation
310
Emulationsprozess (Qemu) für VM 1
311
Emulationsprozess (Qemu) für VM 2
312
Virtuelle Maschine
Applikation
Betriebssystem
Virtuelle Hardware
313
Virtuelle Maschine
Applikation
Betriebssystem
Virtuelle Hardware
307
HW-Treiber
308
HW-Treiber
306
Host-Betriebssystem (Linux mit KVM/Qmemu)
301
302
Core 1
303
Core 2
304
Core 3
305
Core n

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2011/057631

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/455 G06F9/50
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F G05B

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Jan Kiszka: "Towards Linux as a Real-Time Hypervisor", 11th Real-Time Linux Workshop on 2009, September 28 to 30, in Dresden, Germany, 28 September 2009 (2009-09-28), pages 1-10, XP55018265, Retrieved from the Internet: URL:http://old.lwn.net/lwn/images/conf/rtlws11/papers/proc/p18.pdf [retrieved on 2012-02-02] page 1, right-hand column, line 18 - page 2, right-hand column, line 27 page 2, right-hand column, line 33 - page 4, left-hand column, line 4 page 5, left-hand column, lines 1-23; figure 7 page 6, left-hand column, line 1 - page 8, right-hand column, line 20; figure 10 page 9, left-hand column, line 1 - -/--	1-12

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
22 March 2012	17/04/2012

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	de Junca, Irène

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	right-hand column, line 31	
X	JUN ZHANG ET AL: "Performance analysis towards a KVM-Based embedded real-time virtualization architecture", COMPUTER SCIENCES AND CONVERGENCE INFORMATION TECHNOLOGY (ICCIT), 2010 5TH INTERNATIONAL CONFERENCE ON, IEEE, 30 November 2010 (2010-11-30), pages 421-426, XP031912968, DOI: 10.1109/ICCIT.2010.5711095 ISBN: 978-1-4244-8567-3	1,8,11, 12
A	the whole document	2-7,9,10
A	Anonymous: "TenAsys Real-time Hypervisor", TenAsys Corporation , 1 August 2006 (2006-08-01), pages 1-11, XP55018231, Retrieved from the Internet: URL:http://www.tenasys.com/support/files/TenAsysReal-timeHypervisorBackgrounder.pdf [retrieved on 2012-02-02] page 5, line 39 - page 7, line 27 page 8, line 34	1-12
A	Bill Alexander ET AL: "Architected for Performance - Virtualization Support on Nehalem and Westmere Processors", Intel Technology Journal, Volume 14, Issue 3, 2010, 1 September 2010 (2010-09-01), pages 84-103, XP55020418, ISBN: 978-1-93-405333-1 Retrieved from the Internet: URL:http://www.intel.com/technology/itj/2010/v14i3/pdfs/Architected_for_Performance_Virtualization_support.pdf [retrieved on 2012-02-27] page 98, line 10 - page 99, line 3	1-12

INTERNATIONALER RECHERCHENBERICHT

Internationales Aktenzeichen
PCT/EP2011/057631

A. KLASSIFIZIERUNG DES ANMELDUNGSGEGENSTANDES
INV. G06F9/455 G06F9/50
ADD.

Nach der Internationalen Patentklassifikation (IPC) oder nach der nationalen Klassifikation und der IPC

B. RECHERCHIERTE GEBIETE

Recherchierter Mindestprüfstoff (Klassifikationssystem und Klassifikationssymbole)
G06F G05B

Recherchierte, aber nicht zum Mindestprüfstoff gehörende Veröffentlichungen, soweit diese unter die recherchierten Gebiete fallen

Während der internationalen Recherche konsultierte elektronische Datenbank (Name der Datenbank und evtl. verwendete Suchbegriffe)
EPO-Internal, WPI Data

C. ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie*	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
X	Jan Kiszka: "Towards Linux as a Real-Time Hypervisor", 11th Real-Time Linux Workshop on 2009, September 28 to 30, in Dresden, Germany, 28. September 2009 (2009-09-28), Seiten 1-10, XP55018265, Gefunden im Internet: URL:http://old.lwn.net/lwn/images/conf/rtlws11/papers/proc/p18.pdf [gefunden am 2012-02-02] Seite 1, rechte Spalte, Zeile 18 - Seite 2, rechte Spalte, Zeile 27 Seite 2, rechte Spalte, Zeile 33 - Seite 4, linke Spalte, Zeile 4 Seite 5, linke Spalte, Zeilen 1-23; Abbildung 7 Seite 6, linke Spalte, Zeile 1 - Seite 8, rechte Spalte, Zeile 20; Abbildung 10 Seite 9, linke Spalte, Zeile 1 - rechte -/--	1-12

[X] Weitere Veröffentlichungen sind der Fortsetzung von Feld C zu entnehmen [ ] Siehe Anhang Patentfamilie

* Besondere Kategorien von angegebenen Veröffentlichungen :
"A" Veröffentlichung, die den allgemeinen Stand der Technik definiert, aber nicht als besonders bedeutsam anzusehen ist
"E" älteres Dokument, das jedoch erst am oder nach dem internationalen Anmeldedatum veröffentlicht worden ist
"L" Veröffentlichung, die geeignet ist, einen Prioritätsanspruch zweifelhaft erscheinen zu lassen, oder durch die das Veröffentlichungsdatum einer anderen im Recherchenbericht genannten Veröffentlichung belegt werden soll oder die aus einem anderen besonderen Grund angegeben ist (wie ausgeführt)
"O" Veröffentlichung, die sich auf eine mündliche Offenbarung, eine Benutzung, eine Ausstellung oder andere Maßnahmen bezieht
"P" Veröffentlichung, die vor dem internationalen Anmeldedatum, aber nach dem beanspruchten Prioritätsdatum veröffentlicht worden ist
"T" Spätere Veröffentlichung, die nach dem internationalen Anmeldedatum oder dem Prioritätsdatum veröffentlicht worden ist und mit der Anmeldung nicht kollidiert, sondern nur zum Verständnis des der Erfindung zugrundeliegenden Prinzips oder der ihr zugrundeliegenden Theorie angegeben ist
"X" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann allein aufgrund dieser Veröffentlichung nicht als neu oder auf erfinderischer Tätigkeit beruhend betrachtet werden
"Y" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann nicht als auf erfinderischer Tätigkeit beruhend betrachtet werden, wenn die Veröffentlichung mit einer oder mehreren anderen Veröffentlichungen dieser Kategorie in Verbindung gebracht wird und diese Verbindung für einen Fachmann naheliegend ist
"&" Veröffentlichung, die Mitglied derselben Patentfamilie ist

Datum des Abschlusses der internationalen Recherche	Absendedatum des internationalen Recherchenberichts
22. März 2012	17/04/2012

Name und Postanschrift der Internationalen Recherchenbehörde	Bevollmächtigter Bediensteter
Europäisches Patentamt, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	de Junca, Irène

C. (Fortsetzung) ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie*	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
	Spalte, Zeile 31 -----	
X	JUN ZHANG ET AL: "Performance analysis towards a KVM-Based embedded real-time virtualization architecture", COMPUTER SCIENCES AND CONVERGENCE INFORMATION TECHNOLOGY (ICCIT), 2010 5TH INTERNATIONAL CONFERENCE ON, IEEE, 30. November 2010 (2010-11-30), Seiten 421-426, XP031912968, DOI: 10.1109/ICCIT.2010.5711095 ISBN: 978-1-4244-8567-3	1,8,11, 12
A	das ganze Dokument -----	2-7,9,10
A	Anonymous: "TenAsys Real-time Hypervisor", TenAsys Corporation , 1. August 2006 (2006-08-01), Seiten 1-11, XP55018231, Gefunden im Internet: URL:http://www.tenasys.com/support/files/TenAsysReal-timeHypervisorBackgrounder.pdf [gefunden am 2012-02-02] Seite 5, Zeile 39 - Seite 7, Zeile 27 Seite 8, Zeile 34 -----	1-12
A	Bill Alexander ET AL: "Architected for Performance - Virtualization Support on Nehalem and Westmere Processors", Intel Technology Journal, Volume 14, Issue 3, 2010, 1. September 2010 (2010-09-01), Seiten 84-103, XP55020418, ISBN: 978-1-93-405333-1 Gefunden im Internet: URL:http://www.intel.com/technology/itj/2010/v14i3/pdfs/Architected_for_Performance_Virtualization_support.pdf [gefunden am 2012-02-27] Seite 98, Zeile 10 - Seite 99, Zeile 3 -----	1-12