(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 698 33 914 T2** 2006.08.24

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 0 917 057 B1**
(21) Deutsches Aktenzeichen: **698 33 914.2**
(96) Europäisches Aktenzeichen: **98 309 011.9**
(96) Europäischer Anmeldetag: **04.11.1998**
(97) Erstveröffentlichung durch das EPA: **19.05.1999**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **22.03.2006**
(47) Veröffentlichungstag im Patentblatt: **24.08.2006**

(51) Int Cl.[8]: ***G06F 9/46*** (2006.01)

(30) Unionspriorität:

64250	**04.11.1997**	**US**
95543	**10.06.1998**	**US**

(84) Benannte Vertragsstaaten: **DE, FR, GB**

(73) Patentinhaber:
Compaq Computer Corp., Houston, Tex., US

(74) Vertreter:
Grünecker, Kinkeldey, Stockmair & Schwanhäusser, 80538 München

(72) Erfinder:
Zalewski, Stephen H., Nashua, New Hampshire 03062, US; Mason, Andrew H., Hollis, New Hampshire 03049, US; Jordan, Gregory H., Hollis, New Hampshire 03049, US; Noel, Karen L., Pembroke, New Hampshire 03275, US; Kaufman, James R., Nashua, New Hampshire 03062, US; Harter, Paul K., Groton, Massachusetts 01540, US; Kleinsorge, Frederick G., Amherst, New Hampshire 03031, US; Shirron, Stephen F., Acton, Massachusetts 01720, US

(54) Bezeichnung: **Architektur eines Multiprozessorrechners mit mehreren Betriebssysteminstanzen und softwaregesteuerter Betriebsmittelzuteilung**

Beschreibung

GEBIET DER ERFINDUNG

[0001] Diese Erfindung betrifft Mehrprozessor-Computer-Architekturen, in denen Prozessoren und andere Computer-Hardware-Ressourcen in Partitionen gruppiert sind, von denen jede eine Betriebssystem-Instanz aufweist, und insbesondere Verfahren und Einrichtungen zum Reservieren von Computer-Hardware-Ressourcen für Partitionen.

[0002] Der effiziente Betrieb vieler Anwendungen in gegenwärtigen Rechnerumgebungen hängt von schnellen, leistungsstarken und flexiblen Rechnersystemen ab. Die Konfiguration und Auslegung solcher Systeme ist sehr kompliziert geworden, wenn solche Systeme in einer gewerblichen "Unternehmens"-Umgebung eingesetzt werden sollen, in der es viele getrennte Abteilungen, viele verschiedene Problemtypen und ständig wechselnde EDV-Bedürfnisse geben kann. Benutzer in solchen Umgebungen möchten im Allgemeinen in der Lage sein, die Kapazität des Systems, seine Geschwindigkeit und seine Konfiguration rasch und leicht zu ändern. Sie möchten auch die System-Arbeitskapazität erhöhen und Konfigurationen ändern können, um eine bessere Nutzung der Ressourcen zu erzielen, ohne die Ausführung von Anwendungsprogrammen auf dem System zu stoppen. Außerdem möchten sie vielleicht in der Lage sein, das System zu konfigurieren, um die Ressourcenverfügbarkeit so zu maximieren, dass jede Anwendung eine optimale Rechenkonfiguration aufweist.

[0003] Herkömmlicherweise wurde die Rechengeschwindigkeit unter Verwendung einer "Shared-Nothing"-Rechenarchitektur angesteuert, in der Daten, Geschäftslogik und grafische Benutzerschnittstellen unterschiedliche Tiers sind und spezifische Rechenressourcen, die jedem Tier zugeordnet sind, aufweisen. Anfänglich wurde ein einzelner Zentralprozessor verwendet, und die Leistungsstärke und Geschwindigkeit eines solchen Rechensystems wurde erhöht, indem die Taktfrequenz des einzelnen Zentralprozessors erhöht wurde. In letzter Zeit wurden Rechensysteme entwickelt, die mehrere Prozessoren verwenden, die als ein Team arbeiten, statt eines massiven Prozessors, der alleine arbeitet. Auf diese Weise kann eine komplexe Anwendung auf viele Prozessoren aufgeteilt werden, statt darauf zu warten, dass sie von einem einzelnen Prozessor ausgeführt wird. Solche Systeme bestehen typischerweise aus mehreren Zentralprozessoren (CPUs), die durch ein einzelnes Betriebssystem gesteuert werden. In einer Variante eines Mehrprozessor-Systems, das als "symmetrischer Mehrprozessorbetrieb" oder SMP bezeichnet wird, werden die Anwendungen gleichmäßig auf alle Prozessoren verteilt. Die Prozessoren verwenden auch gemeinsam einen Speicher. In einer anderen Variante, die als "asymmetrischer Mehrprozessorbetrieb" oder AMP bezeichnet wird, arbeitet ein Prozessor als ein "Master", und alle anderen Prozessoren arbeiten als "Slaves". Daher müssen alle Operationen, einschließlich des Betriebssystems, über den Master laufen, bevor sie an die Slave-Prozessoren weitergegeben werden. Die Mehrprozessorbetriebs-Architekturen weisen den Vorteil auf, dass die Leistung durch Hinzufügen weiterer Prozessoren gesteigert werden kann, leiden jedoch unter dem Nachteil, dass die auf solchen Systemen laufende Software sorgfältig geschrieben werden muss, um die mehreren Prozessoren nutzen zu können, und es schwierig ist, die Software zu skalieren, wenn sich die Anzahl der Prozessoren erhöht. Gegenwärtige gewerbliche Auslastungen lassen sich über 8–24 CPUs hinaus nicht gut als einzelnes SMP-System skalieren, wobei die genaue Anzahl von Plattform, Betriebssystem und Anwendungs-Mix abhängt.

[0004] Zur Steigerung der Leistung bestand eine andere typische Lösungsmöglichkeit darin, die Computerressourcen (Maschinen) einer Anwendung zuzuordnen, um die Maschinenressourcen optimal auf die Anwendung abzustimmen. Dieser Ansatz wurde jedoch von der Mehrhit der Benutzer nicht übernommen, da die meisten Einsatzorte viele Anwendungen und getrennte Datenbanken aufweisen, die von unterschiedlichen Anbietern entwickelt wurden. Es ist daher schwierig und kostspielig, Ressourcen zu allen der Anwendungen zuzuordnen, insbesondere in Umgebungen, in denen sich der Anwendungs-Mix ständig verändert.

[0005] Alternativ kann ein Rechensystem mit Hardware partitioniert werden, um eine Untergruppe der Ressourcen auf einem Computer zu bilden, die für eine spezifische Anwendung verfügbar sind. Dieser Ansatz vermeidet die permanente Zuordnung der Ressourcen, da die Partitionen geändert werden können, weist aber immer noch Probleme hinsichtlich der Leistungsverbesserung mittels eines Lastausgleichs zwischen den Partitionen und der Ressourcenverfügbarkeit auf.

[0006] Die Probleme der Verfügbarkeit und Verwaltbarkeit wurden durch ein "Shared-Everything"-Modell angegangen, in dem ein großer zentralisierter robuster Server, der den größten Teil der Ressourcen enthält, im Netzwerk mit vielen kleinen, unkomplizierten Client-Netzwerk-Computern verbunden ist und sie bedient. Alternativ werden "Cluster" verwendet, in denen jedes System bzw. jeder "Knoten" seinen eigenen Speicher hat

und von seinem eigenen Betriebssystem gesteuert wird. Die Systeme wirken durch die gemeinsame Benutzung von Disketten und die Übergabe von Nachrichten untereinander über eine Art Kommunikationsnetzwerk zusammen. Ein Cluster-System weist den Vorteil auf, dass zusätzliche Systeme leicht zu einem Cluster hinzugefügt werden können. Allerdings leiden Netzwerke und Cluster unter einem Mangel an gemeinsamem Speicher und einer begrenzten Verbindungsbandbreite, die der Leistung Einschränkungen auferlegen.

[0007] In vielen Unternehmens-Rechenumgebungen ist klar, dass die zwei getrennten Rechenmodelle gleichzeitig integriert werden müssen und jedes Modell optimiert werden muss. Mehrere Ansätze des Stands der Technik sind verwendet wurden, um diese Integrierung zu versuchen. Zum Beispiel verwendet eine Auslegung, die als eine "virtuelle Maschine" oder VM bezeichnet und von International Business Machines Corporation, Armonk, New York, entwickelt und vermarktet wird, eine einzelne physikalische Maschine mit einem oder mehreren physikalischen Prozessoren in Kombination mit Software, die mehrere virtuelle Maschinen simuliert. Jede dieser virtuellen Maschinen besitzt im Prinzip Zugriff auf alle physikalischen Ressourcen des zugrundeliegenden echten Computers. Die Zuweisung von Ressourcen zu jeder virtuellen Maschine wird durch ein Programm gesteuert, das als ein "Hypervisor" bezeichnet wird. Es gibt nur einen Hypervisor in dem System, und er ist für alle physikalischen Ressourcen zuständig. Demzufolge nimmt der Hypervisor, nicht die anderen Betriebssysteme, die Reservierung von physikalischer Hardware vor. Der Hypervisor fängt Ressourcen-Anforderungen von den anderen Betriebssystemen ab und bearbeitet die Anforderungen in einer im Allgemeinen korrekten Art.

[0008] Die VM-Architektur unterstützt das Konzept einer "logischen Partition" bzw. LPAR. Jede LPAR enthält einige der verfügbaren physikalischen CPUs und Ressourcen, die der Partition logisch zugewiesen sind. Die gleichen Ressourcen können zu mehr als einer Partition zugewiesen werden. LPARs werden von einem Administrator statisch eingerichtet, können aber auf Lastenänderungen dynamisch und ohne Neustart auf mehrere Arten reagieren.

[0009] Wenn zum Beispiel zwei logische Partitionen, von denen jede zehn CPUs enthält, auf einem physikalischen System gemeinsam genutzt werden, das zehn physikalische CPUs enthält, und wenn die logischen zehn CPU-Partitionen komplementäre Spitzenlasten aufweisen, kann jede Partition das gesamte physikalische Zehn-CPU-System ohne einen Neustart oder einen Eingriff der Bedienperson übernehmen, wenn sich die Arbeitslast ändert.

[0010] Des Weiteren können die CPUs, die jeder Partition logisch zugewiesen sind, über normale Betriebssystem-Operatorbefehle ohne Neustart dynamisch "Ein"- und "Aus"-geschaltet werden. Die einzige Einschränkung ist, dass die Anzahl der CPUs, die bei Systeminitialisierung aktiv sind, die maximale Anzahl von CPUs ist, die in jeder Partition "Ein"-geschaltet werden können.

[0011] Schließlich können in Fällen, in den der gesamte Arbeitslastbedarf aller Partitionen höher ist als von dem physikalischen System bereitgestellt werden kann, LPAR-Wichtungen verwendet werden, um zu definieren, wie viel von den gesamten CPU-Ressourcen an jede Partition vergeben wird. Diese Wichtungen können von den Bedienpersonen fliegend, ohne Unterbrechung, geändert werden.

[0012] Ein weiteres System des bisherigen Stands der Technik wird als "Paralleles Sysplex" bezeichnet und wird ebenfalls von der International Business Machines Corporation vermarktet und entwickelt. Diese Architektur besteht aus einer Gruppe von Computern, die über eine als "Kopplungseinrichtung" bezeichnete Hardware-Einheit, die an jeder CPU angebracht ist, zu Clustern zusammengefasst werden. Die Kopplungseinrichtungen an jedem Knoten sind über eine faseroptische Verbindung angeschlossen, und jeder Knoten arbeitet wie eine herkömmliche SMP-Maschine mit einer Höchstanzahl von 10 CPUs. Gewisse CPU-Befehle rufen die Kopplungseinrichtung direkt auf. Zum Beispiel registriert ein Knoten eine Datenstruktur in der Kopplungseinrichtung, anschließend sorgt die Kopplungseinrichtung dafür, dass die Datenstrukturen in dem lokalen Speicher jedes Knotens kohärent gehalten werden.

[0013] Der Enterprise 10000 Unix-Server, der von Sun Microsystems, Mountain View, Kalifornien, entwickelt und vermarktet wird, verwendet eine Partitionierung, die als "Dynamische Systemdomänen" bezeichnet wird, um die Ressourcen eines einzelnen physikalischen Servers in mehrfache Partitionen, oder Domänen, von denen jede als ein Stand-Alone-Server arbeitet, logisch zu unterteilen. Jede der Partitionen besitzt CPUs, Speicher und I/O-Hardware. Die dynamische Neukonfiguration gestattet es einem Systemadministrator, Domänen fliegend und ohne Neustart zu erstellen, ihre Größe anzupassen oder zu löschen. Jede Domäne bleibt von jeder anderen Domäne in dem System logisch isoliert, wodurch sie vollständig von jedem Software-Fehler oder CPU-, Speicher- oder I/O-Fehler isoliert wird, der von einer anderen Domäne generiert wird. Unter den Domä-

nen werden keine Ressourcen gemeinsam genutzt.

[0014] Das an der Stanford University durchgeführte Hive-Projekt verwendet eine Architektur, die als eine Gruppe von Zellen strukturiert ist. Wenn das System gestartet wird, wird jeder Zelle ein Bereich von Knoten zugewiesen, zu dem sie durchgehend während der Ausführung zugehörig ist. Jede Zelle verwaltet die Prozessoren, Speicher- und I/O-Einrichtungen auf diesen Knoten so, als ob es sich um ein unabhängiges Betriebssystem handeln würde. Die Zellen arbeiten zusammen, um für Prozesse auf Benutzerebene die Illusion eines einzelnen Systems zu bieten.

[0015] Hive-Zellen sind nicht dafür zuständig, zu entscheiden, wie ihre Ressourcen zwischen lokalen und dezentralen Anforderungen aufgeteilt werden. Jede Zelle ist nur dafür zuständig, ihre internen Ressourcen zu verwalten und die Leistung innerhalb der Ressourcen zu optimieren, die für sie reserviert worden sind. Die globale Ressourcen-Reservierung wird durch einen Benutzerebenen-Prozess mit der Bezeichnung "Wax" ausgeführt. Das Hive-System versucht, eine Datenfälschung zu verhindern, indem gewisse Fehlereindämmungsgrenzen zwischen den Zellen verwendet werden. Um die enge gemeinsame Benutzung zu implementieren, die von einem Mehrprozessorsystem trotz der Fehlereindämmungsgrenzen zwischen den Zellen erwartet wird, wird die gemeinsame Ressourcennutzung durch das Zusammenwirken der verschiedenen Zellenkerne implementiert, doch wird die Richtlinie außerhalb der Kerne in dem Wax-Prozess implementiert. Sowohl Speicher als auch Prozessoren können gemeinsam genutzt werden.

[0016] Ein System mit der Bezeichnung "Cellular IRIX", das von Silicon Graphics Inc., Mountain View, Kalifornien, entwickelt und vermarktet wird, unterstützt modulares Rechnen durch Erweitern herkömmlicher symmetrischer Mehrprozess-Systeme. Die Celluar IRIX-Architektur teilt globalen Kern-Text und -Daten in optimierte Blöcke von SMP-Größe oder "Zellen" auf.

[0017] Die Zellen stellen eine Steuerdomäne dar, die aus einem oder mehreren Maschinenmodulen besteht, wobei jedes Modul aus Prozessoren, Speicher und I/O besteht. Anwendungen, die auf diesen Zellen laufen, stützen sich umfassend auf ein vollständiges Set von lokalen Betriebssystemdiensten, einschließlich lokalen Kopien des Betriebssystemtexts und der Kerndatenstrukturen. Nur eine Instanz des Betriebssystems ist auf dem gesamten System vorhanden. Die Koordinierung zwischen den Zellen ermöglicht es den Anwendungsbildern, die Verarbeitungs-, Speicher- und I/O-Ressourcen von anderen Zellen direkt und transparent zu nutzen, ohne den Overhead von Datenkopien oder zusätzliche Aufgabenumschaltungen zu übernehmen.

[0018] Eine weitere bestehende Architektur mit der Bezeichnung NUMA-Q, die von Sequent Computer Systems, Inc., Beaverton, Oregon, entwickelt und vermarktet wird, verwendet "Quads" bzw. Gruppen von vier Prozessoren pro Speicherabschnitt als den grundlegenden Funktionsbaustein für NUMA_Q SMP-Knoten. Die Erweiterung jedes Quads um I/O verbessert die Leistung zusätzlich. Daher unterteilt die NUMA-Q-Architektur nicht nur physikalischen Speicher, sondern stellt eine vorgegebene Anzahl von Prozessoren und PCI-Slots neben jeden Teil. Der Speicher in jedem Quad ist kein lokaler Speicher im üblichen Sinne. Er ist eher ein Drittel des physikalischen Speicheradressraums und weist einen spezifischen Adressbereich auf. Die Adressabbildung ist gleichmäßig über den Speicher verteilt, wobei jeder Quad einen zusammenhängenden Teil von Adressraum enthält. Es läuft nur eine Kopie des Betriebssystems, und wie in jedem SMP-System ist sie im Speicher resident und führt Prozesse ohne Unterscheidung und gleichzeitig in einem oder mehreren Prozessoren aus.

[0019] Dementsprechend, obwohl viele Versuche unternommen worden sind, ein flexibles Computersystem bereitzustellen, das eine maximale Ressourcenverfügbarkeit und Skalierbarkeit besitzt, weisen die vorhandenen Systeme jeweils beträchtliche Defizite auf. Es wäre daher wünschenswert, eine neue Computersystem-Auslegung zu haben, die eine verbesserte Flexibilität, Ressourcenverfügbarkeit und Skalierbarkeit bereitstellt.

[0020] US-A-5574914 offenbart ein Computersystem, in dem verschiedene Systemressourcen durch einen Verbindungsmechanismus elektrisch verbunden sind.

[0021] WO 97/04388 offenbart ein Computersystem mit einer Vielzahl von Systemressourcen, die Prozessoren, einen Speicher und eine I/O-Schaltung enthalten, wobei das Computersystem umfasst:
einen Verbindungsmechanismus;
einen Software-Mechanismus, der die Systemressourcen in eine Vielzahl von Partitionen unterteilt; und
wenigstens eine Betriebssystem-Instanz, die in einer Vielzahl der Partitionen läuft.

[0022] Gemäß der vorliegenden Erfindung ist ein solches System gekennzeichnet
dadurch, dass der Verbindungsmechanismus die Prozessoren, den Speicher und die I/O-Schaltung elektrisch so verbindet, dass jeder Prozessor elektrischen Zugriff auf den gesamten Speicher und wenigstens einen Teil der I/O-Schaltung hat; und
durch eine Konfigurations-Datenbank, die in dem Speicher gespeichert ist, welche die Partitionen anzeigt, die Teil des Computersystems sind, und welche Informationen enthält, die anzeigen, ob jede Betriebssystem-Instanz aktiv ist.

[0023] Die vorliegende Erfindung stellt des Weiteren ein Verfahren zum Aufbauen eines Computersystems mit einer Vielzahl von Systemressourcen bereit, die Prozessoren, einen Speicher und eine I/O-Schaltung enthalten, wobei das Verfahren die folgenden Schritte umfasst:

(a) elektrisches Verbinden der Prozessoren, des Speichers und der I/O-Schaltung so, dass jeder Prozessor elektrischen Zugang zu dem gesamten Speicher und wenigstens einem Teil der I/O-Schaltung hat;
(b) Unterteilen der Systemressourcen in eine Vielzahl von Partitionen;
(c) Ausführen wenigstens einer Betriebssystem-Instanz in einer Vielzahl der Partitionen; und
d) Erstellen einer Konfigurations-Datenbank, die Informationen dahingehend, welche der Partitionen Teil des Computersystems sind, und Informationen enthält, die anzeigen, ob jede Betriebssystem-Instanz aktiv ist.

[0024] Insbesondere partitioniert die Software logisch und adaptiv CPUs, Speicher und I/O-Ports, indem sie einander zugewiesen werden. Eine Instanz eines Betriebssystems kann dann auf eine Partition geladen werden. Zu unterschiedlichen Zeitpunkten können verschiedene Betriebssystem-Instanzen auf eine bestimmte Partition geladen werden. Diese Partitionierung, die ein System-Manager anweist, ist eine Software-Funktion; es sind keine Hardware-Grenzen erforderlich. Zu jeder einzelner Instanz sind die Ressourcen zugehörig, die sie für eine unabhängige Ausführung benötigt. Ressourcen, wie beispielsweise CPUs und Speicher, können verschiedenen Partitionen dynamisch zugewiesen und von Instanzen des Betriebssystems verwendet werden, die in der Maschine laufen, indem die Konfiguration modifiziert wird. Die Partitionen selbst können ebenfalls ohne Neustart des Systems geändert werden, indem der Konfigurationsbaum modifiziert wird. Das sich daraus ergebende adaptiv partitionierte Mehrprozessor-(APMP) System weist sowohl Skalierbarkeit als auch hohe Leistung auf.

[0025] Die oben genannten und weitere Vorteile der Erfindung lassen sich besser verstehen unter Bezugnahme auf die folgende Beschreibung in Verbindung mit den folgenden begleitenden Zeichnungen:

[0026] Fig. 1 ist ein schematisches Blockschaltbild einer Hardware-Plattform, die mehrere System-Funktionsbausteine darstellt.

[0027] Fig. 2 ist eine schematische Darstellung eines APMP-Computersystems, das in Übereinstimmung mit den Prinzipien der vorliegenden Erfindung aufgebaut ist und mehrere Partitionen zeigt.

[0028] Fig. 3 ist eine schematische Darstellung eines Konfigurationsbaums, der Hardware-Ressourcenkonfigurationen und Software-Konfigurationen und ihre Komponententeile mit Child- und Geschwister-Adressenverweisen (sibling pointers) darstellt.

[0029] Fig. 4 ist eine schematische Darstellung des Konfigurationsbaums, der in Fig. 3 gezeigt und neu angeordnet wurde, um die Zuweisung von Hardware zu Software-Instanzen durch Zugehörigkeits-Adressenverweise (ownership pointer) zu veranschaulichen.

[0030] Fig. 5 ist ein Ablaufdiagramm, das Schritte in einer veranschaulichenden Routine zum Erstellen eines APMP-Computersystems in Übereinstimung mit den Prinzipen der vorliegenden Erfindung skizziert.

[0031] Fig. 6 ist ein Ablaufdiagramm, das die Schritte in einer veranschaulichenden Routine zum Erstellen von Einträgen in einer APMP-Systemverwaltungs-Datenbank darstellt, die Informationen verwaltet, die das APMP-System und seine Konfiguration betreffen.

[0032] Fig. 7A und Fig. 7B bilden, wenn sie zusammengelegt werden, ein Ablaufdiagramm, das im Detail die Schritte in einer veranschaulichenden Routine zum Erstellen eines APMP-Computersystems in Übereinstimmung mit den Prinzipien der vorliegenden Erfindung darstellt.

[0033] Fig. 8A und Fig. 8B bilden, wenn sie zusammengelegt werden, ein Ablaufdiagramm, das die Schritte

in einer veranschaulichenden Routine darstellt, denen eine Betriebssystem-Instanz folgt, um an einem APMP-Computersystem, das bereits erstellt ist, teilzunehmen.

[0034] Eine Computerplattform, die in Übereinstimmung mit den Prinzipien der vorliegenden Erfindung aufgebaut ist, ist ein Mehrprozessorsystem, das partitioniert werden kann, um die gleichzeitige Ausführung mehrerer Instanzen von Betriebssystem-Software zu gestatten. Für das System ist keine Hardware-Unterstützung für die Partitionierung seines Speichers, der CPUs und der I/O-Untersysteme erforderlich, aber einige Hardware lässt sich verwenden, um eine zusätzliche Hardware-Hilfe zum Isolieren von Fehlern und zum Minimieren der Kosten des Software-Engineering bereitzustellen. Die folgende Spezifikation beschreibt die Schnittstellen und Datenstrukturen, die zum Unterstützen der erfinderischen Software-Architektur erforderlich sind. Die beschriebenen Schnittstellen und Datenstrukturen sollen nicht implizieren, dass ein bestimmtes Betriebssystem verwendet werden muss, oder dass nur ein einziger Typ von Betriebssystem eine gleichzeitige Ausführung vornimmt. Jedes Betriebssystem, das die im Folgenden erläuterten Software-Anforderungen implementiert, kann an dem erfinderischen Systembetrieb teilnehmen.

System-Funktionsbausteine

[0035] Die erfinderische Software-Architektur arbeitet auf einer Hardware-Plattform, die mehrere CPUs, einen Speicher und I/O-Hardware integriert. Vorzugsweise wird eine modulare Architektur wie diejenige, die in Fig. 1 gezeigt ist, verwendet, obwohl der Fachmann verstehen wird, dass auch andere Architekturen verwendet werden können, wobei diese Architekturen nicht modular sein müssen. Fig. 1 stellt ein Rechnersystem dar, das aus vier grundlegenden System-Funktionsbausteinen (SBBs) **100**–**106** aufgebaut ist. In der veranschaulichenden Ausführungsform ist jeder Funktionsbaustein, wie beispielsweise Baustein **100**, identisch und umfasst mehrere CPUs **108**–**114**, mehrere Speicher-Slots, (die insgesamt als der Speicher **120** dargestellt werden), einen I/O-Prozessor **118** und einen Port **116**, der einen (nicht gezeigten) Schalter enthält, der das System mit einem anderen solchen System verbinden kann. In anderen Ausführungsformen müssen die Funktionsbausteine jedoch nicht identisch sein. Große Mehrprozessor-Systeme können aufgebaut werden, indem die gewünschte Anzahl von System-Funktionsbausteinen mittels ihrer Ports verbunden werden. Es wird eine Schalter-Technologie statt einer Bus-Technologie zum Verbinden von Funktionsbaustein-Komponenten verwendet, um sowohl die verbesserte Bandbreite zu erhalten als auch nicht-einheitliche Speicher-Architekturen (NUMA) zu gestatten.

[0036] In Übereinstimmung mit den Prinzipien der Erfindung sind die Hardware-Schalter so angeordnet, dass jede CPU alle verfügbaren Speicher und I/O-Ports ansteuern kann, ungeachtet der Anzahl von konfigurierten Funktionsbausteinen, wie dies schematisch durch Leitung **122** dargestellt ist. Des Weiteren können alle CPUs mit irgendeiner oder allen anderen CPUs in allen SBBs mit herkömmlichen Mitteln kommunizieren, wie beispielsweise Interprozessor-Interrupts. Dementsprechend können die CPUs und andere Hardware-Ressourcen nur mit Software verknüpft werden. Eine solche Plattform-Architektur ist inhärent so skalierbar, dass große Mengen von Verarbeitungsleistung, Speicher und I/O in einem einzelnen Computer verfügbar sind.

[0037] Ein APMP-Computersystem **200**, das in Übereinstimmung mit den Prinzipien der vorliegenden Erfindung vom Software-Standpunkt her aufgebaut ist, ist in Fig. 2 dargestellt. In diesem System wurden die Hardware-Komponenten zugewiesen, um die gleichzeitige Ausführung von mehreren Betriebssystem-Instanzen **208**, **210**, **212** zu gestatten.

[0038] In einer bevorzugten Ausführungsform wird diese Reservierung durch ein Software-Programm vorgenommen, das als ein "Konsolenprogramm" bezeichnet wird, weiches, wie hierin im Folgenden im Detail beschrieben wird, beim Hochfahren in den Speicher geladen wird. Konsolenprogramme werden in Fig. 2 schematisch als Programme **213**, **215** und **217** gezeigt. Das Konsolenprogramm kann eine Modifizierung eines bestehenden administrativen Programms oder ein separates Programm sein, das mit einem Betriebssystem zusammenwirkt, um den Betrieb der bevorzugten Ausführungsform zu steuern. Das Konsolenprogramm virtualisiert die Systemressourcen nicht, das heißt, es erstellt keine Software-Schichten zwischen den laufenden Betriebssystemen **208**, **210** und **212** und der physikalischen Hardware, wie beispielsweise dem Speicher und den I/O-Einheiten (in Fig. 2 nicht gezeigt). Auch der Zustand der laufenden Betriebssysteme **208**, **210** und **212** wird nicht gewechselt, um Zugriff auf die gleiche Hardware bereitzustellen. Stattdessen unterteilt das erfinderische System die Hardware logisch in Partitionen. Es liegt in der Zuständigkeit der Betriebssystem-Instanz **208**, **210** und **212**, die Ressourcen entsprechend einzusetzen und eine Koordinierung der Reservierung und gemeinsamen Nutzung der Ressourcen bereitzustellen. Die Hardware-Plattform kann optional eine Hardware-Unterstützung für die Unterteilung der Ressourcen bereitstellen oder kann Fehlersperren bereitstellen, um die Möglichkeit zu minimieren, dass ein Betriebssystem den Speicher beschädigt oder sich auf Einrichtungen auswirkt, die

durch eine andere Betriebssystem-Kopie gesteuert werden.

[0039] Die Ausführungsumgebung für eine einzelne Kopie eines Betriebssystems, wie beispielsweise die Kopie **208**, wird als eine "Partition" **202** bezeichnet, und das ausführende Betriebssystem **208** in der Partition **202** wird als "Instanz" **208** bezeichnet. Jede Betriebssystem-Instanz ist in der Lage, unabhängig von allen anderen Betriebssystem-Instanzen in dem Computersystem zu booten und unabhängig von ihnen zu laufen und kann zusammenwirkend an der gemeinsamen Nutzung von Ressourcen zwischen Betriebssystem-Instanzen teilnehmen, wie im Folgenden beschrieben.

[0040] Zum Ausführen einer Betriebssystem-Instanz muss eine Partition einen Hardware-Neustart-Parameterblock (HWRPB), eine Kopie eines Konsolenprogramms, eine gewisse Speichermenge, eine oder mehrere CPUs und wenigstens einen I/O-Bus umfassen, der einen bestimmten physikalischen Port für die Konsole aufweisen muss. Der HWRPB ist ein Konfigurationsblock, der zwischen dem Konsolenprogramm und dem Betriebssystem übergeben wird.

[0041] Jedes der Konsolenprogramme **213**, **215** und **217** ist an einen Konsolen-Port angeschlossen, der als die entsprechenden Ports **214**, **216** und **218** gezeigt wird. Konsolen-Ports, wie beispielsweise die Ports **214**, **216** und **218** liegen im Allgemeinen in Form eines seriellen Leitungs-Ports oder von angeschlossenen Grafik-, Tastatur- und Maus-Optionen vor. Zum Zweck des erfinderischen Computersystems ist die Fähigkeit, ein spezifisches Betriebssystem oder dazugehörige Eingabeeinrichtungen zu unterstützen, nicht erforderlich, obwohl ein spezifisches Betriebssystem dies erfordern kann. Die grundlegende Voraussetzung ist, dass ein serieller Port für jede Partition ausreichend ist. Während ein separates Endgerät oder eine unabhängige Grafikkonsole verwendet werden könnten, um die von jeder Konsole generierten Informationen anzuzeigen, können vorzugsweise die seriellen Leitungen **220**, **222** und **224** alle mit einem einzelnen Multiplexer **226** verbunden werden, der an eine Workstation, einen PC oder LAT **228** zum Anzeigen von Konsolen-Informationen angeschlossen ist.

[0042] Es ist wichtig zu beachten, dass Partitionen nicht gleichbedeutend mit System-Funktionsbausteinen sind. Zum Beispiel kann die Partition **202** die Hardware in den Funktionsbausteinen **100** und **106** in Fig. 1 umfassen, während die Partitionen **204** und **206** die Hardware in den entsprechenden Funktionsbausteinen **102** und **104** umfassen könnten. Partitionen können auch einen Teil der Hardware in einem Funktionsbaustein enthalten.

[0043] Partitionen können "initialisiert" oder "nichtinitialisiert" sein. Eine initialisierte Partition weist ausreichende Ressourcen auf, um eine Betriebssystem-Instanz auszuführen, besitzt ein geladenes Konsolenprogramm-Bild und eine primäre CPU, die verfügbar und ausführend ist. Eine initialisierte Partition kann unter der Steuerung eines Konsolenprogramms stehen oder kann eine Betriebssystem-Instanz ausführen. In einem initialisierten Zustand weist eine Partition die volle Zugehörigkeit und die Steuerung der Hardware-Komponenten auf, die ihr zugewiesen sind, und nur die Partition selbst kann ihre Komponenten freigeben.

[0044] In Übereinstimmung mit den Prinzipien der Erfindung können Ressourcen von einer initialisierten Partition einer anderen neu zugewiesen werden. Die Neuzuweisung der Ressourcen kann nur durch die initialisierte Partition vorgenommen werden, welcher die Ressource gegenwärtig zugewiesen ist. Wenn sich eine Partition in einem nichtinitialisierten Zustand befindet, können andere Partitionen ihre Hardware-Komponenten neu zuweisen und sie löschen.

[0045] Eine nichtinitialisierte Partition ist eine Partition, die keine primäre CPU besitzt, die entweder unter der Steuerung eines Konsolenprogramms oder eines Betriebssystems ausgeführt wird. Zum Beispiel kann eine Partition beim Hochfahren auf Grund eines Mangels an ausreichenden Ressourcen zum Ausführen einer primären CPU nichtinitialisiert sein, oder wenn ein Systemadministrator das Computersystem neu konfiguriert. Im nichtinitialisierten Zustand kann eine Partition ihre Hardware-Komponenten neu zuweisen und kann von einer anderen Partition gelöscht werden. Nicht zugewiesene Ressourcen können von jeder Partition zugewiesen werden.

[0046] Partitionen können in "Communities" organisiert werden, welche die Basis zum Gruppieren getrennter Ausführungskontexte bereitstellt, um eine zusammenwirkende gemeinsame Ressourcen-Nutzung zu gestatten. Partitionen in der gleichen Community können Ressourcen gemeinsam nutzen. Partitionen, die sich nicht in der gleichen Community befinden, können Ressourcen nicht gemeinsam nutzen. Ressourcen können zwischen den Partitionen, die sich nicht in der gleichen Community befinden, nur manuell durch den Systemadministrator verschoben werden, in dem die Ressourcenzuweisung aufgehoben (und die Nutzung gestoppt)

wird, und die Ressource manuell neu konfiguriert wird. Communities können verwendet werden, um unabhängige Betriebssystem-Domänen zu erstellen oder Benutzerrichtlinien für den Hardware-Einsatz zu implementieren. In Fig. 2 sind die Partitionen **202** und **204** in der Community **230** organisiert worden. Die Partition **206** kann in ihrer eigenen Community **205** liegen. Communities können unter Verwendung des im Folgenden beschriebenen Konfigurationsbaums aufgebaut und durch Hardware durchgesetzt werden.

Das Konsolenprogramm

[0047] Wenn ein Computersystem, das in Übereinstimmung mit den Prinzipien der vorliegenden Erfindung aufgebaut ist, auf einer Plattform aktiviert wird, müssen mehrere HWRPBs erstellt, mehrere Konsolenprogramm-Kopien geladen und Systemressourcen auf eine Weise zugewiesen werden, dass jeder HWRPB zu spezifischen Komponenten des Systems zugeordnet wird. Dazu erstellt das erste auszuführende Konsolenprogramm eine Konfigurationsbaum-Struktur im Speicher, welche die gesamte Hardware im System darstellt. Der Baum enthält auch Software-Partitionierungsinformationen, und die Zuweisungen von Hardware zu Partitionen wird im Folgenden ausführlich erläutert.

[0048] Wenn das APMP-System hochgefahren wird, wird insbesondere eine CPU als eine primäre CPU in einer herkömmlichen Weise von Hardware ausgewählt, die für die Plattform spezifisch ist, auf der das System läuft. Die primäre CPU lädt dann eine Kopie eines Konsolenprogramms in den Speicher. Diese Konsole wird als ein "Master-Konsolen"-Programm bezeichnet. Die primäre CPU arbeitet anfänglich unter der Steuerung des Master-Konsolenprogramms, um den Test- und Prüfvorgang unter der Voraussetzung durchzuführen, dass ein einzelnes System vorhanden ist, zu der die gesamte Maschine zugehörig ist. Anschließend wird ein Set von Umgebungsvariablen geladen, welche die Systempartitionen definieren. Schließlich erstellt die Masterkonsole die Partitionen basierend auf den Umgebungsvariablen. in diesem letzteren Prozess arbeitet die Masterkonsole, um den Konfigurationsbaum zu erstellen, zusätzliche HWRPB-Datenblöcke zu erstellen, zusätzliche Konsolenprogramm-Kopien zu laden und die CPUs auf den alternativen HWRPBs zu starten. Auf jeder Partition läuft dann eine Betriebssystem-Instanz, wobei die Instanz mit einer Konsolenprogramm-Kopie zusammenwirkt, die ebenfalls in der Partition läuft. In einem nichtkonfigurierten APMP-System erstellt das Master-Konsolenprogramm zuerst eine einzelne Partition, welche die primäre CPU, eine Mindest-Speichermenge und eine physikalische Konsole des Systemadministrators enthält, die auf eine plattformspezifische Weise ausgewählt wird. Konsolenprogramm-Befehle gestatten es dem Systemadministrator dann, zusätzliche Partitionen zu erstellen und I/O-Busse, Speicher und CPUs für jede Partition zu konfigurieren.

[0049] Nachdem die Zuordnungen von Ressourcen zu Partitionen durch das Konsolenprogramm vorgenommen worden sind, werden die Zuordnungen in einem nicht-flüchtigen RAM gespeichert, um eine automatische Konfiguration des Systems während anschließender Boot-Vorgänge zu gestatten. Während der anschließenden Boot-Vorgänge muss das Master-Konsolenprogramm die gegenwärtige Konfiguration mit der gespeicherten Konfiguration validieren, um das Entfernen und Hinzufügen von neuen Komponenten zu bearbeiten. Neu hinzugefügte Komponenten werden in einen nicht-zugewiesenen Zustand gesetzt, bis sie durch den Systemadministrator zugewiesen werden. Wenn das Entfernen von Hardware-Komponenten zu einer Partition mit unzureichenden Ressourcen zum Ausführen eines Betriebssystems führt, werden weiterhin Ressourcen zu der Partition hinzugefügt, doch eine Betriebssystem-Instanz kann erst auf ihr laufen, wenn ihr zusätzliche neue Ressourcen zugewiesen werden.

[0050] Wie vorher erwähnt, kommuniziert das Konsolenprogramm mit einer Betriebssystem-Instanz mittels eines HWRPB, der während des Hochfahrens des Betriebssystems an das Betriebssystem übergeben wird. Die grundlegenden Anforderungen für ein Konsolenprogramm sind, dass es in der Lage sein muss, mehrere Kopien von HWRPBs und sich selbst zu erstellen. Jede durch das Konsolenprogramm erstellte HWRPB-Kopie ist in der Lage, eine unabhängige Betriebsystem-Instanz in einen privaten Speicherabschnitt zu booten, und jede auf diese Weise gestartete Betriebssystem-Instanz kann durch einen eindeutigen Wert identifiziert werden, der in den HWRPB gestellt wird. Der Wert gibt die Partition an und wird auch als die Betriebssystem-Instanz-ID verwendet.

[0051] Des Weiteren wird das Konsolenprogramm konfiguriert, um einen Mechanismus zum Entfernen einer CPU aus den verfügbaren CPUs in einer Partition in Reaktion auf eine Anforderung durch ein Betriebssystem bereitzustellen, das in dieser Partition läuft. Jede Betriebssystem-Instanz muss in der Lage sein, auf eine Weise abzuschalten, anzuhalten oder anderweitig abzustürzen, dass die Steuerung an das Konsolenprogramm übergeben wird. Umgekehrt muss jede Betriebssystem-Instanz in der Lange sein, sich unabhängig von jeder anderen Betriebssystem-Instanz erneut in einen betriebsfähigen Modus zu booten.

[0052] Jeder HWRPB, der durch ein Konsolenprogramm erstellt wird, enthält eine slotspezifische CPU-Datenbank für jede CPU, die sich in dem System befindet oder die zu dem System hinzugefügt werden kann, ohne das gesamte System abzuschalten. Jede CPU, die physikalisch vorhanden ist, wird als "present" markiert, doch nur CPUs, die zuerst in einer spezifischen Partition ablaufen, werden in dem HWRPB für die Partition als "available" markiert. Die Betriebssystem-Instanz, die auf einer Partition läuft, ist in der Lage, über ein Bit present (vorhanden) (PP) in Feldern eines Zustands-Flag pro CPU (per-CPU state flag fields) des HWRPB zu erkennen, dass eine CPU zu einem künftigen Zeitpunkt verfügbar sein wird und kann Datenstrukturen aufbauen, um dies wiederzugeben. Wenn es gesetzt ist, zeigt das Bit available (verfügbar) (PA) in den Feldern eines Zustands-Flag pro CPU an, dass die zugehörige CPU gegenwärtig mit der Partition verknüpft ist, und kann eingeladen werden, an dem SMP-Betrieb teilzunehmen.

Der Konfigurationsbaum

[0053] Wie vorher erwähnt, erstellt das Master-Konsolenprogramm einen Konfigurationsbaum, der die Hardware-Konfiguration und die Zuweisung jeder Komponente in dem System zu jeder Partition darstellt. Jedes Konsolenprogramm identifiziert dann den Konfigurationsbaum für seine zugehörige Betriebssystem-Instanz, indem ein Adressenverweis auf den Baum in den HWRPB gestellt wird.

[0054] Unter Bezugnahme auf Fig. 3 stellt der Konfigurationsbaum **300** die Hardware-Komponenten in dem System, die Plattformbedingungen und Mindestvorgaben und die Saftware-Konfiguration dar. Das Master-Konsolenprogramm baut den Baum unter Verwendung von Informationen auf, die durch Prüfen der Hardware erfasst wurden, und von Informationen, die in dem nicht-flüchtigen RAM gespeichert sind, der die Konfigurationsinformationen enthält, die während vorheriger Initialisierungen generiert worden sind.

[0055] Die Master-Konsole kann eine einzelne Kopie des Baums generieren, dessen Kopie von allen Betriebssystem-Instanzen gemeinsam genutzt wird, oder sie kann den Baum für jede Instanz replizieren. Eine einzelne Kopie des Baums weist den Nachteil auf, dass er eine einzelne Fehlerstelle in Systemen mit unabhängigen Speichern erzeugen kann. Plattformen, die mehrere Baum-Kopien generieren, fordern von den Konsolenprogrammen jedoch, dass sie in der Lage sein müssen, Änderungen an dem Baum synchronisiert zu halten.

[0056] Der Konfigurationsbaum umfasst mehrere Knoten, einschließlich Stammknoten, Child-Knoten und Geschwister-Knoten. Jeder Knoten wird aus einem festen Header und einer Erweiterung mit variabler Länge für überlagerte Datenstrukturen gebildet. Der Baum beginnt mit einem Baum-Stammknoten **302**, der die gesamte System-Box darstellt, gefolgt von Zweigen, welche die Hardware-Konfiguration (Hardware-Stammknoten **304**), die Software-Konfiguration (Software-Stammknoten **306**) und die Mindestanforderungen an die Partition (Masken-Stammknoten **308**) beschreiben. In Fig. 3 stellen die Pfeile Child- und Geschwister-Beziehungen dar. Die Children eines Knotens stellen Komponententeile der Hardware- oder Software-Konfiguration dar. Geschwister sind Gleichgestellte einer Komponente, die nicht in Beziehung stehen können, außer dadurch, dass sie den gleichen Stamm haben. Knoten in dem Baum **300** können Informationen über die Software-Communities und Betriebssystem-Instanzen, die Hardware-Konfiguration, Konfigurationsbedingungen, Leistungseinschränkungen und Hot-Swap-Fähigkeiten enthalten. Die Knoten stellen auch die Beziehung von Hardware- zu Software-Zugehörigkeit oder der gemeinsamen Nutzung einer Hardware-Komponente bereit.

[0057] Die Knoten werden zusammenhängend im Speicher gespeichert, und der Adressen-Offset von dem Baum-Stammknoten **320** des Baums **300** zu einem spezifischen Knoten bildet eine "Kennung" (handle), die von jeder Betriebssystem-Instanz verwendet werden kann, um die gleiche Komponente auf jeder Betriebssystem-Instanz eindeutig zu identifizieren. Des Weiteren weist jede Komponente in dem erfinderischen Computersystem eine separate ID auf. Dies kann beispielsweise ein 64-Bit-Wert ohne Vorzeichen sein. Die ID muss eine eindeutige Komponente angeben, wenn sie mit den Typ- und untergeordneten Typ-Werten der Komponente kombiniert wird. Das heißt, die ID muss eine für einen vorgegebenen Komponententyp spezifische Komponente identifizieren. Die ID kann eine einfache Zahl sein, zum Beispiel die CPU-ID, sie kann irgendeine andere eindeutige Verschlüsselung oder eine physikalische Adresse sein. Die Komponenten-ID und die Kennung gestatten es jedem Element des Computersystems, einen bestimmten Teil von Hardware oder Software zu identifizieren. Das heißt, jede Parition muss unter Verwendung eines der Spezifikationsverfahren in der Lage sein, die gleiche Spezifikation zu verwenden und das gleiche Ergebnis zu erzielen.

[0058] Wie oben beschrieben, setzt sich das erfinderische Computersystem aus einer oder mehreren Communities zusammen, die wiederum aus einer oder mehreren Partitionen bestehen. Durch Unterteilen der Partitionen übergreifend über die unabhängigen Communities kann das erfinderische Computersystem in eine

Konfiguration gesetzt werden, in der die gemeinsame Nutzung der Einrichtungen und des Speichers begrenzt werden kann. Die Communities und Partitionen haben IDs, die eng gepackt sind. Die Hardware-Plattform bestimmt die maximale Anzahl von Partitionen basierend auf der Hardware, die in dem System vorhanden ist, sowie auf der Plattform-Obergrenze. Partitions- und Community-IDs überschreiten diesen Wert während der Laufzeit nie. Die IDs werden für gelöschte Partitionen und Communities wiederverwendet. Die maximale Anzahl von Communities ist die gleiche wie die maximale Anzahl von Partitionen. Des Weiteren wird jede Betriebssystem-Instanz durch eine eindeutige Instanz-Kennung identifiziert, beispielsweise eine Kombination der Partitions-ID plus einer Inkarnationsnummer (incarnation number).

[0059] Die Communities und Partitionen werden durch einen Software-Stammknoten **306** dargestellt, der Community-Knoten-Children, (von denen nur der Community-Knoten **310** gezeigt ist), und Partitionsknoten-Grandchildren, (von denen zwei Knoten, **312** und **314**, gezeigt sind), aufweist.

[0060] Die Hardware-Komponenten werden durch einen Hardware-Stammknoten **304** dargestellt, der Children enthält, die eine hierarchische Darstellung der gesamten Hardware zeigen, die gegenwärtig in dem Computersystem vorhanden ist. Die "Zugehörigkeit" einer Handware-Komponente wird durch eine Kennung in dem zugehörigen Hardware-Knoten dargestellt, der auf den entsprechenden Software-Knoten (**310**, **312** oder **314**) verweist. Diese Kennungen sind in Fig. 4 dargestellt und werden im Folgenden ausführlicher erläutert. Komponenten, die zu einer spezifischen Partition zugehörig sind, weisen Kennungen auf, die auf den Knoten verweisen, der die Partition darstellt. Hardware, die von mehreren Partitionen gemeinsam genutzt wird, (zum Beispiel der Speicher), weist Kennungen auf, die auf die Community verweisen, auf welche die gemeinsame Nutzung beschränkt ist. Keiner Zugehörigkeit zugeordnete Hardware weist eine Kennung von Null auf, (die den Baum-Stammknoten **302** darstellt).

[0061] Hardware-Komponenten stellen Konfigurationsbedingungen dahingehend auf, wie eine Zugehörigkeit aufgeteilt werden kann. Eine "config"-Kennung in dem Konfigurationsbaum-Knoten, die mit jeder Komponente verbunden ist, bestimmt, ob die Komponente frei ist, um irgendwo in dem Computersystem durch Verweisen auf den Hardware-Stammknoten **304** zugeordnet zu werden. Einige Hardware-Komponenten können jedoch an einen Ancestor-Knoten gebunden sein und müssen als Teil dieses Knotens konfiguriert werden. Beispiele dafür sind CPUs, die keinerlei Einschränkungen dazu aufweisen können, wo sie ausgeführt werden, die aber einen Komponententeil eines System-Funktionsbausteins (SBB) bilden, wie beispielsweise die SBBs **322** oder **324**. In diesem Fall, auch wenn die CPU ein Child des SBB ist, verweist ihre config-Kennung auf den Hardware-Stammknoten **304**. Ein I/O-Bus kann jedoch nicht zu einer anderen Partition als derjenigen Partition zugehörig sein, zu deren Zugehörigkeit sein I/O-Prozessor gehört. In diesem Fall würde der Konfigurationsbaum-Knoten, der den I/O-Bus darstellt, eine config-Kennung aufweisen, die auf den I/O-Prozessor verweist. Da die Regeln, nach denen sich die Hardware-Konfiguration richtet, plattformspezifisch sind, werden diese Informationen durch die config-Kennung für die Betriebssystem-Instanzen bereitgestellt.

[0062] Jede Hardware-Komponente weist auch eine "Affinitäts"- (affinity) Kennung auf. Die Affinitätskennung ist identisch mit der config-Kennung, außer, dass sie eine Konfiguration darstellt, mit der die beste Leistung der Komponente zu erzielen ist. Zum Beispiel können eine CPU oder ein Speicher eine config-Kennung aufweisen, die es ermöglicht, sie überall in dem Computersystem zu konfigurieren, (sie verweist auf den Hardware-Knoten **304**), doch sollte die CPU bzw. der Speicher so konfiguriert sein, dass sie den System-Funktionsbaustein verwenden, von dem sie ein Teil sind. Das Ergebnis ist, dass der config-Adressenverweis auf den Hardware-Stammknoten **304** verweist, der Affinitäts-Adressenverweis jedoch auf einen SBB-Knoten verweist, wie beispielsweise Knoten **322** oder Knoten **324**. Die Affinität jeder Komponente ist plattformspezifisch und wird durch die Firmware bestimmt. Die Firmware kann diese Informationen verwenden, wenn die Ausbildung "optimaler" automatischer Konfigurationen gewünscht wird.

[0063] Jeder Knoten enthält auch mehrere Flags, die den Typ und Zustand des Knotens anzeigen. Zu diesen Flags gehört ein Flag node_hotswap, das anzeigt, dass die dargestellte Komponente eine "hotswap-fähige" Komponente ist und unabhängig von ihrem Parent- oder ihren Geschwister-Knoten abgeschaltet werden kann. Allerdings müssen alle Children dieses Knotens abgeschaltet werden, wenn sich diese Komponente abschaltet. Wenn sich die Children unabhängig von dieser Komponente abschalten können, muss dieses Bit auch für sie in ihren entsprechenden Knoten gesetzt sein. Ein weiteres Flag ist das Flag node_unavailable, das, wenn es gesetzt ist, anzeigt, dass die durch den Knoten dargestellte Komponente gegenwärtig nicht zur Verwendung zur Verfügung steht. Wenn eine Komponente abgeschaltet wird, (oder niemals eingeschaltet wird), wird sie als nicht verfügbar gekennzeichnet.

[0064] Zwei Flags, node_hardware und node_template, zeigen den Knotentyp an. Weitere Flags, wie bei-

spielsweise node_initialized und node_cpu_primary, können ebenfalls bereitgestellt werden, um anzuzeigen, ob der Knoten eine Partition darstellt, die initialisiert worden ist, oder eine CPU, die gegenwärtig eine primäre CPU ist.

[0065] Der Konfigurationsbaum **300** kann sich auf die Ebene von Einrichtungs-Controllern erstrecken, die es dem Betriebssystem gestatten, Bus- und Einrichtungs-Konfigurationstabellen zu erstellen, ohne die Busse zu prüfen. Der Baum kann jedoch auf jeder Ebene enden, wenn alle darunter liegenden Komponenten nicht unabhängig konfiguriert werden können. System-Software ist noch erforderlich, um Bus- und Einrichtungs-Informationen zu prüfen, die von dem Baum nicht bereitgestellt werden.

[0066] Das Konsolenprogramm implementiert Konfigurationsbedingungen und setzt sie, sofern vorhanden, auf jeder Komponente des Systems durch. Im Allgemeinen können die Komponenten entweder ohne Bedingungen zugewiesen werden, (zum Beispiel weisen CPUs keine Bedingungen auf), oder sie sind nur als Teil einer anderen Komponente konfigurierbar, (ein Einrichtungsadapter ist zum Beispiel nur als ein Teil seines Busses konfigurierbar). Eine Partition, die, wie oben erläutert, eine Gruppierung von CPUs, Speicher und I/O-Einrichtungen zu einer eindeutigen Software-Einheit ist, weist ebenfalls Mindestanforderungen auf. Zum Beispiel sind die Hardware-Mindestanforderungen für eine Partition wenigstens eine CPU, einiger privater Speicher, (plattformabhängiger Mindestwert, einschließlich Konsolenspeicher), und ein I/O-Bus, einschließlich eines physikalischen, nicht gemeinsam genutzten Konsolen-Ports.

[0067] Die Komponenten-Mindestanforderungen für eine Partition werden durch die Informationen bereitgestellt, die in dem Masken-Stammknoten **308** enthalten sind. Der Masken-Stammknoten **308** enthält die Knoten **316**, **318** und **320**, welche die Hardware-Komponenten darstellen, die bereitgestellt werden müssen, um eine Partition zu erstellen, die zur Ausführung eines Konsolenprogramms und einer Betriebssystem-Instanz fähig ist. Konfigurations-Editoren können diese Informationen als die Basis für die Bestimmung verwenden, welche Typen und wie viele Ressourcen verfügbar sein müssen, um eine neue Partition auszubilden.

[0068] Während des Aufbaus einer neuen Partition wird der Masken-Unterbaum "durchgegangen", und für jeden Knoten in dem Masken-Unterbaum muss ein Knoten des gleichen Typs und Untertyps zu der neuen Partition zugehörig sein, so dass sie fähig ist, ein Konsolenprogramm zu laden und eine Betriebssystem-Instanz zu booten. Wenn mehr als ein Knoten des gleichen Typs und Untertyps in dem Masken-Baum vorhanden ist, müssen auch mehrere Knoten in der neuen Partition vorhanden sein. Das Konsolenprogramm verwendet die Masken zum Validieren dessen, dass eine neue Partition die Mindestanforderungen aufweist, bevor versucht wird, ein Konsolenprogramm zu laden und den Betrieb zu initialisieren.

[0069] Folgendes ist ein ausführliches Beispiel einer bestimmten Implementierung von Konfigurationsbaum-Knoten. Es soll nur zu beschreibenden Zwecken dienen und soll nicht einschränkend sein. Jeder HWRPB muss auf einen Konfigurationsbaum verweisen, der die gegenwärtige Konfiguration und die Zuweisungen von Komponenten zu Partitionen bereitstellt. Ein Konfigurations-Adressenverweis (im Feld CONFIG) im HWRPB wird verwendet, um auf den Konfigurationsbaum zu verweisen. Das Feld CONFIG verweist auf einen 64-Byte-Header, der die Größe des Speicher-Pools für den Baum und die ursprüngliche Prüfsumme des Speichers enthält. Unmittelbar nach dem Header befindet sich der Stammknoten des Baums. Der Header und der Stammknoten des Baums sind seitensynchronisiert (page aligned).

[0070] Die Gesamtgröße des für den Konfigurationsbaum reservierten Speichers in Byte befindet sich in dem ersten Quad-Wort des Headers. Es wird sichergestellt, dass die Größe in Vielfachen der Hardware-Seitengröße vorliegt. Das zweite Quad-Wort des Headers ist für eine Prüfsumme reserviert. Um den Konfigurationsbaum prüfen zu können, bildet eine Betriebssystem-Instanz den Baum in ihren lokalen Adressraum ab. Da eine Betriebssystem-Instanz diesen Speicher mit Lesezugriff, der für alle Anwendungen gestattet ist, abbilden kann, müssen einige Vorkehrungen getroffen werden, um eine nicht-privilegierte Anwendung daran zu hindern, Zugriff auf Konsolendaten zu erlangen, auf die sie keinen Zugriff haben soll. Der Zugriff kann eingeschränkt werden, indem der Speicher entsprechend reserviert wird. Zum Beispiel kann der Speicher seitensynchronisiert und in ganzen Seiten reserviert werden. Normalerweise bildet eine Betriebssystem-Instanz die erste Seite des Konfigurationsbaums ab, erhält die Baumgröße und bildet den Speicher, der für die Konfigurationsbaum-Nutzung reserviert ist, nochmals ab. Die Gesamtgröße kann zusätzlichen Speicher umfassen, der von der Konsole für dynamische Änderungen an dem Baum verwendet wird.

[0071] Vorzugsweise werden Konfigurationsbaum-Knoten mit festen Anfangsblöcken gebildet und können optional typspezifische Informationen enthalten, die auf den festen Teil folgen. Das Feld Größe enthält die volle Länge des Knotens, Knoten werden veranschaulichend in Vielfachen von 64 Bytes reserviert und nach Bedarf

aufgefüllt. Die folgende Beschreibung definiert veranschaulichende Felder in dem festen Header für einen Knoten:

```
typedef struct_gct_node {
        unsigned char           type;
        unsigned char           subtype;
        uint16                  size;
        GCT_HANDLE              owner;
        GCT_HANDLE              current_owner;
        GCT_ID                  id;
        union {
            uint64              node_flags:
            struct  {
                unsigned        node_hardware       : 1 ;

        unsigned                node_hotswap        : 1;
            unsigned            node_unavailable    : 1;
            unsigned            node_hw_template    : 1;
            unsigned            node_initialized    : 1;
            unsigned            node_cpu_primary    : 1;

        #define NODE_HARDWARE           0x001
        #define NODE_HOTSWAP            0x002
        #define NODE_UNAVAILABLE        0x004
        #define NODE_HW_TEMPLATE        0x008
        #define NODE_INITIALIZED        0x010
        #define NODE_PRIMARY            0x020

            } flag_bits;

} flag_union;
GCT_HANDLE              config;
GCT_HANDLE              affinity;
GCT_HANDLE              parent;
GCT_HANDLE              next_sib;
GCT_HANDLE              prev_sib;
GCT_HANDLE              child;
GCT_HANDLE              reserved;
uint32                  magic
} GCT_NODE;
```

[0072] In der oben genannten Definition sind die Typ-Definitionen "uint" ganze Zahlen ohne Vorzeichen mit den entsprechenden Bitlängen. Wie vorher erwähnt, werden die Knoten durch eine Kennung aufgefunden und identifiziert, (in der Definition oben identifiziert durch die typedef DCT_HANDLE). Eine veranschaulichende Kennung ist ein 32-Bit-Offset mit Vorzeichen von der Basis des Konfigurationsbaums zu dem Knoten. Der Wert ist über alle Partitionen übergreifend in dem Computersystem eindeutig. Das heißt, eine Kennung, die auf einer Partition erhalten wird, muss zum Suchen eines Knotens oder als eine Eingabe in eine Konsolen-Rückfrage (console callback) auf allen Partitionen gültig sein. Das Feld magic enthält ein vorgegebenes Bit-Muster, das anzeigt, dass der Knoten tatsächlich ein gültiger Knoten ist.

[0073] Der Baum-Stammknoten stellt das gesamte System dar. Seine Kennung ist immer Null. Das heißt, er befindet sich immer an dem ersten physikalischen Speicherplatz in dem Speicher, der für den Konfigurationsbaum nach dem config-Header reserviert ist. Er weist die folgende Definition auf:

```
typedef struct_gct_root_node {
        GCT_NODE                hd:
        uint64                  lock;
        uint64                  transient_level;
        uint64                  current_level;
        uint64                  console_req;
        uint64                  min_alloc;
        uint64                  min_align;
        uint64                  base_alloc;
        uint64                  base_align;
        uint64                  max_phys_address;
        uint64                  mem_size;
        uint64                  platform_type;
        int32                   platform_name;
        GCT_HANDLE              primary_instance;

        GCT_HANDLE              first_free;
        GCT_HANDLE              high_limit;
        GCT_HANDLE              lookaside;
        GCT_HANDLE              available;
        uint32                  max_partition;
        int32                   partitions;
        int32                   communities;
        uint32                  max_platform_partition;
        uint32                  max_fragments;
        uint32                  max_desc;
        char            APMP_id[16];
        char            APMP_id_pad[4];
        int32                   bindings;
}       GCT_ROOT_NODE;
```

[0074] Die Felder in dem Stammknoten werden wie folgt definiert:

lock

[0075] Dieses Feld wird als eine einfache Sperre durch Software verwendet, die Änderungen an der Struktur des Baums und der Software-Konfiguration verhindern möchte. Wenn dieser Wert –1 ist, (alle Bits eingeschaltet), ist der Baum nicht gesperrt; wenn der Wert ≥ 0 ist, ist der Baum gesperrt. Dieses Feld wird unter Verwendung von unterbrechungsfreien Operationen modifiziert. Die Aufrufeinrichtung (caller) der Sperr-Routine übergibt eine Partitions-ID, die in das Feld Sperre geschrieben wird. Dies kann zur Unterstützung bei der Fehlerverfolgung und Wiederherstellung bei Abstürzen verwendet werden.

transient_level

[0076] Dieses Feld wird beim Start einer Baum-Aktualisierung inkrementiert.

current_level

[0077] Dieses Feld wird bei Beendigung einer Baum-Aktualisierung inkrementiert.

console_req

[0078] Dieses Feld gibt den Speicher in Bytes an, der für die Konsole im Grund-Speichersegment einer Partition erforderlich ist.

min_alloc

[0079] Dieses Feld enthält die Mindestgröße eines Speicherfragments und die Reservierungseinheit (die Fragmentgröße muss ein Vielfaches der Reservierung betragen). Es muss eine Potenz von 2 sein.

min_align

[0080] Dieses Feld enthält die Synchronisierungsanforderungen für ein Speicherfragment. Es muss eine Potenz von 2 sein.

base_alloc

[0081] Dieses Feld gibt den Mindestspeicher in Bytes an (einschließlich console_req), der für das Grund-Speichersegment für eine Partition benötigt wird. Dort werden die Konsole, Konsolenstrukturen und das Betriebssystem für eine Partition geladen. Er muss größer oder gleich minAlloc und ein Vielfaches von minAlloc sein.

base_align

[0082] Dieses Feld enthält die Synchronisierungsanforderungen für das Grund-Speichersegment einer Partition. Es muss eine Potenz von 2 sein und eine Synchronisierung von mindestens min-align sein.

max_phys_address

[0083] Dieses Feld enthält die berechnete größte physikalische Adresse, die auf dem System vorhanden sein kann, einschließlich Speicher-Subsystemen, die gegenwärtig nicht eingeschaltet und verfügbar sind.

mem_size

[0084] Dieses Feld enthält den gegenwärtigen Gesamtspeicher im System.

platform_type

[0085] Dieses Feld speichert den Plattformtyp, der aus einem Feld im HWRPB entnommen wurde.

platform_name

[0086] Dieses Feld enthält einen ganzzahligen Offset von der Basis des Baum-Stammknotens zu einer Zeichenfolge, die den Namen der Plattform darstellt.

primary_instance

[0087] Dieses Feld speichert die Partitions-ID der ersten Betriebssystem-Instanz.

first_free

[0088] Dieses Feld enthält den Offset von dem Baum-Stammknoten zu dem ersten freien Byte des Speicher-Pools, das für neue Knoten verwendet wird.

high_limit

[0089] Dieses Feld enthält die höchste Adresse, auf der ein gültiger Knoten in dem Konfigurationsbaum angeordnet werden kann. Es wird für Rückfragen verwendet, um zu validieren, dass eine Kennung gültig ist.

lookaside

[0090] Dieses Feld ist die Kennung einer verknüpften Liste von Knoten, die gelöscht worden sind und die zurückgefordert werden können. Wenn eine Community oder Partition gelöscht werden, wird der Knoten in diese Liste verknüpft, und bei der Erstellung einer neuen Partition oder Community wird diese Liste durchsucht, bevor aus freiem Speicher-Pool reserviert wird.

available

[0091] Dieses Feld enthält die Anzahl von in dem freien Speicher-Pool übrigen Bytes, auf die durch das Feld first_free verwiesen wird.

max_partitions

[0092] Dieses Feld enthält die maximale Anzahl von Partitionen, die durch die Plattform basierend auf der Menge gegenwärtig verfügbarer Hardware-Ressourcen berechnet worden sind.

partitions

[0093] Dieses Feld enthält einen Offset von der Basis des Stammknotens zu einem Array von Kennungen. Jede Partitions-ID wird als ein Indexverweis in dieses Array verwendet, und die Partitionsknotenkennung wird an der indexierten Speicherstelle gespeichert. Wenn eine neue Partition erstellt wird, wird dieses Array geprüft, um die erste Partitions-ID zu finden, die keine entsprechende Partitionsknotenkennung besitzt, und diese Partitions-ID wird als die ID für die neue Partition verwendet.

communities

[0094] Dieses Feld enthält ebenfalls einen Offset von der Basis des Stammknotens zu einem Array von Kennungen. Jede Community-ID wird als ein Indexverweis in dieses Array verwendet, und eine Community-Knotenkennung wird in dem Array gespeichert. Wenn eine neue Community erstellt wird, wird dieses Array geprüft, um die erste Community-ID zu finden, die keine entsprechende Community-Knotenkennung besitzt, und diese Community-ID wird als die ID für die neue Community verwendet. Es kann nicht mehr Communities als Partitionen geben, so dass die Array-Größe basierend auf der maximalen Anzahl von Partitionen bemessen wird.

max_platform_partition

[0095] Dieses Feld enthält die maximale Anzahl von Partitionen, die gleichzeitig auf der Plattform vorhanden sein können, selbst wenn zusätzliche Hardware hinzugefügt wird (potenziell inswapped).

max_fragments

[0096] Dieses Feld enthält eine plattformdefinierte maximale Anzahl von Fragmenten, in die ein Speicher-Deskriptor unterteilt werden kann. Es wird verwendet zur Bemessung der Größe des Arrays von Fragmenten in dem Speicher-Deskriptor-Knoten verwendet.

max_desc

[0097] Dieses Feld enthält die maximale Anzahl von Speicher-Deskriptoren für die Plattform.

APMP_id

[0098] Dieses Feld enthält eine System-ID, die durch die System-Software eingerichtet und in einem nichtflüchtigen RAM gespeichert wird.

APMP_id_pad

[0099] Dieses Feld enthält Auffüll-Bytes für die APMP-ID.

bindings

[0100] Dieses Feld enthält einen Offset zu einem Array von "Bindungen". Jeder Bindungseintrag beschreibt einen Typ von Hardware-Knoten, den Knotentyp, welcher der Parent-Knoten sein muss, die Konfigurationsbindung und die Affinitätsbindung für einen Knotentyp. Bindungen werden von der Software verwendet, um zu bestimmen, welche Knotentypen in Beziehung stehen, und Konfigurations- und Affinitätsregeln.

[0101] Eine Community stellt die Basis für die gemeinsame Nutzung von Ressourcen unter Partitionen bereit.

[0102] Während eine Hardware-Komponente jeder Partition in einer Community zugewiesen werden kann, erfolgt die tatsächliche gemeinsame Nutzung einer Einrichtung, wie beispielsweise des Speichers, nur innerhalb einer Community. Der Community-Knoten **310** enthält einen Adressenverweis zu einem Steuerabschnitt, der als eine APMP-Datenbank bezeichnet wird, der es den Betriebssystem-Instanzen gestattet, den Zugriff und die Mitgliedschaft in der Community zum Zweck der gemeinsamen Nutzung des Speichers und von Kommunikationen zwischen den Instanzen zu steuern. Die APMP-Datenbank und die Erstellung von Communities werden im Folgenden ausführlich erläutert. Die Konfigurations-ID für die Community ist ein ganzzahliger 16-Bit-Wert mit Vorzeichen, der durch das Konsolenprogramm zugewiesen wird. Der ID-Wert ist niemals größer als die maximale Anzahl von Partitionen, die auf der Plattform erstellt werden können.

[0103] Ein Partitionsknoten, wie beispielsweise der Knoten **312** oder **314**, stellt eine Sammlung von Hardware dar, die in der Lage ist, eine unabhängige Kopie des Konsolenprogramms und eine unabhängige Kopie eines Betriebssystems auszuführen. Die Konfigurations-ID für diesen Knoten ist ein ganzzahliger 16-Bit-Wert mit Vorzeichen, der auf der Plattform erstellt werden kann. Der Knoten weist die folgende Definition auf:

```
typedef struct_gct_partition_node {
        GCT_NODE                hd;
        uint64                  hwrpb;
        uint64                  incarnation;
        uint64                  priority;
        int32                   os_type;
        uint32                  partition_reserved_1;

        uint64                  instance_name_format;
        char                    instance_name[128];
} GCT_PARTITION_NODE;
```

[0104] Die definierten Felder haben die folgenden Definitionen:

hwrpb

[0105] Dieses Feld enthält die physikalische Adresse des Hardware-Neustart-Parameterblocks für diese Partition. Zum Minimieren von Änderungen an dem HWRPB enthält der HWRPB keinen Adressenverweis auf die Partition oder die Partitions-ID. Stattdessen enthalten die Partitionsknoten einen Adressenverweis auf den HWRPB. Die System-Software kann dann die Partitions-ID der Partition bestimmen, in der sie läuft, indem die Partitionsknoten nach der Partition durchsucht werden, welche die physikalische Adresse ihres HWRPB enthält.

incarnation

[0106] Dieses Feld enthält einen Wert, der jedes Mal inkrementiert wird, wenn die primäre CPU der Partition einen Boot- oder Neustart-Vorgang auf der Partition ausführt.

priority

[0107] Dieses Feld enthält eine Partitions-Priorität.

os_type

[0108] Dieses Feld enthält einen Wert, der den Typ des Betriebssystems angibt, das in die Partition geladen wird.

partition_reserved_1

[0109] Dieses Feld ist für künftige Verwendung reserviert.

instance_name_format

[0110] Dieses Feld enthält einen Wert, der das Format der Instanznamen-Zeichenfolge beschreibt.

instance_name

[0111] Dieses Feld enthält eine formatierte Zeichenfolge, die unter Verwendung des Felds instance_name_format interpretiert wird. Der Wert in diesem Feld stellt einen Pfadnamen auf höchster Ebene für die Betriebssystem-Instanz bereit, welche in der Partition ausgeführt wird. Dieses Feld wird durch die System-Software geladen und wird nicht leistungszyklusübergreifend gespeichert. Dieses Feld wird beim Einschalten und Erstellen und Löschen von Partitionen gelöscht.

[0112] Ein System-Funktionsbaustein-Knoten, wie beispielsweise der Knoten **322** oder **324**, stellt ein frei wählbares Hardware-Teil oder eine konzeptionelle Gruppierung dar, die von System-Plattformen mit modularen Auslegungen verwendet werden, wie derjenigen, die in Fig. 2 dargestellt ist. Ein QBB (Quad-Funktionsbaustein) ist ein spezifisches Beispiel eines SBB und entspricht Einheiten, wie beispielsweise den Einheiten **100**, **102**, **104** und **106** in Fig. 1. Children der SBB-Knoten **322** und **324** umfassen die Eingabe/Ausgabe-Prozessorknoten **326** und **340**.

[0113] Von CPU-Knoten, wie beispielsweise den Knoten **328**–**332** und **342**–**346**, wird angenommen, dass sie als eine primäre CPU für den SMP-Betrieb arbeiten können. In dem seltenen Fall, in dem eine CPU nicht primärfähig ist, besitzt sie einen SUBTYPE-Code, der angibt, dass sie nicht als eine primäre CPU im SMP-Betrieb verwendet werden kann. Diese Information ist entscheidend, wenn Ressourcen zum Erstellen einer neuen Partition konfiguriert werden. Der CPU-Knoten enthält auch Informationen darüber, wo die CPU gegenwärtig ausgeführt wird. Für die Primär-CPU für eine Partition ist das Flag NODE_CPU_PRIMARY in dem Feld NODE_FLAGS gesetzt. Der CPU-Knoten weist die folgende Definition auf:

```
typedef struct_gct_cpu_node {
        GCT_NODE                hd;
} GCT_CPU_NODE;
```

[0114] Ein Speicher-Subsystemknoten, wie beispielsweise der Knoten **334** oder **348**, ist ein "Pseudo"-Knoten, der Knoten zusammengruppiert, welche die physikalischen Speicher-Controller und die Zuweisungen des Speichers darstellen, die von den Controllern bereitgestellt werden. Die Children dieses Knotens bestehen aus einem oder mehreren Speicher-Controller-Knoten, (wie beispielsweise den Knoten **336** und **350**), welche die Konsole so konfiguriert hat, das sie (verschachtelt) zusammenarbeiten, und einem oder mehreren Speicher-Deskriptor-Knoten, (wie beispielsweise den Knoten **338** und **352**), die physikalisch zusammenhängende Speicherbereiche beschreiben.

[0115] Ein Speicher-Controller-Knoten, (wie beispielsweise die Knoten **336** oder **350**), wird verwendet, um eine physikalische Hardware-Komponente auszudrücken, und sie sind typischerweise zu der Partition zugehörig, die Fehler und die Initialisierung bearbeitet. Speicher-Controller können nicht zu Communities zugewiesen werden, da sie eine spezifische Betriebssystem-Instanz für Initialisierung, Prüfung und Fehler benötigen. Eine Speicherbeschreibung, die durch einen Speicher-Deskriptor-Knoten definiert wird, kann jedoch in "Fragmente" aufgeteilt werden, um es verschiedenen Partitionen oder Communities zu ermöglichen, dass spezifische Speicherbereiche in dem Speicher-Deskriptor zu ihnen zugehörig sind. Der Speicher unterscheidet sich dadurch von anderen Hardware-Ressourcen, dass er gleichzeitig gemeinsam genutzt oder in "private" Bereiche aufgeteilt werden kann. Jeder Speicher-Deskriptor-Knoten enthält eine Liste von Untergruppen-Bereichen, die es gestatten, den Speicher unter den Partitionen aufzuteilen sowie gemeinsam von Partitionen, (die zu einer Community zugehörig sind), nutzen zu lassen. Ein Speicher-Deskriptor-Knoten, (wie beispielsweise die Knoten **338** oder **352**), werden definiert als:

```
typedef struct_gct_mem_desc_node {
        GCT_NODE                hd;
        GCT_MEM_INFO            mem_info;
        int32                   mem_frag;
} GCT_MEM_DESC_NODE;
```

[0116] Die Struktur mem_info besitzt die folgende Definition:

```
typedef struct_gct_mem_info {
        unint64                 base_pa;
        unint64                 base_size;
        unint32                 desc_count;
        unint32                 info_fill;
} GCT_MEM_INFO;
```

[0117] Das Feld mem_frag enthält einen Offset von der Basis des Speicher-Deskriptor-Knotens zu einem Array der GCT_MEM_DESC-Strukturen, welche die folgende Definition haben:

```
typedef struct_gct_mem_desc {
        unint64                 pa;
        unint64                 size;
        GCT_HANDLE                      mem_owner;

        GCT_HANDLE                      mem_current_owner;
        union {
             unint32            mem_flags;
             struct {
                    unsigned            mem_console     : 1;
                    unsigned            mem_private     : 1;
                    unsigned            mem_shared      : 1;
                    unsigned            base            : 1;

        #define CGT_MEM_CONSOLE         0x1
        #define CGT_MEM_PRIVATE         0x2
        #define CGT_MEM_SHARED          0x4
        #define CGT_MEM_CONSOLE         0x8

            } flag_bits;
        }   flag_union;
        uint32                  mem_fill;
} GCT_MEM_DESC;
```

[0118] Die Anzahl von Fragmenten in einem Speicher-Deskriptor-Knoten, (Knoten **338** oder **352**), wird durch die Plattform-Firmware begrenzt. Dies erzeugt eine Obergrenze hinsichtlich der Speicherunterteilung und schränkt die unbegrenzte Erweiterung des Konfigurationsbaums ein. Die Software kann die maximale Anzahl von Fragmenten aus dem Feld max_fragments in dem Baum-Stammknoten **302** bestimmen (oben erläutert), oder durch Aufrufen einer entsprechenden Konsolen-Rückfragefunktion zum Zurückgeben des Werts. Jedes Fragment kann jeder Partition zugewiesen werden, vorausgesetzt, die config-Bindung und die Zugehörigkeit des Speicher-Deskriptors und des Speicher-Subsystems lassen dies zu. Jedes Fragment enthält ein Feld für grundlegende physikalische Adresse, Größe und Zugehörigkeit sowie Flags, die den Typ der Verwendung angeben.

[0119] Um den Zugriff zur gemeinsamen Speichernutzung zu gestatten, müssen der Speicher-Subsystem-Parent-Knoten und der Speicher-Deskriptor-Knoten zu einer Community zugehörig sein. Die Fragmente in dem Speicher-Deskriptor können zu der Community (gemeinsame Nutzung) oder zu jeder Partition in der Community zugehörig sein.

[0120] Fragmente können Mindest-Reservierungsgrößen und in dem Baum-Stammknoten **302** bereitgestellte Synchronisierungen (alignments) besitzen. Der grundlegende Speicher für eine Partition, (die Fragmente, in denen die Konsole und das Betriebssystem geladen werden), können eine größere Reservierung und Synchronisierung als andere Fragmente aufweisen, (siehe Definition des Baum-Stammknotens oben). Wenn das Feld Zugehörigkeit des Speicher-Deskriptor-Knotens eine Partition ist, dann können die Fragmente nur zu dieser Partition zugehörig sein.

[0121] Fig. 4 veranschaulicht den Konfigurationsbaum, der in Fig. 3 gezeigt ist, wenn er von der Zugehörigkeits-Perspektive aus betrachtet wird. Das Konsolenprogramm für eine Partition überlässt Zugehörigkeit und

Steuerung der Partitions-Ressourcen der Betriebssystem-Instanz, die in dieser Partition läuft, wenn die primäre CPU für diese Partition mit der Ausführung beginnt. Das "Zugehörigkeits"-Konzept bestimmt, wie die Hardware-Ressourcen und CPUs zu Software-Partitionen und Communities zugewiesen werden. Der Konfigurationsbaum weist Zugehörigkeits-Adressenverweise auf, die in Fig. 3 dargestellt sind, welche die Abbildung von Hardware-Einrichtungen auf die Software bestimmen, wie beispielsweise Partitionen (ausschließlich Zugriff) und Communities (gemeinsamer Zugriff). Eine Betriebssystem-Instanz verwendet die Informationen in dem Konfigurationsbaum, um zu bestimmen, über welche Hardware-Ressourcen sie Zugriffs- und Neukonfigurations-Kontrolle besitzt.

[0122] Passive Hardware-Ressourcen, die keine Zugehörigkeit aufweisen, stehen für den Einsatz erst zur Verfügung, wenn die Zugehörigkeit eingerichtet worden ist. Sobald die Zugehörigkeit durch Verändern des Konfigurationsbaums eingerichtet worden ist, können die Betriebssystem-Instanzen beginnen, die Ressourcen zu verwenden. Wenn eine Instanz eine Anfangsanforderung stellt, kann die Zugehörigkeit geändert werden, indem das Betriebssystem mit der Zugehörigkeit veranlasst wird, die Verwendung einer Ressource zu stoppen, oder indem ein Konsolenprogramm Maßnahmen ergreift, um die Verwendung einer Ressource in einer Partition zu stoppen, in der keine Betriebssystem-Instanz ausgeführt wird. Der Konfigurationsbaum wird dann geändert, um die Zugehörigkeit der Ressource auf eine andere Betriebssystem-Instanz zu übertragen. Der erforderliche Vorgang, um ein Betriebssystem zu veranlassen, die Verwendung einer Hardware-Ressource zu stoppen, ist betriebssystemspezifisch und kann einen Neustart der Betriebssystem-Instanzen erfordern, die von der Änderung betroffen sind.

[0123] Zum Verwalten des Übergangs einer Ressource von einem zugehörigen und aktiven Zustand in einen nicht zugehörigen und inaktiven Zustand sind in jedem Knoten des Baums zwei Felder vorgesehen. Das Feld owner stellt die Zugehörigkeit einer Ressource dar und wird mit der Kennung Software-Partition oder Community mit der Zugehörigkeit geladen. Beim Hochfahren eines APMP-Systems werden die Felder owner der Hardware-Knoten aus den Inhalten des nicht-flüchtigen RAM geladen, um eine Anfangskonfiguration einzurichten.

[0124] Zum Ändern der Zugehörigkeit einer Ressource wird der Kennungswert in dem Feld owner der Hardware-Komponente und in den Feldern owner von irgendwelchen Abkömmlingen der Hardware-Komponente modifiziert, die durch ihre config-Kennungen an die Komponente gebunden sind. Das Feld current_owner stellt den gegenwärtigen Benutzer der Ressource dar. Wenn die Felder owner und current_owner den gleichen Nicht-Null-Wert enthalten, ist die Ressource zugehörig und ist aktiv. Nur über die Zugehörigkeit einer Ressource kann die Zuweisung der Ressource aufgehoben werden, (das Feld owner auf Null setzen). Eine Ressource, deren Felder owner und current_owner Null sind, befindet ist nicht zugehörig und ist inaktiv. Nur Ressourcen, deren Felder owner und current_owner Null sind, können einer neuen Partition oder Community zugewiesen werden.

[0125] Wenn die Zuweisung einer Ressource aufgehoben wird, kann über die Zugehörigkeit entschieden werden, die Zuweisung des Felds owner oder beider Felder, owner und current_owner, aufzuheben. Die Entscheidung basiert auf der Fähigkeit der besitzenden Betriebssystem-Instanz, die in der Partition läuft, die Verwendung der Ressource vor der Aufhebung der Zugehörigkeits-Zuweisung zu unterbrechen. In dem Fall, indem ein Neustart erforderlich ist, um die Zugehörigkeit aufzugeben, wird das Feld owner gelöscht, doch das Feld current_owner bleibt unverändert. Wenn die besitzende Betriebssystem-Instanz neu startet, kann das Konsolenprogramm alle Felder current_owner für Ressourcen löschen, die während der Initialisierung keine Zugehörigkeit aufgewiesen haben.

[0126] Während der Initialisierung modifiziert das Konsolenprogramm das Feld current_owner, um das Feld owner für jeden Knoten abzugleichen, der zu ihm zugehörig ist und für den das Feld current_owner Null ist. Die System-Software sollte nur Hardware verwenden, die gegenwärtig zu ihr zugehörig ist. Im Fall einer Zuweisungsaufhebung einer Ressource, die zu einer Community zugehörig ist, liegt es in der Zuständigkeit der System-Software, den Übergang zwischen den Zuständen zu verwalten. In einigen Ausführungsformen kann eine Ressource an eine andere Partition ausgeliehen werden. In diesem Fall sind die Felder owner und current_owner beide gültig, aber nicht gleich. Die folgende Tabelle fasst die möglichen Ressourcen-Zustände und die Werte der Felder owner und current_owner zusammen:

TABELLE 1

Wert von Feld **owner**	Wert von Feld **current_owner**	Ressourcen-Status
keiner	keiner	nicht zugehörig und inaktiv
keiner	gültig	nicht zugehörig, aber noch aktiv
gültig	keiner	zugehörig, noch nicht aktiv
gültig	gleich mit Zugehörigkeit	zugehörig und aktiv
gültig	ist nicht gleich mit Zugehörigkeit	ausgeliehen

[0127] Da CPUs aktive Einrichtungen sind und die gemeinsame Nutzung von CPUs bedeutet, dass eine CPU im Kontext einer Partition ausgeführt werden könnte, zu der sie eventuell nicht "zugehörig" ist, unterscheidet sich die Zugehörigkeit einer CPU von der Zugehörigkeit einer passiven Ressource. Der CPU-Knoten in dem Konfigurationsbaum stellt zwei Felder bereit, die angeben, zu welcher Partition eine CPU nominell zugehörig ist, und in welcher Partition die CPU gegenwärtig ausgeführt wird. Das Feld owner enthält einen Wert, der die nominelle Zugehörigkeit der CPU bzw. insbesondere der Partition angibt, in der die CPU beim Hochfahren des Systems ausgeführt wird.

[0128] Bis ein anfängliche Zugehörigkeit eingerichtet ist, (das heißt, wenn das Feld owner nicht zugewiesen ist), werden die CPUs in einen HWRPB-Kontext gestellt, der von der Master-Konsole entschieden wurde, doch wird das HWRPB-Bit available für die CPU noch in keinem HWRPB gesetzt. Diese Kombination verhindert, dass die CPU an irgendeiner Betriebssystem-Instanz im SMP-Betrieb teilnimmt. Wenn die Zugehörigkeit einer CPU eingerichtet ist, (das Feld owner ist mit einer gültigen Partitionskennung gefüllt), migriert die CPU, falls erforderlich, zu der zugehörigen Partition, setzt das Bit available in dem HWRPB, der mit dieser Partition verbunden ist, und fordert die Teilnahme an dem SMP-Betrieb der Instanz an, die in dieser Partition läuft, oder nimmt an dem Konsolenprogramm im SMP-Modus teil. Die Kombination der Bits present und available in dem HWRPB melden der Betriebssystem-Instanz, dass die CPU für den Einsatz im SMP-Betrieb zur Verfügung steht, und dass die Betriebssystem-Instanz diese Bits zum Aufbauen entsprechender Datenstrukturen pro CPU und zum Senden einer Nachricht an die CPU verwenden kann, um sie aufzufordern, an dem SMP-Betrieb teilzunehmen.

[0129] Wenn eine CPU das Bit available in einem HWRPB setzt, trägt sie auch einen Wert in das Feld current_owner ihres entsprechenden CPU-Knotens in dem Konfigurationsbaum ein. Das Feld current_owner ist die Kennung der Partition, in der die CPU das HWRPB-Bit active gesetzt hat und an dem SMP-Betrieb teilnehmen kann. Das Feld current_owner für eine CPU wird nur durch das Konsolenprogramm gesetzt. Wenn eine CPU von einer Partition zu einer anderen Partition migriert oder in einem nicht-zugewiesenen Zustand angehalten wird, wird das Feld current_owner gelöscht, (oder in den Kennungswert der neuen Partition geändert), zum gleichen Zeitpunkt, zu dem das Bit available in dem HWRPB gelöscht wird. In das Feld current_owner sollten keine direkten Einträge durch die System-Software vorgenommen werden, und es gibt nur wieder, welcher HWRPB das Bit available für die CPU gesetzt hat.

[0130] Während der Laufzeit kann eine Betriebssystem-Instanz eine CPU temporär an eine andere Partition "ausleihen", ohne die nominelle Zugehörigkeit der CPU zu ändern. Das herkömmliche Zugehörigkeitskonzept unter Verwendung der HWRPB-Bits present und available wird verwendet, um den gegenwärtigen Ausführungskontext der CPU durch Modifizieren des HWRPB und des Konfigurationsbaums in unterbrechungsfreien Operationen wiederzugeben. Das Feld current_owner kann des Weiteren durch die System-Software in einer der Partitionen verwendet werden, um zu bestimmen, in welcher Partition die CPU gegenwärtig ausgeführt wird, (andere Instanzen können die Speicherstelle einer bestimmten CPU bestimmen, indem der Konfigurationsbaum geprüft wird).

[0131] Es ist auch möglich, die Zuweisung einer CPU aufzuheben und in einen Zustand zurückzusetzen, in dem das Bit available in keinem HWRPB gesetzt ist, und das Feld current_owner in dem Konfigurationsbaum-Knoten für die CPU wird gelöscht. Dies wird erreicht, indem die Ausführung der CPU angehalten und das Konsolenprogramm veranlasst wird, das Feld owner in dem Konfigurationsbaum-Knoten sowie das Feld current_owner und das HWRPB-Bit available zu löschen. Die CPU wird dann im Konsololenmodus ausgeführt und fragt das Feld owner ab, das darauf wartet, dass eine gültige Partitionskennung eingetragen wird. Die Sys-

tem-Software kann dann eine neue Zugehörigkeit einrichten und die CPU die Ausführung in der neuen Partition beginnen.

[0132] Veranschaulichende Zugehörigkeits-Adressenverweise werden in Fig. 4 durch Pfeile dargestellt. Jedem der Knoten in Fig. 4, der einem ähnlichen Knoten in Fig. 3 entspricht, wird eine entsprechende Nummer zugeteilt. Zum Beispiel wird der Software-Stammknoten, der in Fig. 3 als Knoten **306** bezeichnet wird, in Fig. 4 als Knoten **406** bezeichnet. Wie in Fig. 4 gezeigt, ist die Community **410** zur "Zugehörigkeit" des Software-Stammknotens **406** zugehörig. In ähnlicher Weise sind die System-Funktionsbausteine 1 und 2 (**422** und **425**) zur Zugehörigkeit der Community **410** zugehörig. In ähnlicher Weise sind die Partitionen **412** und **414** ebenfalls zur Zugehörigkeit der Community **410** zugehörig.

[0133] Zur Partition **412** sind die CPUs **428**–**432** und der I/O-Prozessor **426** zugehörig. Der Speicher-Controller **436** ist ebenfalls ein Teil der Partition 1 (**412**). In ähnlicher Weise sind zur Partition 2 (**414**) die CPUs **442**–**446**, der I/O-Prozessor **440** und der Speicher-Controller **450** zugehörig.

[0134] Der gemeinsame oder gemeinsam genutzte Speicher in dem System besteht aus den Speicher-Subsystemen **434** und **448** und den Speicher-Deskriptoren **438** und **452**. Diese sind zu der Community **410** zugehörig. Daher beschreibt Fig. 4 den Aufbau des Systems, wie er sich den Betriebssystem-Instanzen darstellen würde.

Betriebssystem-Charaktenstiken

[0135] Wie vorher erwähnt, kann das veranschaulichende Computersystem mit mehreren verschiedenen Betriebssystemen in verschiedenen Partitionen arbeiten. Herkömmliche Betriebssysteme müssen jedoch unter Umständen in einigen Gesichtspunkten modifiziert werden, um sie mit dem erfinderischen System kompatibel zu machen, je nachdem, wie das System konfiguriert ist. Einige Beispiel-Modifizierungen für die veranschaulichende Ausführungsform sind im Folgenden aufgelistet:

1. Instanzen müssen unter Umständen modifiziert werden, um einen Mechanismus zum Auswählen einer "primären" CPU in der Partition zu enthalten, um die Konsole zu betreiben und ein Ziel für die Kommunikation von anderen Instanzen zu sein. Die Auswahl einer primären CPU kann in einer herkömmlichen Weise unter Verwendung von Entscheidungsmechanismen oder anderen herkömmlichen Einrichtungen erfolgen.
2. Für jede Instanz können Modifizierungen erforderlich sein, die es ihr gestatten, mit dem Konsolenprogramm zu kommunizieren und zusammenzuwirken, das für die Erstellung eines Konfigurations-Datenblocks zuständig ist, der die Ressourcen beschreibt, die der Partition zur Verfügung stehen, in welcher die Partition läuft. Zum Beispiel sollte die Instanz nicht die darunter liegende Hardware prüfen, um zu bestimmen, welche Ressourcen zur Verwendung durch die Instanz verfügbar sind. Stattdessen muss sie, wenn ihr ein Konfigurations-Datenblock übergeben wird, der beschreibt, auf welche Ressourcen diese Instanz zugreifen darf, mit dem angegebenen Ressourcen arbeiten.
3. Eine Instanz muss in der Lage sein können, an einer frei wählbaren physikalischen Adresse zu beginnen und kann unter Umständen nicht in der Lage sein, irgendeine spezifische physikalische Adresse zu reservieren, um eine Konfliktbildung mit anderen Betriebssystemen zu vermeiden, die auf dieser bestimmten Adresse laufen.
4. Eine Instanz muss in der Lage sein können, mehrere frei wählbare physikalische Löcher in ihrem Adressraum zu unterstützen, wenn sie Teil einer System-Konfiguration ist, in dem Speicher zwischen Partitionen gemeinsam genutzt wird. Des Weiteren muss eine Instanz physikalische Löcher in ihrem Adressraum bearbeiten können, um "hot inswap" von Speicher zu unterstützen.
5. Eine Instanz muss Nachrichten übergeben und Benachrichtigungen empfangen können, dass neue Ressourcen für Partitionen und Instanzen zur Verfügung stehen. Insbesondere wird ein Protokoll benötigt, um eine Instanz zu informieren, nach einer neuen Ressource zu suchen. Andernfalls erkennt die Instanz möglicherweise nie, dass die Ressource vorhanden und einsatzfähig ist.
6. Eine Instanz muss in der Lage sein können, völlig in ihrem "privaten Speicher" zu laufen, wenn sie in einem System verwendet wird, in dem Instanzen den Speicher nicht gemeinsam nutzen. Alternativ muss eine Instanz in der Lage sein können, physikalischen "gemeinsam genutzten Speicher" für die Kommunikation oder gemeinsame Nutzung von Daten mit anderen Instanzen zu verwenden, die in dem Computer laufen, wenn die Instanz Teil eines Systems ist, in dem Speicher gemeinsam genutzt wird. In einem solchen System mit gemeinsam genutzten Speicher muss eine Instanz in der Lage sein können, physikalischen "gemeinsam genutzten Speicher" wie im Konfigurationsbaum identifiziert in einen virtuellen Adressraum und die virtuellen Adressräume der "Prozesse" abzubilden, die in dieser Betriebssystem-Instanz laufen.
7. Jede Instanz benötigt unter Umständen einen Mechanismus zum Kontaktieren einer anderen CPU in dem Computersystem, um mit ihr zu kommunizieren.

8. Eine Instanz muss ebenfalls in der Lage sein können, andere CPUs zu erkennen, die mit ihren Operationen kompatibel sind, selbst wenn die CPUs ihrer Partition gegenwärtig nicht zugewiesen sind. Zum Beispiel muss die Instanz in der Lage sein können, CPU-Parameter zu ermitteln, wie beispielsweise die Konsolenversionsnummer und Taktgeschwindigkeit, um zu bestimmen, ob sie mit dieser CPU laufen könnte, wenn die CPU der Partition, in dem die Instanz läuft, neu zugewiesen würde.

Den Konfigurationsbaum ändern

[0136] Jedes Konsolenprogramm stellt eine Anzahl von Rückfragefunktionen bereit, um es der dazugehörigen Betriebssystem-Instanz zu gestatten, die Konfiguration des APMP-Systems zu ändern, zum Beispiel durch Erstellen einer neuen Community oder Partition oder durch Ändern der Zugehörigkeit von Speicherfragmenten. Des Weiteren stellen andere Rückfragefunktionen die Möglichkeit bereit, eine Community oder Partition zu löschen oder den Betrieb einer neu erstellten Partition zu starten.

[0137] Allerdings veranlassen Rückfragefunktionen nicht, dass auf den laufenden Betriebssystem-Instanzen Änderungen stattfinden. Auf alle an dem Konfigurationsbaum vorgenommenen Änderungen muss von jeder von der Änderung betroffenen Instanz eingewirkt werden. Die Vorgangsart, die in einer Instanz stattfinden muss, wenn der Konfigurationsbaum geändert wird, ist eine Funktion der Art von Änderung und der Fähigkeiten der Betriebssystem-Instanz. Zum Beispiel kann das Verschieben eines Eingabe/Ausgabe-Prozessors von einer Partition in eine andere erfordern, dass beide Partitionen neu gestartet werden. Die Änderung der Speicherreservierung von Fragmenten könnte andererseits durch eine Betriebssystern-Instanz bearbeitet werden, ohne dass ein Neustart erforderlich ist.

[0138] Die Konfiguration eines APMP-Systems bedingt die Erstellung von Communities und Partitionen sowie die Zuweisung von nicht-zugewiesenen Komponenten. Wenn eine Komponente von einer Partition in eine andere verschoben wird, entfernt sich die gegenwärtige Zugehörigkeit selbst als zu der Ressource zugehörig und gibt dann die neue Zughörigkeit der Ressource an. Die neue Zugehörigkeit kann die Ressource dann verwenden. Wenn eine in einer Partition laufende Instanz eine Komponente freigibt, darf die Instanz nicht mehr auf die Komponente zugreifen. Diese einfache Prozedur lässt die komplexe Synchronisierung entfallen, die erforderlich ist, um ein blindes Entwenden (blind stealing) einer Komponente von einer Instanz und mögliche Wettlaufsituationen beim Booten einer Instanz während einer Neukonfiguration zu gestatten.

[0139] Sobald sie initialisiert sind, werden Konfigurationsbaum-Knoten nie gelöscht oder verschoben, das heißt, ihre Kennungen sind immer gültig. Daher können Hardware-Knotenadressen durch die Software im Cache gespeichert werden. Rückfragefunktionen, die vorgeben, eine Partition oder eine Community zu löschen, löschen den dazugehörigen Knoten nicht wirklich oder entfernen ihn aus dem Baum, sondern kennzeichnen den Knoten als UNAVAILABLE und löschen die Zugehörigkeits-Felder jeder Hardware-Ressource, die zu der Zugehörigkeit der Software-Komponente zugehörig gewesen sind.

[0140] Um Änderungen an dem Konfigurationsbaum zu synchronisieren, führt der Stammknoten des Baums zwei Zähler (transient_level und current_level). Der Zähler transient_level wird zu Beginn einer Aktualisierung des Baums inkrementiert, und der Zähler current_level wird inkrementiert, wenn die Aktualisierug abgeschlossen ist. Die Software kann diese Zähler verwenden, um zu bestimmen, wann eine Änderung an dem Baum eingetreten ist oder gerade eintritt. Wenn eine Aktualisierung durch eine Konsole abgeschlossen wird, kann ein Interrupt für alle CPUs in dem APMP-System generiert werden. Dieser Interrupt kann dazu verwendet werden, die System-Software zu veranlassen, ihren Zustand basierend auf den Änderungen an dem Baum zu aktualisieren.

Erstellung eines APMP-Computersystems

[0141] Fig. 5 ist ein Ablaufdiagramm, das eine Übersicht des Aufbaus des veranschaulichenden adaptiv partitionierten Mehrprozessor- (AMPM) Computersystems darstellt. Die Routine beginnt in Schritt **500** und fährt mit Schritt **502** fort, in dem ein Master-Konsolenprogramm gestartet wird. Wenn das APMP-Computersystem beim Hochfahren erstellt wird, wird die CPU, auf der die Master-Konsole läuft, durch einen vorgegebenen Mechanismus, wie beispielsweise Entscheidung, oder einen anderen Hardware-Mechanismus ausgewählt. Wenn das APMP-Computersystem auf Hardware erstellt wird, die bereits läuft, führt eine CPU in der ersten Partition, die versucht, an den (nicht-vorhandenen) Systemen teilzunehmen, das Master-Konsolenprogramm aus, wie im Folgenden erläutert wird.

[0142] Als Nächstes prüft das Master-Konsolenprogramm in Schritt **504** die Hardware und erstellt den Konfi-

gurationsbaum in Schritt **506**, wie oben erläutert wurde. Wenn mehr als eine Partition in dem APMP-System beim Hochfahren vorhanden ist, wird jede Partition initialisiert und ihr Konsolenprogramm gestartet (Schritt **508**).

[0143] Schließlich wird eine Betriebssystem-Instanz in wenigstens einer der Partitionen gebootet, wie in Schritt **510** angegeben. Die erste zu bootende Betriebssystem-Instanz erstellt eine APMP-Datenbank und füllt die Einträge wie im Folgenden beschrieben aus. APMP-Datenbanken speichern Informationen, die sich auf den Zustand von aktiven Betriebssystem-Instanzen in dem System beziehen. Es ist zu anzumerken, dass die Teilnahme einer Instanz in einem APMP-System nicht erforderlich ist. Die Instanz kann zu einem Zeitpunkt, der lange nach dem Booten liegt, wählen, ob sie nicht teilnimmt oder teilnimmt. Diejenigen Instanzen, die teilnehmen, bilden ein "Mitbenutzungs-Set". Die erste Instanz, die entscheidet, an einem Mitbenutzungs-Set teilzunehmen, muss es erstellen. Es können mehrere Mitbenutzungs-Sets vorhanden sein, die auf einem einzelnen APMP-System arbeiten, und jedes Mitbenutzungs-Set besitzt seine eigene APMP-Datenbank.

Entscheidung zum Erstellen eines neuen APMP-Systems oder zur Teilnahme an einem vorhandenen APMP-System

[0144] Eine Betriebssystem-Instanz, die auf einer Plattform läuft, auf der auch das APMP-Computersystem läuft, muss nicht notwendigerweise ein Mitglied des APMP-Computersystems sein. Die Instanz kann zu jeder Zeit nach dem Booten versuchen, ein Mitglied des APMP-Systems zu werden. Dies kann entweder automatisch nach einem Booten erfolgen oder nachdem ein Benutzerbefehl die Teilnahme explizit initiiert. Nachdem das Betriebssystem zum Boot-Zeitpunkt geladen worden ist, wird die Betriebssystem-Initialisierungsroutine aufgerufen und prüft einen gespeicherten Parameter, um festzustellen, ob er eine unmittelbare Teilnahme angibt, und falls dies der Fall ist, führt das System eine Teilnahmeroutine aus, die Teil des APMP-Computersystems ist. Ein Benutzerbefehl würde zu einer Ausführung der gleichen Routine führen.

APMP-Datenbank

[0145] Eine wichtige Datenstruktur, welche die erfinderische Software-Reservierung von Ressourcen unterstützt, ist die APMP-Datenbank, die Betriebssystem-Instanzen verfolgt, die Mitglieder eines Mitbenutzungs-Sets sind. Die erste Betriebssystem-Instanz, die versucht, das APMP-Computersystem einzurichten, initialisiert eine APMP-Datenbank, wodurch die erfinderischen Software-Ressourcenreservierungen für das anfängliche Mitbenutzungs-Set erstellt oder instantiiert werden. Spätere Instanzen, die Teil des Mitbenutzungs-Sets werden wollen, nehmen durch Registrierung in der APMP-Datenbank teil, die mit diesem Mitbenutzungs-Set verknüpft ist. Die APMP-Datenbank ist eine gemeinsam genutzte Datenstruktur, welche die zentralisierten Informationen enthält, die für die Verwaltung von gemeinsam genutzten Ressourcen des Mitbenutzungs-Sets erforderlich sind. Eine APMP-Datenbank wird ebenfalls initialisiert, wenn das APMP-Computersystem in Reaktion auf einen nicht behebbaren Fehler neu aufgebaut wird.

[0146] Insbesondere ist jede APMP-Datenbank eine dreiteilige Struktur. Der erste Teil ist ein Header-Teil mit einer festen Größe, der grundlegende Synchronisierungsstrukturen für die Erstellung des APMP-Computersystems, Adressenabbildungs-Informationen für die Datenbank und Offsets zu den dienstspezifischen Segmenten enthält, die den zweiten Teil ausmachen. Der zweite Teil ist ein Array von Datenblöcken, wobei jeder potenziellen Instanz ein Block zugewiesen wird. Die Datenblöcke werden als "Knotenblöcke" bezeichnet. Der dritte Teil ist in Segmente unterteilt, die von jedem der untergeordneten Betriebsmittel des Computersystems verwendet werden. Jedes untergeordnete Betriebsmittel ist für den Inhalt seines eigenen Segments und die Synchronisierung des Zugriffs darauf zuständig.

[0147] Der Header-Teil einer APMP-Datenbank ist der erste Teil der APMP-Datenbank, der von einer teilnehmenden Betriebssystem-Instanz abgebildet wird. Auf Teile des Headers wird zugegriffen, bevor eine Instanz an dem Mitbenutzungs-Set teilnimmt und eigentlich bevor die Instanz weiß, dass das APMP-Computersystem vorhanden ist.

[0148] Der Header-Abschnitt enthält:

1. ein Mitgliedschafts- und Erstellungssynchronierungs-Quad-Wort
2. eine Computersystem-Software-Version
3. Zustandsinformationen, Erstellungszeit, Inkamationszählung usw.
4. einen Adressenverweis (Offset) zu einer Mitgliedschafts-Maske
5. Absturz-Instanz, Absturz-Bestätigungs-Bits usw.
6. Validierungsmasken, einschließlich eines Bits für jeden Dienst

7. Speicherabbildungs-Informationen (Seiten-Frame-Nummer-Informationen) für die gesamte APMP-Datenbank
8. Offset/Länge-Paare, die jedes der Dienst-Segmente beschreiben (Längen in Bytes gerundet auf volle Seiten von Seiten und Offsets), einschließlich:
gemeinsam genutzte Speicherdienste
CPU-Kommunikationsdienste
mitgliedschaftsdienste (falls erforderlich)
Sperrdienste

[0149] Das Array von Knotenblöcken wird durch eine System-Partitions-ID, (pro Instanz ist auf der gegenwärtigen Plattform eine möglich), indexiert, und jeder Block enthält:
Instanz-Software-Version
Interruptgrund-Maske
Instanz-Zustand
Instanz-Inkarnation
Instanz-Heartbeat
Instanz-Mitgliedschafts-Zeitstempel
Little-Brother-Instanz-ID und Inaktiv-Zeit; Big-Brother-Instanz-ID
Bit Instanz-Validieruang ausgeführt.

[0150] Eine APMP-Datenbank wird im gemeinsam genutzten Speicher gespeichert. Der feste Anfangsteil von N physikalisch zusammenhängenden Seiten belegt die ersten N Seiten von einem oder zwei Speicherbereichen, die von der ersten teilnehmenden Instanz während des anfänglichen Partitionierens der Hardware reserviert werden. Die Instanz weist die Konsole an, die physikalischen Start-Adressen dieser Bereiche in dem Konfigurationsbaum zu speichern. Der Zweck der Reservierung von zwei Bereichen besteht darin, eine Ausfallsicherung im Fall eines Hardware-Speicherausfalls zu ermöglichen. Die Speicherverwaltung ist für die Abbildung des physikalischen Speichers in virtuellen Adressraum für die APMP-Datenbank zuständig.

[0151] Die einzelnen Maßnahmen, die von einer Betriebssystem-Instanz ergriffen werden, sind in Fig. 6 dargestellt. Insbesondere wenn eine Betriebssystem-Instanz ein Mitglied eines Mitbenutzungs-Sets werden möchte, muss sie vorbereitet sein, das APMP-Computersystem zu erstellen, wenn sie die erste Instanz ist, die versucht, an einem nicht-vorhandenen System "teilzunehmen". Damit die Instanz bestimmen kann, ob ein APMP-System bereits vorhanden ist, muss die Instanz in der Lage sein, den Zustand des gemeinsam genutzten Speichers wie oben beschrieben zu prüfen. Ferner muss sie in der Lage sein, sich mit anderen Instanzen zu synchronisieren, die unter Umständen versuchen, an dem APMP-System und dem Mitbenutzungs-Set zum gleichen Zeitpunkt teilzunehmen, um konfliktbildende Versuche zu verhindern. Die Master-Konsole erstellt den Konfigurationsbaum wie oben erläutert. Anschließend wird ein Speicherbereich von der ersten oder primären zu bootenden Betriebssystem-Instanz initialisiert, und dieser Speicherbereich kann für eine APMP-Datenbank verwendet werden.

Abbildung des APMP-Datenbank-Headers

[0152] Das Ziel der ersten Maßnahmen, die von allen Betriebssystem-Instanzen ergriffen werden, besteht dann, den Header-Teil der APMP-Datenbank abzubilden und eine primitive Inter-Instanz-Interrupt-Bearbeitung zu initialisieren, um die Grundlage für eine Erstellungs- oder Teilnahme-Entscheidung zu schaffen. Die verwendete Routine ist in Fig. 6 dargestellt und beginnt in Schritt **600**. Die erste Maßnahme, die von jeder Instanz ergriffen wird, (Schritt **602**), besteht darin, die Speicherverwaltung zu beauftragen, das Anfangssegment der APMP-Datenbank abzubilden, wie oben beschrieben. Zu diesem Zeitpunkt wird das Array von Knotenblöcken in dem zweiten Datenbank-Abschnitt ebenfalls abgebildet. Die Speicherverwaltung bildet die ersten und zweiten Segmente der APMP-Datenbank in den primären Betriebssystem-Adressraum ab und gibt die Startadresse und Länge zurück. Die Instanz informiert dann die Konsole, die Speicherstelle und Größe der Segmente in dem Konfigurationsbaum zu speichern.

[0153] Als Nächstes wird in Schritt **604** die erste virtuelle Adresse der APMP-Datenbank verwendet, um es der Initialisierungs-Routine zu gestatten, Interruptursachen-Masken in dem Knotenblock, welcher der gegenwärtigen Instanz zugewiesen ist, auf Null zu setzen.

[0154] Ein Null-Anfangswert wird dann in dem Heartbeat-Feld für die Instanz in dem Knotenblock und anderen Knotenblockfeldern gespeichert. In einigen Fällen war die Instanz, die versucht hat, ein neues APMP-Computersystem zu erstellen, vorher ein Mitglied eines APMP-Systems und hat sich aus dem APMP-System nicht

zurückgezogen. Wenn diese Instanz neu gebootet wird, bevor die anderen Instanzen sie entfernt haben, bleiben ihre Bits immer noch in der System-Mitgliedschaftsmaske "eingeschaltet". Andere ungewöhnliche oder Fehler-Fälle können auch dazu führen, dass "Müll" in der System-Mitgliedschaftsmaske gespeichert wird.

[0155] Als Nächstes wird in Schritt **608** die virtuelle Adresse (VA) der APMP-Datenbank in einer privaten Zelle gespeichert, die durch einen Inter-Prozessor-Interrupt-Handler geprüft wird. Der Handler prüft diese Zelle, um zu ermitteln, ob die Interruptursachen-Maske pro Instanz in dem APMP-Datenbank-Header auf auszuführende Arbeit getestet werden soll. Wenn diese Zelle Null ist, ist die APMP-Datenbank nicht abgebildet, und es wird nichts weiter von dem Handler getan. Wie vorher erläutert, wird die gesamte APMP-Datenbank, einschließlich dieser Maske, so initialisiert, dass der Handler nichts tut, bevor die Adresse gespeichert ist. Des Weiteren kann ein Takt-Interrupt-Handler die gleiche private Zelle prüfen, um zu bestimmen, ob das instanzspezifische Heartbeat-Feld für diese Instanz in dem entsprechenden Knotenblock inkrementiert werden soll. Wenn die private Zelle Null ist, inkrementiert der Inerrupt-Handler das Heartbeat-Feld nicht.

[0156] An diesem Punkt ist die Routine beendet, (Schritt **610**), und auf den APMP-Datenbank-Header kann zugegriffen werden, und die teilnehmende Instanz ist in der Lage, den Header zu prüfen und zu entscheiden, ob das APMP-Computersystem nicht vorhanden ist und die Instanz es daher erstellen muss, oder ob die Instanz an einem bereits vorhandenen APMP-System teilnimmt.

[0157] Sobald der APMP-Header abgebildet ist, wird der Header geprüft, um zu bestimmen, ob das APMP-Computersystem eingerichtet ist und funktioniert, und falls dies nicht der Fall ist, ob die gegenwärtige Instanz die APMP-Datenbank initialisieren und das APMP-Computersystem erstellen sollte. Das Problem einer Teilnahme an einem vorhandenen APMP-System wird zum Beispiel schwieriger, wenn das APMP-Computersystem zu einem bestimmten Zeitpunkt erstellt worden ist, aber keine Mitglieder hat, oder wenn das APMP-System nach einem Fehler neu aufgebaut wird. In diesem Fall ist der Zustand des APMP-Datenbankspeichers nicht im Voraus bekannt, und ein einfacher Speichertest ist nicht ausreichend. Eine Instanz, die versucht, an einem möglicherweise vorhandenen APMP-System teilzunehmen, muss in der Lage sein, bestimmen zu können, ob ein APMP-System vorhanden ist oder nicht, und falls dies nicht der Fall ist, muss die Instanz in der Lage sein, ein neues APMP-System ohne Beeinträchtigung durch andere Instanzen zu erstellen. Diese Beeinträchtigung könnte von Threads stammen, die entweder auf der gleichen Instanz oder einer anderen Instanz laufen.

[0158] Um eine solche Beeinträchtigung zu verhindern, wird die Erstellungs-/Teilnahme-Entscheidung getroffen, indem die APMP-Datenbank zuerst gesperrt und dann der APMP-Header geprüft wird, um zu bestimmen, ob ein funktionierendes APMP-Computersystem vorhanden ist. Wenn ein einwandfrei funktionierendes APMP-System vorhanden ist, nimmt die Instanz an dem System teil und hebt die Sperre auf der APMP-Datenbank auf. Wenn andererseits keine APMP-System vorhanden ist, oder wenn ein APMP-System vorhanden ist, das aber nicht funktioniert, dann erstellt die Instanz ein neues APMP-System mit sich selbst als Mitglied und hebt die Sperre auf der APMP-Datenbank auf.

[0159] Wenn es den Anschein hat, als befinde sich ein APMP-System im Übergang, dann wartet die Instanz, bis das APMP-System wieder betriebsfähig oder tot ist und geht dann wie oben beschrieben vor. Wenn ein System nicht erstellt werden kann, schlägt die Teilnahme fehl.

Ein neues APMP-Computersystem erstellen

[0160] Angenommen, ein neues APMP-System muss erstellt werden, dann ist die Ersteller-Instanz für die Reservierung des Rests der APMP-Datenbank, die Initialisierung des Headers und den Aufruf von Systemdiensten zuständig. Angenommen, die APMP-Datenbank ist gesperrt, wie oben beschreiben, dann werden die folgenden Schritte von der Ersteller-Instanz unternommen, um das APMP-System zu initialisieren (diese Schritte sind in den **Fig. 7A** und **Fig. 7B** gezeigt):
Schritt **702** Die Ersteller-Instanz setzt den Zustand des APMP-Systems und seinen Knotenblock-Zustand auf "Initialisieren".
Schritt **704** Die Ersteller-Instanz ruft eine Größenroutine für jeden Systemdienst mit der Adresse seines Längenfelds im Header auf.
Schritt **706** Die sich daraus ergebenden Längenfelder werden summiert, und die Ersteller-Instanz ruft die Speicherverwaltung auf, um Raum für die gesamte APMP-Datenbank zuzuweisen, indem eine neue Abbildung erstellt und die alte Abbildung gelöscht wird.
Schritt **708** Die Ersteller-Instanz füllt die Offsets zu den Anfängen jedes Systemdienst-Segments auf.
Schritt **710** Die Initialisierungs-Routine für jeden Dienst wird mit den virtuellen Adressen der APMP-Datenbank,

des Dienst-Segments und der Segmentlänge aufgerufen.
Schritt **712** Die Ersteller-Instanz initialisiert eine Mitgliedschafts-Maske, um sich selbst zum einzigen Mitglied zu machen und inkrementiert eine Inkamationszählung. Dann richtet sie die Erstellungszeit, Software-Version und andere Erstellungsparameter ein.
Schritt **714** Die Instanz setzt sich dann selbst als ihr eigener Big Brother und Little Brother (zu Heartbeat-Überwachungszwecken, wie im Folgenden beschrieben) ein.
Schritt **716** Dann trägt die Instanz ihren Instanz-Zustand als "Mitglied" und den Zustand des APMP-Systems als "betriebsfähig" ein.
Schritt **718** Schließlich hebt die Instanz die APMP-Datenbanksperre auf.

[0161] Die Routine endet mit Schritt **720**.

Teilnahme an einem bestehenden APMP-Computersystem

[0162] Angenommen, die APMP-Datenbank ist für die Instanz gesperrt, dann werden von der Instanz die folgenden (in den **Fig. 8A** und **Fig. 8B** gezeigten) Schritte unternommen, um ein Mitglied eines bestehenden APMP-System zu werden:
Schritt **802** Die Instanz prüft, um sicherzustellen, dass ihr Instanz-Name eindeutig ist. Wenn ein anderes gegenwärtiges Mitglied den vorgeschlagenen Namen der Instanz besitzt, wird die Teilnahme abgebrochen.
Schritt **804** Die Instanz setzt den Zustand des APMP-Systems und seinen Knotenblock-Zustand auf "Instanz-Teilnahme".
Schritt **806** Die Instanz ruft eine Speicherverwaltungs-Routine auf, um den variablen Teil der APMP-Datenbank in ihren lokalen Adressraum abzubilden.
Schritt **808** Die Instanz ruft Systemteilnahme-Routinen für jeden Systemdienst mit den virtuellen Adressen der APMP-Datenbank und seinem Segment und seiner Segmentlänge auf.
Schritt **810** Wenn alle Systemdienst-Teilnahmeroutinen Erfolg melden, wird die Instanzteilnahme-Routine fortgesetzt. Wenn irgendeine Systemdienst-Teilnahmeroutine fehlschlägt, muss der Instanz-Teilnahmeprozess nochmals von vorne beginnen und möglicherweise ein neues APMP-Computersystem erstellen.
Schritt **812** Angenommen, der Schritt **810** war erfolgreich, dann fügt sich die Instanz selbst zu der Systemmitgliedschafts-Maske hinzu.
Schritt **814** Die Instanz wählt einen Big Brother, um ihre Instanz-Unversehrtheit zu überwachen, wie im Folgenden dargelegt.
Schritt **816** Die Instanz trägt ihren Instanz-Zustand als "Mitglied" ein und setzt ein lokales Mitgliedschafts-Flag
Schritt **818** Die Instanz hebt die Konfigurationsdatenbanksperre auf.

[0163] Die Routine endet dann mit Schritt **820**.

[0164] Der Verlust einer Instanz, entweder durch Inaktivitäts-Zeitüberschreitung oder einen Absturz, wird mittels eines "Heartbeat"-Mechanismus erfasst, der in der APMP-Datenbank implementiert ist. Die Instanzen versuchen, eine minimale Prüfung und Löschung vorzunehmen und benachrichtigen den Rest des APMP-Systems während eines Instanz-Absturzes. Wenn dies nicht möglich ist, erfassen die Systemdienste das Verschwinden einer Instanz über einen Software-Heartbeat-Mechanismus. Insbesondere ist ein Feld "Heartbeat" in der APMP-Datenbank für jede aktive Instanz reserviert. In dieses Feld werden durch die entsprechende Instanz in Zeitintervallen, die kleiner sind als ein vorgegebener Wert, beispielsweise alle zwei Millisekunden, Einträge geschrieben.

[0165] Jede Instanz kann das Heartbeat-Feld einer anderen Instanz prüfen, um eine direkte Entscheidung zu irgendeinem spezifischen Zweck zu treffen. Eine Instanz liest das Heartbeat-Feld der anderen Instanz, indem sie ihr Hearbeat-Feld zweimal im Abstand einer Zeitdauer von zwei Millisekunden liest. Wenn der Heartbeat zwischen den zwei Ablesungen nicht inkrementiert worden ist, wird die Instanz als inaktiv (verloren, an Kontroll-P (control-P) angehalten oder auf oder über Takt-Interrupt-Prioritätsebene aufgehängt) betrachtet. Wenn die Instanz über die Dauer einer vorgegebenen Zeit inaktiv bleibt, wird die Instanz als tot oder desinteressiert betrachtet.

[0166] Des Weiteren wird eine spezielle Anordnung verwendet, um alle Instanzen zu überwachen, weil es nicht für jede Instanz machbar ist, jede andere Instanz zu überwachen, vor allem, wenn das APMP-System groß wird. Diese Anordnung verwendet ein "Big Brother – Little Brother"-Konzept. Insbesondere wenn eine Instanz an dem APMP-System teilnimmt, bevor die Sperre auf der APMP-Datenbank aufgehoben worden ist, wählt sie eines der gegenwärtigen Mitglieder als ihren Big Brother aus, der die teilnehmende Instanz bewachen soll. Die teilnehmende Instanz übernimmt zunächst die Aufgaben des Big Brother für den gegenwärtigen Little

Brother ihres gewählten Big Brother und weist sich dann selbst als der neue Little Brother der ausgewählten Instanz zu. Wenn umgekehrt eine Instanz das APMP-Computersystem verlässt, während sie noch in Betrieb ist, so dass sie die Beendigungs-Verarbeitung durchführen kann, und während sie die Sperre auf der APMP-Datenbank enthält, weist sie ihre Big-Brother-Aufgaben ihrem gegenwärtigen Big Brother zu, bevor sie aufhört, ihren Heartbeat zu inkrementieren.

[0167] Bei jedem Takt-Tick liest jede Instanz nach dem Inkrementieren ihres eigenen Heartbeat den Heartbeat ihres Little Brother und vergleicht ihn mit dem Wert, der beim letzten Takt-Tick gelesen worden ist. Wenn der neue Wert größer ist, oder sich die ID des Little Brother geändert hat, wird der Little Brother als aktiv betrachtet. Wenn die ID des Little Brother und sein Heartbeat-Wert jedoch die gleichen sind, wird der Little Brother als inaktiv betrachtet, und die gegenwärtige Instanz beginnt, den Little Brother ihres Little Brother ebenfalls zu beobachten. Diese Ansammlung von Zuständigkeit wird bis zu einem vorgegeben Höchstwert fortgesetzt und stellt sicher, dass der Ausfall einer Instanz nicht zum Übersehen des Ausfalls ihres Little Brother führt. Wenn der Little Brother beginnt, seinen Heartbeat wieder zu inkrementieren, werden alle zusätzlichen Zuständigkeiten eingestellt.

[0168] Wenn eine Mitglied-Instanz als tot oder desinteressiert betrachtet wird, und sie das APMP-Computersystem nicht über ihre Absicht, sich abzuschalten oder ihren Absturz benachrichtigt hat, wird die Instanz aus dem APMP-System entfernt. Dies kann zum Beispiel durch Setzen des "Bugcheck"-Bits in der primitiven Interrupt-Maske der Instanz (instance primitive interrupt mask) und Senden eines IP-Interrupts an alle CPUs der Instanz erfolgen. Im Regelfall kann auf einen gemeinsam genutzten Speicher nur unterhalb der Hardware-Priorität des IP-Interrupt zugegriffen werden. Dies stellt sicher, dass, wenn die CPUs in der Instanz versuchen sollten, auf einer Priorität unterhalb derjenigen des IP-Interrupt zu arbeiten, der IP-Interrupt zuerst eintritt und die CPU daher das "Bugcheck"-Bit sieht, bevor irgendwelche Threads mit niedrigerer Priorität ausgeführt werden. Dies stellt sicher, dass die Betriebssystem-Instanz abstürzt und keine gemeinsam genutzten Ressourcen in Mitleidenschaft zieht, wie beispielsweise den Speicher, der zu anderen Zwecken neu reserviert worden sein kann, als die Instanzen als tot beurteilt wurden. Als ein zusätzlicher oder alternativer Mechanismus kann eine Konsolen-Rückfrage, (sofern vorhanden), aufgerufen werden, um die Instanz zu entfernen. Des Weiteren führen die restlichen Instanzen in Übereinstimmung mit einer bevorzugten Ausführungsform, immer wenn eine Instanz verschwindet oder ohne Warnung aus dem APMP-Computersystem fällt, einige Unversehrtheitsprüfungen durch, um zu bestimmen, ob sie fortfahren können. Zu diesen Prüfungen gehört das Überprüfen, dass auf alle Seiten in der APMP-Datenbank immer noch zugegriffen werden kann, d.h. dass kein Speicherfehler vorgelegen hat.

Zuweisung von Ressourcen nach der Teilnahme

[0169] Eine CPU kann höchstens eine zugehörige Partition zu jedem Zeitpunkt während der Hochfahrzeit eines APMP-Systems aufweisen. Jedoch kann sich die Wiedergabe dieser Zugehörigkeit und der Einheit, die für deren Steuerung zuständig ist, als Ergebnis von Konfigurations- und Zustands-Übergängen, denen die Ressource selbst unterliegt, der Partition, in der sie resident ist, und der Instanz, die in dieser Partition läuft, ändern.

[0170] Die CPU-Zugehörigkeit wird auf eine Reihe von Arten in einer Reihe von Strukturen angegeben, die von der Einheit vorgegeben werden, welche die Ressource zu dem Zeitpunkt verwaltet. Im grundlegendsten Fall kann die CPU in einem nicht-zugewiesenen Zustand sein, verfügbar für alle Partitionen, die in dem gleichen Mitbenutzungs-Set resident sind wie die CPU. Schließlich wird diese CPU einer spezifischen Partition zugewiesen, die eine Betriebssystem-Instanz ausführen kann oder nicht. In jedem Fall gibt die Partition ihre Zugehörigkeit für alle anderen Partitionen über die Konfigurationsbaumstruktur und für alle Betriebssystem-Instanzen, die in dieser Partition ausgeführt werden können, über das Bit AVAILABLE in dem HWRPB-Feld Flags pro CPU wieder.

[0171] Wenn die besitzende Partition keine Betriebssystem-Instanz aufweist, die auf ihr läuft, ist ihre Konsole dafür zuständig, auf Übergangsereignisse in den darin enthaltenen Ressourcen zu reagieren und diese zu initialisieren. Die Konsole entscheidet, ob sich die Ressource in einem Zustand befindet, die eine Migration zu einer anderen Partition oder ihre Zurücksetzung in den nicht-zugewiesenen Zustand gestattet.

[0172] Wenn in der Partition jedoch gegenwärtig eine Partition läuft, gibt die Konsole die Zuständigkeit für die Initialisierung von Ressourcen-Übergängen auf und ist zuständig für die Benachrichtigung der ausführenden primären CPU der Instanz, wenn eine Konfigurationsänderung stattgefunden hat. Sie ist immer noch der Vermittler (facilitator) des grundlegenden Hardware-Übergangs, doch die Steuerung von Ressourcen-Übergängen geht auf eine Ebene höher an die Betriebssystem-Instanz über. Der Zuständigkeits-Transfer findet statt,

wenn die primäre CPU ihren ersten Befehl außerhalb des Konsolenmodus in einem Systemstart ausführt.

[0173] Betriebssystem-Instanzen können Zugehörigkeitszustands-Informationen auf jede Reihe von Arten verwalten, die intern den effizientesten Einsatz der Informationen fördern. Zum Beispiel kann eine Hierarchie von Zustands-Bit-Vektoren verwendet werden, welche die instanzspezifischen Informationen sowohl intern als auch global (für andere Mitglieder, die eine APMP-Datenbank gemeinsam nutzen) wiedergeben.

[0174] Die internen Darstellung sind strikt für die Verwendung der Instanz. Sie werden zum Startzeitpunkt aus dem grundlegenden Konfigurationsbaum und HWRPB-Informationen aufgebaut, aber als strikte Software-Konstrukte für die Laufzeit der Instanz verwaltet. Sie stellen die Software-Sicht der für die Instanz verfügbaren Partitions-Ressourcen dar und können – über Software-Regelsets – die Konfiguration weiter auf eine Untergruppe von derjenigen einschränken, die durch die physikalischen Konstrukte angegeben wird. Gleichwohl sind alle Ressourcen in der Partition zu der Instanz zugehörig und werden von dieser – unter Verwendung der Konsolen-Mechanismen zum Anweisen von Zustands-Übergängen – verwaltet, bis dieser Betriebssystem-Aufruf keine durchführbare Einheit mehr ist. Dieser Zustand wird angegeben, indem die primäre CPU, die wieder in den Konsolenmodus zurückgekehrt ist, angehalten wird, ohne die Möglichkeit, ohne Neustart wieder zurückzukehren.

[0175] Die Zugehörigkeit von CPU-Ressourcen erstreckt sich nie über die Instanz hinaus. Die Zustands-Informationen jeder einzelnen Instanz werden in einer APMP-Datenbank für Nur-Lese-Entscheidungszwecke dupliziert, doch kann keine andere Instanz ein Zustands-Übergangsereignis für eine Ressource einer anderen CPU erzwingen. Jede Instanz ist dafür zuständig, ihre eigenes Ressourcen-Set zu verstehen und zu steuern; sie kann externe Anforderungen für ihre Ressourcen empfangen, aber nur sie kann die Entscheidung treffen, dass gestattet wird, die Ressourcen zu transferieren.

[0176] Wenn jede derartige CPU betriebsfähig wird, setzt sie ihr Bit AVAILABLE in den Flags pro CPU nicht. Wenn das Bit AVAILABLE nicht gesetzt ist, wird keine Instanz versuchen, zu starten oder erwarten, dass die CPU an dem SMP-Betrieb teilnimmt. Stattdessen fragt die CPU im Konsolenmodus das Feld Zugehörigkeit in dem Konfigurationsbaum ab, das auf die Zuweisung einer gültigen Partition wartet. Sobald eine gültige Partition durch die primäre CPU als Zugehörigkeit zugewiesen worden ist, beginnt die CPU mit ihrem Betrieb in dieser Partition.

[0177] Während der Laufzeit gibt das Feld current_owner die Partition wieder, in der eine CPU sich in Ausführung befindet. Das Bit AVAILABLE in dem Feld Flags pro CPU in dem HWRPB bleibt der endgültige Indikator, ob eine CPU tatsächlich verfügbar ist oder sich in Ausführung befindet, für den SMP-Betrieb mit einer Betriebssystem-Instanz und hat die gleiche Bedeutung wie in herkömmlichen SMP-Systemen.

[0178] Es ist anzumerken, dass eine Instanz kein Mitglied eines Mitbenutzungs-Sets sein muss, um an vielen der Neukonfigurationsfunktionen eines APMP-Computersystems teilzunehmen. Eine Instanz kann ihre Ressourcen zu einer anderen Instanz in dem APMP-System transferieren, so dass eine Instanz, die nicht Teil eines Mitbenutzungs-Sets ist, eine Ressource zu einer Instanz transferieren kann, die Teil des Mitbenutzungs-Sets ist. Auf ähnliche Weise kann die Instanz, die nicht Teil des Mitbenutzungs-Sets ist, eine Ressource von einer Instanz empfangen, die Teil des Mitbenutzungs-Sets ist.

[0179] Eine Software-Implementierung der oben beschriebenen Ausführungsform kann eine Reihe von Computerbefehlen umfassen, die entweder auf einem konkreten Medium, wie beispielsweise einem computerlesbaren Medium, z.B. einer Diskette, einer CD-ROM, einem ROM-Speicher oder einer Festplatte fixiert sind, oder zu einem Computersystem über ein Modem oder eine andere Schnittstelleneinrichtung über ein Medium übertragen werden können. Das Medium kann entweder ein konkretes Medium sein, einschließlich, aber nicht darauf beschränkt, optischen oder analogen Kommunikationsleitungen, oder kann mit drahtlosen Techniken implementiert werden, einschließlich, aber nicht darauf beschränkt, Mikrowellen-, Infrarot- oder anderen Übertragungstechniken. Es kann auch das Internet sein. Die Reihe von Computerbefehlen verkörpert die gesamte oder einen Teil der Funktionalität, die hierin vorher unter Bezugnahme auf die Erfindung beschrieben worden ist. Der Fachmann wird verstehen, dass solche Computerbefehle in einer Reihe von Programmiersprachen für den Einsatz in vielen Computer-Architekturen oder Betriebssystemen geschrieben werden können. Ferner können solche Befehle unter Verwendung jeder gegenwärtigen oder künftigen Speichertechnologie gespeichert werden, einschließlich, aber nicht darauf beschränkt, Halbleiter-, Magnet-, Optik- oder anderen Speichereinrichtungen, oder unter Verwendung jeder gegenwärtigen oder künftigen Kommunikationstechnologie übertragen werden, einschließlich, aber nicht darauf beschränkt, optischen, Infrarot-, Mikrowellen- oder anderen Übertragungstechnologien. Es wird in Erwägung gezogen, dass ein solches Computerprogramm-Produkt als

ein entfernbares Medium mit begleitender gedruckter oder elektronischer Dokumentation vertrieben werden kann, z.B. in Folie eingeschweißte Software, vorabgeladen mit einem Computersystem, z.B. auf einem System-ROM oder einer Festplatte, oder von einem Server oder elektronischen schwarzen Brett aus über ein Netzwerk, z.B. das Internet oder World Wide Web, vertrieben.

[0180] Obwohl eine beispielhafte Ausführungsform der Erfindung offenbart worden ist, wird es für den Fachmann offensichtlich sein, dass verschiedene Änderungen und Modifizierungen vorgenommen werden können, die einige der Vorteile der Erfindung erzielen können, ohne vom Umfang der Erfindung anzuweichen. Zum Beispiel wird es für den durchschnittlichen Fachmann offenkundig sein, dass, obwohl die Beschreibung sich auf ein bestimmtes Hardware-System und Betriebssystem bezogen hat, andere Hardware und Betriebssystem-Software in der gleichen Weise wie beschrieben verwendet werden könnten. Andere Gesichtspunkte, wie beispielsweise die spezifischen Befehle, die zum Erzielen einer bestimmten Funktion verwendet werden, sowie andere Modifizierungen an dem erfinderischen Konzept sollen durch die Ansprüche im Anhang abgedeckt werden.

[0181] Beansprucht wird Folgendes:

Patentansprüche

1. Computer-System (**200**) mit einer Vielzahl von System-Ressourcen, die Prozessoren (**108**–**114**), einen Speicher (**120**) und eine I/O-Schaltung (**118**) enthalten, wobei das Computer-System umfasst:
einen Verbindungsmechanismus;
einen Software-Mechanismus, der die System-Ressourcen in eine Vielzahl von Partitionen (**202**, **204**, **206**) unterteilt; und
wenigstens eine Betriebssystem-Instanz (**208**), die in einer Vielzahl der Partitionen läuft;
gekennzeichnet
dadurch, dass der Verbindungsmechanismus die Prozessoren, den Speicher und die I/O-Schaltung elektrisch so verbindet, dass jeder Prozessor elektrischen Zugriff auf den gesamten Speicher und wenigstens einen Teil der I/O-Schaltung hat; und
durch eine Konfigurations-Datenbank, die in dem Speicher (**120**) gespeichert ist und die Partitionen (**202**, **204**, **206**) anzeigt, die Teil des Computer-Systems (**200**) sind, und die Informationen enthält, die anzeigen, ob jede Betriebssystem-Instanz (**208**, **210**, **212**) aktiv ist.

2. Computer-System nach Anspruch 1, des Weiteren gekennzeichnet durch wenigstens eine Instanz (**208**) eines anderen Betriebssystems, die in wenigstens einer der Vielzahl von Partitionen (**202**) läuft.

3. Computer-System nach Anspruch 1, wobei wenigstens ein Teil des Speichers exklusiv jeder der Partitionen (**202**, **204**, **206**) zugewiesen ist.

4. Computer-System nach Anspruch 1, wobei die Vielzahl von Prozessoren (**108**–**114**) physisch zwischen Partitionen (**202**) aufgeteilt ist und jede Partition ein Konsolenprogramm (**213**, **215**, **217**) umfasst, das die Prozessoren in der Partition steuert.

5. Computer-System nach Anspruch 1, des Weiteren gekennzeichnet durch eine Einrichtung, die Konfigurations-Informationen führt, die anzeigen, welche der Vielzahl von System-Ressourcen jeder Partition (**202**) zugewiesen ist.

6. Computer-System nach Anspruch 5, wobei einer der Prozessoren (**108**–**114**) ein Master-Konsolenprogramm ausführt, das die Konfigurations-Informationen erzeugt; jede Partition (**202**) ein Konsolenprogramm (**213**, **215**, **217**) umfasst, das die Prozessoren in der Partition steuert, und das Konsolenprogramm in jeder Partition so ausgestattet ist, dass es mit dem Master-Konsolenprogramm kommuniziert, um Konfigurations-Informationen auszutauschen.

7. Computer-System nach Anspruch 1, wobei der Verbindungsmechanismus einen Schalter umfasst.

8. Computer-System nach Anspruch 1, des Weiteren gekennzeichnet durch eine Master-Konsole, die eine Einrichtung umfasst, die die Konfigurationsdatenbank während einer Hochfahr-Sequenz des Computer-Systems erzeugt.

9. Computer-System nach Anspruch 1, wobei die Betriebssystem-Instanzen (**208**) Einrichtungen zum kon-

tinuierlichen gegenseitigen Überwachen auf Aktivität umfassen, um eine Fehlfunktion in einer Betrieb-Instanz zu erfassen, und jede Betriebssystem-Instanz eine Einrichtung zum Überwachen einer anderen Betriebssystem-Instanz mittels eines Heartbeat-Mechanismus umfasst.

10. Verfahren zum Aufbauen eines Computer-Systems mit einer Vielzahl von Systemressourcen, die Prozessoren (**108**–**114**), einen Speicher (**120**) und eine I/O-Schaltung (**118**) enthalten, wobei das Verfahren die folgenden Schritte umfasst:
a) elektrisches Verbinden der Prozessoren, des Speichers und der I/O-Schaltung so, dass jeder Prozessor elektrischen Zugang zu dem gesamten Speicher und wenigstens einem Teil der I/O-Schaltung hat;
b) Unterteilen der System-Ressourcen in eine Vielzahl von Partitionen (**202**, **204**, **206**);
c) Ausführen wenigstens einer Betriebssystem-Instanz (**208**) in einer Vielzahl der Partitionen; und
d) Erzeugen einer Konfigurations-Datenbank, die Informationen dahingehend, welche der Partitionen Teil des Computer-Systems sind, und Informationen enthält, die anzeigen, ob jede Betriebssystem-Instanz aktiv ist.

11. Verfahren nach Anspruch 10, wobei Schritt c) den folgenden Schritt umfasst:
c1) Ausführen wenigstens zwei verschiedener Betriebssystem-Instanzen (**208**) in der Vielzahl von Partitionen (**202**, **204**, **206**).

12. Verfahren nach Anspruch 10, wobei Schritt b) den folgenden Schritt umfasst:
b1) Zuweisen wenigstens eines Teils des Speichers (**120**) zu jeder der Partitionen (**202**, **204**, **206**).

13. Verfahren nach Anspruch 10, wobei Schritt b) die folgenden Schritte umfasst:
b2) physisches Aufteilen der Prozessoren (**108**–**114**) zwischen Partitionen (**202**, **204**, **206**); und
b3) Ausführen eines Konsolenprogramms (**213**, **215**, **217**) auf einem Prozessor in jeder Partition (**202**), wobei das Konsolenprogramm die Prozessoren in der Partition steuert.

14. Verfahren nach Anspruch 13, wobei Schritt b) den folgenden Schritt umfasst:
b4) Bestimmen eines primären Prozessors in jeder Partition; und
wobei jeder Schritt c) die folgenden Schritte umfasst:
c1) Ausführen jeder Betriebssystem-Instanz (**208**) auf einem primären Prozessor in einer der Partitionen (**202**, **204**, **206**); und
c2) Steuern jeder Betriebssystem-Instanz (**208**) so, dass sie mit dem Konsolenprogramm für die Partition kommuniziert.

15. Verfahren nach Anspruch 10, das des Weiteren den folgenden Schritt umfasst:
e) Führen von Konfigurations-Informationen, die anzeigen, welche der Vielzahl von System-Ressourcen jeder Partition (**202**) zugewiesen ist.

16. Verfahren nach Anspruch 15, wobei Schritt (e) die folgenden Schritte umfasst:
e1) Ausführen eines Master-Konsolenprogramms auf einem der Prozessoren (**108**–**114**), wobei das Master-Konsolenprogramm die Konfigurations-Informationen erzeugt;
e2) Ausführen eines Konsolenprogramms (**213**, **215**, **217**) in jeder Partition, das die Prozessoren in der Partition (**202**) steuert; wobei Schritt e2) den folgenden Schritt umfasst:
e2a) Verwenden des Konsolenprogramms in jeder Partition zum Kommunizieren mit dem Master-Konsolenprogramm, um Konfigurations-Informationen auszutauschen; und
e3) Senden der Konfigurations-Informationen von dem Master-Konsolenprogramm zu jedem der anderen Konsolenprogramme.

17. Verfahren nach Anspruch 10, wobei Schritt a) den folgenden Schritt umfasst:
a1) Verwenden eines Schalters, um die Prozessoren (**108**–**114**), den Speicher (**120**) und die I/O-Schaltung (**118**) miteinander zu verbinden.

18. Verfahren nach Anspruch 10, wobei Schritt c) den folgenden Schritt umfasst:
c3) Verwenden der Betriebssystem-Instanzen (**208**, **210**, **212**), um einander kontinuierlich zu überwachen und eine Fehlfunktion in einer Betriebs-Instanz mittels eines Heartbeat-Mechanismus zu erfassen.

Es folgen 10 Blatt Zeichnungen

Anhängende Zeichnungen

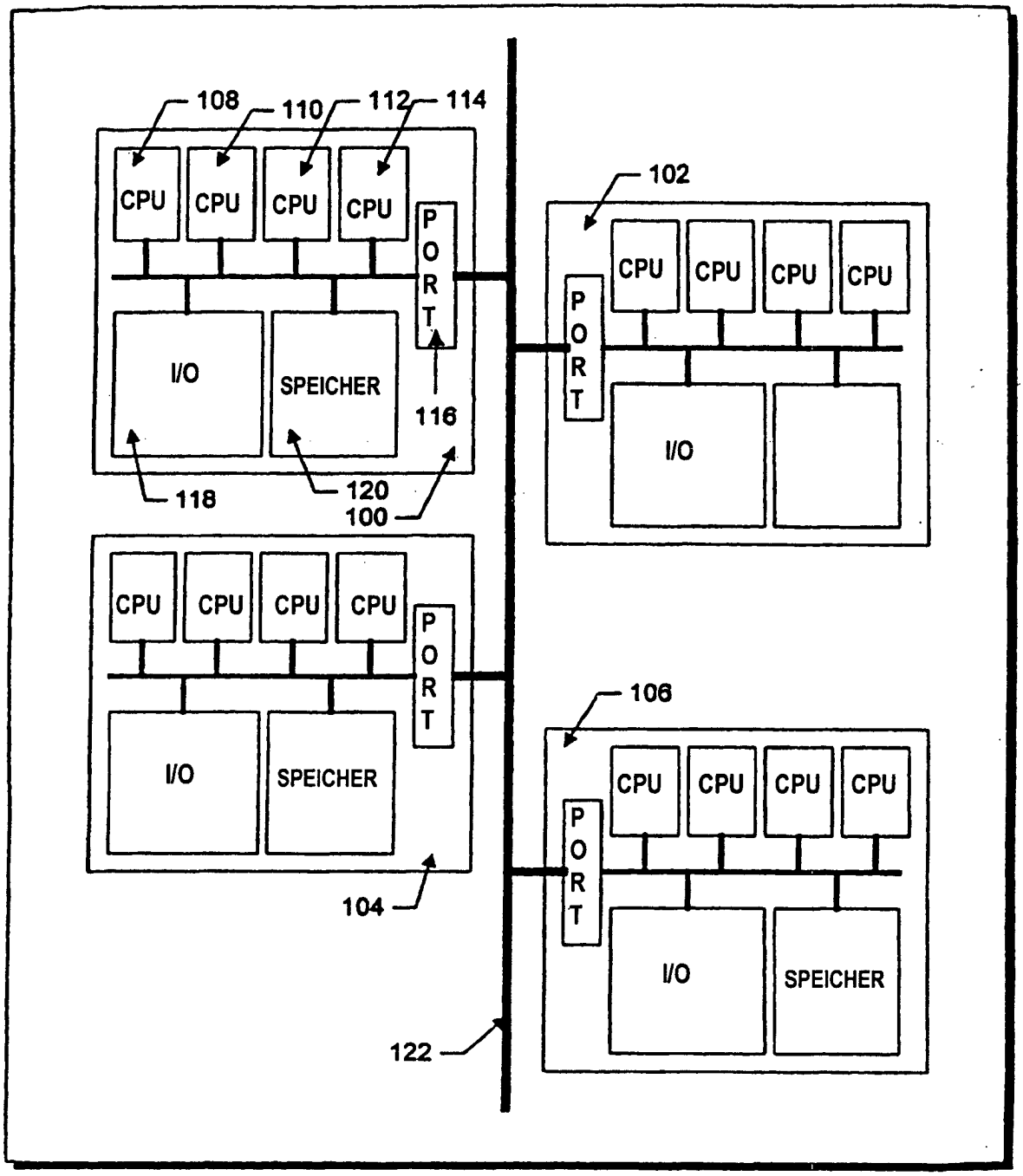

FIG. 1

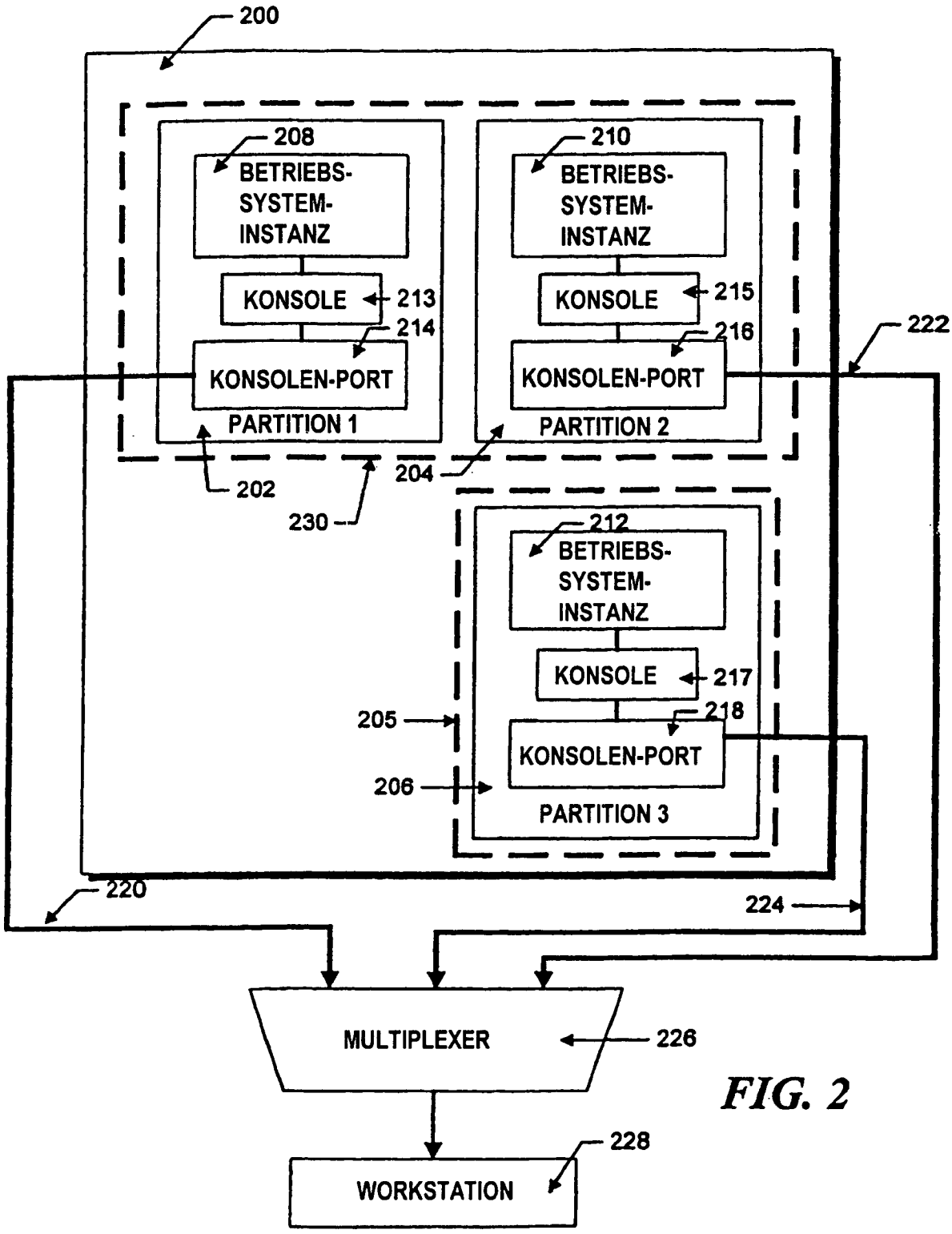
200
208
BETRIEBS-SYSTEM-INSTANZ
KONSOLE
213
214
KONSOLEN-PORT
PARTITION 1
210
BETRIEBS-SYSTEM-INSTANZ
KONSOLE
215
216
KONSOLEN-PORT
PARTITION 2
222
202
230
204
212
BETRIEBS-SYSTEM-INSTANZ
KONSOLE
217
218
KONSOLEN-PORT
PARTITION 3
205
206
220
224
MULTIPLEXER
226
228
WORKSTATION

FIG. 2

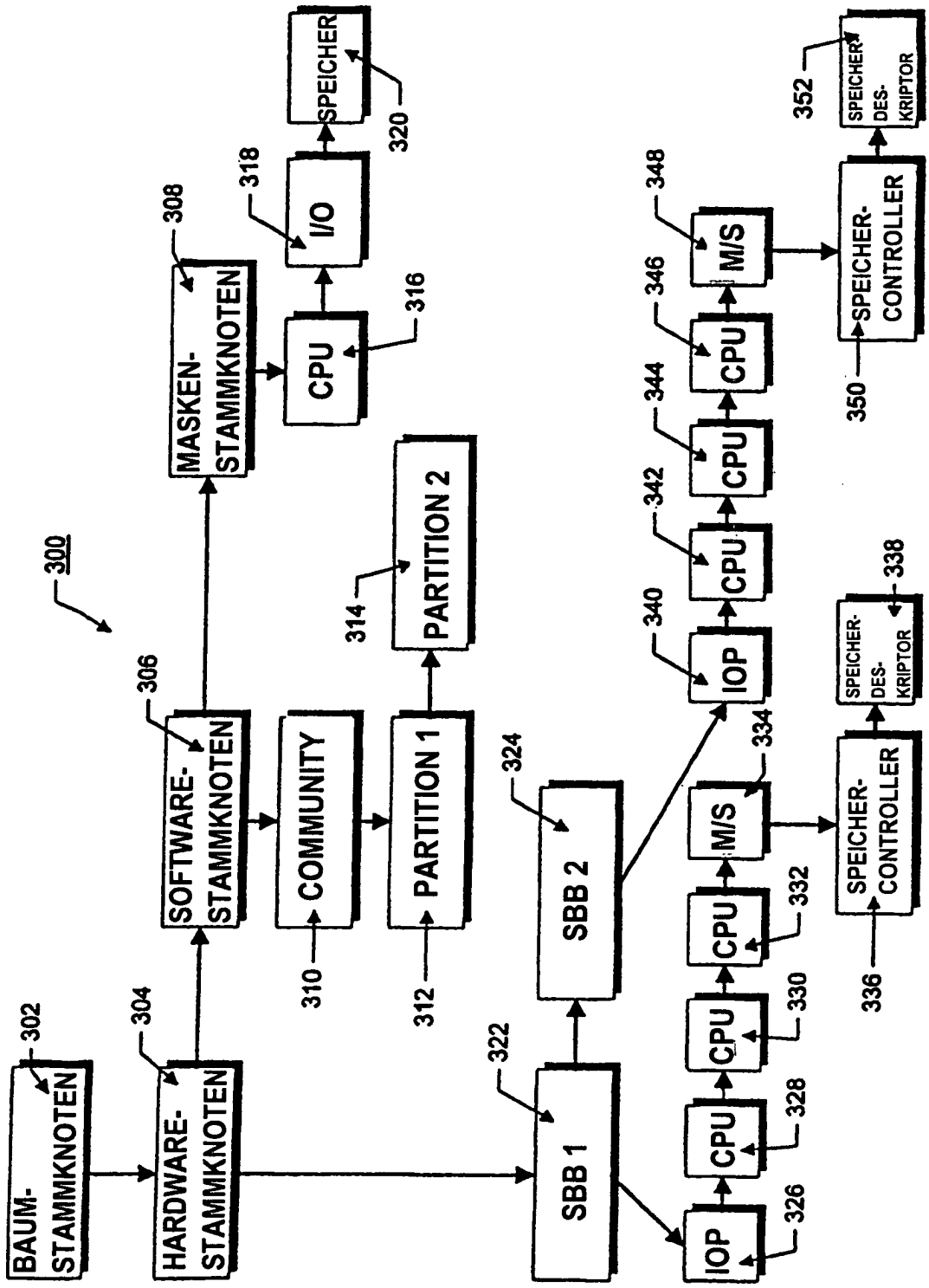
300
BAUM-STAMMKNOTEN
302
HARDWARE-STAMMKNOTEN
304
SOFTWARE-STAMMKNOTEN
306
MASKEN-STAMMKNOTEN
308
COMMUNITY
310
PARTITION 1
312
PARTITION 2
314
CPU
316
I/O
318
SPEICHER
320
SBB 1
322
SBB 2
324
IOP
326
CPU
328
CPU
330
CPU
332
M/S
334
IOP
340
CPU
342
CPU
344
CPU
346
M/S
348
SPEICHER-CONTROLLER
350
SPEICHER-DES-KRIPTOR
352
SPEICHER-CONTROLLER
336
SPEICHER-DES-KRIPTOR
338

FIG. 3

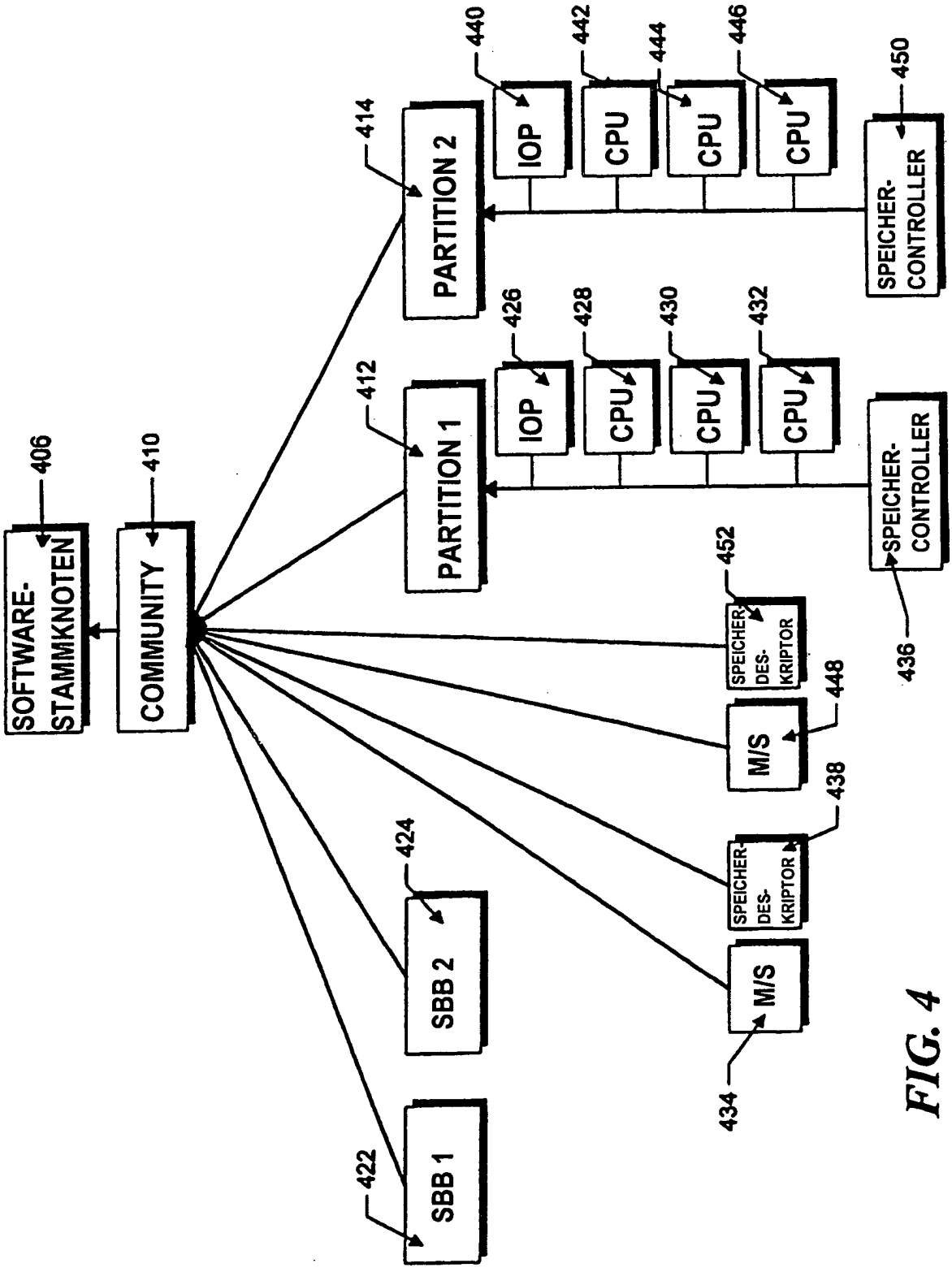
SOFTWARE-STAMMKNOTEN
406
COMMUNITY
410
PARTITION 1
412
PARTITION 2
414
IOP
426
CPU
428
CPU
430
CPU
432
IOP
440
CPU
442
CPU
444
CPU
446
SPEICHER-CONTROLLER
436
SPEICHER-CONTROLLER
450
SPEICHER-DES-KRIPTOR
452
M/S
448
SPEICHER-DES-KRIPTOR
438
M/S
434
SBB 1
422
SBB 2
424

FIG. 4

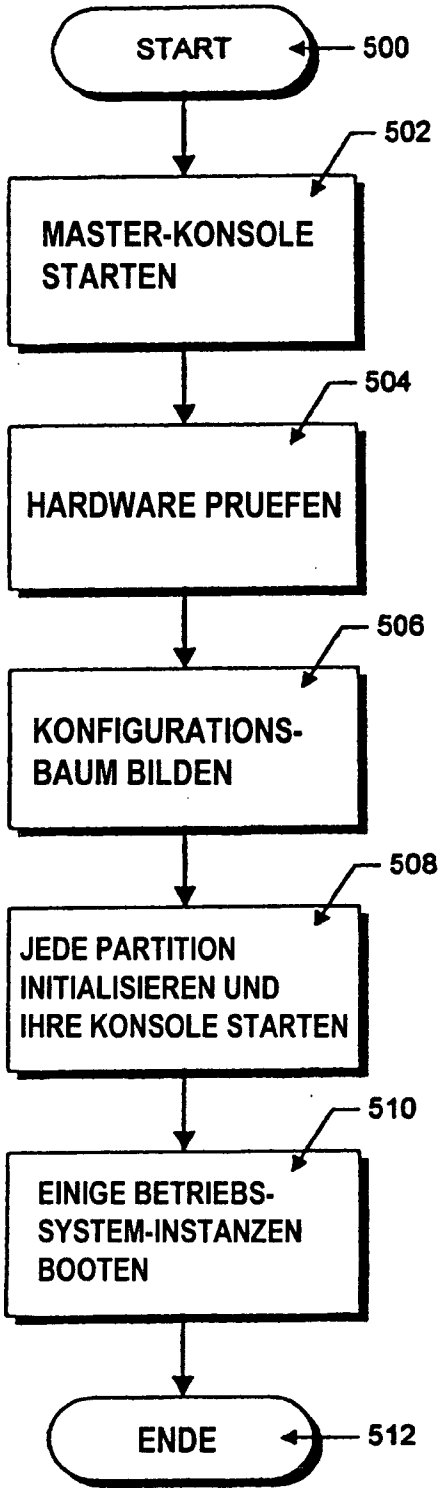
START
500
502
MASTER-KONSOLE STARTEN
504
HARDWARE PRUEFEN
506
KONFIGURATIONS-BAUM BILDEN
508
JEDE PARTITION INITIALISIEREN UND IHRE KONSOLE STARTEN
510
EINIGE BETRIEBS-SYSTEM-INSTANZEN BOOTEN
ENDE
512

FIG. 5

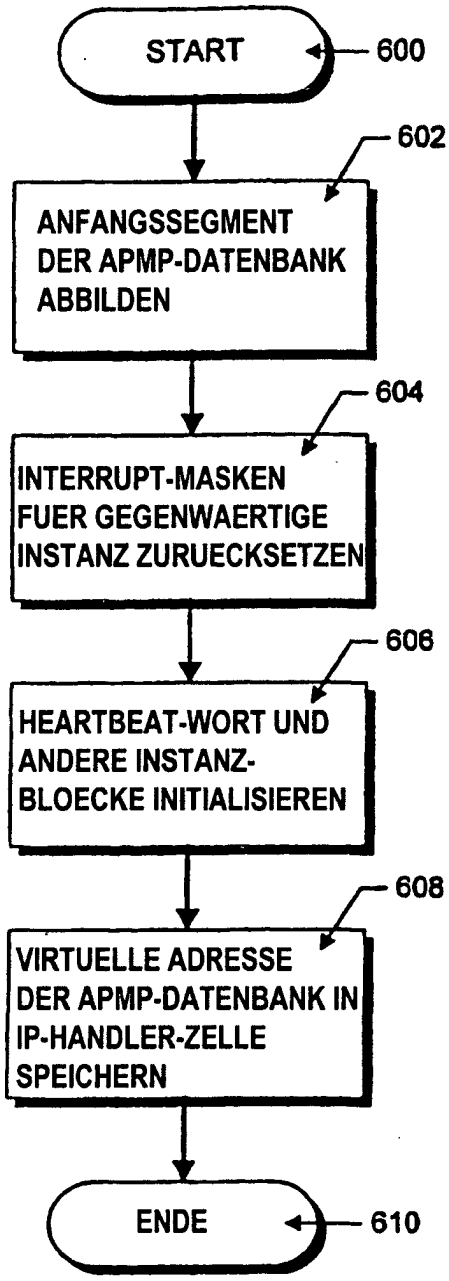
START
600
602
ANFANGSSEGMENT DER APMP-DATENBANK ABBILDEN
604
INTERRUPT-MASKEN FUER GEGENWAERTIGE INSTANZ ZURUECKSETZEN
606
HEARTBEAT-WORT UND ANDERE INSTANZ-BLOECKE INITIALISIEREN
608
VIRTUELLE ADRESSE DER APMP-DATENBANK IN IP-HANDLER-ZELLE SPEICHERN
ENDE
610

FIG. 6

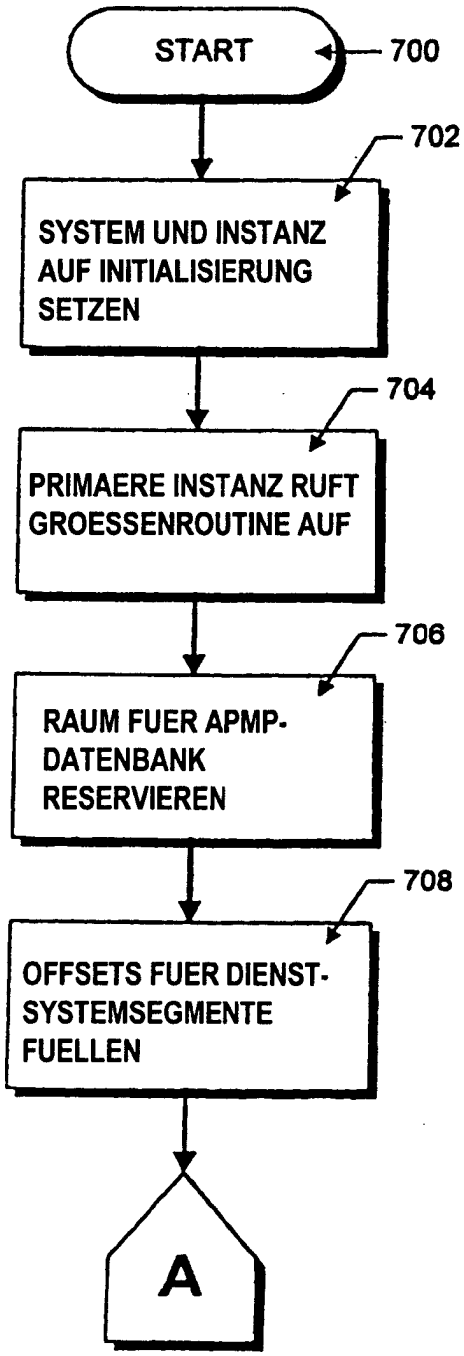
START
700
702
SYSTEM UND INSTANZ AUF INITIALISIERUNG SETZEN
704
PRIMAERE INSTANZ RUFT GROESSENROUTINE AUF
706
RAUM FUER APMP-DATENBANK RESERVIEREN
708
OFFSETS FUER DIENST-SYSTEMSEGMENTE FUELLEN
A

FIG. 7A

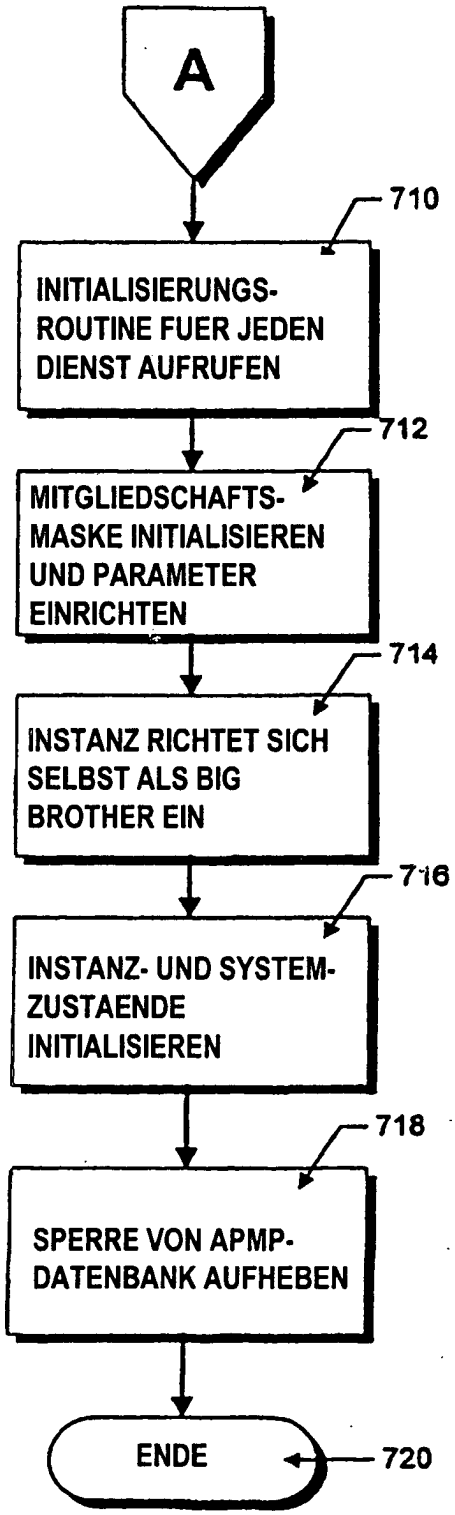
A
710
INITIALISIERUNGS-ROUTINE FUER JEDEN DIENST AUFRUFEN
712
MITGLIEDSCHAFTS-MASKE INITIALISIEREN UND PARAMETER EINRICHTEN
714
INSTANZ RICHTET SICH SELBST ALS BIG BROTHER EIN
716
INSTANZ- UND SYSTEM-ZUSTAENDE INITIALISIEREN
718
SPERRE VON APMP-DATENBANK AUFHEBEN
ENDE
720

FIG. 7B

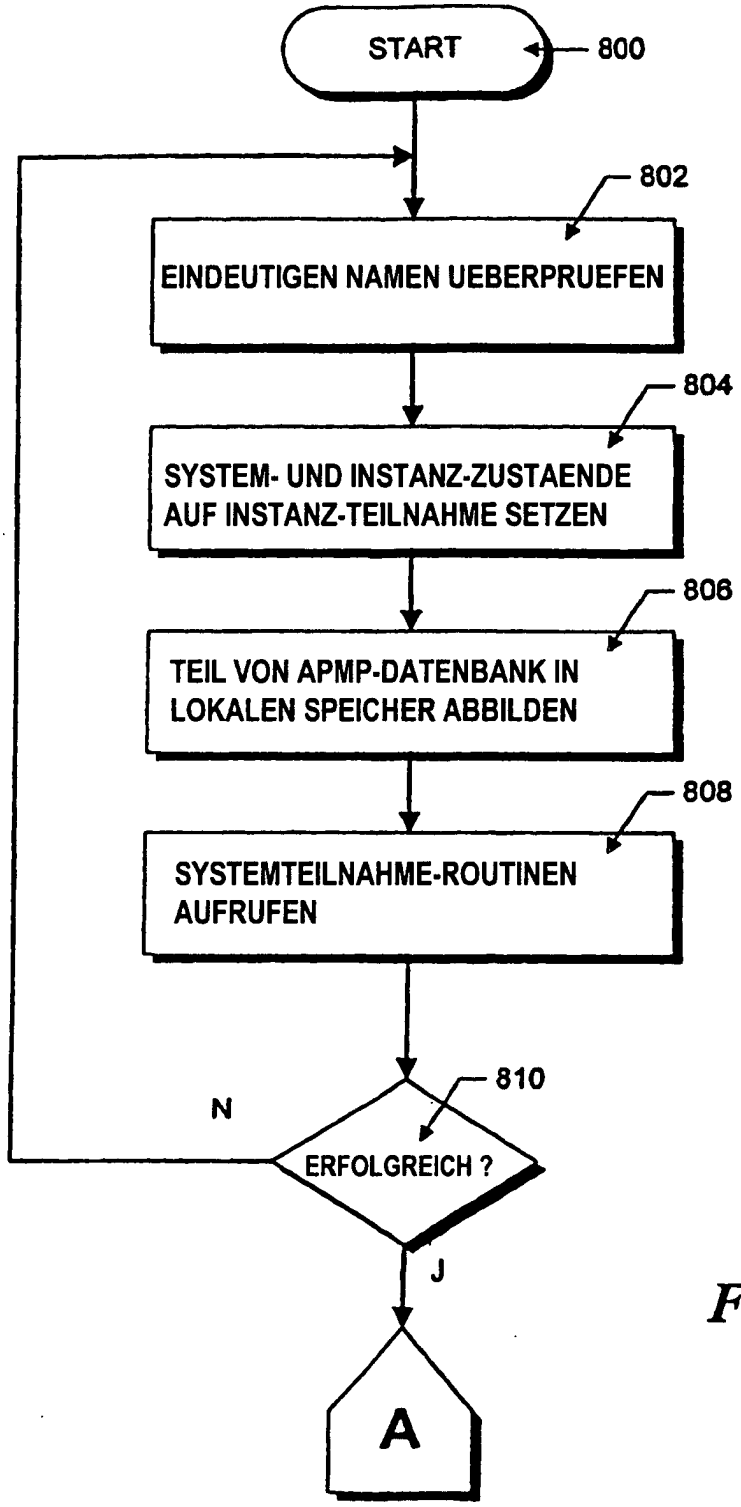
START
800
802
EINDEUTIGEN NAMEN UEBERPRUEFEN
804
SYSTEM- UND INSTANZ-ZUSTAENDE AUF INSTANZ-TEILNAHME SETZEN
806
TEIL VON APMP-DATENBANK IN LOKALEN SPEICHER ABBILDEN
808
SYSTEMTEILNAHME-ROUTINEN AUFRUFEN
810
N
ERFOLGREICH ?
J
A

FIG. 8A

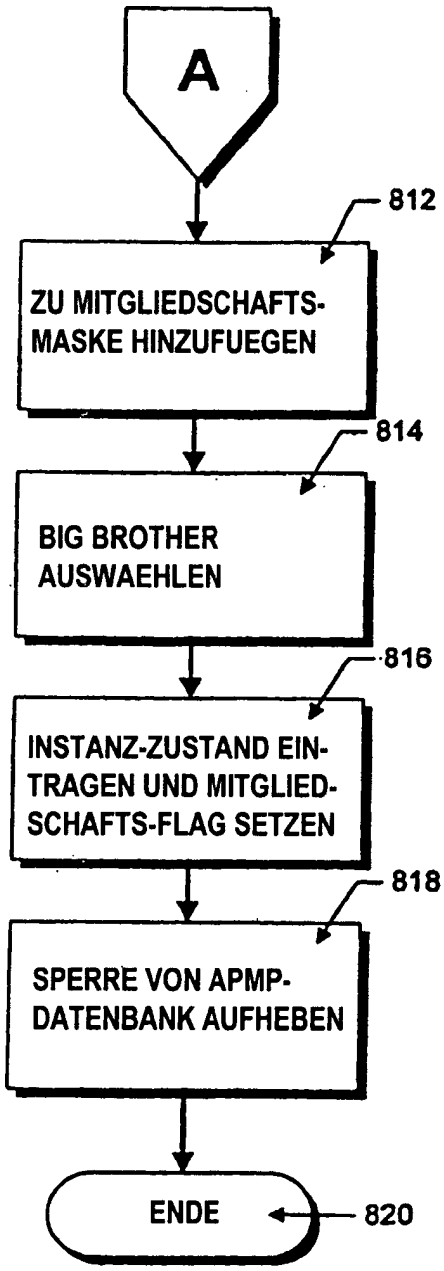
A
812
ZU MITGLIEDSCHAFTS-MASKE HINZUFUEGEN
814
BIG BROTHER AUSWAEHLEN
816
INSTANZ-ZUSTAND EINTRAGEN UND MITGLIEDSCHAFTS-FLAG SETZEN
818
SPERRE VON APMP-DATENBANK AUFHEBEN
ENDE
820

FIG. 8B