(19) Deutsches Patent- und Markenamt

(10) **DE 11 2011 100 714 B4** 2018.07.19

(12) # Patentschrift

(21) Deutsches Aktenzeichen: **11 2011 100 714.0**
(86) PCT-Aktenzeichen: **PCT/JP2011/059833**
(87) PCT-Veröffentlichungs-Nr.: **WO 2011/142227**
(86) PCT-Anmeldetag: **21.04.2011**
(87) PCT-Veröffentlichungstag: **17.11.2011**
(43) Veröffentlichungstag der PCT Anmeldung in deutscher Übersetzung: **17.01.2013**
(45) Veröffentlichungstag der Patenterteilung: **19.07.2018**

(51) Int Cl.: ***G06F 9/48*** (2006.01)

Innerhalb von neun Monaten nach Veröffentlichung der Patenterteilung kann nach § 59 Patentgesetz gegen das Patent Einspruch erhoben werden. Der Einspruch ist schriftlich zu erklären und zu begründen. Innerhalb der Einspruchsfrist ist eine Einspruchsgebühr in Höhe von 200 Euro zu entrichten (§ 6 Patentkostengesetz in Verbindung mit der Anlage zu § 2 Abs. 1 Patentkostengesetz).

(30) Unionspriorität:
2010-112117 **14.05.2010** **JP**

(73) Patentinhaber:
International Business Machines Corporation, Armonk, N.Y., US

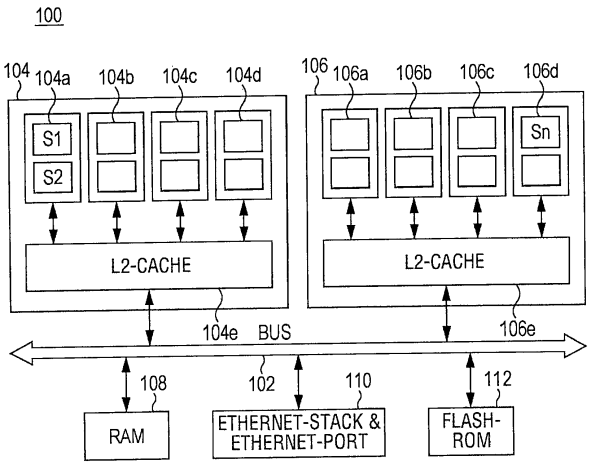
(74) Vertreter:
Richardt Patentanwälte PartG mbB, 65185 Wiesbaden, DE

(72) Erfinder:
Maeda, Kumiko, Yamato-shi, Kanagawa, JP;
Murase, Masana, Yamato-shi, Kanagawa, JP;
Yoshizawa, Takeo, Yamato-shi, Kanagawa, JP;
Doi, Munehiro, Yamato-shi, Kanagawa, JP;
Komatsu, Hideaki, Yamato-shi, Kanagawa, JP

(56) Ermittelter Stand der Technik:
siehe Folgeseiten

(54) Bezeichnung: **Computersystem, Verfahren und Programm**

(57) Zusammenfassung: Zum Verringern einer Systemunterbrechungsdauer infolge der Änderung einer Systemkonfiguration in einem System zur computergesteuerten dynamischen Ressourcenzuweisung in Abhängigkeit von den Umständen.
Während das System in Betrieb ist werden Verkehrsdaten für eine bestimmte Zeit als eine Vorverarbeitung gesammelt. Typische Muster werden aus den gesammelten Verkehrsdaten extrahiert. Anschließend werden für die einzelnen typischen Muster Stream-Programme erzeugt und für eine spätere Bezugnahme gespeichert. Anschließend werden die Kennungen von alternativen Tasks für einen Übergang zwischen verschiedenen Stream-Programmen gespeichert. Im eigentlichen Systembetrieb misst das System Verkehrsdaten regelmäßig oder jederzeit, vergleicht die resultierenden Muster mit den typischen Mustern und wählt ein Stream-Programm in Übereinstimmung mit dem ähnlichsten typischen Muster als die nächste Phase. Gemäß der vorliegenden Erfindung kann eine Programmunterbrechungszeitdauer, wenn von dem Stream-Programm in der gegenwärtigen Phase zu der nächsten Phase gewechselt wird, verringert werden, indem leere Tasks in der gegenwärtigen Phase zu dem nächsten Stream-Programm unter Berücksichtigung der Kosten der Umschaltung zwischen Tasks, der Kosten der Übertragung von Daten zwischen Ressourcen usw. schrittweise verschoben werden.



(19) Deutsches Patent- und Markenamt

(10) **DE 11 2011 100 714 B4** 2018.07.19

(56) Ermittelter Stand der Technik:

US **6 711 607** **B1**

Bochmann, G. v. et al.: Architectural Design of Adaptive Distributed Multimedia Systems. In: Proceedings of the International Workshop on Multimedia Software Development, Berlin, Germany, 1996. pp. 31 - 40

Chang, F. et al.: Automatic Configuration and Run-time Adaptation of Distributed Applications. In: Proceedings of the Ninth International Symposium on High-Performance Distributed Computing, Pittsburgh, PA, USA, 2000. pp. 11 - 20

Beschreibung

Technisches Gebiet

[0001] Die vorliegende Erfindung bezieht sich auf eine Technik zum dynamischen Zuweisen von Ressourcen in einem Computersystem in Abhängigkeit vom Zustand des Netzwerkverkehrs usw. und bezieht sich insbesondere auf eine Technik zum Umschalten der Konfiguration der Ressourcenzuweisung.

Technischer Hintergrund

[0002] Bisher wurde ein dynamisches Umschalten der Ressourcenzuweisung ausgeführt, um die Computerverarbeitung in Abhängigkeit vom Betriebszustand von aktiven Anwendungsprogrammen und vom Zustand des Netzwerkverkehrs zu optimieren. Für diesen Zweck werden die folgenden Techniken vorgeschlagen.

[0003] Die Veröffentlichungen der japanischen ungeprüften Patentanmeldungen Nr. 2005-174201 und 2007-48315 offenbaren ein System zum Umschalten der Ressourcenzuweisung, das Folgendes enthält: eine Ressourcenliste, die einen Server, ein Netzwerk und eine Speichereinheit enthält; eine Anwendungsliste, die die Komponenten von Anwendungen zeigt, die auf den Ressourcen betrieben werden; eine Ressourcenanwendungs-Zuweisungsliste; eine Leistungsinformations-Messelementmaske, die Leistungsinformations-Messelemente aus den oben genannten Konfigurationsinformationen erzeugt; eine Leistungsmessmaschine, die die Leistungsinformations-Messelemente misst; und Mittel zum Erzeugen von Ressourcenzuweisungs-Änderungsregeln, die zum Ändern der Ressourcenzuweisung verwendet werden, aus den Konfigurationsinformationen, wobei eine Ressourcenzuweisung in Übereinstimmung mit den erzeugten Ressourcenzuweisungs-Änderungsregeln und den Konfigurationsinformationen geändert wird. Die Ressourcenzuweisungs-Änderungsregeln enthalten Schwellenwerte von gemessenen Leistungsindizes.

[0004] Um ein Anwendungsausführungssystem bereitzustellen, das einen stabilen Systembetrieb ausführen kann und das nicht durch die Serverleistung und Leitungsqualität beeinflusst wird und keinen zusätzlichen Raum benötigt, offenbart die japanische Veröffentlichung der ungeprüften Patentanmeldung Nr. 2008-191737 ein Anwendungsausführungssystem, bei dem ein Client-Endgerät und eine Anwendungsausführungseinheit mit einem ersten Netzwerk verbunden sind, wobei die Anwendungsausführungseinheit und ein Proxy-Server mit einem zweiten Netzwerk verbunden sind, wobei die Anwendungsausführungseinheit einen Ausführungsabschnitt, der eine Anwendung ausführt, und einen Umschaltabschnitt, der unabhängig von dem Ausführungsabschnitt arbeitet, enthält, um eine empfangene Ausführungsanforderung in Abhängigkeit vom Zustand des Ausführungsabschnitts zu dem Ausführungsabschnitt oder zu dem Proxy-Server zu übertragen. Der Proxy-Server führt in Reaktion auf die Ausführungsanforderung eine alternative Anwendung aus.

[0005] Obwohl im Stand der Technik Techniken zum Ändern der Systemkonfiguration durch Ändern der Leistung am Server und zum Verbessern der Verarbeitungskapazität in Abhängigkeit von dem Ergebnis beschrieben werden, besteht bei ihnen das Problem, dass das System unterbrochen wird, wenn die Systemkonfiguration geändert wird.

[0006] Dienste, die auf Cloud-Computing basieren, das durch viele Firmen kürzlich eingeführt wurde, verwenden das Konzept der Dienstgütevereinbarung (service level agreement, SLA); wenn die Systemunterbrechungsdauer lang ist, erleidet der Dienstanbieter einen Verlust.

[0007] Nach dem Stand der Technik wird jedoch keine bestimmte Lösung zur Verringerung der Systemunterbrechungsdauer während des Umschaltens der Systemkonfiguration vorgeschlagen.

[0008] In Chang, F. et al., «Automatic Configuration and Run-time Adaptation of Distributed Applications », Proceedings of the Ninth International Symposium on High-Performance Distributed Computing, Pittsburgh, US, S. 11 - 20 wird die Entwicklung von Applikationen unter Berücksichtigung der zur Verfügung stehenden Ressourcen beschrieben.

[0009] Aus Bochmann, G. et al., „Architectural Design of Adaptive Distributed Multimedia Systems“, Proceedings of the International Workshop on Multimedia Software Development, Berlin, Deutschland, 1996, S. 31 - 40, ist ein System bekannt, mit dem ein Fernzugriff auf Multimedia-Datenbanken ermöglicht wird.

[0010] Aus US 6 711 607 B1 ist die dynamische Planung von Task-Streams in einem System mit Mehrfachressourcen bekannt, bei der die Dienstqualität gewährleistet bleibt. Dabei werden die Ressourcen auf die verschiedenen Streams verteilt, um die Dienstqualität beizubehalten.

[0011] Es ist dementsprechend eine Aufgabe der vorliegenden Erfindung, eine Systemunterbrechungsdauer infolge einer Änderung der Systemkonfiguration in einem computergesteuerten System zur dynamischen Ressourcenzuweisung in Abhängigkeit von den Umständen zu verringern.

[0012] Die Aufgabe wird erfindungsgemäß gelöst durch ein Verfahren mit den Merkmalen von Anspruch 1, durch ein Computersystem mit den Merkmalen von Anspruch 4 und ein Programmprodukt mit den Merkmalen von Anspruch 7. Bevorzugte Ausführungsformen der Erfindung sind Gegenstand der jeweiligen Unteransprüche.

[0013] Das System gemäß der vorliegenden Erfindung wird im folgenden mit seinen Merkmalen kurz dargestellt. Das System sammelt zunächst Verkehrsdaten, während das System für eine bestimmte Dauer in Betrieb ist, als eine Vorverarbeitung und extrahiert typische Muster aus den gesammelten Verkehrsdaten.

[0014] Das System erzeugt dann Stream-Programme für die einzelnen typischen Muster und speichert sie für eine spätere Bezugnahme z.B. unter Verwendung einer Technik, die in der japanischen Patentanmeldung Nr. 2009-271308 beschrieben ist, die durch den Anmelder eingereicht wurde, wobei keine Beschränkung hierauf besteht.

[0015] Das System speichert dann die Kennungen von alternativen Tasks für den Übergang zwischen unterschiedlichen Stream-Programmen.

[0016] Im eigentlichen Systembetrieb misst dann das System regelmäßig oder zu irgendeinem Zeitpunkt Verkehrsdaten, vergleicht die sich ergebenden Muster mit den typischen Mustern und wählt ein Stream-Programm, das dem typischen Muster am ähnlichsten ist, als die nächste Phase.

[0017] Dabei kann die Programmunterbrechungsdauer, wenn der Übergang von dem Stream-Programm in der aktuellen Phase zu der nächsten Phase erfolgt, verringert werden, indem leere Tasks in der aktuellen Phase zu dem nächsten Stream-Programm als alternative Tasks unter Berücksichtigung des Aufwands (der Kosten) zum Umschalten zwischen Tasks, des Aufwands der Übertragung von Daten zwischen Ressourcen usw. schrittweise verschoben werden.

[0018] Zu diesem Zeitpunkt werden die alternativen Tasks ausgewählt, indem die Pipeline-Pitches von betreffenden Tasks, ein task-bezogener Umschaltaufwand und ein Aufwand, der die Zeit für die Übertragung und den Empfang von Daten zwischen den Ressourcen in den betreffenden Phasen enthält, im Voraus gemessen und gespeichert werden, so dass der Aufwand unter Berücksichtigung der aktuellen Phase und der nächsten Phase, des Übergangs von verwendete Ressourcen und des Übergangs von ausgeführten Tasks verringert wird.

[0019] Es wird ein System zur computergesteuerten dynamischen Ressourcenzuweisung bereitgestellt mit dem Vorteil der Verringerung einer Programmunterbrechungsdauer, wenn von der ursprünglichen Konfiguration zur nächsten Konfiguration in Abhängigkeit von den Umständen übergegangen wird, indem eine Leerlaufzeit zwischen Prozessen vermindert wird, indem alternative Tasks von der ursprünglichen Konfiguration ausgewählt werden, um einen Zwischenprozess auszuführen.

Figurenliste

Fig. 1 ist ein Blockschaubild, das eine Hardware-Konfiguration zum Ausführen der vorliegenden Erfindung zeigt.

Fig. 2 ist ein Funktionsblockschaubild zum Ausführen der vorliegenden Erfindung.

Fig. 3 ist ein Schaubild, das ein Beispiel einer Konfigurationstabelle zeigt.

Fig. 4 ist ein Schaubild, das ein Beispiel eines Stream-Programms zeigt.

Fig. 5 ist ein Ablaufplan zur Vorverarbeitung.

Fig. 6 ist ein Schaubild, das Verkehrsinformationen zeigt, die sich zeitlich ändern.

Fig. 7 ist ein Schaubild, das Beispiele einer extrahierten Phase zeigt.

Fig. 8 ist ein Schaubild, das ein Beispiel eines Stream-Programms zeigt.

Fig. 9 ist ein Schaubild, das ein Beispiel eines Stream-Programms zeigt.

Fig. 10 ist ein Schaubild, das ein Beispiel eines Stream-Programms zeigt.

Fig. 11 ist ein Schaubild, das einen Ablaufplan für den Prozess zum Zuweisen von Berechnungsressourcen zu UDOPs zeigt.

Fig. 12 ist ein Schaubild, das ein Beispiel eines Stream-Graphen und verfügbare Ressourcen zeigt.

Fig. 13 ist ein Schaubild, das ein Beispiel von angeforderten Ressourcen zeigt, nachdem Berechnungsressourcen zu UDOPs zugewiesen wurden.

Fig. 14 ist ein Schaubild, das ein Beispiel eines Zuweisungsänderungsprozesses zeigt.

Fig. 15 ist ein Schaubild, das einen Ablaufplan für einen Prozess zum Bestimmen alternativer Ressourcen von Tasks zeigt.

Fig. 16 ist ein Schaubild, das Tabellen der aktuellen Tasks für die einzelnen Ressourcen und alternative Tasks zeigt.

Fig. 17 ist ein Schaubild, das einen Ablaufplan eines Prozesses zum Bestimmen einer Phasenverschiebung zeigt.

Fig. 18 ist ein Schaubild, das einen Ablaufplan für einen Ausführungsprozess durch alternative Tasks zeigt.

Fig. 19 ist ein Schaubild, das ein Beispiel der Topologie einer Berechnungsressource zeigt.

Fig. 20 ist ein Schaubild, das ein Beispiel der Phasenumschaltung und der Ausführung durch alternative Tasks zeigt.

Fig. 21 ist ein Schaubild, das ein Beispiel eines Übergangszyklus von Ausführungs-Tasks durch ein alternatives Verfahren zeigt.

Beschreibung von Ausführungsformen

[0020] Eine Ausführungsform der vorliegenden Erfindung wird im Folgenden unter Bezugnahme auf die Zeichnungen beschrieben. Falls nicht anders angegeben bezeichnen in sämtlichen Zeichnungen gleiche Bezugszeichen die gleichen Komponenten. Es sollte klar sein, dass die folgende Beschreibung eine Ausführungsform der vorliegenden Erfindung darstellt und die Erfindung nicht auf die Beschreibung der Ausführungsform beschränkt ist.

[0021] **Fig. 1** ist ein Blockschaubild, das eine Hardware-Konfiguration zum Ausführen der vorliegenden Erfindung zeigt. Diese Ausführungsform verwendet eine Multikern-Multiprozessor-Routervorrichtung **100** wie etwa ein PRISM; die vorliegende Erfindung ist jedoch nicht darauf beschränkt.

[0022] In **Fig. 1** ist ein Bus **102** mit einem Multikern-Prozessor **104**, einem Multikern-Prozessor **106**, einem RAM **108**, einem Ethernet-Stack & Ethernet-Port **110** und einem Flash-ROM **112** verbunden.

[0023] Zu Beispielen der Multikern-Prozessoren **104** und **106** gehören ein Netzwerk-Prozessor wie etwa ein Intel(R)-IXP-425-Netzwerk-Prozessor, wobei jedoch keine Beschränkung darauf besteht. Der Netzwerk-Prozessor besitzt die Funktionen des Ethernet(R)-MAC, der Ziffernverarbeitung usw.

[0024] Da der Multikern-Prozessor **104** und der Multikern-Prozessor **106** im Wesentlichen die gleiche Konfiguration aufweisen, wird der Multikern-Prozessor **104** stellvertretend beschrieben. Der Multikern-Prozessor **104** enthält mehrere Kerne 104a, 104b, 104c und 104d. Die Kerne 104a, 104b, 104c und 104d sind über einen L2-Cache 104e mit dem Bus **102** verbunden.

[0025] Die einzelnen Kerne 104a, 104b, 104c und 104d ermöglichen, dass mehrere Threads ablaufen. Die Kästen S1 und S2, die im Kern **104** a gezeigt sind, sind z.B. unabhängige Threads S1 und S2. Da das gleiche für die Kerne 104b, 104c und 104d gilt, werden individuelle Beschreibungen weggelassen.

[0026] Wie in der Zeichnung dargestellt läuft in dieser Ausführungsform ein Verwaltungs-Thread Sn, der die Hauptfunktionen ausführt, auf dem Kern 106d des Multikern-Prozessors **106**.

[0027] Der RAM **108** wird für den Multikern-Prozessor **104** und den Multikern-Prozessor **106** verwendet, um die Werte von Verarbeitungsergebnissen vorübergehend zu halten oder zu lesen.

[0028] Obwohl nicht dargestellt verbindet der Ethernet-Stack & Ethernet-Port **110** mit einem weiteren Computersystem, einer am Netzwerk angeschlossenen Speichereinheit (NAS), einem Speicherbereichs-Netzwerk (SAN), einem weiteren Router usw. Die Multikern-Multiprozessor-Routervorrichtung **100** hat die Funktion zum Datenaustausch mit diesen Einheiten.

[0029] Der Flash-ROM **112** enthält ein Netzwerk-Betriebssystem wie etwa Junos(R) von Juniper Networks oder IOS von Cisco Systems Inc. und die Verarbeitungsmodule, die später beschrieben werden sollen, die bewirken, dass der Multikern-Prozessor 104 und der Multikern-Prozessor **106** als Router betrieben werden.

[0030] Im Folgenden werden Verarbeitungsmodule, die in dem Flash-ROM **112**, im RAM 108 usw. gehalten werden, unter Bezugnahme auf ein Funktionsblockschaubild in **Fig. 2** beschrieben.

[0031] In **Fig. 2** hat ein Modul **202** zum Sammeln statistischer Informationen die Funktion zum Sammeln von Verkehr, der in bestimmten Intervallen am Ethernet-Stack & Ethernet-Port **110** eintrifft. Ein herkömmliches typisches Verfahren zum Sammeln von Verkehr verwendet ein SNMP-Protokoll. Alternativ kann ein Paketanalysebefehl mit der Bezeichnung tcpdump verwendet werden. Des Weiteren kann ein kommerzielles Werkzeug wie etwa NetFlow Tracker, das von Fluke Networks erhalten werden kann, verwendet werden. Die hier gesammelten Informationen stellen das Verhältnis von Verkehr dar, wie etwa Mail-Verkehr, FTP, bewegte Bilder und Web-Verkehr.

[0032] Das Modul **202** zum Sammeln statistischer Informationen sammelt typischerweise Verkehrsinformationen eines Wochentags oder eines Feiertags und speichert sie in einem Festplattenlaufwerk oder dergleichen eines (nicht gezeigten) Computersystems.

[0033] Ein Phasenmuster-Extraktionsmodul **204** analysiert die Verkehrsinformationen, die in dem Festplattenlaufwerk oder dergleichen gespeichert sind, extrahiert eine Vielzahl typischer Muster und speichert sie üblicherweise in dem RAM **108** als eine Phasenmustergruppe **206**. Die Phasenmustergruppe **206** kann durch das Sammeln von Verkehrsmustern in regelmäßigen Intervallen oder durch Clusterbildung wie etwa k-Means-Clusterbildung festgelegt werden.

[0034] Eine Konfigurationstabelle 208 enthält Einträge, die den einzelnen Verkehrsmustern der Verkehrsmustergruppe **206** entsprechen, und ist gleichbedeutend mit einer Tabelle in **Fig. 6**, die in der japanischen Patentanmeldung Nr. 2009-271308 beschrieben ist, die durch den Anmelder eingereicht wurde, und ist deswegen in **Fig. 3** nochmals dargestellt.

[0035] In **Fig. 3** bezeichnet UDOP benutzerdefinierte Operatoren im Stream-Programming; z.B. einen Ethernet (eine Handelsmarke)-Protokollstapel, einen IP-Protokollstapel, einen TCP-Protokollstapel, einen UDP-Protokollstapel, einen SSL-Protokollstapel, Virusscan und einen XML-Beschleuniger in dieser Ausführungsform, wobei darauf keine Beschränkung besteht.

[0036] In **Fig. 3** bezeichnet Kernel ein Modul oder eine Vielzahl von Modulen, die für die einzelnen UDOPs hergestellt wurden. Wenn eine Vielzahl von Modulen vorhanden ist, unterscheiden sich die Größen der eindimensionalen Arrays von Paketen.

[0037] Ein Ausführungsmuster wird in Übereinstimmung mit den folgenden Regeln ausgedrückt, zum Beispiel:

```
Als Schleife ausführen: A+A+A....A => loop(n, A)
wobei A+A+A....A eine serielle Verarbeitung von A bezeichnet und loop(n, A)
eine Schleife bezeichnet, bei der A n-mal wiederholt wird.
Abrollen der Schleife: loop(n, A) => A+A+A....A
Serielles Aufrollen: split_join(A, A... A) => loop(n, A)
Das bezeichnet, dass A, A... A parallel zu loop(n, A) aufgerollt wird.
Paralleles Abrollen der Schleife: loop(n, A) => split_joing(A, A, A........A)
Das bezeichnet, dass loop(n, A) zu parallel A, A... A abgerollt wird.
Schleifenteilung loop(n, A) => loop(x, A) + loop(n-x, A)
Parallele Schleifenteilung: loop(n, A) => split_join(loop(x, A), loop(n-x, A))
```

```
Schleifenfusion: loop(n, A) + loop(n, B) => loop(n, A+B)
Serielle Schleifenfusion: split_join(loop(n, A), loop(n, B)) => loop(n, A+B)
Schleifenverteilung: loop(n, A+B) => loop(n, A) + loop(n, B)
Parallele Schleifenverteilung: loop(n, A+B) => split_join(loop(n, A), loop(n, B))
Knotenverschmelzung: A+B => {A,B}
Knotenteilung: {A,B} => A+B
Schleifenersetzung: loop(n,A) => X /* X ist geringerer Aufwand */
Knotenersetzung: A => X /* X ist geringerer Aufwand */
```

[0038] In **Fig. 3** bezeichnet Pitch den Pipeline-Pitch, das heißt die Verarbeitungszeit für eine Stufe der Pipeline-Verarbeitung. Ressource bezeichnet die Anzahl der verwendeten CPUs. In dieser Ausführungsform wird die Anzahl von Threads in dem System von **Fig. 1** in der Spalte Ressource von **Fig. 3** beschrieben.

[0039] Die Einträge in der Konfigurationstabelle 208 werden durch ein im Voraus festgelegtes Verarbeitungsmodul (nicht gezeigt) auf der Grundlage einer Systemumgebung **209** erzeugt, die Hardware und Software enthält, die mit dem Ethernet-Stack & Ethernet-Port **110** verbunden sind. Das wird erreicht, indem für alle verwendeten Ressourcengruppen der Prozess zum Erhalten einer Kernel-Definition zum Erreichen jedes UDOP, zum Erhalten einer Soll-Hardwarekonfiguration, zum Herstellen einer Gruppe von verwendeten Ressourcen durch Kombinieren von verwendeten Architekturen, zum Auswählen eines ausführbaren Kernel hierfür und zum Messen eines Pipeline-Pitch ausgeführt wird. Für weitere Einzelheiten über die Verarbeitung siehe **Fig. 3** und eine entsprechende Beschreibung der Spezifikation der japanischen Patentanmeldung Nr. 2009-271308, die durch den Anmelder eingereicht wurde.

[0040] Der Compiler **210** erzeugt Stream-Formatcodes für die einzelnen Phasenmuster der Phasenmustergruppe **206** unter Bezugnahme auf die Einträge in der Konfigurationstabelle 208.

[0041] Zu bekannten Beispielen der Stream-Programming-Sprachen zum Beschreiben der Stream-Formatcodes gehören SPADE von International Business Machines Corporation und Streamlt von Massachusetts Institute of Technology. Streamlt beschreibt den in **Fig. 4** gezeigten Stream-Graphen in dem folgenden Code.

```
add splitjoin {
        split roundrobin();
        add pipeline {
                add A();
                add B();
                add C();
        } add D();
        add pipeline {
                add E();
                add F();
        }
} join roundrobin();
```

[0042] Für weitere Einzelheiten über Streamlt siehe
http://groups.csail.mit.edu/cag/streamit/
oder
http-//groups.csail.mit.edu/cag/streamit/papers/streamit-cookbook.pdf

[0043] In Bezug auf SPADE siehe
http://domino.research.ibm.com/comm/research_projects.nsf/pages/esps.spad e.html

[0044] Der Compiler **210** erzeugt Stream-Formatcodes auf der Grundlage der einzelnen Phasenmuster. Weitere Einzelheiten über die Verarbeitung werden später beschrieben. Der Stream-Formatcode beschreibt Tasks zum Ausführen des Prozesses und Hardwareressourcen zum Ausführen der Tasks in einem Graphformat.

[0045] Wenn die Stream-Formatcodes für die einzelnen Phasenmuster auf diese Weise erzeugt werden, erzeugt ein spezielles Verarbeitungsprogramm (nicht gezeigt) eine Übergangstabelle 214 von Stream-Formatcodes, die den einzelnen Phasenmustern entsprechen. Die Übergangstabelle 214 kann durch den Com-

piler **210** erzeugt werden. Im Folgenden wird eine Gruppe von Stream-Formatcodes als eine Stream-Formatcodegruppe **212** bezeichnet.

[0046] Die oben beschriebenen Module dienen dazu, die Phasenmustergruppe **206**, die Stream-Formatcodegruppe **212** und die Übergangstabelle 214 im Voraus zu bilden. Im Folgenden wird eine Modulgruppe beschrieben, die während des tatsächlichen Betriebs der Routervorrichtung **100** betrieben wird.

[0047] Das Phasenmuster-Messmodul **216** misst Phasenmuster vorzugsweise durch den Prozess des Zählens von Datenelementen für jede Art, die die Tasks am Kopf der Stream-Formatcodes verarbeiten, die leichter sind als die des Moduls **202** zur Sammlung statistischer Informationen während des Betriebs der Routervorrichtung 100. Die Messung wird z.B. in Intervallen ausgeführt, die für den Verkehrsaufwand, den die Routervorrichtung **100** bewältigt, geeignet sind.

[0048] Ein Phasenmuster-Vergleichsmodul **218** hat die Funktion zum Vergleichen der Phasenmuster, die durch das Phasenmuster-Messmodul **216** gemessen werden, und der einzelnen Phasenmuster der Phasenmustergruppe **206**. Als Ergebnis des Vergleichs wird ein Stream-Formatcode, der dem naheliegendsten Phasenmuster entspricht, aus der Stream-Formatcodegruppe **212** durch ein Stream-Formatauswahlmodul **220** ausgewählt.

[0049] Das Umschaltmodul **222** hat die Funktion zum Umschalten von einem Stream-Formatcode, der durch die Ausführungsumgebung **224** ausgeführt wurde, zu dem ausgewählten Stream-Formatcode auf der Grundlage des Vergleichsergebnisses durch das Phasenmuster-Vergleichsmodul **218** und seiner Ausführung durch die Ausführungsumgebung **224**.

[0050] Zu diesem Zeitpunkt verringert das Umschaltmodul **222** die Programmunterbrechungsdauer durch Einrichten eines geeigneten alternativen Task, wenn von dem gegenwärtigen Stream-Formatcode zu dem nächsten Stream-Formatcode umgeschaltet wird. Die Einzelheiten dieses Prozesses werden unter Verwendung eines Ablaufplans usw. später beschrieben.

[0051] Im Folgenden wird der Ablauf der Sammlung statistischer Informationen und der Vorverarbeitung unter Bezugnahme auf einen Ablaufplan in **Fig. 5** beschrieben. Dieser Prozess wird durch eine Benutzeroperation gestartet, z.B. bevor die Routervorrichtung **100** betrieben wird.

[0052] Im Schritt 502 sammelt das Modul **202** zur Sammlung statistischer Informationen Verkehr, der an dem Ethernet-Stack & Ethernet-Port **110** eintrifft, in bestimmten Intervallen. Das Modul **202** zur Sammlung statistischer Informationen verwendet ein SNMP-Protokoll, einen Paketanalyse-Befehl mit der Bezeichnung tcpdump, ein kommerzielles Werkzeug wie etwa „NetFlow Tracker“, das von Fluke Networks erhalten werden kann, oder dergleichen. Die dabei gesammelte Information ist das Verhältnis von Verkehr, wie etwa Mail, FTP, bewegte Bilder und Web-Verkehr. Das Modul **202** zur Sammlung statistischer Informationen sammelt typischerweise Verkehrsinformationen an wenigstens einem Wochentag oder einem Feiertag und speichert sie in einem Festplattenlaufwerk oder dergleichen eines (nicht gezeigten) Computersystems. **Fig. 6** veranschaulicht schematisch einen Zustand, bei dem sich Verkehrsinformationen zeitlich ändern. Obwohl dabei Mail, FTP, bewegte Bilder und Web-Verkehr beispielhaft gezeigt sind, sollte klar sein, dass sie lediglich Beispiele darstellen und tatsächlich verschiedene Arten von Verkehr möglich sind.

[0053] Im Schritt 504 extrahiert das Phasenmuster-Extraktionsmodul **204** eine Vielzahl von typischen Phasenmustern **702** und **704** bis **706** wie in **Fig. 7** gezeigt aus den Verkehrsinformationen, die durch das Modul **202** zur Sammlung statistischer Informationen gesammelt werden.

[0054] Die Phasenmuster **702** und **704** bis **706** können Muster von Verkehrsinformationen sein, die in regelmäßigen Intervallen extrahiert werden oder in einem typischen Intervall aus einem Cluster extrahiert werden, das durch k-Mean-Clusterbildung der Muster von in regelmäßigen Intervallen extrahierten Verkehrsinformationen erhalten wird. Die Phasenmuster (Phasen) **702** und **704** bis **706** werden vorzugsweise in dem RAM **108** als die Phasenmustergruppe **206** gespeichert. Den auf diese Weise erzeugten Phasenmustern **702** und **704** bis **706** werden eindeutige Phasenkennungen (Phasen-IDs) einzeln zugewiesen.

[0055] Im Schritt 506 erzeugt der Compiler **210** Stream-Formatcodes für einzelne Phasenmuster (Phasen) in der Phasenmustergruppe **206** unter Bezugnahme auf die Elemente in der Konfigurationstabelle 208. Mit anderen Worten leitet der Compiler 210 eine Ressourcenabbildung auf Grundlage der Stream-Verarbeitung für jede Phase ab.

[0056] **Fig. 8** veranschaulicht ein Beispiel eines grundlegenden Stream-Programms, bei dem wie in der Zeichnung dargestellt ein IP-Protokollstapel mit einem Ethernet-Protokollstapel verbunden ist. Der IP-Protokollstapel spaltet sich in einen TCP-Protokollstapel und einen UDP-Protokollstapel. Der TCP-Protokollstapel und der UDP-Protokollstapel sind mit einer Virus-Abtastung und außerdem mit einem SSL-Protokollstapel verbunden. Der SSL-Protokollstapel ist mit der Virus-Abtastung verbunden. Die Virus-Abtastung ist mit einem XML-Beschleuniger (Akzelerator) verbunden.

[0057] Obwohl Belastungen an den Verarbeitungselementen des Stream-Programms in **Fig. 8** nicht berücksichtigt sind, ändern sich die Belastungen an den Verarbeitungselementen des Stream-Programms mit der Phase spezieller Verkehrsinformationen wie in Phase 902 von **Fig. 9**. Das heißt, in **Fig. 9** geben die Größen der Kästen, die die Verarbeitungselemente umgeben, die Belastungen an.

[0058] **Fig. 10** veranschaulicht eine andere Verteilung von Belastungen an den Verarbeitungselementen in der Phase 1002 von anderen Verkehrsinformationen. Das heißt, wenn sich die Phase von Verkehrsinformationen ändert, werden übermäßige Belastungen an bestimmten Verarbeitungselementen des Stream-Programms ausgeübt, was als ein Flaschenhals zur Verringerung der gesamten Verarbeitungsgeschwindigkeit wirkt.

[0059] Deswegen wird der Prozess des Stream-Programms durch Ressourcenabbildung optimiert. Das Folgende stellt hierfür ein Verfahren dar.

- Parallelisierung von Daten und Festlegung einer Pipeline

[0060] Zu diesem Zeitpunkt bietet auch eine Verwendung der Task-Parallelisierung die Vorteile der Verbesserung des Speicherzugriffsorts, der Unterdrückung eines Kommunikationswettbewerbs und der Verdeckung einer Kommunikationsverzögerung.

- Ausgleichen einer Belastung pro Ressource und des Pipeline-Pitch durch Teilen eines Protokollstapels, der bei der Verarbeitung schwer wiegt, in mehrere Stufen oder durch Integrieren leichter Protokollstapel in eine Stufe.

[0061] Die Einzelheiten des Prozesses werden deswegen später beschrieben, da sie etwas kompliziert sind. Die erzeugte Stream-Formatcodegruppe **212** wird vorzugsweise in dem RAM **108** gespeichert.

[0062] Im Schritt 508 wird die Übergangstabelle 214 der Stream-Formatcodes der auf diese Weise erzeugten Stream-Formatcodegruppe **212** erzeugt. Die Übergangstabelle 214 enthält die derzeitigen Tasks in Stream-Formatcodes der einzelnen Prozessoren für jede Phase des Profils und alternative Tasks zum Umschalten von einem Stream-Formatcode zu einem anderen Stream-Formatcode. Die Einzelheiten der Übergangstabelle 214 werden ebenfalls später beschrieben.

[0063] Im Folgenden wird ein Prozess zum Erzeugen von Stream-Formatcodes unter Bezugnahme auf die **Fig. 11** usw. beschrieben.

[0064] In **Fig. 11** werden die Systemumgebung **209**, d.h. Ressourcenbegrenzung (Hardwarekonfiguration), und die Konfigurationstabelle 208 im Voraus gebildet. Ein Beispiel eines Stream-Graphen, der Funktionsblöcke A, B, C und D und eine Ressourcenbegrenzung enthält, ist in **Fig. 12** gezeigt. Die Systemumgebung **209** gibt dabei eine Systemkonfiguration an, die mit dem Ethernet-Stack & Ethernet-Port **110** in **Fig. 1** verbunden ist.

[0065] Der Compiler **210** führt im Schritt 1102 eine Filterung aus. Das heißt, der Compiler 210 extrahiert lediglich ausführbare Muster aus der vorgegebenen Hardwarekonfiguration und der Konfigurationstabelle 208, um eine Optimierungstabelle (A) zu erzeugen.

[0066] Im Schritt 1104 erzeugt der Compiler **210** eine Ausführungsmustergruppe (B), in der ein Ausführungsmuster mit dem verkürzten Pipeline-Pitch einzelnen UDOPs in dem Stream-Graphen unter Bezugnahme auf die Optimierungstabelle (A) zugewiesen wird. Ein Beispiel, in dem Ausführungsmuster zu den einzelnen Blöcken des Stream-Graphen zugewiesen sind, ist in **Fig. 13** gezeigt.

[0067] Im Folgenden bestimmt der Compiler **210** im Schritt 1106, ob die Ausführungsmustergruppe (B) die vorgegebene Ressourcenbeschränkung erfüllt.

[0068] Wenn der Compiler **210** im Schritt 1106 feststellt, dass die Ausführungsmustergruppe (B) die vorgegebene Ressourcenbeschränkung erfüllt, wird dieser Prozess abgeschlossen.

[0069] Wenn der Compiler **210** im Schritt 1106 feststellt, dass die Ausführungsmustergruppe (B) die vorgegebene Ressourcenbeschränkung nicht erfüllt, geht der Prozess zum Schritt 1108, in dem er eine Liste (C) erzeugt, in der die Ausführungsmuster in der Ausführungsmustergruppe (B) in der Reihenfolge des Pipeline-Pitch sortiert sind.

[0070] Im Folgenden geht der Prozess zum Schritt 1110, in dem der Compiler **210** einen UDOP (D), der ein Ausführungsmuster mit dem kürzesten Pipeline-Pitch aufweist, aus der Liste (C) auswählt.

[0071] Im Folgenden geht der Prozess zum Schritt 1112, in dem der Compiler **210** für den UDOP (D) feststellt, ob ein Ausführungsmuster, das einen geringeren Ressourcenverbrauch (nächster Kandidat) (E) aufweist, in der Optimierungstabelle (A) vorhanden ist.

[0072] Wenn eine positive Feststellung erfolgt, geht der Prozess zum Schritt 1114, in dem der Compiler **210** für den UDOP (D) feststellt, ob das Pipeline-Pitch (nächster Kandidat) (E) kleiner als der maximale Längenwert in der Liste (C) ist.

[0073] Wenn eine positive Feststellung erfolgt, geht der Prozess zum Schritt 1116, in dem der Compiler **210** das Ausführungsmuster (nächster Kandidat) (E) als ein neues Ausführungsmuster des UDOP (D) zuweist, um die Ausführungsmustergruppe (B) zu aktualisieren.

[0074] Der Prozess kehrt vom Schritt 116 zu der Feststellung im Schritt 1106 zurück.

[0075] Wenn die Feststellung im Schritt 1112 negativ ist, geht der Prozess zum Schritt 1118, in dem der Compiler **210** den relevanten UDOP aus der Liste (C) entfernt.

[0076] Anschließend geht der Prozess zum Schritt 1120, in dem der Compiler **210** feststellt, ob ein Element in der Liste (C) vorhanden ist. Wenn eine positive Feststellung erfolgt, kehrt der Prozess zum Schritt 1108 zurück.

[0077] Wenn im Schritt 1120 festgestellt wird, dass in der Liste (C) kein Element vorhanden ist, geht der Prozess zum Schritt 1122, in dem der Compiler **210** eine Liste (F) erzeugt, in der die Ausführungsmuster in der Ausführungsmustergruppe (B) in der Reihenfolge der Differenz zwischen dem längsten Pipeline-Pitch der Ausführungsmustergruppe (B) und dem Pipeline-Pitch des nächsten Kandidaten sortiert sind.

[0078] Anschließend stellt der Compiler **210** im Schritt 1124 fest, ob Ressourcen, die von einem Ausführungsmuster (G) gefordert werden, bei dem die Differenz zwischen den Pipeline-Pitches in der Liste (F) am kleinsten ist, kleiner als gebündelte vorhandene Ressourcen sind.

[0079] Wenn eine positive Feststellung erfolgt, geht der Prozess zum Schritt 1126, in dem der Compiler **210** das Ausführungsmuster (G) als ein neues Ausführungsmuster zuweist, um die Ausführungsmustergruppe (B) zu aktualisieren, und geht zum Schritt 1106. Wenn eine negative Feststellung erfolgt, entfernt der Compiler **210** den relevanten UDOP im Schritt 1128 aus der Liste (F) und kehrt zum Schritt 1122 zurück.

[0080] **Fig. 14** ist ein Schaubild, das ein Beispiel einer derartigen Optimierung durch die Ersetzung der Ausführungsmustergruppe zeigt. In **Fig. 14** wird D4 durch D5 ersetzt, um die Ressourcenbegrenzung aufzuheben.

[0081] Nachdem die Ressourcenzuweisung auf diese Weise ausgeführt wurde, werden die einzelnen Stream-Formatcodes als die Stream-Formatcodegruppe **212** vorzugsweise in dem RAM **108** gespeichert. Den einzelnen Tasks mit den Stream-Formatcodes werden eindeutige Task-Kennungen zugewiesen.

[0082] Anschließend wird der Prozess zum Auswählen von alternativen Tasks, wenn zwischen Phasen umgeschaltet wird, unter Bezugnahme auf einen Ablaufplan in **Fig. 15** beschrieben. Dieser Prozess wird durch einen vorgegebenen Thread, der in dem Prozessor **104** oder **106** läuft, ausgeführt.

[0083] Vor der Beschreibung des Ablaufpans in **Fig. 15** werden im Folgenden die Definitionen von Zeichen oder mathematischen Ausdrücken beschrieben.

Definition

task(b,r): Task der Ressource r in Phase b

Pre(t): eine Gruppe von vorhergehenden Tasks des Task t

Post(t): eine Gruppe von nachfolgenden Tasks des Task t

*1: Startzeitpunkt des Tasks t in Phase b

start(b,t) = max{start(b,p):p∈Pre(t)} + Pitch

Pitch: Pipeline-Pitch (Task-Ausführungszeitpunkt)

*2: cost(t,r) = D + C + T

D = max{0,start(a,task(a,r)) - start(b,task(t)}

wobei D Kosten sind, die eine Leerlaufzeit bis zum Beginn der Ausführung eines alternativen Tasks enthalten.

C = max{delay(i,r),delay(r,j):i∈Deputize(a,b,s),s∈Pre(t),j∈Resource(b,u),u∈Post(t)}

Resource(b,t): eine Gruppe von Ressourcen, die für den Task t in Phase b zuständig sind

Deputize(a,b,t): eine Gruppe von Ressourcen, die für den Task b wirken, wenn die Phase a zu Phase b umgeschaltet wird

delay(a,b): die Zeit, nachdem eine Ressource a eine Datenübertragung beginnt, bis eine Ressource b den Datenempfang beendet

T = change(task(a,r),t) + change(t,task(b,r))

change(t1,12): 0, wenn der Task t1 gleich der Zeit t2 ist, wenn das nicht der Fall ist, TC (die Kosten einer Task-Umschaltung, eine Konstante)

[0084] In dem Ablaufplan von **Fig. 15** werden im Schritt 1502 eine Gruppe von Ressourcen R und Phasen a und b eingegeben. Zu diesem Zeitpunkt enthält die Gruppe von Ressourcen R Ressourcenkennungen und Kommunikationsverzögerungen zwischen den Ressourcen. Die Phasen a und b enthalten Task-Kennungen, Ressourcenkennungen, die für die einzelnen Tasks zuständig sind, Gruppen von vorhergehenden und nachfolgenden Tasks der einzelnen Tasks und Task-Ausführungszeitpunkte. Das entspricht dem Schritt 508 von **Fig. 5**.

[0085] Im Schritt 1504 wird eine Liste Tb erzeugt, in der Tasks in der Phase b (genauer Tasks in einem Stream-Graphen, der der Phase b entspricht) in aufsteigender Reihenfolge des Task-Startzeitpunkts sortiert sind. Der Task-Startzeitpunkt ist durch *1, das oben beschrieben wurde, definiert.

[0086] Im Schritt 1506 wird der erste Task auf t gesetzt.

[0087] Im Folgenden sind der Schritt 1508, der Schritt 1510, der Schritt 1512, der Schritt 1514 und der Schritt 1516 Prozesse für Ressourcen rb, die in Phase b für t zuständig sind.

[0088] Im Schritt 1510 wird die Ressource r, die unter den in R enthaltenen Ressourcen r die geringsten Kosten cost(t, r) aufweisen und von der gleichen Art ist wie rb, auf rb' gesetzt. Die Art der Ressource gibt einen Mehrzweck-Prozessor, einen Akzelerator, der eine grafische Verarbeitungseinheit darstellt usw. an. Cost(t, r) ist durch *2, das oben beschrieben wurde, definiert.

[0089] Im Schritt 1512 wird der alternative Task von rb' auf t gesetzt. Anschließend wird im Schritt 1514 rb' aus R gelöscht.

[0090] Dadurch kehrt der Prozessor zum Schritt 1508 zurück und der Schritt 1510, der Schritt 1512, der Schritt 1514 und der Schritt 1516 werden wiederholt, solange die Ressource rb, die in Phase b für t zuständig ist, vorhanden ist.

[0091] Im Schritt 1518 wird das erste Element t aus Tb gelöscht und im Schritt 1520 wird festgestellt, ob Tb leer ist. Wenn eine negative Feststellung erfolgt, kehrt der Prozess zum Schritt 1506 zurück. Das erste Element im Schritt 1506 bezieht sich auf das nächste Element, nachdem das erste Element t im Schritt 1518 aus Tb gelöscht wurde.

[0092] Wenn im Schritt 1520 festgestellt wurde, dass Tb leer ist, wird eine Gruppe alternativer Ressourcen des einzelnen Tasks erhalten. Das heißt, da alternative Tasks t der Ressource rb' im Schritt 1512 festgelegt werden, kann eine Gruppe derartiger Tasks (rb', t) erhalten werden {Schritt 1522}.

[0093] Der Prozess in dem Ablaufplan von **Fig. 15** wird an allen Kombinationen von verschiedenen Phasen a und b ausgeführt. Wie in **Fig. 16** gezeigt werden als ein Ergebnis eine Korrelationstabelle der Phasenkennungen und des vorhandenen Tasks und eine Korrelationstabelle von Phasenkennungen und Kennungen alternativer Tasks für alle verfügbaren Ressourcen, die mit der Routervorrichtung 100 verbunden sind, erzeugt. Die Korrelationstabellen von Phasenkennungen und Kennungen alternativer Tasks werden als zweidimensionale Tabellen zum Ausdrücken eines zweistufigen Übergangs präsentiert. Diese Korrelationstabellen werden in **Fig. 2** als die Übergangstabelle 214 gezeigt.

[0094] Das heißt, die Korrelationstabellen von Phasenkennungen und Kennungen vorhandener Tasks werden im Schritt 506 von **Fig. 5** erzeugt und die Korrelationstabellen von Phasenkennungen und Kennungen alternativer Tasks werden im Schritt 508 von **Fig. 5** erzeugt.

[0095] Programme, die an den einzelnen Ressourcen ausgeführt werden, können umgeschaltet werden, indem ein Wrapper gebildet wird, der zwischen Funktionen umschaltet, die auf der Grundlage einer Task-Kennung aufgerufen werden.

[0096] Obwohl die oben beschriebenen Tabellen Tabellen allgemeiner Tasks sind, kann außerdem eine Tabelle von Überwachungs-Tasks bereitgestellt werden. Diese hält Schwellenwerte zur Verwendung beim Festlegen der Phasenumschaltung, die Gruppen von Zeitpunkten und Phasenkennungen zum Umschalten auf Grundlage einer Zeitzone und Gruppen von Schwerpunkten und Phasenkennungen für eine k-Means-Clusterbildung darstellen.

[0097] **Fig. 17** ist ein Ablaufplan, der die Operationen des Phasenmuster-Messmoduls **216** und des Phasenmuster-Vergleichsmoduls **218** zeigt, nachdem die Stream-Formatcodegruppe **212** und die Übergangstabellen 214 gebildet wurden.

[0098] In **Fig. 17** misst das Phasenmuster-Messmodul **216** im Schritt 1702 die Belastungen, d.h. sammelt Verkehr, der an dem Ethernet-Stack & Ethernet-Port **110** eingetroffen ist. Das Phasenmuster-Vergleichsmodul **218** stellt im Schritt 1704 fest, ob zwischen den Phasen umgeschaltet wird, indem der Abstand zwischen der Phase des gemessenen Verkehrs und einer Phase, die gegenwärtig ausgewählt ist, berechnet wird. Das heißt, wenn der Abstand in einem vorgegebenen Schwellenwert liegt, legt das Phasenmuster-Vergleichsmodul **218** fest, die Phase nicht umzuschalten, und kehrt zum Schritt 1702 zurück. Wenn der Abstand größer als der vorgegebene Schwellenwert oder gleich diesem ist, legt das Phasenmuster-Vergleichsmodul **218** fest, die Phase umzuschalten und geht zum Schritt 1706.

[0099] Wie in **Fig. 7** gezeigt kann die Phase als ein Merkmalvektor mit dem Verhältnis des Verkehrs betrachtet werden, d.h. Mail, FTP, bewegte Bilder und Web-Verkehr, und deswegen können solche Werte wie Euklidscher Abstand, Manhattanabstand und inneres Produkt zwischen den Merkmalvektoren definiert werden. Es sollte deswegen festgestellt werden, ob ein derartiges inneres Produkt oder Abstand innerhalb eines Schwellenwerts liegt.

[0100] Im Schritt 1706 sendet das Phasenmuster-Vergleichsmodul **218** einen Befehl zum Umschalten der Phase der ersten Ressource in dem Graphen an das Stream-Formatauswahlmodul **220**. Dann kehrt der Prozess zum Schritt 1702 zurück.

[0101] **Fig. 18** ist ein Schaubild, das einen Ablaufplan für die Operationen des Stream-Formatauswahlmoduls **220**, des Umschaltmoduls **222** und der Ausführungsumgebung **224** zeigt.

[0102] Im Schritt 1802 führt die Ausführungsumgebung **224** den gegenwärtigen Task in einem momentan ausgewählten Stream-Formatcode aus. Im Schritt 1804 stellt das Umschaltmodul **222** fest, ob ein Phasenumschaltbefehl von allen vorhergehenden Ressourcen empfangen wurde. Wenn eine negative Feststellung er-

folgt, kehrt der Prozess zum Schritt 1802 zurück, in dem die Ausführungsumgebung **224** den gegenwärtigen Task weiter ausführt.

[0103] Wenn im Schritt 1804 das Umschaltmodul **222** einen Phasenumschaltbefehl von allen vorhergehenden Ressourcen empfängt, geht der Prozess zum Schritt 1806, in dem ein Stream-Formatcode, der bereits durch einen Überwachungs-Task ausgewählt wurde, als ein nachfolgender Stream-Formatcode eingesetzt wird.

[0104] Da die Kennung des nachfolgenden Stream-Formatcodes in dem Phasenumschaltbefehl enthalten ist, fügt das Umschaltmodul **222** im Schritt 1806 den Phasenumschaltbefehl an das Ende der Datenwarteschlange aller nachfolgenden Ressourcen an.

[0105] Anschließend geht der Prozess zum Schritt 1808, in dem das Umschaltmodul **222** feststellt, ob der Zeitpunkt zum Starten des gegenwärtigen Tasks mit dem nachfolgenden Stream-Formatcodes erreicht wurde. Wenn im Schritt 1810 eine negative Feststellung erfolgt, führt die Ausführungsumgebung **224** einen alternativen Task in der Phasenkennung des Phasenumschaltbefehls aus und kehrt zum Schritt 1808 zurück. Der alternative Task kann aus der unmittelbar vorgehenden Phasenkennung und der gegenwärtigen Phasenkennung der aktiven Ressource bestimmt werden wie in **Fig. 16** gezeigt.

[0106] Wenn das Umschaltmodul **222** im Schritt 1808 feststellt, dass der Zeitpunkt zum Starten des gegenwärtigen Tasks mit dem nachfolgenden Stream-Formatcode erreicht wurde, geht der Prozess zum Schritt 1812.

[0107] Im Schritt 1812 stellt das Umschaltmodul **222** fest, ob Daten des alternativen Task in der Warteschlange verbleiben. Wenn eine negative Feststellung erfolgt, kehrt der Prozess im Schritt 1802 zu der Ausführung des vorhandenen Tasks zurück.

[0108] Wenn das Umschaltmodul **222** im Schritt 1812 feststellt, dass die Daten des alternativen Task in der Warteschlange verbleiben, geht der Prozess zum Schritt 1814, in dem das Umschaltmodul **222** bewirkt, dass die Ausführungsumgebung **224** den alternativen Task ausführt und die Ausgabe an eine Ressource sendet, die den nachfolgenden Task ausführen soll.

[0109] Die Phasenumschaltoperation unter Verwendung des alternativen Tasks wird hier unter Verwendung eines schematischen Beispiels anschaulicher beschrieben. Zunächst zeigt **Fig. 19** die Topologie einer Berechnungsressource, die mit dem Ethernet-Stack & Ethernet-Port **110** der Routervorrichtung **100** verbunden ist. Diese kann nicht dynamisch konfiguriert werden, da sie eine physische Topologie darstellt und keine Software ist.

[0110] **Fig. 20** zeigt die Verringerung einer Systemunterbrechungsdauer infolge einer Phasenumschaltung der Berechnungsressource in der physischen Topologie, die die Ressourcen 1 bis 6 enthält, durch das Durchlaufen eines Zustands wie in Fig. 20(2) gezeigt, in dem alternative Tasks verwendet werden, wenn von einem in Fig. 20(1) gezeigten Zustand, in dem die Ressourcen in einem Stream-Format in Phase α zugewiesen sind, zu einem in Fig. 20(3) gezeigten Zustand, in dem die Ressourcen in einem Stream-Format in Phase β zugewiesen sind, gewechselt wird.

[0111] Das heißt, um von dem Zustand in Phase α von Fig. 20(1), in dem die Ressource 1 den Task a ausführt, die Ressource 2 den Task c ausführt, die Ressource 3 den Task d ausführt, die Ressource 4 den Task e ausführt, die Ressource 5 den Task f ausführt und die Ressource 6 den Task b ausführt, zu dem Zustand in Phase β von Fig. 20(3), in dem die Ressource 1 den Task A ausführt, die Ressource 2 den Task B ausführt, die Ressource 3 den Task C ausführt, die Ressource 4 den Task E ausführt, die Ressource 5 den Task D ausführt und die Ressource 6 den Task F ausführt, zu wechseln, führt die Ressource 3 den Task D aus und sendet ihn in Fig. 20(2) wie im Schritt 1810 an die Ressource 5; führt die Ressource 3 den Task D aus und sendet ihn wie im Schritt 1814 an die Ressource 6; und anschließend führt in Fig. 20(3) die Ressource 3 den Task C aus und sendet ihn wie im Schritt 1802 an die Ressource 5. In ähnlicher Weise führt die Ressource 6 außerdem alternativ den Task C aus und sendet ihn in Fig. 20(2) zu den Ressourcen 4 und 3.

[0112] **Fig. 21** ist ein Schaubild, das einen Übergangszyklus eines Ausführungs-Task zeigt. In der Zeichnung sind die Ausführung des Task C durch die Ressource 6 und die Ausführung des Task D durch die Ressource 3 Ausführungen als alternative Tasks.

[0113] Das heißt, da die alternativen Tasks in einer Pipeline angeordnet sind, werden die Tasks von dem ersten Task abgeschlossen. Somit wirken gemäß dem grundlegenden Schema Ressourcen, die die Phase α

früher abgeschlossen haben, für die ersten Ressourcen in Phase β, und nachdem alle Ressourcen Tasks in Phase α abgeschlossen haben, werden geplante Tasks gestartet.

Bezugszeichenliste

100: Routervorrichtung
102: Bus
104: Mehrkern-Prozessor
106: Mehrkern-Prozessor
108: RAM
110: Ethernet-Port
112: Flash-ROM
202: Modul zur Sammlung statistischer Informationen
204: Phasenmuster-Extraktionsmodul
206: Phasenmustergruppe
208: Konfigurationstabelle
209: Systemumgebung
210: Compiler
212: Stream-Formatcodegruppe
214: Übergangstabelle
216: Phasenmuster-Messmodul
218: Phasenmuster-Vergleichsmodul
220: Stream-Formatauswahlmodul
222: Umschaltmodul
224: Ausführungsumgebung
702: Phasenmuster
704: Phasenmuster
706: Phasenmuster

Patentansprüche

1. Verfahren zum Steuern eines Computersystems zur dynamischen Änderung der Konfiguration des Systems durch Computerverarbeitung, so dass es für den Inhalt der Netzwerk-Verkehrsinformationen geeignet ist, wobei das Verfahren die folgenden Schritte umfasst:
im Voraus Speichern von Informationen, die Tasks und die Zuweisung von Ressourcen, mit denen die Tasks ausgeführt werden, in einem Stream-Format-Code angeben, so dass die Tasks in dem entsprechenden Stream-Format für Phasen angepasst sind, die Mustern einer Vielzahl von unterschiedlichen Inhalten der Netzwerk-Verkehrsinformationen entsprechen; Messen der Netzwerk-Verkehrsinformationen;
Auswählen von Stream-Format-Ressourcenzuweisungsinformationen, die den gemessenen Netzwerk-Verkehrsinformationen verhältnismäßig ähnlich sind, als die nächste Phase; und
Umschalten von der Stream-Format-Ressourcenzuweisung in der ausgewählten vorhandenen Phase zur Stream-Format-Ressourcenzuweisung in der nächsten Phase durch ein Umschaltmodul,
Feststellen durch das Umschaltmodul, ob der Zeitpunkt zum Starten eines Tasks mittels einer aktiven Ressource mit einem Stream-Formatcode einer gegenwärtigen Phase B noch nicht erreicht wurde,
solange durch das Umschaltmodul festgestellt wird, dass der Zeitpunkt noch nicht erreicht wurde,
Ausführen eines alternativen Tasks, einer aus der unmittelbar vorhergehenden Phase A und der gegenwärtigen Phase B der aktiven Ressource bestimmt wird.

2. Verfahren nach Anspruch 1, ferner umfassend den Schritt zum Berechnen alternativer Tasks beim Phasenwechsel für die einzelnen verfügbaren Ressourcen und zum Speichern der alternativen Tasks als eine Tabelle, wobei in dem Umschaltschritt die alternativen Tasks, die in Übereinstimmung mit den Ressourcen gespeichert sind, von der gegenwärtigen Phase und der nächsten Phase gefunden werden und alternativ ausgeführt werden.

3. Verfahren nach Anspruch 2, wobei die alternativen Tasks beim Phasenwechsel ausgewählt werden, um task-bezogene Umschaltkosten, Kosten, die eine Leerlaufzeit enthalten, bis die Ausführung des alternativen Task gestartet wird, Kosten, die die Zeit für die Übertragung und den Empfang von Daten zwischen den Ressourcen in den betreffenden Phasen enthalten, zu verringern.

4. Computersystem zum dynamischen Ändern der Konfiguration des Systems durch Computerverarbeitung, so dass es für den Inhalt von Netzwerk-Verkehrsinformationen angepasst ist, wobei das System Folgendes umfasst:
Speichermittel;
Mittel, um in den Speichermitteln Informationen zu speichern, die Tasks und die Zuweisung von Ressourcen, mit denen die Tasks ausgeführt werden, in einem Stream-Format-Code angeben, so dass die Tasks in dem entsprechenden Stream-Format für Phasen angepasst sind, die Mustern einer Vielzahl von unterschiedlichen Inhalten der Netzwerk-Verkehrsinformationen entsprechen;
Mittel zum Messen der Netzwerk-Verkehrsinformationen Mittel zum Auswählen von Informationen der Stream-Format-Ressourcenzuweisung, die den gemessenen Netzwerk-Verkehrsinformationen verhältnismäßig ähnlich sind, als die nächste Phase; und
Mittel zum Umschalten von der Stream-Format-Ressourcenzuweisung in der ausgewählten gegenwärtigen Phase zur Stream-Format-Ressourcenzuweisung in der nächsten Phase, ;
wobei die Mittel zum Umschalten feststellen, ob der Zeitpunkt zum Starten eines Tasks mittels einer aktiven Ressource mit einem Stream-Formatcode einer gegenwärtigen Phase B noch nicht erreicht wurde, und solange der Zeitpunkt noch nicht erreicht wurde, einen alternativen Task ausführen, der aus einer unmittelbar vorhergehenden Phase A und der gegenwärtigen Phase B der aktiven Ressource bestimmt wird.

5. System nach Anspruch 4, ferner umfassend Mittel zum Berechnen alternativer Tasks beim Phasenwechsel für die einzelnen verfügbaren Ressourcen und zum Speichern der alternativen Tasks als eine Tabelle, wobei die Umschaltmittel die alternativen Tasks, die in Übereinstimmung mit den Ressourcen von der gegenwärtigen Phase und der nächsten Phase gespeichert sind, finden und die alternativen Tasks alternativ ausführen.

6. System nach Anspruch 5, wobei die alternativen Tasks beim Phasenwechsel ausgewählt werden, um task-bezogene Umschaltkosten, Kosten, die eine Leerlaufzeit enthalten, bis die Ausführung des alternativen Task gestartet wird, Kosten, die die Zeit für die Übertragung und den Empfang von Daten zwischen den Ressourcen in den betreffenden Phasen enthalten, zu vermindern.

7. Programmprodukt zum Steuern eines Computersystems zur dynamischen Änderung der Konfiguration des Systems durch Computerverarbeitung, so dass es für den Inhalt von Netzwerk-Verkehrsinformationen geeignet ist, wobei das Programmprodukt Folgendes umfasst:
computernutzbaren Programmcode, um im Voraus Informationen zu speichern, die Tasks und die Zuweisung von Ressourcen, mit denen die Tasks ausgeführt werden, in einem Stream-Format-Code anzugeben, so dass die Tasks in dem entsprechenden Stream-Format für Phasen angepasst sind, die Mustern einer Vielzahl von unterschiedlichen Inhalten der Netzwerk-Verkehrsinformationen entsprechen;
computernutzbaren Programmcode zum Messen der Netzwerk-Verkehrsinformationen;
computernutzbaren Programmcode zum Auswählen von Informationen zur Stream-Format-Ressourcenzuweisung, die den gemessenen Netzwerk-Verkehrsinformationen verhältnismäßig ähnlich sind, als die nächste Phase; und
computernutzbaren Programmcode zum Umschalten von der Stream-Format-Ressourcenzuweisung in der ausgewählten gegenwärtigen Phase zur Stream-Format-Ressourcenzuweisung in der nächsten Phase, zum Feststellen, ob der Zeitpunkt zum Starten eines Tasks mittels einer aktiven Ressource mit einem Stream-Formatcode einer gegenwärtigen Phase B noch nicht erreicht wurde, und solange der Zeitpunkt noch nicht erreicht wurde, zum Ausführen eines alternativen Tasks, der aus einer unmittelbar vorhergehenden Phase A und der gegenwärtigen Phase B der aktiven Ressource bestimmt wird.

8. Programmprodukt nach Anspruch 7, das ferner computernutzbaren Programmcode zum Berechnen alternativer Tasks beim Phasenwechsel für die einzelnen verfügbaren Ressourcen und zum Speichern der alternativen Tasks als eine Tabelle umfasst, wobei bei dem Umschalten die alternativen Tasks, die in Übereinstim-

mung mit den Ressourcen gespeichert sind, von der gegenwärtigen Phase und der nächsten Phase gefunden werden und alternativ ausgeführt werden.

9. Programmprodukt nach Anspruch 8, wobei die alternativen Tasks beim Phasenwechsel ausgewählt werden, um task-bezogene Umschaltkosten, Kosten, die eine Leerlaufzeit enthalten, bis die Ausführung des alternativen Task gestartet wird, Kosten, die die Zeit für die Übertragung und den Empfang von Daten zwischen den Ressourcen in den betreffenden Phasen enthalten, zu verringern.

Es folgen 17 Seiten Zeichnungen

Anhängende Zeichnungen

FIG. 1

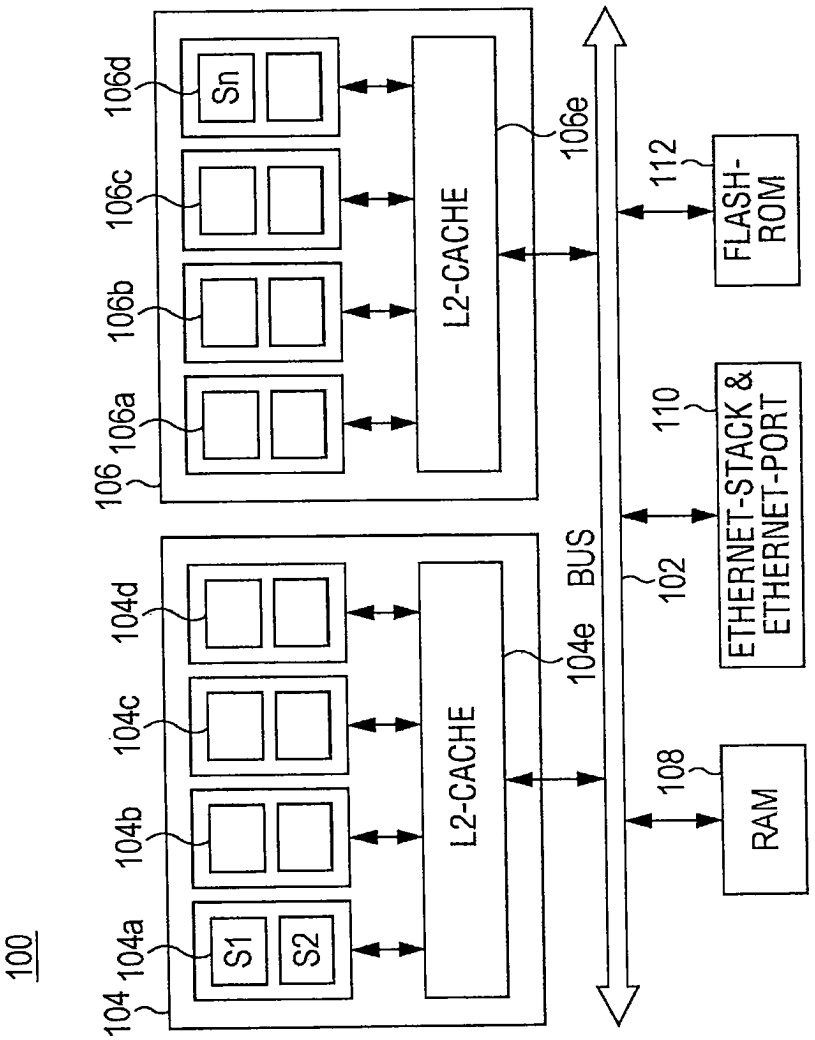


FIG. 2

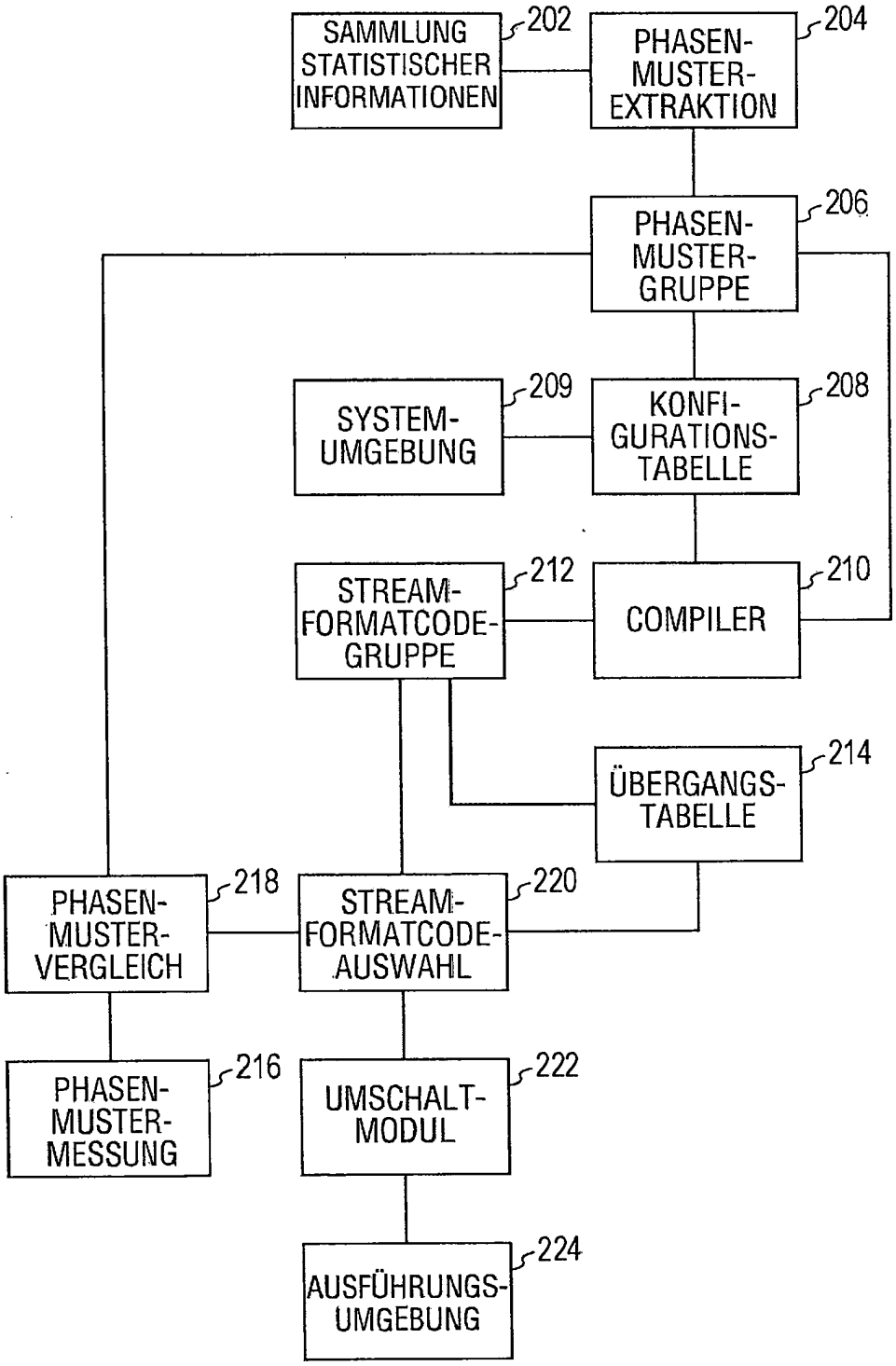

SAMMLUNG STATISTISCHER INFORMATIONEN
202
PHASEN-MUSTER-EXTRAKTION
204
PHASEN-MUSTER-GRUPPE
206
SYSTEM-UMGEBUNG
209
KONFI-GURATIONS-TABELLE
208
STREAM-FORMATCODE-GRUPPE
212
COMPILER
210
ÜBERGANGS-TABELLE
214
PHASEN-MUSTER-VERGLEICH
218
STREAM-FORMATCODE-AUSWAHL
220
PHASEN-MUSTER-MESSUNG
216
UMSCHALT-MODUL
222
AUSFÜHRUNGS-UMGEBUNG
224

FIG. 3

UDOP	KERNEL	AUSFÜHRUNGSMUSTER	PITCH	RESSOURCE
A	A1	loop (1, A_x86)		x86 x 1
B	B1	loop (1, B_x86)		x86 x 1
B	B2	loop (1, B_cu)		cuda x 1
C	C1	loop (1, C_x86)		x86 x 1
D	D1	loop (3, D_x86)		x86 x 1
D	D2	parellel (3, D_x86)		x86 x 3
D	D3	loop (3, D_cu)		cuda x 1
D	D4	parellel (3, D_cu)		cuda x 3
D	D5	splitjoin(loop (1, D_x86), loop (2, D_cu))		x86 x 1, cuda x 1

FIG. 4

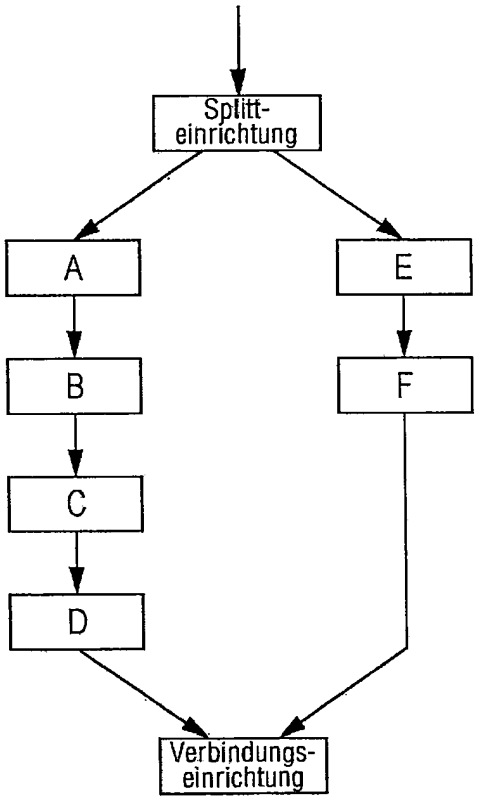


FIG. 5

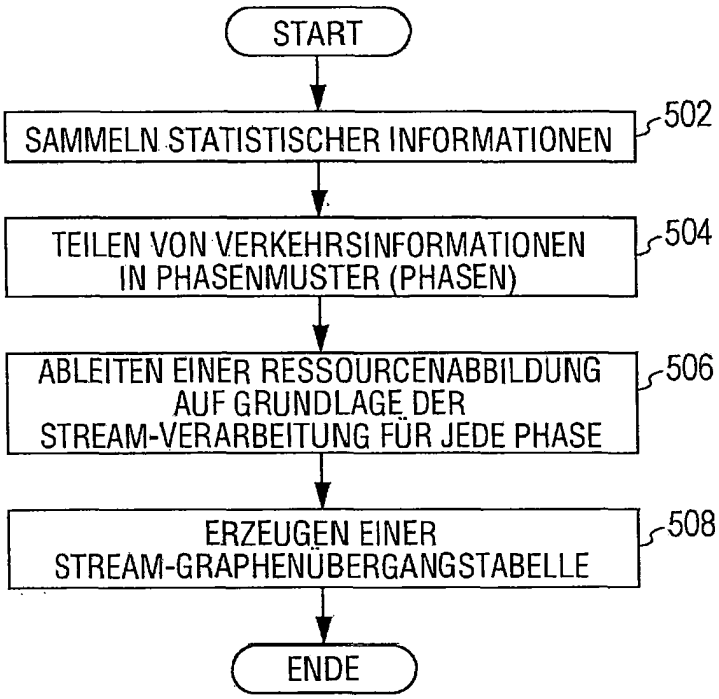

START
SAMMELN STATISTISCHER INFORMATIONEN
502
TEILEN VON VERKEHRSINFORMATIONEN IN PHASENMUSTER (PHASEN)
504
ABLEITEN EINER RESSOURCENABBILDUNG AUF GRUNDLAGE DER STREAM-VERARBEITUNG FÜR JEDE PHASE
506
ERZEUGEN EINER STREAM-GRAPHENÜBERGANGSTABELLE
508
ENDE

FIG. 6

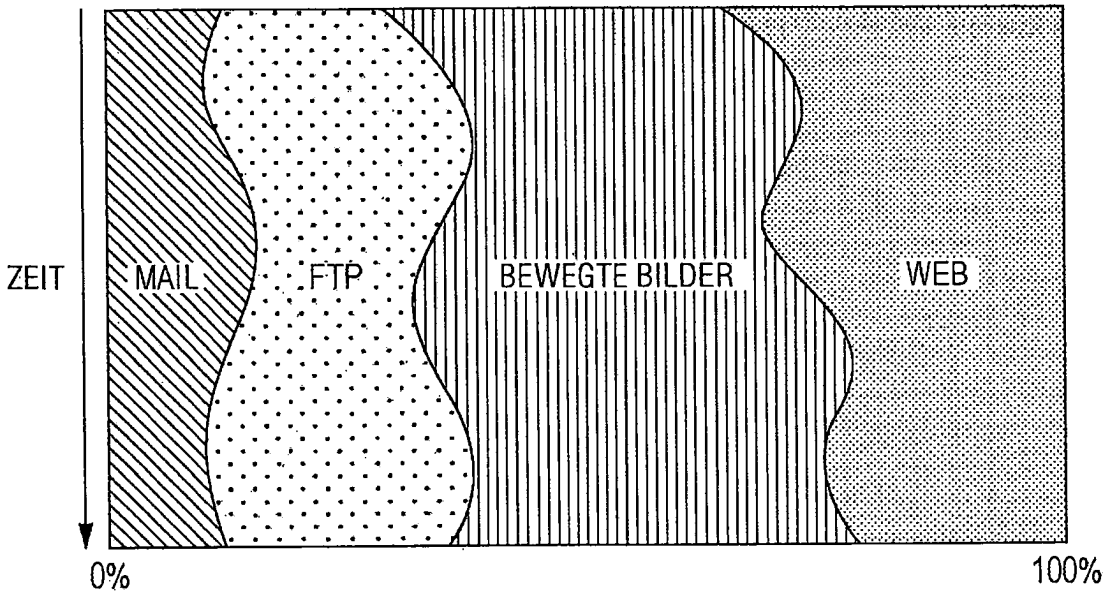


FIG. 7

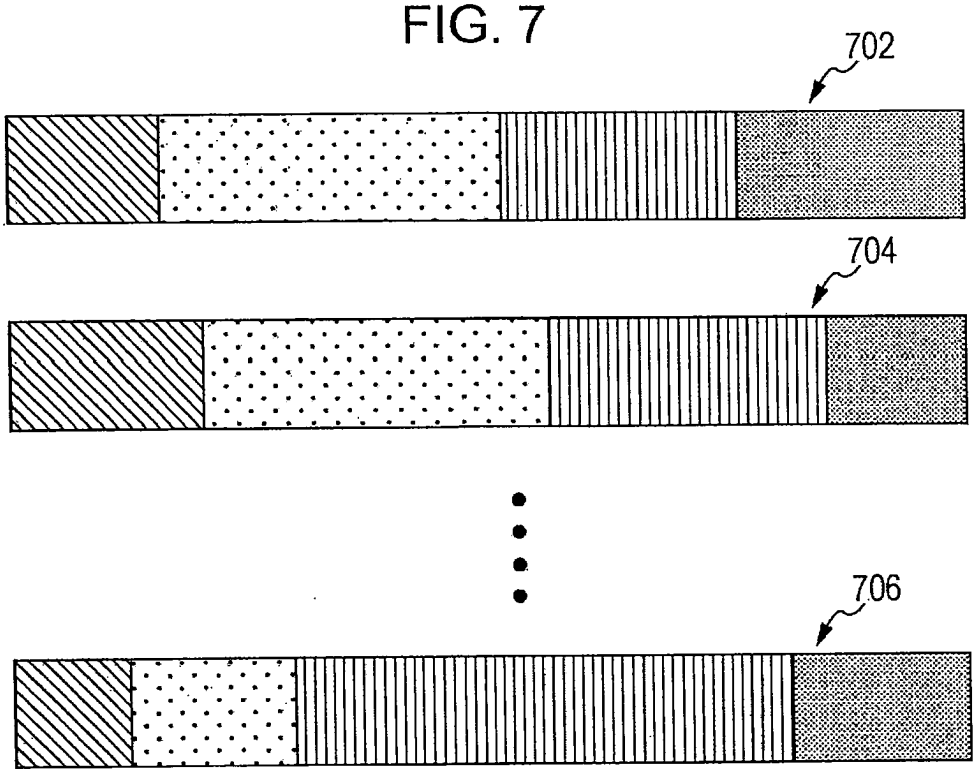


FIG. 8

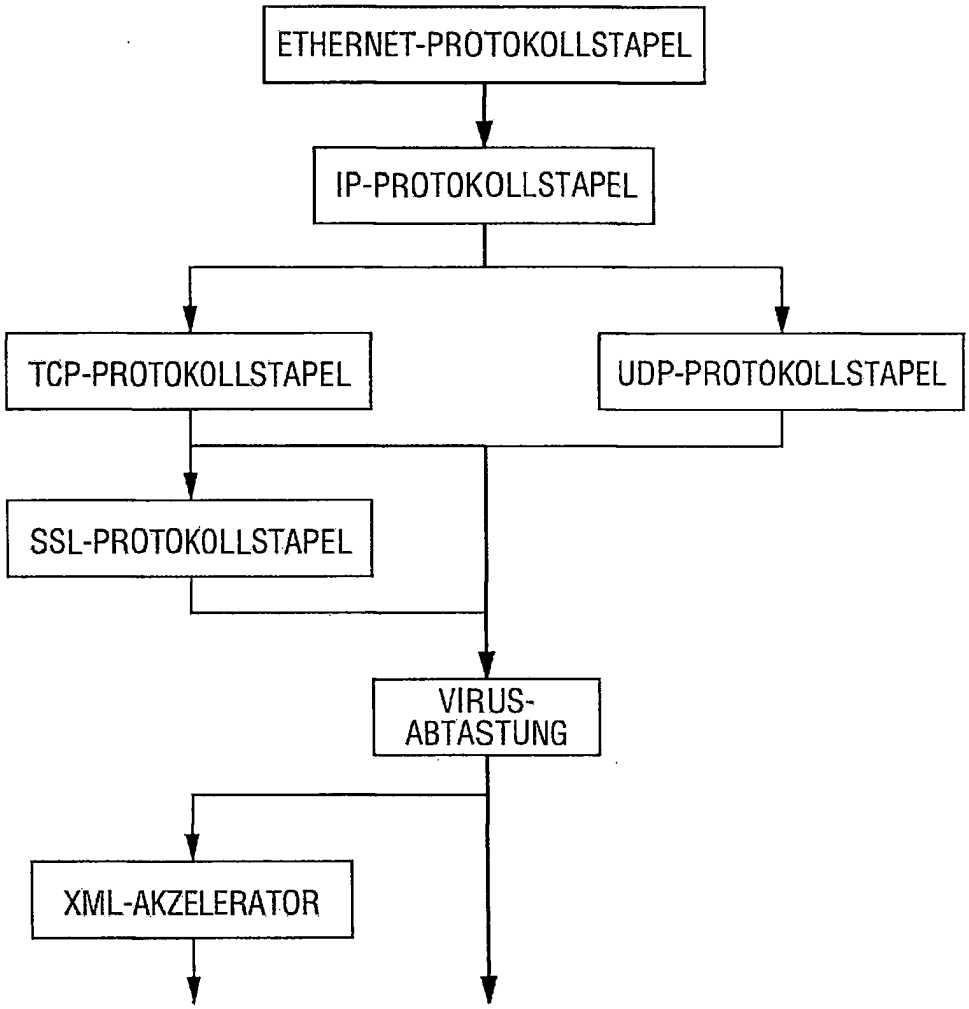

ETHERNET-PROTOKOLLSTAPEL
IP-PROTOKOLLSTAPEL
TCP-PROTOKOLLSTAPEL
UDP-PROTOKOLLSTAPEL
SSL-PROTOKOLLSTAPEL
VIRUS-
ABTASTUNG
XML-AKZELERATOR

FIG. 9

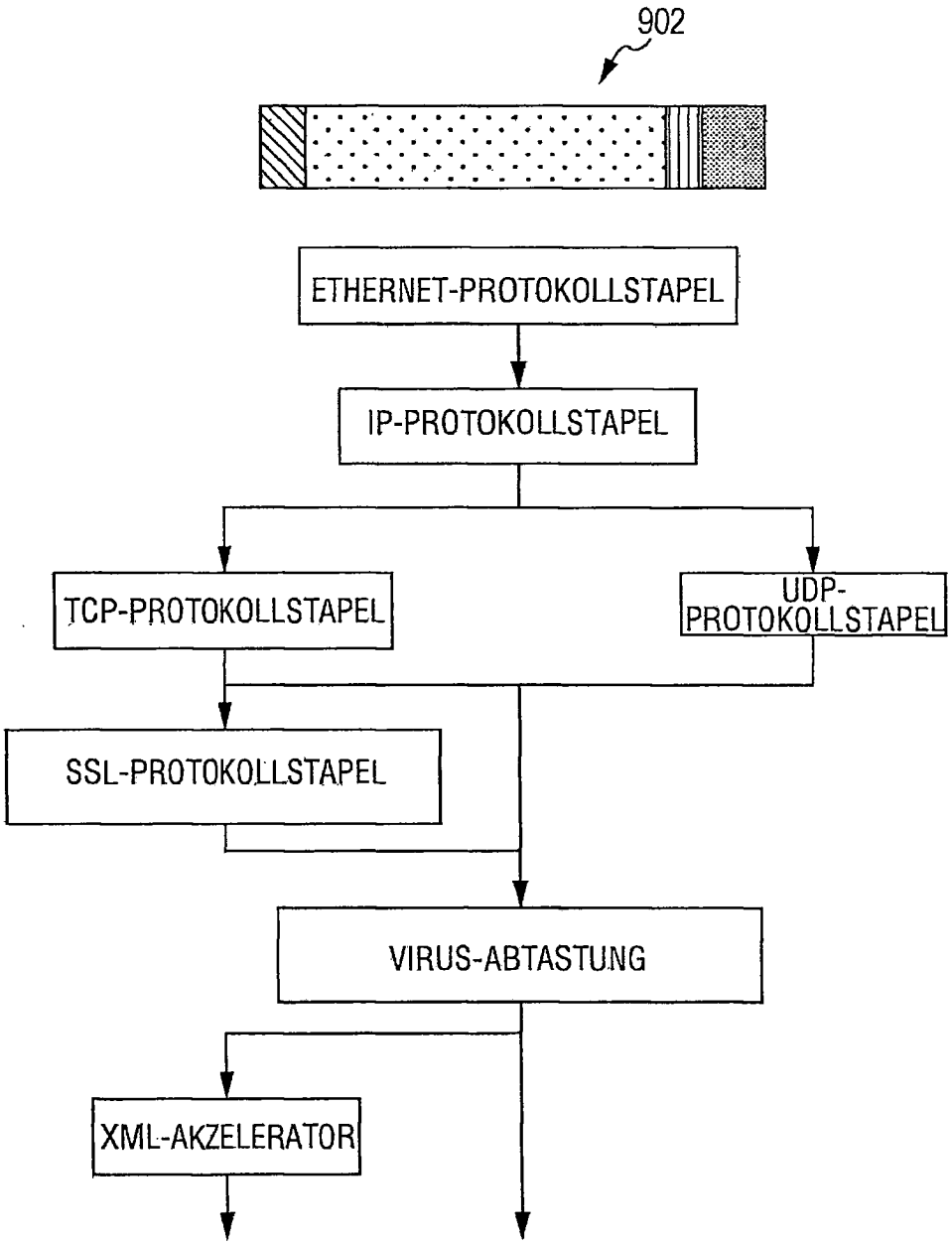

902
ETHERNET-PROTOKOLLSTAPEL
IP-PROTOKOLLSTAPEL
TCP-PROTOKOLLSTAPEL
UDP-PROTOKOLLSTAPEL
SSL-PROTOKOLLSTAPEL
VIRUS-ABTASTUNG
XML-AKZELERATOR

FIG. 10

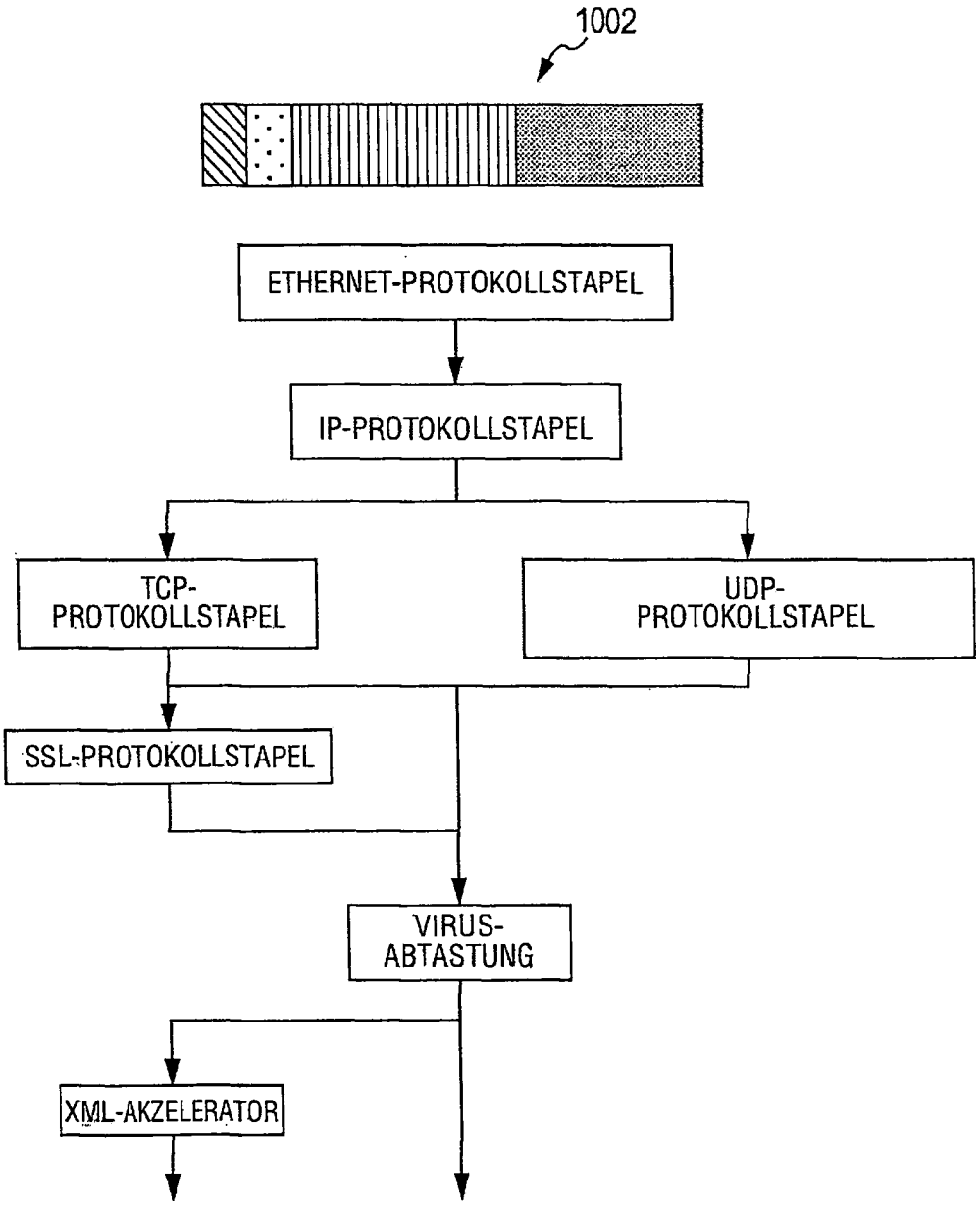

1002
ETHERNET-PROTOKOLLSTAPEL
IP-PROTOKOLLSTAPEL
TCP-
PROTOKOLLSTAPEL
UDP-
PROTOKOLLSTAPEL
SSL-PROTOKOLLSTAPEL
VIRUS-
ABTASTUNG
XML-AKZELERATOR

FIG. 11

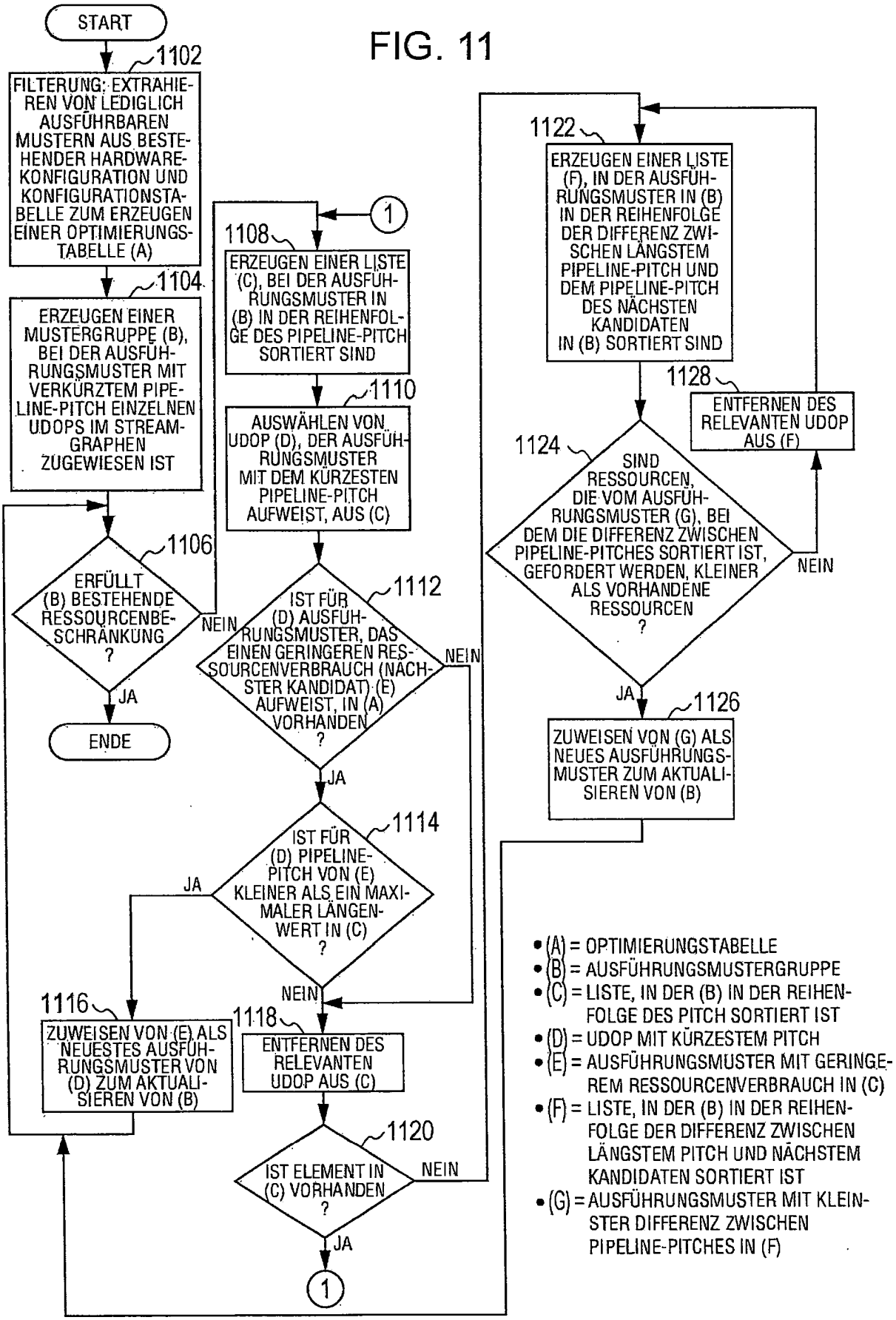

START
1102
FILTERUNG: EXTRAHIEREN VON LEDIGLICH AUSFÜHRBAREN MUSTERN AUS BESTEHENDER HARDWARE-KONFIGURATION UND KONFIGURATIONSTABELLE ZUM ERZEUGEN EINER OPTIMIERUNGSTABELLE (A)
1104
ERZEUGEN EINER MUSTERGRUPPE (B), BEI DER AUSFÜHRUNGSMUSTER MIT VERKÜRZTEM PIPELINE-PITCH EINZELNEN UDOPS IM STREAM-GRAPHEN ZUGEWIESEN IST
1106
ERFÜLLT (B) BESTEHENDE RESSOURCENBESCHRÄNKUNG ?
NEIN
JA
ENDE
1
1108
ERZEUGEN EINER LISTE (C), BEI DER AUSFÜHRUNGSMUSTER IN (B) IN DER REIHENFOLGE DES PIPELINE-PITCH SORTIERT SIND
1110
AUSWÄHLEN VON UDOP (D), DER AUSFÜHRUNGSMUSTER MIT DEM KÜRZESTEN PIPELINE-PITCH AUFWEIST, AUS (C)
1112
IST FÜR (D) AUSFÜHRUNGSMUSTER, DAS EINEN GERINGEREN RESSOURCENVERBRAUCH (NÄCHSTER KANDIDAT) (E) AUFWEIST, IN (A) VORHANDEN ?
NEIN
JA
1114
IST FÜR (D) PIPELINE-PITCH VON (E) KLEINER ALS EIN MAXIMALER LÄNGENWERT IN (C) ?
JA
NEIN
1116
ZUWEISEN VON (E) ALS NEUESTES AUSFÜHRUNGSMUSTER VON (D) ZUM AKTUALISIEREN VON (B)
1118
ENTFERNEN DES RELEVANTEN UDOP AUS (C)
1120
IST ELEMENT IN (C) VORHANDEN ?
NEIN
JA
1
1122
ERZEUGEN EINER LISTE (F), IN DER AUSFÜHRUNGSMUSTER IN (B) IN DER REIHENFOLGE DER DIFFERENZ ZWISCHEN LÄNGSTEM PIPELINE-PITCH UND DEM PIPELINE-PITCH DES NÄCHSTEN KANDIDATEN IN (B) SORTIERT SIND
1128
ENTFERNEN DES RELEVANTEN UDOP AUS (F)
1124
SIND RESSOURCEN, DIE VOM AUSFÜHRUNGSMUSTER (G), BEI DEM DIE DIFFERENZ ZWISCHEN PIPELINE-PITCHES SORTIERT IST, GEFORDERT WERDEN, KLEINER ALS VORHANDENE RESSOURCEN ?
NEIN
JA
1126
ZUWEISEN VON (G) ALS NEUES AUSFÜHRUNGSMUSTER ZUM AKTUALISIEREN VON (B)
• (A) = OPTIMIERUNGSTABELLE
• (B) = AUSFÜHRUNGSMUSTERGRUPPE
• (C) = LISTE, IN DER (B) IN DER REIHENFOLGE DES PITCH SORTIERT IST
• (D) = UDOP MIT KÜRZESTEM PITCH
• (E) = AUSFÜHRUNGSMUSTER MIT GERINGEREM RESSOURCENVERBRAUCH IN (C)
• (F) = LISTE, IN DER (B) IN DER REIHENFOLGE DER DIFFERENZ ZWISCHEN LÄNGSTEM PITCH UND NÄCHSTEM KANDIDATEN SORTIERT IST
• (G) = AUSFÜHRUNGSMUSTER MIT KLEINSTER DIFFERENZ ZWISCHEN PIPELINE-PITCHES IN (F)

FIG. 12

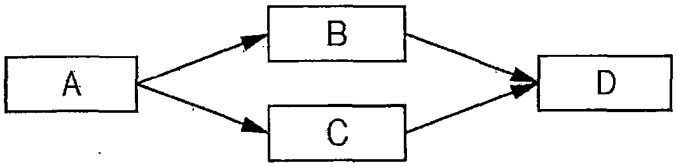


FIG. 13

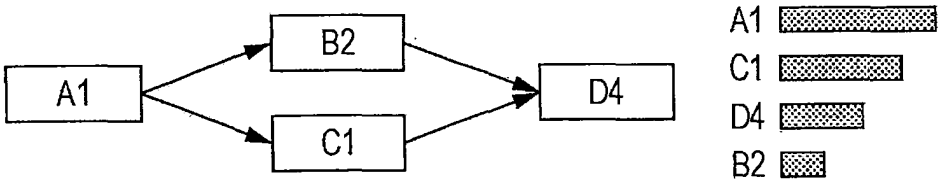


FIG. 14

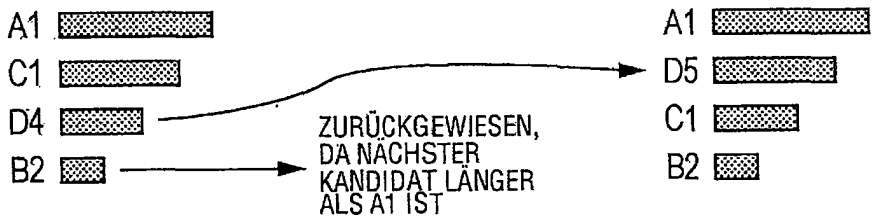


FIG. 15

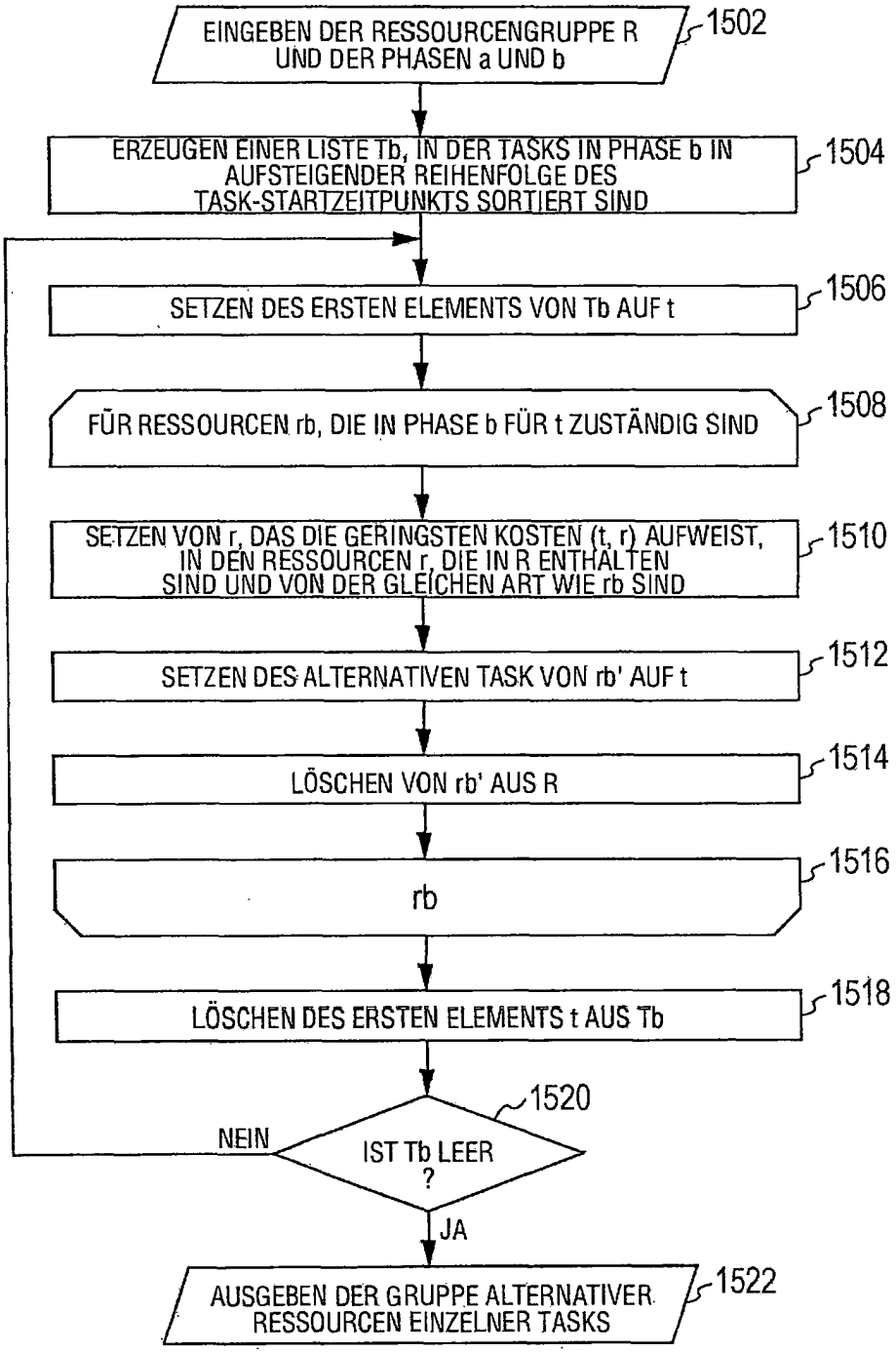

EINGEBEN DER RESSOURCENGRUPPE R UND DER PHASEN a UND b
1502
ERZEUGEN EINER LISTE Tb, IN DER TASKS IN PHASE b IN AUFSTEIGENDER REIHENFOLGE DES TASK-STARTZEITPUNKTS SORTIERT SIND
1504
SETZEN DES ERSTEN ELEMENTS VON Tb AUF t
1506
FÜR RESSOURCEN rb, DIE IN PHASE b FÜR t ZUSTÄNDIG SIND
1508
SETZEN VON r, DAS DIE GERINGSTEN KOSTEN (t, r) AUFWEIST, IN DEN RESSOURCEN r, DIE IN R ENTHALTEN SIND UND VON DER GLEICHEN ART WIE rb SIND
1510
SETZEN DES ALTERNATIVEN TASK VON rb' AUF t
1512
LÖSCHEN VON rb' AUS R
1514
rb
1516
LÖSCHEN DES ERSTEN ELEMENTS t AUS Tb
1518
IST Tb LEER ?
1520
NEIN
JA
AUSGEBEN DER GRUPPE ALTERNATIVER RESSOURCEN EINZELNER TASKS
1522

FIG. 16

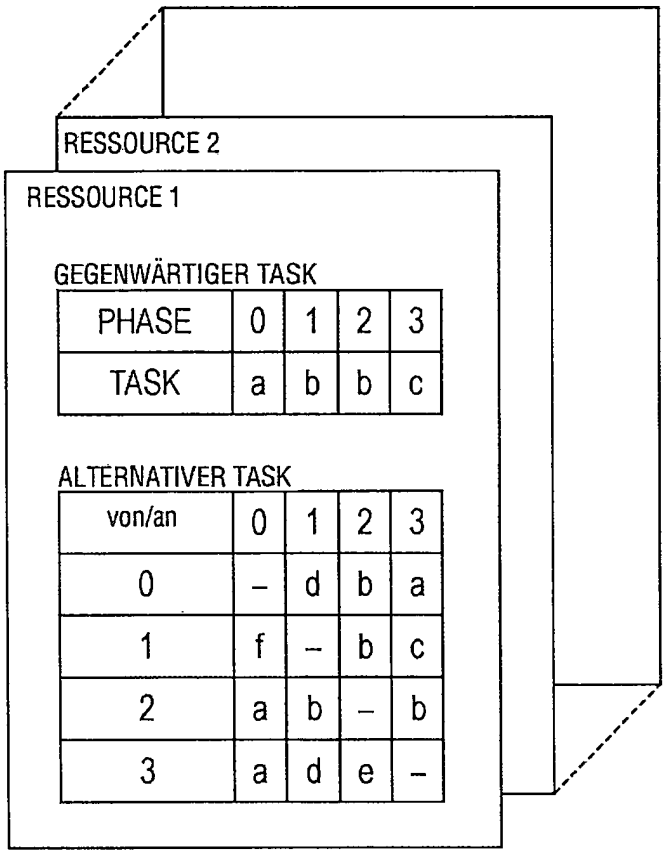

RESSOURCE 2
RESSOURCE 1
GEGENWÄRTIGER TASK
PHASE 0 1 2 3
TASK a b b c
ALTERNATIVER TASK
von/an 0 1 2 3
0 – d b a
1 f – b c
2 a b – b
3 a d e –

FIG. 17

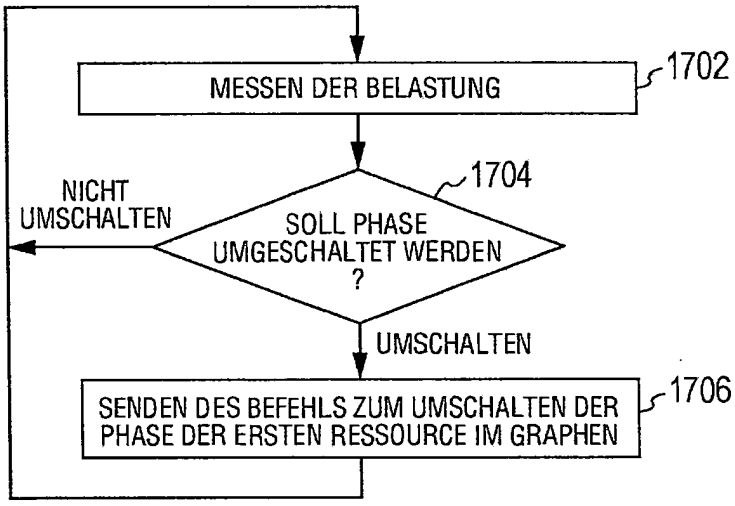

MESSEN DER BELASTUNG
1702
1704
NICHT UMSCHALTEN
SOLL PHASE UMGESCHALTET WERDEN ?
UMSCHALTEN
SENDEN DES BEFEHLS ZUM UMSCHALTEN DER PHASE DER ERSTEN RESSOURCE IM GRAPHEN
1706

FIG. 18

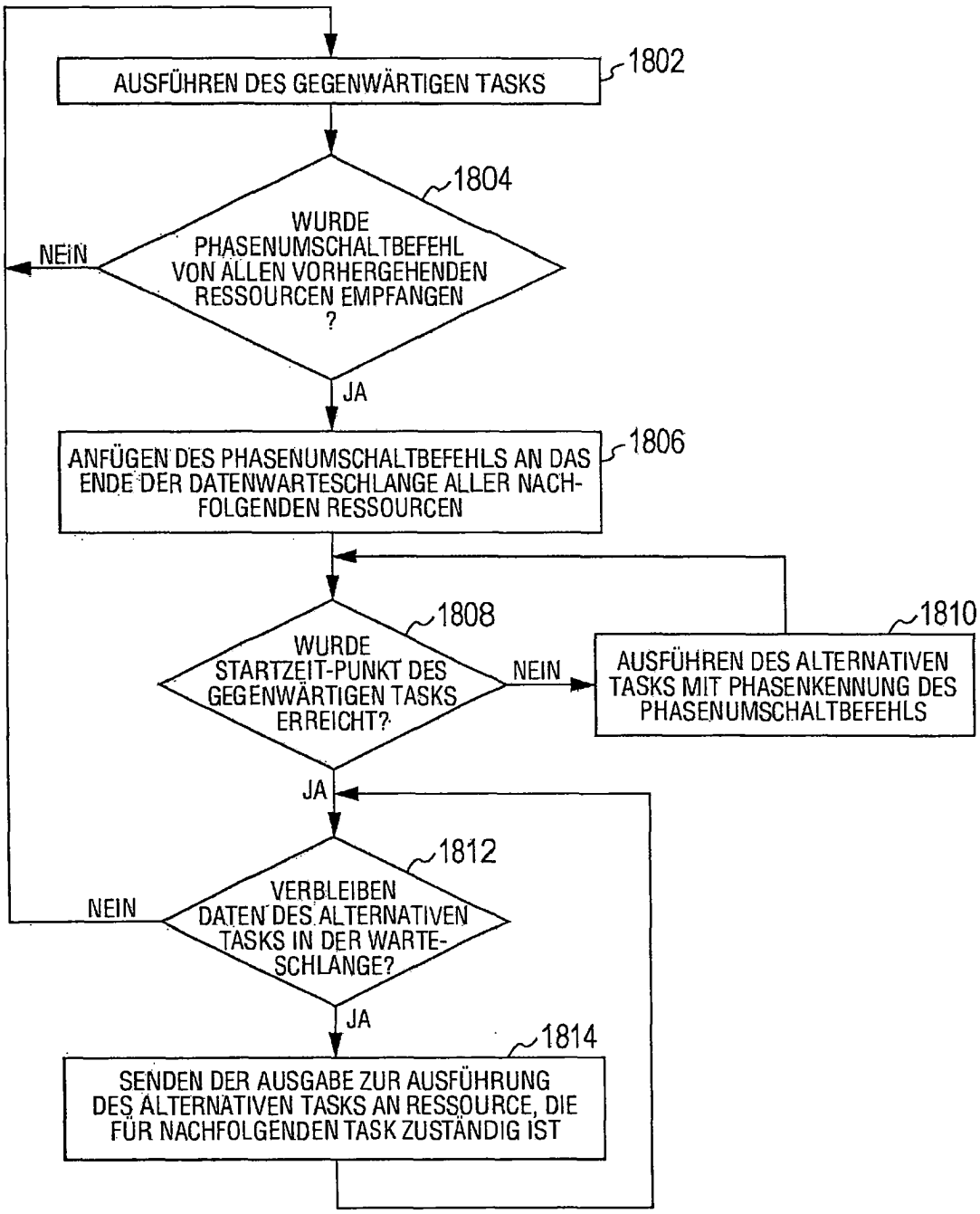

AUSFÜHREN DES GEGENWÄRTIGEN TASKS
1802
WURDE PHASENUMSCHALTBEFEHL VON ALLEN VORHERGEHENDEN RESSOURCEN EMPFANGEN ?
1804
NEIN
JA
ANFÜGEN DES PHASENUMSCHALTBEFEHLS AN DAS ENDE DER DATENWARTESCHLANGE ALLER NACHFOLGENDEN RESSOURCEN
1806
WURDE STARTZEIT-PUNKT DES GEGENWÄRTIGEN TASKS ERREICHT?
1808
NEIN
AUSFÜHREN DES ALTERNATIVEN TASKS MIT PHASENKENNUNG DES PHASENUMSCHALTBEFEHLS
1810
JA
VERBLEIBEN DATEN DES ALTERNATIVEN TASKS IN DER WARTESCHLANGE?
1812
NEIN
JA
SENDEN DER AUSGABE ZUR AUSFÜHRUNG DES ALTERNATIVEN TASKS AN RESSOURCE, DIE FÜR NACHFOLGENDEN TASK ZUSTÄNDIG IST
1814

FIG. 19

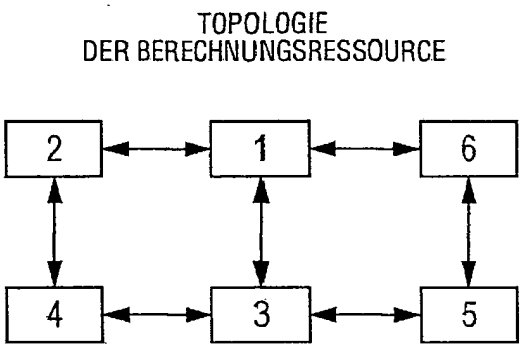


FIG. 20

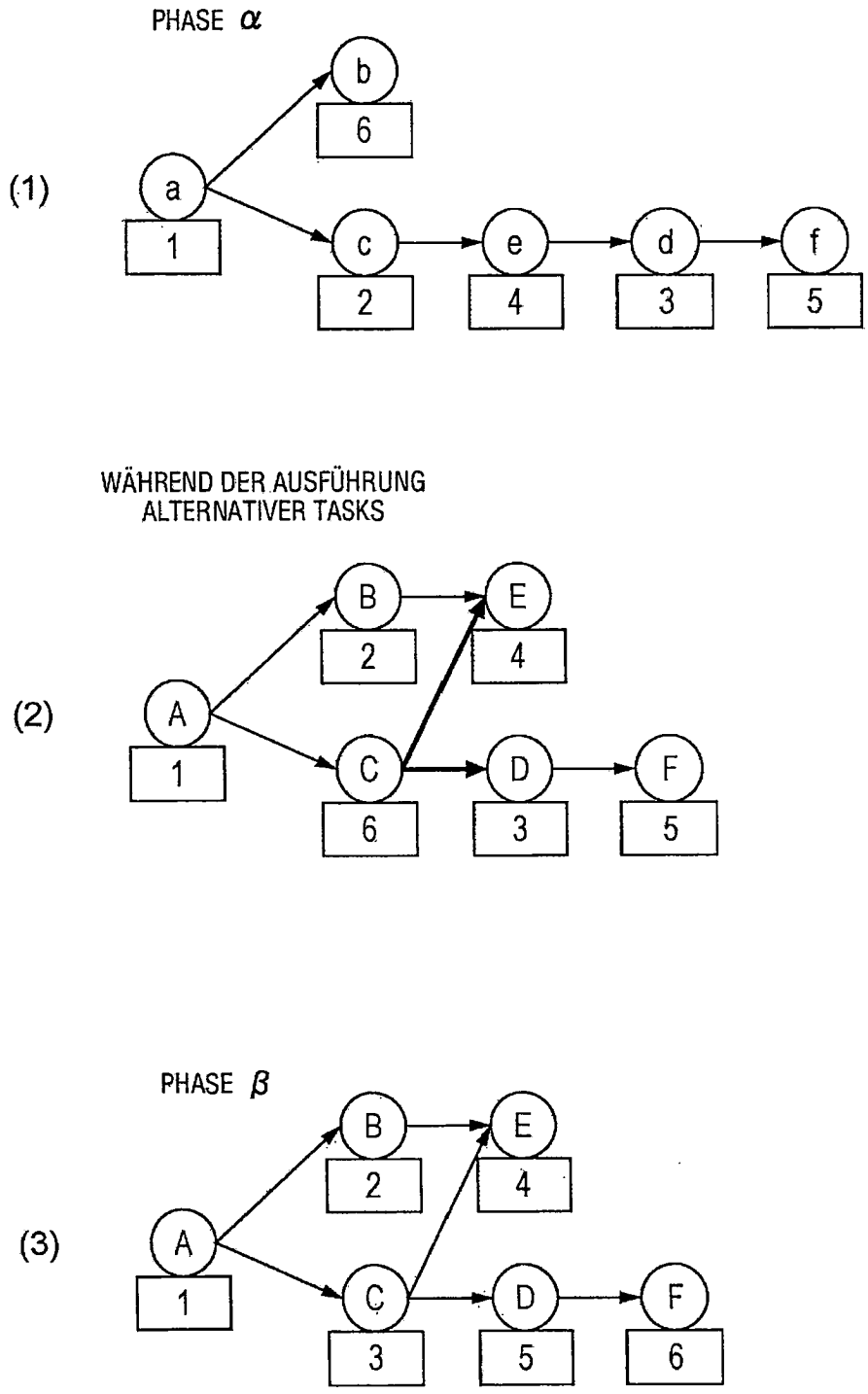
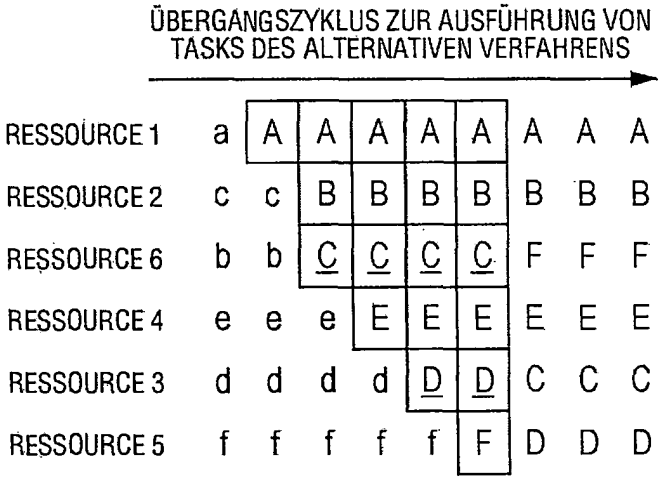

PHASE α
(1)
a
1
b
6
c
2
e
4
d
3
f
5
WÄHREND DER AUSFÜHRUNG
ALTERNATIVER TASKS
(2)
A
1
B
2
C
6
E
4
D
3
F
5
PHASE β
(3)
A
1
B
2
C
3
E
4
D
5
F
6

FIG. 21


ÜBERGANGSZYKLUS ZUR AUSFÜHRUNG VON TASKS DES ALTERNATIVEN VERFAHRENS
RESSOURCE 1
RESSOURCE 2
RESSOURCE 6
RESSOURCE 4
RESSOURCE 3
RESSOURCE 5
a A A A A A A A A
c c B B B B B B B
b b C C C C F F F
e e e E E E E E E
d d d d D D C C C
f f f f f F D D D