(19) Deutsches Patent- und Markenamt

(10) **DE 11 2011 101 469 T5** 2013.03.14

(12) # Veröffentlichung

der internationalen Anmeldung mit der
(87) Veröffentlichungs-Nr.: **WO 2011/134689**
in deutscher Übersetzung (Art. III § 8 Abs. 2 IntPatÜG)
(21) Deutsches Aktenzeichen: **11 2011 101 469.4**
(86) PCT-Aktenzeichen: **PCT/EP2011/052105**
(86) PCT-Anmeldetag: **14.02.2011**
(87) PCT-Veröffentlichungstag: **03.11.2011**
(43) Veröffentlichungstag der PCT Anmeldung
in deutscher Übersetzung: **14.03.2013**

(51) Int Cl.: ***G06F 9/45*** (2013.01)

(30) Unionspriorität:
12/770,353 **29.04.2010** **US**

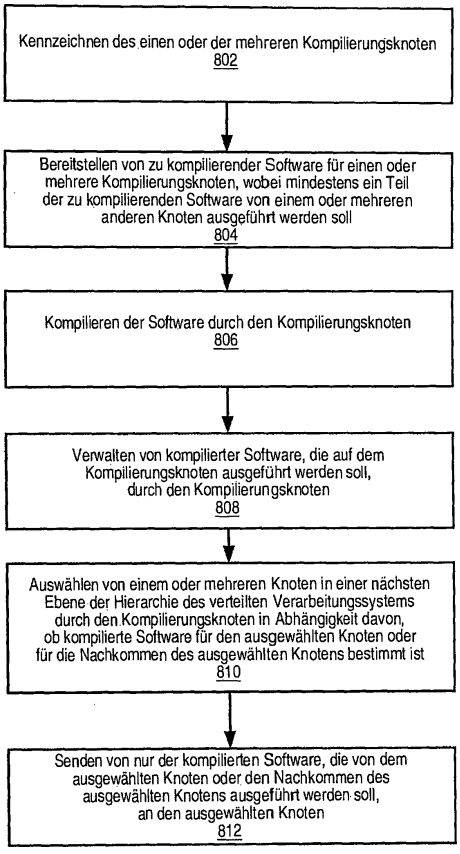
(71) Anmelder:
International Business Machines Corp., Armonk, N.Y., US

(74) Vertreter:
RICHARDT PATENTANWÄLTE GbR, 65185, Wiesbaden, DE

(72) Erfinder:
Rattermann, Joseph, Rochester, Minn., US;
Archer, Charles Jens, Rochester, Minn., US;
Smith, Brian Edward, Rochester, Minn., US;
Blocksome, Michael Alan, Rochester, Minn., US

(54) Bezeichnung: **Kompilieren von Software für ein hierarchisches verteiltes Verarbeitungssystem**

(57) Zusammenfassung: Kompilieren von Software für ein hierarchisches verteiltes Verarbeitungssystem, wobei das Kompilieren das Bereitstellen von zu kompilierender Software für einen oder mehrere Kompilierungsknoten, wobei mindestens ein Teil der zu kompilierenden Software von einem oder mehreren anderen Knoten ausgeführt werden soll; das Kompilieren der Software durch den Kompilierungsknoten; das Verwalten von kompilierter Software, die auf dem Kompilierungsknoten ausgeführt werden soll, durch den Kompilierungsknoten; das Auswählen von einem oder mehreren Knoten in einer nächsten Ebene der Hierarchie des verteilten Verarbeitungssystems durch den Kompilierungsknoten in Abhängigkeit davon, ob kompilierte Software für den ausgewählten Knoten oder für die Nachkommen des ausgewählten Knotens bestimmt ist; und das Senden von nur der kompilierten Software, die von dem ausgewählten Knoten oder von dem Nachkommen des ausgewählten Knotens ausgeführt werden soll, an den ausgewählten Knoten, beinhaltet.



Beschreibung

DER ERFINDUNG ZUGRUNDE LIEGENDER ALLGEMEINER STAND DER TECHNIK

Bereich der Erfindung

[0001] Der Bereich der Erfindung ist die Datenverarbeitung oder, genauer gesagt, Verfahren, Vorrichtungen und Produkte, um Software für ein hierarchisches verteiltes Verarbeitungssystem zu kompilieren.

Beschreibung der verwandten Technik

[0002] Die Entwicklung des EDVAC-Systems (EDVAC steht für Electronic Discrete Variable Automatic Computer) von 1948 wird oft als der Beginn der Computer-Ära bezeichnet. Seit dieser Zeit haben sich Computersysteme zu äußerst komplizierten Einheiten entwickelt. Die heutigen Computer sind den ersten Systemen wie zum Beispiel dem EDVAC deutlich überlegen. Computersysteme enthalten üblicherweise eine Kombination aus Hardware- und Software-Komponenten, Anwendungsprogramme, Betriebssysteme, Prozessoren, Busse, Speicher, Eingabe/Ausgabe-Einheiten und so weiter. Da Fortschritte in der Halbleiter-Verarbeitung und bei den Computerarchitekturen die Leistungsfähigkeit des Computers immer weiter steigern, ist eine für höhere Ansprüche ausgelegte Computer-Software entstanden, die die höhere Leistungsfähigkeit der Hardware vorteilhaft nutzt, so dass wir heute über Computersysteme verfügen, die weitaus leistungsstärker sind als noch vor ein paar Jahren.

[0003] Die verteilte Datenverarbeitung ist ein Bereich der Computertechnologie, in dem Fortschritte gemacht wurden. Die verteilte Datenverarbeitung bezieht sich im Allgemeinen auf eine Datenverarbeitung mit mehreren halbautonomen Computersystemen, die über ein Datenübertragungsnetzwerk Daten austauschen. Die halbautonomen Computersysteme treten miteinander in Dialogverkehr, um ein gemeinsames Ziel zu erreichen. Ein Computerprogramm oder eine Computeranwendung, die in einem verteilten Datenverarbeitungssystem ausgeführt wird, kann als ein verteiltes Programm bezeichnet werden. Die verteilte Datenverarbeitung kann sich auch auf die Verwendung von verteilten Datenverarbeitungssystemen zur Lösung von Berechnungsproblemen beziehen. Bei der verteilten Datenverarbeitung kann ein Problem in viele Aufgaben (Tasks) unterteilt werden, von denen jede von einem der halbautonomen Computersysteme gelöst werden kann.

[0004] Manche verteilte Datenverarbeitungssysteme werden zur Durchführung einer parallelen Datenverarbeitung optimiert. Unter der parallelen Datenverarbeitung versteht man die gleichzeitige Ausführung derselben Task (aufgeteilt und speziell angepasst) auf mehreren Prozessoren, damit man Ergebnisse schneller erhält. Die parallele Datenverarbeitung beruht darauf, dass der Prozess des Lösens eines Problems gewöhnlich in kleinere Tasks aufgeteilt werden kann, die gleichzeitig koordiniert ausgeführt werden können.

[0005] Parallele Computer führen parallele Algorithmen aus. Ein paralleler Algorithmus kann aufgeteilt werden, damit er auf vielen verschiedenen Verarbeitungseinheiten jeweils in Teilen ausgeführt werden kann, und anschließend kann er am Ende wieder zusammengesetzt werden, damit man ein Datenverarbeitungsergebnis erhält. Manche Algorithmen können problemlos in Teile gegliedert werden. Die Aufteilung der Aufgabe, alle Zahlen von eins bis einhunderttausend zu prüfen, um festzustellen, welches Primzahlen sind, könnte beispielsweise durchgeführt werden, indem man jedem vorhandenen Prozessor eine Teilmenge der Zahlen zuweist und die Liste der positiven Ergebnisse dann wieder zusammensetzt. In dieser Beschreibung werden die mehreren Verarbeitungseinheiten, die die einzelnen Teile eines parallelen Programms ausführen, als ”Computerknoten” bezeichnet. Ein paralleler Computer besteht sowohl aus Computerknoten als auch aus anderen Verarbeitungsknoten, zu denen beispielsweise auch Eingabe/Ausgabe-(E/A-)Knoten und Service-Knoten gehören.

[0006] Parallele Algorithmen sind wertvoll, da sich bestimmte Arten von großen Rechenaufgaben aufgrund der Funktionsweise moderner Prozessoren mittels eines parallelen Algorithmus schneller durchführen lassen als mittels eines seriellen (nichtparallelen) Algorithmus. Es ist weitaus schwieriger, einen Computer mit einem einzigen schnellen Prozessor als einen mit vielen langsamen Prozessoren und dem gleichen Durchsatz zu bauen. Es gibt auch bestimmte theoretische Grenzen für die mögliche Geschwindigkeit von seriellen Prozessoren. Andererseits hat jeder parallele Algorithmus einen seriellen Teil, und folglich haben parallele Algorithmen einen Sättigungspunkt. Nach diesem Punkt erzielt man mit dem Hinzufügen weiterer Prozessoren keinen höheren Durchsatz, man erhöht lediglich den Verarbeitungsaufwand und die Kosten.

[0007] Parallele Algorithmen sind auch so ausgelegt, dass sie eine mehrere Ressourcen die Datenübertragungsanforderungen unter den Knoten eines parallelen Computers optimieren. Es gibt zwei Arten, in der par-

allele Prozessoren kommunizieren, der gemeinsam genutzte Speicher oder den Nachrichtenaustausch. Die Verarbeitung mit gemeinsam genutztem Speicher erfordert ein zusätzliches Sperren der Daten, ist mit einem Mehraufwand in Form von zusätzlichen Prozessor- und Buszyklen verbunden und serialisiert darüber hinaus einen Teil des Algorithmus.

[0008] Bei der Verarbeitung des Nachrichtenaustausch werden Datenübertragungsnetzwerke hoher Geschwindigkeit und Nachrichten-Pufferspeicher verwendet, doch ist der Transportaufwand für die Datenübertragungsnetzwerke bei dieser Übertragung höher, für die Nachrichten-Pufferspeicher entsteht zusätzlicher Speicherbedarf und bei den Datenübertragungen zwischen den Knoten ist die Latenzzeit größer. Beim Entwurf von parallelen Computern werden speziell konzipierte Datenübertragungs-Verbindungsleitungen verwendet, so dass der zusätzliche Übertragungsaufwand gering ist, doch ist es der parallele Algorithmus, der über das Volumen des Datenverkehrs entscheidet.

[0009] Viele Architekturen von Datenübertragungsnetzwerken werden zum Nachrichtenaustausch zwischen Knoten in parallelen Computern verwendet. Computerknoten können in einem Netzwerk beispielsweise als ”Torus” oder als ”Masche” aufgebaut sein. Auch können Computerknoten in einem Netzwerk als ein Baum aufgebaut sein. Ein Torus-Netzwerk verbindet die Knoten in einer dreidimensionalen Masche mit Wrap-around-Verbindungsleitungen. Jeder Knoten ist mit seinen sechs Nachbarn über dieses Torus-Netzwerk verbunden, und jeder Knoten wird von seiner x-, y-, z-Koordinate in der Masche adressiert. Auf diese Weise eignet sich ein Torus-Netzwerk für Punkt-zu-Punkt-Operationen. In einem Baumnetzwerk werden die Knoten üblicherweise zu einem Binärbaum verbunden: Jeder Knoten hat einen Elternknoten und zwei Kindknoten (obgleich manche Knoten in Abhängigkeit von der Hardware-Konfiguration gar keine Kinder oder nur ein Kind haben dürfen). Zwar ist ein Baumnetzwerk bei der Punkt-zu-Punkt-Übertragung gewöhnlich wenig leistungsfähig, doch bietet es eine hohe Bandbreite und eine geringe Latenzzeit bei bestimmten kollektiven Operationen, Nachrichtenaustausch-Operationen, an denen alle Computerknoten gleichzeitig teilnehmen wie zum Beispiel eine Allgather-Operation. Bei Computern, die ein Torus- und ein Baumnetzwerk verwenden, werden die beiden Netzwerke üblicherweise unabhängig voneinander realisiert, mit getrennten Weiterleitungsschaltungen, getrennten physischen Verbindungsleitungen und getrennten Nachrichten-Pufferspeichen.

KURZDARSTELLUNG DER ERFINDUNG

[0010] Verfahren, Vorrichtungen und Produkte zum Kompilieren von Software für ein hierarchisches verteiltes Verarbeitungssystem einschließlich der Bereitstellung von zu kompilierender Software für einen oder mehrere Kompilierungsknoten, wobei mindestens ein Teil der zu kompilierenden Software von einem oder mehreren anderen Knoten ausgeführt werden soll; Kompilieren der Software durch den Kompilierungsknoten; Verwalten von kompilierter Software, die auf dem Kompilierungsknoten ausgeführt werden soll, durch den Kompilierungsknoten; Auswählen von einem oder mehreren Knoten in einer nächsten Ebene der Hierarchie des verteilten Verarbeitungssystems durch den Kompilierungsknoten in Abhängigkeit davon, ob kompilierte Software für den ausgewählten Knoten oder für die Nachkommen des ausgewählten Knotens bestimmt ist; Senden von nur der kompilierten Software, die von dem ausgewählten Knoten oder von dem Nachkommen des ausgewählten Knotens ausgeführt werden soll, an den ausgewählten Knoten; Empfangen von kompilierter Software durch einen ausgewählten Knoten; Feststellen, ob die kompilierte Software für den ausgewählten Knoten oder für einen seiner Nachkommen bestimmt ist; wenn die kompilierte Software für den ausgewählten Knoten bestimmt ist, Verwalten der Software durch den ausgewählten Knoten zur Ausführung; und wenn die kompilierte Software für einen der Nachkommen bestimmt ist, Auswählen eines anderen Knotens in einer nächsten Ebene des hierarchischen verteilten Verarbeitungssystems in Abhängigkeit von einem Nachkommen für die kompilierte Software und Senden der kompilierten Software an den ausgewählten anderen Knoten.

[0011] Die vorstehenden und andere Aufgaben, Merkmale und Vorteile der Erfindung gehen aus den folgenden ausführlicheren Beschreibungen von beispielhaften Ausführungsformen der Erfindung, die in den beigefügten Zeichnungen veranschaulicht sind, hervor, in denen gleiche Bezugszahlen im Allgemeinen gleiche Teile von beispielhaften Ausführungsformen der Erfindung darstellen.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

[0012] Fig. 1 zeigt ein beispielhaftes verteiltes Datenverarbeitungssystem, um Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung zu kompilieren.

[0013] Fig. 2 zeigt ein Blockschaltbild eines beispielhaften Computerknotens, der in einem parallelen Computer von Nutzen ist, welcher Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung kompilieren kann.

[0014] Fig. 3A zeigt einen beispielhaften Punkt-zu-Punkt-Adapter, der in Systemen von Nutzen ist, die Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung kompilieren können.

[0015] Fig. 3B zeigt einen beispielhaften Punkt-zu-Punkt-Adapter, der in Systemen von Nutzen ist, die Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung kompilieren können.

[0016] Fig. 4 zeigt eine Zeichnung mit Linien, die ein beispielhaftes Datenübertragungsnetzwerk darstellt, das für Punkt-zu-Punkt-Operationen optimiert ist, welche in Systemen von Nutzen sind, die Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung kompilieren können.

[0017] Fig. 5 zeigt eine Zeichnung mit Linien, die ein beispielhaftes Datenübertragungsnetzwerk darstellt, das für kollektive Operationen optimiert ist, welche in Systemen von Nutzen sind, die Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung kompilieren können.

[0018] Fig. 6 zeigt ein weiteres beispielhaftes verteiltes Datenverarbeitungssystem zum Kompilieren von Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung, in dem das verteilte Datenverarbeitungssystem als eine hybride Datenverarbeitungsumgebung ausgeführt ist.

[0019] Fig. 7 zeigt ein beispielhaftes Verfahren zum Kompilieren von Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung.

[0020] Fig. 8 zeigt einen Ablaufplan, der ein weiteres beispielhaftes Verfahren zum Kompilieren von Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung veranschaulicht.

[0021] Fig. 9 zeigt ein Schaubild eines beispielhaften Anwendungsfalls eines Systems zum Kompilieren von Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung.

AUSFÜHRLICHE BESCHREIBUNG VON BEISPIELHAFTEN AUSFÜHRUNGSFORMEN

[0022] Beispielhafte Verfahren, Vorrichtungen und Produkte zum Kompilieren von Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung werden mit Bezug auf die beigefügten Zeichnungen beschrieben, wobei mit Fig. 1 begonnen wird. Fig. 1 zeigt ein beispielhaftes verteiltes Datenverarbeitungssystem, um Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung zu kompilieren. Das System von Fig. 1 enthält einen parallelen Computer (**100**), einen nichtflüchtigen Speicher für den Computer in Form der Datenspeichereinheit (**118**), eine Ausgabeeinheit für den Computer in Form des Druckers (**120**) und eine Eingabe/Ausgabe-Einheit für den Computer in Form des Computerendgeräts (**122**). Der parallele Computer (**100**) in dem Beispiel von Fig. 1 enthält eine Vielzahl von Computerknoten (**102**).

[0023] Die Computerknoten (**102**) sind für Datenübertragungen über mehrere unabhängige Datenübertragungsnetzwerke verbunden, darunter ein Netzwerk vom Typ ”Joint Test Action Group” (JTAG) (**104**), ein globales Kombinationsnetzwerk (**106**), das für kollektive Operationen optimiert ist, und ein Torus-Netzwerk (**108**), das für Punkt-zu-Punkt-Operationen optimiert ist. Das globale Kombinationsnetzwerk (**106**) ist ein Datenübertragungsnetzwerk, das Datenübertragungs-Verbindungsleitungen enthält, die mit den Computerknoten verbunden sind, um die Computerknoten als einen Baum aufzubauen. Jedes Datenübertragungsnetzwerk ist mit Datenübertragungs-Verbindungsleitungen zwischen den Computerknoten (**102**) realisiert. Die Datenübertragungs-Verbindungsleitungen ermöglichen Datenübertragungen für parallele Operationen zwischen den Computerknoten des parallelen Computers. Die Verbindungsleitungen zwischen den Computerknoten sind bidirektionale Verbindungsleitungen, die üblicherweise mit zwei getrennten gerichteten Datenübertragungspfaden realisiert werden.

[0024] Überdies sind die Computerknoten (**102**) des parallelen Computers in mindestens eine operative Gruppe (**132**) von Computerknoten für kollektive parallele Operationen auf dem parallelen Computer (**100**) gegliedert. Bei einer operativen Gruppe von Computerknoten handelt es sich um den Satz von Computerknoten, auf dem eine kollektive parallele Operation ausgeführt wird. Kollektive Operationen werden mit Datenübertragungen zwischen den Computerknoten einer operativen Gruppe durchgeführt. Kollektive Operationen sind diejenigen Funktionen, an denen alle Computerknoten einer operativen Gruppe beteiligt sind. Eine kollektive Operation ist eine Operation, ein Befehl eines Computerprogramms zum Nachrichtenaustausch, der von allen Computerknoten in einer operativen Gruppe von Computerknoten gleichzeitig, das heißt, zu ungefähr dem gleichen Zeitpunkt, ausgeführt wird. Eine solche operative Gruppe kann alle Computerknoten in einem parallelen Computer (**100**) oder eine Teilmenge aller Computerknoten beinhalten. Kollektive Operationen werden oft um Punkt-zu-Punkt-Operationen aufgebaut. Eine kollektive Operation setzt voraus, dass alle Prozesse auf allen Computerknoten in einer operativen Gruppe dieselbe kollektive Operation mit übereinstimmenden Argumenten aufrufen. Eine „Rundsendung" (broadcast) ist ein Beispiel für eine kollektive Operation, um Daten zwischen Computerknoten einer operativen Gruppe zu übertragen. Eine „Reduktions"-(Reduce-)Operation ist ein Beispiel für eine kollektive Operation, die arithmetische oder logische Funktionen an Daten ausführt, die zwischen den Computerknoten einer operativen Gruppe verteilt sind. Eine operative Gruppe kann zum Beispiel als ein MPI-"Kommunikator" realisiert sein.

[0025] "MPI" bezieht sich auf die "Message Passing Interface", eine Bibliothek für parallele Übertragungen nach dem Stand der Technik, ein Modul mit Computerprogrammbefehlen für Datenübertragungen auf parallelen Computern. Zu Beispielen für Bibliotheken für parallele Übertragungen nach dem Stand der Technik, die zur Verwendung mit Systemen gemäß Ausführungsformen der vorliegenden Erfindung verbessert werden können, gehören MPI und die Bibliothek "Parallel Virtual Machine" (PVM). PVM wurde von der University of Tennessee, The Oak Ridge National Laboratory und der Emory University entwickelt. MPI wird vom MPI-Forum verbreitet, einer offenen Gruppe mit Repräsentanten aus vielen Unternehmen und Organisationen, die den MPI-Standard definieren und pflegen. MPI ist zum Zeitpunkt der Erstellung dieses Schriftstücks der de-facto-Standard für die Übertragung zwischen Computerknoten, die ein paralleles Programm auf einem parallelen Computer mit verteiltem Speicher ausführen. Diese Beschreibung verwendet der einfacheren Erklärung halber manchmal die MPI-Terminologie, obgleich die Verwendung von MPI als solches weder ein Erfordernis noch eine Beschränkung der vorliegenden Erfindung darstellt.

[0026] Manche kollektiven Operationen verfügen über einen einzigen Ursprungs- oder Empfangsprozess, der auf einem bestimmten Computerknoten in einer operativen Gruppe ausgeführt wird. Bei einer kollektiven Operation "Rundsendung" zum Beispiel ist der Prozess auf dem Computerknoten, der die Daten an alle anderen Computerknoten verteilt, ein Ursprungsprozess. Bei einer „Sammel"-(Gather-)Operation beispielsweise ist der Prozess auf dem Computerknoten, der alle Daten von den anderen Computerknoten empfangen hat, ein Empfangsprozess. Der Computerknoten, auf dem ein solcher Ursprungs- oder Empfangsprozess ausgeführt wird, wird als lokaler Wurzelknoten bezeichnet.

[0027] Die meisten kollektiven Operationen sind Variationen oder Kombinationen von vier Basisoperationen: Rundsenden (Broadcast), Sammeln (Gather), Streuen (Scatter) und Reduzieren (Reduce). Die Schnittstellen für diese kollektiven Operationen sind in den vom MPI-Forum verbreiteten MPI-Standards definiert. Algorithmen zur Ausführung von kollektiven Operationen sind in den MPI-Standards jedoch nicht definiert. Bei einer Rundsende-Operation geben alle Prozesse denselben Wurzelprozess an, von dem der Inhalt seines Pufferspeichers gesendet wird. Von dem Wurzelprozess verschiedene Prozesse geben Empfangspufferspeicher an. Nach der Operation enthalten alle Pufferspeicher die Nachricht von dem Wurzelprozess.

[0028] Bei einer Streuoperation teilt die logische Wurzel Daten in der Wurzel in Segmente und verteilt an jeden Computerknoten in der operativen Gruppe ein anderes Segment. Bei einer Streuoperation geben alle Prozesse gewöhnlich denselben Empfangszählstand an. Die Sendeargumente sind nur für den Wurzelprozess von Bedeutung, dessen Pufferspeicher den Sendezählstand * N Elemente eines bestimmten Datentyps enthält, wobei N die Anzahl der Prozesse in der bestimmten Gruppe der Computerknoten ist. Der Sendepufferspeicher wird aufgeteilt und an alle Prozesse (einschließlich des Prozesses auf der logischen Wurzel) verteilt. Jedem Computerknoten wird eine fortlaufende Kennung mit der Bezeichnung "Rang" zugewiesen. Nach der Operation hat die Wurzel jedem Prozess in aufsteigender Rangfolge Datenelemente des Sendezählstands gesendet. Der Rang 0 empfängt die ersten Datenelemente des Sendezählstands aus dem Sendepufferspeicher. Der Rang 1 empfängt die zweiten Datenelemente des Sendezählstands aus dem Sendepufferspeicher und so weiter.

[0029] Eine Sammeloperation ist eine kollektive Viele-zu-eins-Operation, die das komplette Gegenteil der Beschreibung der Streuoperation ist. Das heißt, eine Sammeloperation ist eine kollektive Viele-zu-eins-Operation,

bei der Elemente eines Datentyps von den nach ihrem Rang sortierten Computerknoten eingesammelt und in einen Empfangspufferspeicher in einem Wurzelknoten eingegeben werden.

[0030] Eine Reduktionsoperation ist ebenfalls eine kollektive Viele-zu-eins-Operation, die eine arithmetische oder logische Funktion enthält, welche an zwei Datenelementen ausgeführt wird. Alle Prozesse geben denselben „Zählstand" (count) und dieselbe arithmetische oder logische Funktion an. Nach der Reduktion haben alle Prozesse Datenelemente des Zählstands aus den Sendepufferspeichern der Computerknoten an den Wurzelprozess gesendet. Bei einer Reduktionsoperation werden Datenelemente von entsprechenden Speicherplätzen im Sendepufferspeicher mittels arithmetischer oder logischer Operationen paarweise verknüpft, um ein einziges entsprechendes Element im Empfangspufferspeicher des Wurzelprozesses zu erzeugen. Die Anwendung von bestimmten Reduktionsoperationen kann zur Laufzeit festgelegt werden. Bibliotheken für parallele Übertragungen können vordefinierte Operationen unterstützen. MPI stellt beispielsweise die folgenden vordefinierten Reduktionsoperationen zur Verfügung:

MPI_MAX	Maximum
MPI_MIN	Minimum
MPI_SUM	Summe
MPI_PROD	Produkt
MPI_LAND	Logisches UND
MPI_BAND	Bitweises UND
MPI_LOR	Logisches ODER
MPI_BOR	Bitweises ODER
MPI_LXOR	Logisches exklusives ODER
MPI_BXOR	Bitweises exklusives ODER

[0031] Neben Computerknoten enthält der parallele Computer (**100**) auch Eingabe/Ausgabe-(E/A-)Knoten (**110**, **114**), die über das globale Kombinationsnetzwerk (**106**) mit den Computerknoten (**102**) verbunden sind. Die Computerknoten in dem parallelen Computer (**100**) werden in Verarbeitungsgruppen unterteilt, so dass jeder Computerknoten in einer Verarbeitungsgruppe für Datenübertragungen mit demselben E/A-Knoten verbunden ist. Jede Verarbeitungsgruppe besteht folglich aus einem E/A-Knoten und einer Teilmenge von Computerknoten (**102**). Das Verhältnis zwischen der Anzahl der Computerknoten und der Anzahl der E/A-Knoten in dem ganzen System hängt üblicherweise von der für den parallelen Computer gewählten Hardware-Konfiguration ab. Bei manchen Konfigurationen kann jede Verarbeitungsgruppe zum Beispiel aus acht Computerknoten und einem E/A-Knoten bestehen. Bei anderen Konfigurationen kann jede Verarbeitungsgruppe aus vierundsechzig Computerknoten und einem E/A-Knoten bestehen. Dieses Beispiel dient jedoch lediglich der Erklärung und ist nicht als Einschränkung zu verstehen. Jeder E/A-Knoten stellt E/A-Dienste zwischen Computerknoten (**102**) seiner Verarbeitungsgruppe und einer Gruppe von E/A-Einheiten bereit. In dem Beispiel von Fig. 1 sind die E/A-Knoten (**110**, **114**) über ein lokales Netzwerk (LAN) (**130**), das mittels eines Hochgeschwindigkeits-Ethernet-Netzwerks realisiert ist, für Datenübertragungen mit den E/A-Einheiten (**118**, **120**, **122**) verbunden.

[0032] Der parallele Computer (**100**) von Fig. 1 enthält auch einen Service-Knoten (**116**), der über eines der Netzwerke (**104**) mit den Computerknoten verbunden ist. Der Service-Knoten (**116**) stellt Dienste bereit, die einer Vielzahl von Computerknoten gemein sind; er verwaltet die Konfiguration von Computerknoten, er lädt Programme in die Computerknoten, er startet die Ausführung von Programmen auf den Computerknoten, er ruft Ergebnisse von Programmoperationen auf den Computerknoten ab und so weiter. Der Service-Knoten (**116**) führt eine Service-Anwendung (**124**) aus und kommuniziert mit Benutzern (**128**) über eine Schnittstelle (**126**) der Service-Anwendung, die auf dem Computerendgerät (**122**) ausgeführt wird.

[0033] In dem Beispiel von Fig. 1 ist auf einem der Computerknoten ein hierarchischer verteilter Compiler (**155**), ein Modul einer automatisierten Datenverarbeitungsmaschine, installiert, der Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung kompilieren kann. Der hierarchische verteilte Compiler (**155**) enthält einen Befehl eines Computerprogramms, um die Software durch den Kompilierungsknoten zu kompilieren; um kompilierte Software, die auf dem Kompilierungsknoten ausgeführt werden soll, durch den Kompilierungsknoten zu verwalten; um einen oder mehrere Knoten in einer nächsten Ebene der Hierarchie des verteilten Verarbeitungssystems durch den Kompilierungsknoten in Abhängigkeit davon, ob kompilierte Software für den ausgewählten Knoten oder für die Nachkommen des ausgewählten Knotens bestimmt ist, auszuwählen; um nur die kompilierte Software, die von dem ausgewählten

Knoten oder von dem Nachkommen des ausgewählten Knotens ausgeführt werden soll, an den ausgewählten Knoten zu senden. Jeder der anderen Computerknoten (**102**) von Fig. 1 kann auch kompilierte Software empfangen; Feststellen, ob die kompilierte Software für diesen Knoten oder für einen seiner Nachkommen bestimmt ist; die Software zur Ausführung verwalten, wenn die kompilierte Software für diesen Knoten bestimmt ist; und einen anderen Knoten in einer nächsten Ebene des hierarchischen verteilten Verarbeitungssystems in Abhängigkeit von einem Nachkommen für die kompilierte Software auswählen, wenn die kompilierte Software für einen der Nachkommen bestimmt ist, und die kompilierte Software an den ausgewählten anderen Knoten senden.

[0034] Die Anordnung der Knoten, Netzwerke und E/A-Einheiten, die das beispielhafte in Fig. 1 dargestellte System bilden, dient lediglich der Erklärung und ist nicht als Einschränkung der vorliegenden Erfindung zu verstehen. Datenverarbeitungssysteme, die Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung kompilieren können, können weitere Knoten, Netzwerke, Einheiten und Architekturen enthalten, die in Fig. 1 nicht gezeigt sind, wie für den Fachmann zu erkennen ist. Der parallele Computer (**100**) in dem Beispiel von Fig. 1 enthält zwar sechzehn Computerknoten (**102**), doch wird der Leser bemerken, dass parallele Computer, die Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung kompilieren können, eine beliebige Anzahl von Computerknoten enthalten können. Neben Ethernet und JTAG können Netzwerke in solch einem Datenverarbeitungssystem viele Datenübertragungsprotokolle unterstützen, darunter beispielsweise TCP (Transmission Control Protocol), IP (Internet Protocol) und andere, wie für den Fachmann zu erkennen ist. Verschiedene Ausführungsformen der vorliegenden Erfindung können neben den in Fig. 1 veranschaulichten auch auf vielen verschiedenen Hardware-Plattformen realisiert werden.

[0035] Das Kompilieren von Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung kann im Allgemeinen auf einem parallelen Computer durchgeführt werden, der eine Vielzahl von Computerknoten enthält. Tatsächlich können solche Computer Tausende von solchen Computerknoten enthalten. Jeder Computerknoten ist selbst wiederum eine Art Computer, der aus einem oder mehreren Computerprozessoren (oder Prozessorkernen), seinem eigenen Computerspeicher und seinen eigenen Eingabe/Ausgabe-Adaptern besteht. Zur näheren Erklärung zeigt Fig. 2 daher ein Blockschaltbild eines beispielhaften Computerknotens, der in einem parallelen Computer von Nutzen ist, welcher Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung kompilieren kann. Der Computerknoten (**152**) von Fig. 2 enthält einen oder mehrere Verarbeitungskerne (**164**) sowie einen Direktzugriffspeicher (RAM) (**156**). Die Verarbeitungskerne (**164**) sind mit dem RAM (**156**) über einen Hochgeschwindigkeits-Speicherbus (**154**) und über einen Busadapter (**194**) sowie einen Erweiterungsbus (**168**) mit anderen Komponenten des Computerknotens (**152**) verbunden. Im RAM (**156**) ist ein Anwendungsprogramm (**158**), ein Modul mit Computerprogrammbefehlen, gespeichert, das eine parallele Datenverarbeitung auf Benutzerebene unter Verwendung von parallelen Algorithmen durchführt.

[0036] Ebenfalls im RAM (**156**) gespeichert ist ein Nachrichtenaustauschmodul (**160**), eine Bibliothek mit Computerprogrammbefehlen, die parallele Übertragungen zwischen Computerknoten einschließlich Punkt-zu-Punkt-Operationen sowie kollektive Operationen durchführen. Das Anwendungsprogramm (**158**) führt kollektive Operationen aus, indem es Software-Routinen in dem Nachrichtenaustauschmodul (**160**) aufruft. Eine Bibliothek mit parallelen Übertragungsroutinen kann zur Verwendung in Systemen gemäß Ausführungsformen der vorliegenden Erfindung von Grund auf entwickelt werden, wobei eine herkömmliche Programmiersprache wie zum Beispiel die Programmiersprache C und herkömmliche Programmierverfahren verwendet werden, um parallele Übertragungsroutinen zu schreiben, die Daten zwischen Knoten in zwei unabhängigen Datenübertragungsnetzwerken senden und empfangen. Alternativ können vorhandene Bibliotheken nach dem Stand der Technik verbessert werden, damit sie gemäß Ausführungsformen der vorliegenden Erfindung arbeiten. Zu Beispielen für Bibliotheken für parallele Übertragungen nach dem Stand der Technik gehören die Bibliothek ”Message Passing Interface” (MPI) und die Bibliothek ”Parallel Virtual Machine” (PVM).

[0037] Ebenfalls im RAM gespeichert ist ein hierarchischer verteilter Compiler (**155**), ein Modul einer automatisierten Datenverarbeitungsmaschine, der Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung kompilieren kann. Der hierarchische verteilte Compiler (**155**) enthält einen Befehl eines Computerprogramms, um die Software durch den Kompilierungsknoten zu kompilieren; um kompilierte Software, die auf dem Kompilierungsknoten ausgeführt werden soll, durch den Kompilierungsknoten zu verwalten; um einen oder mehrere Knoten in einer nächsten Ebene der Hierarchie des verteilten Verarbeitungssystems durch den Kompilierungsknoten in Abhängigkeit davon, ob kompilierte Software für den ausgewählten Knoten oder für die Nachkommen des ausgewählten Knotens bestimmt ist, auszuwählen; um nur die kompilierte Software, die von dem ausgewählten Knoten oder von dem Nachkommen

des ausgewählten Knotens ausgeführt werden soll, an den ausgewählten Knoten zu senden. Jeder der anderen Computerknoten in einem parallelen Computer kann auch kompilierte Software empfangen; Feststellen, ob die kompilierte Software für diesen Knoten oder für einen seiner Nachkommen bestimmt ist; die Software zur Ausführung verwalten, wenn die kompilierte Software für diesen Knoten bestimmt ist; und einen anderen Knoten in einer nächsten Ebene des hierarchischen verteilten Verarbeitungssystems in Abhängigkeit von einem Nachkommen für die kompilierte Software auswählen, wenn die kompilierte Software für einen der Nachkommen bestimmt ist, und die kompilierte Software an den ausgewählten anderen Knoten senden.

[0038] Darüber hinaus sind im RAM (**156**) ein Betriebssystem (**162**), ein Modul mit Computerprogrammbefehlen und Routinen für den Zugriff eines Anwendungsprogramms auf andere Ressourcen des Computerknotens, gespeichert. Es ist für ein Anwendungsprogramm und eine Bibliothek für parallele Übertragungen in einem Computerknoten eines parallelen Computers üblich, einen einzelnen Ausführungs-Thread ohne Benutzeranmeldung und ohne Sicherheitsaspekte auszuführen, weil der Thread über die Berechtigung zum vollständigen Zugriff auf alle Ressourcen des Knotens verfügt. Die Menge und die Komplexität der von einem Betriebssystem auf einem Computerknoten in einem parallelen Computer auszuführenden Tasks ist folglich geringer als die Menge und die Komplexität der Tasks eines Betriebssystems auf einem seriellen Computer, auf dem viele Threads gleichzeitig ausgeführt werden. Überdies gibt es auf dem Computerknoten (**152**) von Fig. 2 keinen Video-E/A, ein weiterer Faktor, der die Anforderungen an das Betriebssystem verringert. Bei dem Betriebssystem kann es sich im Vergleich zu Betriebssystemen von Universalcomputern daher um ein abgespecktes Betriebssystem handeln, eine im Funktionsumfang reduzierte Variante sozusagen, oder um ein Betriebssystem, das speziell für Operationen auf einem bestimmten parallelen Computer entwickelt worden ist. Zu Betriebssystemen, die zur Verwendung in einem Computerknoten auf sinnvolle Weise verbessert, vereinfacht werden, gehören UNIX$_{TM}$, Linux$_{TM}$, Microsoft XP$_{TM}$, AIX$_{TM}$, i5/OS von IBM$_{TM}$ und andere, wie für den Fachmann zu erkennen ist.

[0039] Der beispielhafte Computerknoten (**152**) von Fig. 2 enthält mehrere Übertragungsadapter (**172**, **176**, **180**, **188**), um den Austausch von Daten mit anderen Knoten eines parallelen Computers durchzuführen. Solche Datenübertragungen können seriell über RS-232-Verbindungen, über externe Busse wie zum Beispiel den Universal Serial Bus (USB), über Datenübertragungsnetzwerke wie zum Beispiel IP-Netzwerke und auf andere Weise durchgeführt werden, wie für den Fachmann zu erkennen ist. Übertragungsadapter realisieren die Hardware-Ebene von Datenübertragungen, über die ein Computer entweder direkt oder über ein Netzwerk Datenübertragungen an einen anderen Computer sendet. Zu Beispielen für Übertragungsadapter, die in Systemen nützlich sind, welche Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung kompilieren, gehören Modems für drahtgebundene Übertragungen, Ethernet-(IEEE 802.3-)Adapter für drahtgebundene Netzwerkübertragungen und 802.11b-Adapter für drahtlose Netzwerkübertragungen.

[0040] Die Datenübertragungsadapter in dem Beispiel von Fig. 2 beinhalten einen Gigabit-Ethernet-Adapter (**172**), der den als Beispiel dienenden Rechnerknoten (**152**) für Datenübertragungen mit einem Gigabit-Ethernet-Netzwerk (**174**) verbindet. Gigabit Ethernet ist ein Netzwerk-Übertragungsstandard, der in dem Standard IEEE 802.3 festgelegt ist, die eine Datenrate von 1 Milliarde Bits pro Sekunde (ein Gigabit) vorsieht. Gigabit Ethernet ist eine Variante von Ethernet, die über Mehrmoden-Lichtwellenleiterkabel, Einmoden-Lichtwellenleiterkabel oder nicht abgeschirmte, verdrillte Zwillingskabel betrieben wird.

[0041] Die Datenübertragungsadapter in dem Beispiel von Fig. 2 beinhalten eine untergeordnete JTAG-Schaltung (JTAG-Slave-Schaltung) (**176**), die den als Beispiel dienenden Computerknoten (**152**) für den Datenaustausch mit einer übergeordneten JTAG-Schaltung (JTAG-Master-Schaltung) (**178**) verbindet. JTAG ist der für den Standard IEEE 1149.1 mit dem Titel ”Standard Test Access Port and Boundary-Scan Architecture for test access ports used for testing printed circuit boards using boundary scan” gebräuchliche Name. JTAG ist in weiten Teilen so angepasst, dass Boundary-Scan derzeit mehr oder weniger gleichbedeutend mit JTAG ist. JTAG wird nicht nur für Leiterplatten, sondern auch für Boundary-Scan-Tests von integrierten Schaltungen eingesetzt und ist auch als ein Mechanismus zur Fehlersuche und -beseitigung in eingebetteten Systemen hilfreich, da er eine praktische ”Hintertür” in das System bietet. Auf den beispielhaften Computerknoten von Fig. 2 kann zum Beispiel alles drei von Folgendem zutreffen: Er enthält üblicherweise eine oder mehrere integrierte Schaltungen, die auf einer Leiterplatte installiert sind, und kann als ein eingebettetes System realisiert werden, das über seinen eigenen Prozessor, seinen eigenen Speicher und seine eigene E/A-Funktionalität verfügt. JTAG-Boundary-Scan-Tests durch den JTAG-Slave (**176**) können Prozessorregister und Speicher in dem Computerknoten (**152**) zur Verwendung beim Kompilieren von Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung wirksam konfigurieren.

[0042] Die Datenübertragungsadapter in dem Beispiel von Fig. 2 beinhalten einen Punkt-zu-Punkt-Adapter (**180**), der den beispielhaften Computerknoten (**152**) für Datenübertragungen mit einem Netzwerk (**108**) verbindet, das für Punkt-zu-Punkt-Nachrichtenaustauschoperationen wie zum Beispiel ein Netzwerk, das als ein dreidimensionaler Torus oder als eine dreidimensionale Masche konfiguriert ist, optimal geeignet ist. Der Punkt-zu-Punkt-Adapter (**180**) ermöglicht Datenübertragungen in sechs Richtungen auf drei Übertragungsachsen, x, y und z, über sechs bidirektionale Verbindungsleitungen: +x (**181**), –x (**182**), +y (**183**), –y (**184**), +z (**185**) und –z (**186**).

[0043] Die Datenübertragungsadapter in dem Beispiel von Fig. 2 beinhalten einen Global-Combining-Network-Adapter (**188**), der den beispielhaften Computerknoten (**152**) für den Austausch von Daten mit einem Netzwerk (**106**) verbindet, das für kollektive Nachrichtenaustauschoperationen in einem globalen Kombinationsnetzwerk, das zum Beispiel als ein Binärbaum konfiguriert ist, optimal geeignet ist. Der Global-Combining-Network-Adapter (**188**) ermöglicht Datenübertragungen über drei bidirektionale Verbindungsleitungen: zwei zu Kindknoten (**190**) und eine zu einem Elternknoten (**192**).

[0044] Der beispielhafte Computerknoten (**152**) enthält zwei Rechenwerke (arithmetic logic units – ALUs). Die ALU (**166**) ist eine Komponente eines jeden Verarbeitungskerns (**164**), und eine gesonderte ALU (**170**) ist für den ausschließlichen Gebrauch durch den Global-Combining-Network-Adapter (**188**) zur Verwendung bei der Durchführung der arithmetischen und logischen Funktionen von Reduktionsoperationen vorgesehen. Computerprogrammbefehle einer Reduktionsroutine in der Bibliothek für parallele Übertragungen (**160**) können einen Befehl für eine arithmetische oder logische Funktion im Befehlsregister (**169**) zwischenspeichern. Wenn die arithmetische oder logische Funktion einer Reduktionsoperation zum Beispiel eine ”Summe” oder ein ”logisches ODER” ist, kann der Global-Combining-Network-Adapter (**188**) die arithmetische oder logische Operation mittels der ALU (**166**) im Prozessor (**164**) oder, was üblicherweise weitaus schneller ist, mittels der fest zugeordneten ALU (**170**) ausführen.

[0045] Der beispielhafte Computerknoten (**152**) von Fig. 2 enthält eine Steuereinheit für den direkten Speicherzugriff (direct memory access – DMA) (**195**), bei der es sich um Computer-Hardware für einen direkten Speicherzugriff handelt, und eine DMA-Komponente (**197**), bei der es sich um Computer-Software für einen direkten Speicherzugriff handelt. Die DMA-Komponente (**197**) von Fig. 2 wird üblicherweise im Computerspeicher der DMA-Steuereinheit (**195**) gespeichert. Der Direktspeicherzugriff beinhaltet Lese- und Schreiboperationen aus dem beziehungsweise in den Speicher von Computerknoten, während die Arbeitslast der Zentraleinheiten (**164**) gleichzeitig verringert wird. Bei einer DMA-Übertragung wird im Wesentlichen ein Block des Speichers von einem Speicherplatz an einen anderen kopiert, gewöhnlich von einem Computerknoten an einen anderen. Die CPU kann die DMA-Übertragung zwar einleiten, führt sie jedoch nicht aus.

[0046] Zur näheren Erklärung zeigt Fig. 3A einen beispielhaften Punkt-zu-Punkt-Adapter (**180**), der in Systemen von Nutzen ist, die Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung kompilieren können. Der Punkt-zu-Punkt-Adapter (**180**) ist zur Verwendung in einem für Punkt-zu-Punkt-Operationen optimierten Datenübertragungsnetzwerk vorgesehen, einem Netzwerk, das Computerknoten in einem dreidimensionalen Torus oder in einer dreidimensionalen Masche aufbaut. In dem Beispiel von Fig. 3A ermöglicht der Punkt-zu-Punkt-Adapter (**180**) Datenübertragungen auf einer x-Achse über vier einseitig gerichtete Datenübertragungs-Verbindungsleitungen, an den und von dem nächsten Knoten in der –x-Richtung (**182**) und an den und von dem nächsten Knoten in der +x-Richtung (**181**). In dem Beispiel von Fig. 3A ermöglicht der Punkt-zu-Punkt-Adapter (**180**) Datenübertragungen auf einer x-Achse über vier einseitig gerichtete Datenübertragungs-Verbindungsleitungen, an den und von dem nächsten Knoten in der –x-Richtung (**184**) und an den und von dem nächsten Knoten in der +x-Richtung (**183**). Der Punkt-zu-Punkt-Adapter (**180**) in Fig. 3A ermöglicht auch Datenübertragungen auf einer z-Achse über vier einseitig gerichtete Datenübertragungs-Verbindungsleitungen, an den und von dem nächsten Knoten in der –z-Richtung (**186**) und an den und von dem nächsten Knoten in der +z-Richtung (**185**).

[0047] Zur näheren Erklärung zeigt Fig. 3B einen beispielhaften Adapter (**188**) eines globales Kombinationsnetzwerks, der in Systemen von Nutzen ist, die Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung kompilieren können. Der Adapter (**188**) des globalen Kombinationsnetzwerks ist zur Verwendung in einem für kollektive Operationen optimierten Netzwerk vorgesehen, einem Netzwerk, das Computerknoten eines parallelen Computers als Binärbaum aufbaut. In dem Beispiel von Fig. 3B ermöglicht der Adapter (**188**) des globalen Kombinationsnetzwerks Datenübertragungen an und von zwei Kindknoten über vier einseitig gerichtete Datenübertragungs-Verbindungsleitungen (**190**). Der Adapter (**188**) des globalen Kombinationsnetzwerks ermöglicht auch Datenübertragungen an einen und von einem Elternknoten über zwei einseitig gerichtete Datenübertragungs-Verbindungsleitungen (**192**).

[0048] Zur näheren Erklärung zeigt Fig. 4 eine Zeichnung mit Linien, die ein beispielhaftes Datenübertragungsnetzwerk (**108**) darstellt, das für Punkt-zu-Punkt-Operationen optimiert ist, welche in Systemen von Nutzen sind, die Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung kompilieren können. In dem Beispiel von Fig. 4 stellen Punkte Computerknoten (**102**) eines parallelen Computers dar, und die punktierten Linien zwischen den Punkten stellen Datenübertragungs-Verbindungsleitungen (**103**) zwischen Computerknoten dar. Die Datenübertragungs-Verbindungsleitungen sind mit Punkt-zu-Punkt-Datenübertragungsadaptern realisiert, die ähnlich dem für das Beispiel in Fig. 3A gezeigten Adapter mit Datenübertragungs-Verbindungsleitungen auf drei Achsen, x, y und z, und in und aus sechs Richtungen +x (**181**), –x (**182**), +y (**183**), –y (**184**), +z (**185**) und –z (**186**) sind. Die Verbindungsleitungen und die Computerknoten werden von diesem für Punkt-zu-Punkt-Operationen optimierten Datenübertragungsnetzwerk zu einer dreidimensionalen Masche (**105**) zusammengefasst. Die Masche (**105**) hat Wrap-around-Verbindungsleitungen auf jeder Achse, die die äußersten Computerknoten in der Masche (**105**) auf gegenüberliegenden Seiten der Masche (**105**) verbinden. Diese Wrap-around-Verbindungsleitungen bilden einen Teil eines Torus (**107**). Jeder Computerknoten in dem Torus hat einen Speicherplatz in dem Torus, der von einem Satz von x-, y-, z-Koordinaten eindeutig angegeben wird. Der Leser wird bemerken, dass die Wrap-around-Verbindungsleitungen in der y- und in der z-Richtung aus Gründen der Übersichtlichkeit weggelassen wurden, jedoch in ähnlicher Weise wie die in der x-Richtung gezeigte Wrap-around-Verbindungsleitung konfiguriert sind. Um die Erklärung verständlicher zu machen, ist das Datenübertragungsnetzwerk von Fig. 4 mit nur 27 Computerknoten gezeigt, aber der Leser wird erkennen, dass ein für Punkt-zu-Punkt-Operationen optimiertes Datenübertragungsnetzwerk zur Verwendung beim Kompilieren von Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung einige wenige Computerknoten oder aber Tausende von Computerknoten enthalten kann.

[0049] Zur näheren Erklärung zeigt Fig. 5 eine Zeichnung mit Linien, die ein beispielhaftes für kollektive Operationen optimiertes Datenübertragungsnetzwerk (**106**) darstellt, das in Systemen von Nutzen ist, die Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung kompilieren können. Das beispielhafte Datenübertragungsnetzwerk von Fig. 5 beinhaltet Datenübertragungs-Verbindungsleitungen, die mit den Computerknoten verbunden sind, um die Computerknoten als einen Baum aufzubauen. In dem Beispiel von Fig. 5 stellen Punkte Computerknoten (**102**) eines parallelen Computers dar, und die punktierten Linien (**103**) zwischen den Punkten stellen Datenübertragungs-Verbindungsleitungen zwischen Computerknoten dar. Die Datenübertragungs-Verbindungsleitungen sind mit globalen Kombinationsnetzwerk-Adaptern realisiert, die ähnlich dem für das Beispiel in Fig. 3B gezeigten Adapter sind, wobei jeder Knoten üblicherweise Datenübertragungen an und von zwei Kindknoten und Datenübertragungen an einen und von einem Elternknoten, mit ein paar Ausnahmen, ermöglicht. Knoten in einem Binärbaum (**106**) können als ein physischer Wurzelknoten (**202**), Zweigknoten (**204**) und Blattknoten (**206**) beschrieben werden. Der Wurzelknoten (**202**) hat zwei Kind-, aber keinen Elternknoten. Die Blattknoten (**206**) haben jeweils einen Elternknoten, aber Blattknoten haben keine Kindknoten. Die Zweigknoten (**204**) haben jeweils einen Eltern- und zwei Kindknoten. Die Verbindungsleitungen und die Computerknoten werden von diesem für kollektive Operationen optimierten Datenübertragungsnetzwerk dabei zu einem Binärbaum (**106**) aufgebaut. Um die Erklärung verständlicher zu machen, ist das Datenübertragungsnetzwerk von Fig. 5 mit nur 31 Computerknoten gezeigt, aber der Leser wird erkennen, dass ein für kollektive Operationen optimiertes Datenübertragungsnetzwerk zur Verwendung in Systemen, um Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung zu kompilieren, einige wenige Computerknoten oder aber Tausende von Computerknoten enthalten kann.

[0050] In dem Beispiel von Fig. 5 wird jedem Knoten in dem Baum eine Einheitenkennung zugewiesen, die als ein ”Rang” (**250**) bezeichnet wird. Der Rang eines Knoten kennzeichnet eindeutig den Speicherplatz des Knotens in dem Baumnetzwerk zur Verwendung sowohl bei Punkt-zu-Punkt- als auch bei kollektiven Operationen in dem Baumnetzwerk. In diesem Beispiel werden die Ränge als ganze Zahlen zugewiesen, wobei mit 0 begonnen wird, die dem Wurzelknoten (**202**) zugewiesen wird, 1 wird dem ersten Knoten in der zweiten Ebene des Baumes zugewiesen, 2 wird dem zweiten Knoten in der zweiten Ebene des Baumes zugewiesen, 3 wird dem ersten Knoten in der dritten Ebene des Baumes zugewiesen, 4 wird dem zweiten Knoten in der dritten Ebene des Baumes zugewiesen und so weiter. Zur einfacheren Darstellung werden hier nur die Ränge der ersten drei Ebenen des Baums gezeigt, doch wird allen Computerknoten in dem Baumnetzwerk ein eindeutiger Rang zugewiesen.

[0051] Zur näheren Erklärung zeigt Fig. 6 ein weiteres beispielhaftes verteiltes Datenverarbeitungssystem zum Kompilieren von Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung, in dem das verteilte Datenverarbeitungssystem als eine hybride Datenverarbeitungsumgebung ausgeführt ist. Entsprechend der Verwendung des Begriffs in dieser Beschreibung ist

eine ”hybride Datenverarbeitungsumgebung” insofern eine Datenverarbeitungsumgebung, als dass sie Computerprozessoren enthält, die mit dem Computerspeicher funktionsmäßig verbunden sind, um eine Datenverarbeitung in der Form einer Ausführung von Computerprogrammbefehlen durchzuführen, die in dem Speicher abgelegt und auf den Prozessoren ausgeführt werden. Die hybride Datenverarbeitungsumgebung (**600**) von Fig. 6 enthält einen Computerknoten (**603**), der eine kleine gesonderte hybride Datenverarbeitungsumgebung darstellt, welche in Verbindung mit anderen ähnlichen Computerknoten (**602**) eine größere hybride Datenverarbeitungsumgebung bildet.

[0052] Der beispielhafte Computerknoten (**603**) von Fig. 6 kann die Ausführung eines Computerhauptprogramms auf Benutzerebene vornehmen, indem er administrative Dienste wie zum Beispiel das einleitende Programmladen und dergleichen von einer Service-Anwendung übernimmt, die auf einem Service-Knoten ausgeführt wird, welcher mit dem Computerknoten (**603**) über ein Datenübertragungsnetzwerk verbunden ist. Der beispielhafte Computerknoten kann für Datenübertragungen auch mit einem oder mehreren Eingabe/Ausgabe-(E/A-)Knoten verbunden sein, die es dem Computerknoten ermöglichen, den Zugriff auf den Datenspeicher und andere E/A-Funktionen zu erhalten. Die E/A-Knoten und der Service-Knoten können mit dem beispielhaften Computerknoten (**603**), mit anderen Computerknoten (**602**) in der größeren hybriden Datenverarbeitungsumgebung und mit E/A-Einheiten über ein lokales Netzwerk (LAN) verbunden werden, das mittels eines Hochgeschwindigkeits-Ethernet oder eines Datenübertragungsnetzwerks eines anderen Netzwerk-Typs realisiert wird, wie für den Fachmann zu erkennen ist. Zu E/A-Einheiten, die in einer größeren hybriden Datenverarbeitungsumgebung nützlich sind, welche den Computerknoten (**603**) enthält, können ein nichtflüchtiger Speicher für die Datenverarbeitungsumgebung in Form von einer Datenspeichereinheit, eine Ausgabeeinheit für die hybride Datenverarbeitungsumgebung in Form von einem Drucker und eine E/A-Einheit für den Benutzer in Form von einem Computerendgerät gehören, welches eine Schnittstelle einer Service-Anwendung ausführt, die einem Benutzer eine Schnittstelle bereitstellt, um Computerknoten in der hybriden Datenverarbeitungsumgebung zu konfigurieren und um die Ausführung von Befehlen eines Computerhauptprogramms auf Benutzerebene durch die Computerknoten einzuleiten.

[0053] Der Computerknoten (**603**) in dem Beispiel von Fig. 6 ist in einer erweiterten Ansicht gezeigt, um eine hybride Datenverarbeitungsumgebung (**600**) ausführlicher erklären zu können, die mit anderen hybriden Datenverarbeitungsumgebungen wie zum Beispiel den anderen Computerknoten (**602**) kombiniert werden kann, um eine größere hybride Datenverarbeitungsumgebung zu bilden. Der Computerknoten (**603**) in dem Beispiel von Fig. 6 enthält einen Hostcomputer (**610**). Ein Host-Computer (**610**) ist ein ”Host” in dem Sinn, dass es sich dabei um den Host-Computer handelt, welcher Schnittstellenfunktionen zwischen einem Computerknoten und anderen Komponenten der hybriden Datenverarbeitungsumgebung ausführt, die sich außerhalb eines bestimmten Computerknotens befindet. Das heißt, es ist der Host-Computer, der einleitende Urlade-Prozeduren, Selbsttests beim Einschalten, grundlegende E/A-Funktionen ausführt, Programm-Ladeoperationen auf Benutzerebene von Service-Knoten übernimmt und so weiter.

[0054] Der Host-Computer (**610**) in dem Beispiel von Fig. 6 enthält einen Computerprozessor (**652**), der über einen Hochgeschwindigkeits-Speicherbus (**653**) mit dem Computerspeicher, dem Direktzugriffspeicher (RAM) (**642**), funktionsmäßig verbunden ist. Der Prozessor (**652**) in jedem Hostcomputer (**610**) verfügt über einen Satz von Architekturregistern (**654**), der die Architektur des Hostcomputers bestimmt. Der beispielhafte Computerknoten (**603**) von Fig. 6 enthält auch einen oder mehrere Beschleuniger (**604**, **605**). Ein Beschleuniger (**604**) ist insofern ein ”Beschleuniger”, als jeder Beschleuniger eine Beschleuniger-Architektur hat, die in Bezug auf die Architektur des Hostcomputers hinsichtlich der Ausführungsgeschwindigkeit einer bestimmten Klasse von Berechnungsfunktionen optimiert ist. Zu solchen beschleunigten Berechnungsfunktionen gehören zum Beispiel die Vektorverarbeitung, Gleitkomma-Operationen und andere, wie für den Fachmann zu erkennen ist. Jeder Beschleuniger (**604**, **605**) in dem Beispiel von Fig. 6 enthält einen Computerprozessor (**648**), der über einen Hochgeschwindigkeits-Speicherbus (**651**) mit dem RAM (**640**) funktionsmäßig verbunden ist. Im RAM (**640**, **642**) des Hostcomputers und in den Beschleunigern (**604**, **605**) ist ein Betriebssystem (**645**) gespeichert. Zu Betriebssystemen, die in Hostcomputern und Beschleunigern von hybriden Datenverarbeitungsumgebungen gemäß Ausführungsformen der vorliegenden Erfindung von Nutzen sind, gehören $UNIX_{TM}$, $Linux_{TM}$, Microsoft XP_{TM}, Microsoft $Vista_{TM}$, Microsoft NT_{TM}, AIX_{TM}, das Betriebssystem i5/OS von IBM_{TM} und andere, wie für den Fachmann zu erkennen ist. Es besteht keine Notwendigkeit, dass das Betriebssystem in den Hostcomputern dasselbe wie das in den Beschleunigern verwendete Betriebssystem ist.

[0055] Der Prozessor (**648**) jedes Beschleunigers (**604**, **605**) verfügt über einen Satz von Architekturregistern (**650**), der die Architektur der Beschleuniger bestimmt. Die Architekturregister (**650**) des Prozessors (**648**) jedes Beschleunigers unterscheiden sich von den Architekturregistern (**654**) des Prozessors (**652**) in dem Hostcomputer (**610**). Bei den Architekturregistern handelt es sich um Register, auf die Befehle eines Computer-

programms zugreifen können, die auf jeder Architektur ausgeführt werden, Register wie zum Beispiel ein Befehlsregister, ein Programmzähler, Speicherindexregister, Zeiger auf die aktuelle Adresse im Stapelspeicher (stack pointer) und dergleichen. Bei voneinander abweichenden Architekturen wäre es zwar möglich, aber unüblich, dass ein Hostcomputer und ein Beschleuniger dieselben Befehlssätze unterstützen. Als solches würde man im Allgemeinen nicht erwarten, dass für die Ausführung auf dem Prozessor (**648**) eines Beschleunigers (**604**) kompilierte Befehle eines Computerprogramms nativ auf dem Prozessor (**652**) des HostComputers (**610**) ausgeführt werden und umgekehrt. Überdies würde man aufgrund der üblichen Unterschiede bei den Hardware-Architekturen zwischen Host-Prozessoren und Beschleunigern im Allgemeinen nicht erwarten, dass für die Ausführung auf dem Prozessor (**652**) eines Hostcomputers (**610**) kompilierte Befehle eines Computerprogramms nativ auf dem Prozessor (**648**) eines Beschleunigers (**604**) ausgeführt werden, selbst wenn der Beschleuniger den Befehlssatz des Hostcomputers unterstützen würde. Ein Beschleuniger (**604**) ist insofern ein ”Beschleuniger”, als jeder Beschleuniger eine Beschleuniger-Architektur hat, die in Bezug auf die Architektur des Hostcomputers hinsichtlich der Ausführungsgeschwindigkeit einer bestimmten Klasse von Berechnungsfunktionen optimiert ist. Das heißt, dass die Ausführung einer Funktion oder mehrerer Funktionen, für die der Beschleuniger optimiert ist, auf dem Beschleuniger schneller vonstatten geht, als wenn diese Funktionen auf dem Prozessor des Hostcomputers ausgeführt würden.

[0056] Zu Beispielen für hybride Datenverarbeitungsumgebungen gehören ein Datenverarbeitungssystem, das wiederum einen oder mehrere Hostcomputer enthält, wobei jeder Hostcomputer über einen x86-Prozessor verfügt, und Beschleuniger, deren Architekturregister den PowerPC-Befehlssatz ausführen. Für die Ausführung auf den x86-Prozessoren in den Hostcomputern kompilierte Befehle eines Computerprogramms können nicht von den PowerPC-Prozessoren in den Beschleunigern nativ ausgeführt werden. Der Leser wird außerdem erkennen, dass einige der in dieser Beschreibung beschriebenen beispielhaften hybriden Datenverarbeitungsumgebungen auf der Architektur des Supercomputers des Los Alamos National Laboratory (”LANL”) beruhen, die im Rahmen des Projekts ”LANL-Roadrunner” (benannt nach dem Staatsvogel von New Mexico) entwickelt wurde, jener Supercomputer-Architektur, die dafür bekannt ist, dass sie zum ersten Mal ein ”Petaflop”, eine Billiarde Gleitkomma-Operationen pro Sekunde, erzeugt hat. Die Supercomputer-Architektur des LANL enthält viele Hostcomputer mit Zweikern-Opteron-Prozessoren von AMD, die mit vielen Beschleunigern mit Cell-Prozessoren von IBM verbunden sind, wobei die Opteron-Prozessoren und die Cell-Prozessoren eine andere Architektur haben.

[0057] In dem Beispiel von Fig. 6 sind der Hostcomputer (**610**) und die Beschleuniger (**604**, **605**) für Datenübertragungen über ein Nachrichtenübermittlungsmodul auf Systemebene (system level message passing module – SLMPM) (**646**) und zwei Datenübertragungsnetzwerke (**628**, **630**) mit mindestens zwei verschiedenen Arten von Netzwerken aufeinander abgestimmt. Ein Datenübertragungsnetzwerk (**628**, **630**) stellt eine Konfiguration einer Datenübertragungs-Hardware und -Software dar, die eine Datenübertragungsverbindung zwischen einem Hostcomputer und einem Beschleuniger realisiert. Zu Beispielen für Arten von Datenübertragungsnetzwerken gehören Peripheral Component Interconnect (PCI), PCI express (PCIe), Ethernet, Infiniband, Fibre Channel, Small Computer System Interface (”SCSI”), External Serial Advanced Technology Attachment (eSATA), Universal Serial Bus (USB) und so weiter, wie für den Fachmann zu erkennen ist. In dem Beispiel von Fig. 6 sind der Hostcomputer (**610**) und die Beschleuniger (**604**, **605**) für Datenübertragungen über ein PCIe-Netzwerk (**630**) durch PCIe-Datenübertragungsadapter (**660**) und über ein Ethernet-Netzwerk (**628**) durch Ethernet-Datenübertragungsadapter (**661**) aufeinander abgestimmt. Die Verwendung von PCIe und Ethernet dient lediglich der Erklärung und ist nicht als Einschränkung der Erfindung zu verstehen. Der Fachleser erkennt sofort, dass zu hybriden Datenverarbeitungsumgebungen gemäß Ausführungsformen der vorliegenden Erfindung auch Netzwerke anderer Art wie zum Beispiel PCI, Infiniband, Fibre Channel, SCSI, eSATA, USB und so weiter gehören können.

[0058] Ein SLMPM (**646**) ist ein Modul oder eine Bibliothek mit Computerprogrammbefehlen, das beziehungsweise die Anwendungen auf Benutzerebene eine Anwendungsprogrammierschnittstelle (API) zur Verfügung stellt, um Datenübertragungen auf der Grundlage von Nachrichten zwischen dem Hostcomputer (**610**) und dem Beschleuniger (**604**, **605**) durchzuführen. Zu Beispielen für Bibliotheken für Datenübertragungen auf der Grundlage von Nachrichten, die zur Verwendung als ein SLMPM gemäß Ausführungsformen der vorliegenden Erfindung verbessert werden können, gehören Folgende:

- die Message Passing Interface oder ”MPI”, eine in zwei Versionen erhältliche Schnittstelle in Industriestandard, die erstmalig auf der Supercomputing-Konferenz 1994 vorgestellt wurde und von keiner großen Standardisierungsorganisation unterstützt wird,
- die Data Communication and Synchronization interface (DACS) des LANL-Supercomputers,

- die POSIX-Threads-Bibliothek (Pthreads), ein IEEE-Standard für verteilte Multithread-Verarbeitung,
- die Schnittstelle ”Open Multi-Processing” (OpenMP), eine von der Industrie unterstützte Spezifikation für die parallele Programmierung, und
- andere für den Fachmann erkennbare Bibliotheken.

[0059] Um Datenübertragungen auf der Grundlage von Nachrichten zwischen dem Hostcomputer (**610**) und dem Beschleuniger (**604**) zu unterstützen, verfügen sowohl der Hostcomputer (**610**) als auch der Beschleuniger (**604**) in diesem Beispiel über ein SLMPM (**646**), so dass Übertragungen auf der Grundlage von Nachrichten auf beiden Seiten einer jeden Verbindung für Datenübertragungen gesendet und empfangen werden können.

[0060] Das SLMPM (**646**) wird in diesem Beispiel im Allgemeinen zur Datenverarbeitung in einer hybriden Datenverarbeitungsumgebung (**600**) betrieben, indem die Datenübertragungsleistung für eine Vielzahl von Datenübertragungsmodi zwischen dem Hostcomputer (**610**) und den Beschleunigern (**604**, **605**) überwacht wird, eine Anforderung (**668**) für das Senden von Daten entsprechend einem Datenübertragungsmodus von dem Hostcomputer an einen Beschleuniger empfangen wird, festgestellt wird, ob die Daten entsprechend dem angeforderten Datenübertragungsmodus gesendet werden sollen, und wenn die Daten nicht entsprechend dem angeforderten Datenübertragungsmodus gesendet werden sollen: Auswählen eines anderen Datenübertragungsmodus und Senden der Daten entsprechend dem ausgewählten Datenübertragungsmodus. In dem Beispiel von Fig. 6 wird die überwachte Leistung in Form von Daten der überwachten Leistung (**674**) dargestellt, die während des Betriebs des Computerknotens (**603**) von dem SLMPM (**646**) im RAM (**642**) des Hostcomputers (**610**) gespeichert werden.

[0061] Ein Datenübertragungsmodus gibt einen Typ eines Datenübertragungsnetzwerks, eine Datenübertragungs-Verbindungsleitung und ein Datenübertragungsprotokoll (**678**) an. Eine Datenübertragungs-Verbindungsleitung (**656**) ist eine Datenübertragungsverbindung zwischen einem Hostcomputer und einem Beschleuniger. In dem Beispiel von Fig. 6 kann eine Verbindungsleitung (**656**) zwischen dem Hostcomputer (**610**) und dem Beschleuniger **604**) die PCIe-Verbindung (**638**) oder die Ethernet-Verbindung (**631**, **632**) über das Ethernet-Netzwerk (**606**) beinhalten. Eine Verbindungsleitung (**656**) zwischen dem Hostrechner (**610**) und dem Beschleuniger (**605**) in dem Beispiel von Fig. 6 kann die PCIe-Verbindung (**636**) oder die Ethernet-Verbindung (**631**, **634**) über das Ethernet-Netzwerk (**606**) beinhalten. In dem Beispiel von Fig. 6 ist für jede Art von Netzwerk zwar nur eine Verbindungsleitung zwischen dem Hostcomputer und dem Beschleuniger gezeigt, doch erkennt der Fachleser sofort, dass es für jede Art von Netzwerk beliebig viele Verbindungsleitungen geben kann.

[0062] Ein Datenübertragungsprotokoll ist ein Satz von Standard-Regeln für die Darstellung und Übertragung von Daten, die Identitätsprüfung und Fehlererkennung, die erforderlich sind, um Informationen von einem Hostcomputer (**610**) an einen Beschleuniger (**604**) zu senden. In dem Beispiel von Fig. 6 kann das SLMPM (**646**) eines von mehreren Protokollen (**678**) für Datenübertragungen zwischen dem Hostcomputer (**610**) und dem Beschleuniger auswählen. Zu Beispielen für solche Protokolle (**678**) gehören der Austausch von Daten mit gemeinsam genutzten Speichern (shared memory tansfer – SMT) (**680**), der mit einer Sende- und einer Empfangsoperation (**681**) ausgeführt wird, und der direkte Speicherzugriff (direct memory access – DMA) (**682**), der mit PUT- und GET-Operationen (**683**) ausgeführt wird.

[0063] ”Shared Memory Transfer” ist ein Datenübertragungsprotokoll, das dazu dient, Daten zwischen einem Hostcomputer und einem Beschleuniger in einen gemeinsam genutzten Speicherbereich (**658**) zu leiten, der für diesen Zweck zugeordnet wurde, so dass sich jeweils nur eine Instanz der Daten in dem Speicher befindet. Betrachten wir das Folgende als eine Übertragung mit gemeinsam genutztem Speicher zwischen dem Hostcomputer (**610**) und dem Beschleuniger (**604**) von Fig. 6. Eine Anwendung (**669**) stellt eine Anforderung (**668**) für eine Übertragung von Daten (**676**) vom Hostcomputer (**610**) an den Beschleuniger (**604**) gemäß dem Protokoll ”SMT” (**680**). Eine solche Anforderung (**668**) kann eine Speicheradresse enthalten, die für einen solchen gemeinsam genutzten Speicher zugeordnet ist. In diesem Beispiel ist das gemeinsam genutzte Speichersegment (**658**) in einem Speicherplatz auf dem Beschleuniger (**604**) gezeigt, aber der Leser erkennt, dass sich gemeinsam genutzte Speichersegmente auf dem Beschleuniger (**604**), auf dem Hostcomputer (**610**), sowohl auf dem Hostcomputer als auch auf dem Beschleuniger oder auch ganz außerhalb des lokalen Computerknotens (**603**) befinden können – solange der Hostcomputer und der Beschleuniger nach Bedarf auf das Segment zugreifen können. Um eine Übertragung mit gemeinsam genutzten Speicher durchzuführen, stellt das SLMPM (**646**) auf dem Hostcomputer (**610**) mittels eines Quittungsbetriebs, der ähnlich dem TCP-Protokoll ist, eine Datenübertragungsverbindung mit dem SLMPM (**646**) her, das auf dem Beschleuniger (**604**) ausgeführt wird. Das SLMPM (**646**) erzeugt dann eine Nachricht (**670**), die einen Kopfbereich und Nutzdaten enthält, und stellt die Nachricht in eine Nachrichten-Sendewarteschlange für eine bestimmte Verbindungsleitung eines bestimmten Netzwerks. Bei der Erzeugung der Nachricht fügt das SLMPM eine Kennzeichnung des Beschleunigers

und eine Kennzeichnung eines Prozesses, der auf dem Beschleuniger ausgeführt wird, in den Kopfbereich der Nachricht ein. Das SLMPM fügt auch die Speicheradresse aus der Anforderung (**668**) in die Nachricht ein, entweder in den Kopfbereich oder als Teil der Nutzdaten. Das SLMPM fügt auch die Daten (**676**), die in der Nachricht (**670**) übertragen werden sollen, als Teil der Nutzdaten der Nachricht ein. Die Nachricht wird dann von einem Datenübertragungsadapter (**660**, **661**) über ein Netzwerk (**628**, **630**) an das SLMPM übertragen, das auf dem Beschleuniger (**604**) ausgeführt wird, wobei das SLMPM die Nutzdaten, die Daten (**676**), die übertragen wurden, in dem gemeinsam genutzten Speicherbereich (**658**) im RAM (**640**) entsprechend der Speicheradresse in der Nachricht speichert.

[0064] Bei dem direkten Speicherzugriff (direct memory access – DMA) handelt es sich um ein Datenübertragungsprotokoll zum Austausch von Daten zwischen einem Hostcomputer und einem Beschleuniger, während die Arbeitslast des Computerprozessors (**652**) gleichzeitig verringert wird. Bei einer DMA-Übertragung wird im Wesentlichen ein Block des Speichers von einem Speicherplatz an einen anderen kopiert, gewöhnlich von einem Hostcomputer an einen Beschleuniger oder umgekehrt. Ein Hostcomputer oder ein Beschleuniger oder aber beide können eine DMA-Steuereinheit und eine DMA-Komponente, bei der es sich um eine Verbindung von Computer-Hardware und -Software für den direkten Speicherzugriff handelt, enthalten. Der direkte Speicherzugriff beinhaltet Lese- und Schreiboperationen aus dem beziehungsweise in den Speicher von Beschleunigern und Hostcomputern, während die Arbeitslast ihrer Prozessoren gleichzeitig verringert wird. Eine DMA-Komponente eines Beschleunigers kann zum Beispiel Daten in den für DMA-Zwecke zugeordneten Speicher schreiben oder Daten aus diesem Speicher lesen, während der Prozessor des Beschleunigers Befehle eines Computerprogramms ausführt oder seinen Betrieb in anderer Weise fortsetzt. Das heißt, ein Computerprozessor kann einen Befehl zum Ausführen einer DMA-Übertragung ausgeben, jedoch führt die DMA-Komponente und nicht der Prozessor die Übertragung aus.

[0065] In dem Beispiel von Fig. 6 enthält nur der Beschleuniger (**604**) eine DMA-Steuereinheit (**685**) und eine DMA-Komponente (**684**), während der Hostcomputer keine von beiden enthält. In dieser Ausführungsform leitet der Prozessor (**652**) auf dem Hostcomputer eine DMA-Übertragung von Daten von dem Hostcomputer an den Beschleuniger ein, indem er eine Nachricht gemäß dem SMT-Protokoll an den Beschleuniger schickt, mit der er den Beschleuniger anweist, eine ferne ”GET”-Operation durchzuführen. Die in dem Beispiel von Fig. 6 gezeigte Konfiguration, bei der der Beschleuniger (**604**) die einzige Einheit ist, die eine DMA-Komponente enthält, dient lediglich der Erklärung und ist nicht als Einschränkung zu verstehen. Der Fachleser erkennt sofort, dass in vielen Ausführungsformen sowohl ein Hostcomputer als auch ein Beschleuniger eine DMA-Steuereinheit und eine DMA-Komponente enthalten können, während in noch anderen Ausführungsformen nur ein Hostcomputer eine DMA-Steuereinheit und eine DMA-Komponente enthält.

[0066] Um in der hybriden Datenverarbeitungsumgebung von Fig. 6 ein DMA-Protokoll auszuführen, wird für den Zugriff durch die DMA-Komponente ein bestimmter Speicherbereich zugeordnet. Die Zuordnung eines solchen Speichers kann unabhängig von anderen Beschleunigern oder Hostcomputern vorgenommen oder von einem anderen Beschleuniger oder Hostcomputer eingeleitet und in Zusammenarbeit mit einem anderen Beschleuniger oder Hostcomputer abgeschlossen werden. Gemeinsam genutzte Speicherbereiche, die beispielsweise gemäß dem SMA-Protokoll zugeordnet werden, können Speicherbereiche sein, die einer DMA-Komponente zur Verfügung gestellt werden. Das heißt, der anfängliche Aufbau und die Durchführung von DMA-Datenübertragungen in der hybriden Datenverarbeitungsumgebung (**600**) von Fig. 6 kann zumindest teilweise mittels des Shared-Memory-Transfer- oder mittels eines anderen Außerband-Datenübertragungsprotokolls vorgenommen werden, wobei sich ”Außerband” auf die DMA-Komponente bezieht. Die Zuordnung von Speicher zur Durchführung von DMA-Übertragungen ist mit einer verhältnismäßig hohen Latenzzeit verbunden, aber sobald er einmal zugeordnet worden ist, ermöglicht das DMA-Protokoll Datenübertragungen hoher Bandbreite, die den Prozessor weniger belasten als viele andere Datenübertagungsprotokolle.

[0067] Eine direkte ”PUT”-Operation ist eine Betriebsweise zur Übertragung von Daten von einer DMA-Komponente auf einer Ursprungseinheit an eine DMA-Komponente auf einer Zieleinheit. Eine direkte ”PUT”-Operation gestattet die Übertragung und Speicherung von Daten auf der Zieleinheit mit nur geringer Beteiligung des Prozessors der Zieleinheit. Um die Beteiligung des Prozessors der Zieleinheit an der direkten ”PUT”-Operation gering zu halten, überträgt die Ursprungs-DMA-Komponente die auf der Zieleinheit zu speichernden Daten zusammen mit einer bestimmten Kennzeichnung eines Speicherorts auf der Zieleinheit. Die Ursprungs-DMA kennt den bestimmten Speicherort auf der Zieleinheit, da die Ziel-DMA-Komponente den bestimmten Speicherort zur Speicherung der Daten auf der Zieleinheit der Ursprungs-DMA-Komponente zuvor zur Verfügung gestellt hat.

[0068] Eine ferne ”GET”-Operation, die manchmal auch als ”rGET” bezeichnet wird, ist eine weitere Betriebsweise zur Übertragung von Daten von einer DMA-Komponente auf einer Ursprungseinheit an eine DMA-Komponente auf einer Zieleinheit. Eine ferne ”GET”-Operation gestattet die Übertragung und Speicherung von Daten auf der Zieleinheit mit nur geringer Beteiligung des Prozessors der Ursprungseinheit. Um die Beteiligung des Prozessors der Ursprungseinheit an der fernen ”GET”-Operation gering zu halten, speichert die Ursprungs-DMA-Komponente die Daten an einem Speicherplatz, auf den die Ziel-DMA-Komponente zugreifen kann, sie benachrichtigt die Ziel-DMA-Komponente direkt oder außerhalb des Bandbereichs mittels einer Übertragung mit gemeinsam genutztem Speicher über den Speicherplatz und den Umfang der zur Übertragung bereiten Daten, und die Ziel-DMA-Komponente ruft die Daten aus dem Speicherplatz ab.

[0069] Die Überwachung der Datenübertragungsleistung für eine Vielzahl von Datenübertragungsmodi kann die Überwachung von mehreren Anforderungen (**668**) in einer Nachrichten-Sendeanforderungswarteschlange (**662** bis **165**) für eine Datenübertragungs-Verbindungsleitung (**656**) beinhalten. In dem Beispiel von Fig. 6 ist jeder Nachrichten-Sendeanforderungswarteschlange (**662** bis **165**) eine bestimmte Datenübertragungs-Verbindungsleitung (**656**) zugeordnet. Jede Warteschlange (**662** bis **165**) enthält Einträge für Nachrichten (**670**), die Daten (**676**) enthalten, welche von den Datenübertragungsadaptern (**660**, **661**) auf einer Datenübertragungs-Verbindungsleitung (**656**), die der Warteschlange zugeordnet ist, übertragen werden sollen.

[0070] Die Überwachung der Datenübertragungsleistung für eine Vielzahl von Datenübertragungsmodi kann auch die Überwachung der Auslastung eines gemeinsam genutzten Speicherbereichs (**658**) beinhalten. In dem Beispiel von Fig. 6 ist der gemeinsam genutzte Speicherbereich (**658**) im RAM (**640**) des Beschleunigers zugeordnet. Die Auslastung stellt den Anteil des zugeordneten gemeinsam benutzten Speicherbereichs dar, in den Daten zum Versenden an eine Zieleinheit gespeichert wurden, die von der Zieleinheit aber noch nicht gelesen oder empfangen wurden, wobei die Auslastung überwacht wird, indem die Schreib- und Leseoperationen in den und aus dem zugeordneten gemeinsam genutzten Speicher erfasst werden. In der hybriden Datenverarbeitungsumgebung (**600**) von Fig. 6 ist der gemeinsam benutzte Speicherbereich – praktisch jeder Speicher – begrenzt. Als solches besteht die Möglichkeit, dass ein gemeinsam genutzter Speicherbereich (**658**) während der Ausführung eines Anwendungsprogramms (**669**) gefüllt wird, so dass die Übertragung von Daten vom Hostcomputer (**610**) an einen Beschleuniger aufgrund von Bereichsbeschränkungen des gemeinsam benutzten Speicherbereichs gegebenenfalls verlangsamt oder sogar angehalten wird.

[0071] In manchen Ausführungsformen der vorliegenden Erfindung kann die hybride Datenverarbeitungsumgebung (**600**) von Fig. 6 so konfiguriert werden, dass sie als eine parallele Datenverarbeitungsumgebung betrieben werden kann, in der zwei oder mehr Instanzen des Anwendungsprogramms (**669**) auf zwei oder mehr Hostcomputern (**610**) in der parallelen Datenverarbeitungsumgebung ausgeführt werden. In solchen Ausführungsformen kann die über alle Datenübertragungsmodi hinweg stattfindende Überwachung der Datenübertragungsleistung auch das Zusammentragen von Informationen (**674**) über die Datenübertragungsleistung über eine Vielzahl von Instanzen des Anwendungsprogramms (**669**) beinhalten, das auf zwei oder mehr Hostcomputern in einer parallelen Datenverarbeitungsumgebung ausgeführt wird. Die zusammengetragenen Informationen (**674**) über die Leistung können zur Berechnung der durchschnittlichen Übertragungslatenzzeiten bei den Datenübertragungsmodi, der durchschnittlichen Anzahl der Anforderungen auf Datenübertragungs-Verbindungsleitungen eines bestimmten Netzwerk-Typs, der durchschnittlichen Auslastung des gemeinsam benutzten Speichers unter der Vielzahl der Hostcomputer und Beschleuniger in der parallelen Datenverarbeitungsumgebung und so weiter verwendet werden, wie für den Fachmann zu erkennen ist. Jede beliebige Kombination dieser Messwerte kann von dem SLMPM verwendet werden, um festzustellen, ob die Daten entsprechend dem angeforderten Datenübertragungsmodus übertragen werden sollen, und auch, um einen anderen Datenübertragungsmodus zur Übertragung der Daten auszuwählen, wenn die Daten nicht entsprechend dem angeforderten Datenübertragungsmodus übertragen werden sollen.

[0072] Das SLMPM (**646**) von Fig. 6 empfängt von einem Anwendungsprogramm (**669**) auf dem Hostcomputer (**610**) eine Anforderung (**668**), Daten (**676**) entsprechend einem Datenübertragungsmodus von dem Hostcomputer (**610**) an den Beschleuniger (**604**) zu übertragen. Diese Daten (**676**) können Befehle eines Computerprogramms enthalten, die zur Ausführung durch den Beschleuniger (**604**) kompiliert werden, wie zum Beispiel eine Programmdatei eines Anwendungsprogramms des Beschleunigers, Daten über eine Arbeitseinheit für ein Anwendungsprogramm des Beschleunigers, Dateien, die für die Ausführung eines Anwendungsprogramms des Beschleunigers erforderlich sind, wie zum Beispiel Bibliotheken, Datenbanken, Treiber und dergleichen. Das Empfangen einer Anforderung (**668**) für die Übertragung von Daten (**676**) entsprechend einem Datenübertragungsmodus kann den Empfang einer Anforderung für die Übertragung von Daten durch einen angegebenen Netzwerk-Typ, den Empfang einer Anforderung für die Übertragung von Daten über eine angegebene Datenübertragungs-Verbindungsleitung von dem Hostcomputer an den Beschleuniger oder den

Empfang einer Anforderung für die Übertragung von Daten von dem Hostcomputer an den Beschleuniger gemäß einem Protokoll beinhalten.

[0073] Eine Anforderung (**668**) für die Übertragung von Daten (**676**) entsprechend einem Datenübertragungsmodus kann als ein Funktionsaufruf einer Anwendung auf Benutzerebene über eine API an das SLMPM (**646**) erfolgen, als ein Aufruf, der einen Datenübertragungsmodus entsprechend einem Protokoll, einem Netzwerk-Typ und einer Verbindungsleitung ausdrücklich angibt. Eine als ein Funktionsaufruf realisierte Anforderung kann ein Protokoll entsprechend der Operation des Funktionsaufrufs selbst angeben. Ein Funktionsaufruf dacs_put() beispielsweise kann einen Aufruf über eine API darstellen, die von einem SLMPM zur Verfügung gestellt wird, das als eine DACS-Bibliothek ausgeführt ist, um Daten im Standard-Modus einer DMA-"PUT"-Operation zu übertragen. Ein solcher Aufruf stellt aus der Sicht der aufrufenden Anwendung und des Programmierers, der die aufrufende Anwendung geschrieben hat, eine Anforderung an die SLMPM-Bibliothek dar, Daten entsprechend dem Standard-Modus zu übertragen, die dem Programmierer als der Standard-Modus bekannt ist, die dem API-Schnellaufruf zugeordnet ist. Die aufgerufene Funktion, in diesem Beispiel dacs_put(), kann in Ausführungsformen mit mehreren Netzwerk-Typen, Protokollen und Verbindungsleitungen codiert werden, damit sie selbst feststellen kann, ob die Daten entsprechend dem angeforderten Datenübertragungsmodus, das heißt, entsprechend dem Standard-Modus der aufgerufenen Funktion, übertragen werden sollen. In einem weiteren Beispiel kann ein Befehl dacs_send() einen Aufruf über eine API darstellen, die von einem SLMPM zur Verfügung gestellt wird, das als eine DACS-Bibliothek ausgeführt ist, um Daten im Standard-Modus einer SMT-"Sende"-Operation zu übertragen, wobei die aufgerufene Funktion dacs_send() in Ausführungsformen mit mehreren Netzwerk-Typen, Protokollen und Verbindungsleitungen wieder codiert wird, damit sie selbst feststellen kann, ob die Daten entsprechend dem angeforderten Modus übertragen werden sollen.

[0074] Eine Kennzeichnung eines bestimmten Beschleunigers in einem Funktionsaufruf kann praktisch einen Netzwerk-Typ angeben. Ein solcher Funktionsaufruf kann eine Kennzeichnung eines bestimmten Beschleunigers als Aufruf-Parameter enthalten. Eine Kennzeichnung eines bestimmten Beschleunigers, beispielsweise durch Verwendung der Kennung (ID) eines PCIe, gibt praktisch den Netzwerk-Typ "PCI" an. In einem weiteren, ähnlichen Beispiel gibt eine Kennzeichnung eines bestimmten Beschleunigers durch Verwendung einer Medienzugriffssteuerungs-(MAC-)Adresse eines Ethernet-Adapters praktisch den Netzwerk-Typ "Ethernet" an. Statt die ID des Beschleunigers von dem Funktionsaufruf einer Anwendung, die auf dem Hostcomputer ausgeführt wird, so auszuführen, dass sie einen Netzwerk-Typ angibt, kann der Funktionsaufruf auch nur eine global eindeutige Kennzeichnung des bestimmten Beschleunigers als einen Parameter des Aufrufs enthalten und damit statt eines Netzwerk-Typs nur eine Verbindungsleitung von dem Hostcomputer zu dem Beschleuniger angeben. In diesem Fall kann die aufgerufene Funktion einen standardmäßigen Netzwerk-Typ zur Verwendung mit einem bestimmten Protokoll einsetzen. Wenn die in dem SLMPM aufgerufene Funktion zum Beispiel mit PCIe als einem standardmäßigen Netzwerk-Typ zur Verwendung mit dem DMA-Protokoll konfiguriert wird und das SLMPM eine Anforderung empfängt, Daten entsprechend dem DMA-Protokoll, einer DMA-PUT- oder einer fernen DMA-GET-Operation an den Beschleuniger (**604**) zu übertragen, gibt die aufgerufene Funktion ausdrücklich den standardmäßigen Netzwerk-Typ für DMA, den Netzwerktyp "PCIe", an.

[0075] In hybriden Datenverarbeitungsumgebungen, in denen nur eine Verbindungsleitung eines jeden Netzwerk-Typs einen einzelnen Hostcomputer auf einen einzelnen Beschleuniger abstimmt, kann die Kennzeichnung eines bestimmten Beschleunigers in einem Parameter eines Funktionsaufrufs im Grunde ebenfalls eine Verbindungsleitung angeben. In hybriden Datenverarbeitungsumgebungen, in denen mehr als eine Verbindungsleitung eines jeden Netzwerk-Typs einen Hostcomputer und einen Beschleuniger aufeinander abstimmt, so zum Beispiel zwei PCIe-Verbindungsleitungen, die den Hostcomputer (**610**) mit dem Beschleuniger (**604**) verbinden, kann die aufgerufene SLMPM-Funktion eine Standard-Verbindungsleitung für den im Parameter des Funktionsaufrufs gekennzeichneten Beschleuniger für den Netzwerk-Typ realisieren, der von der Kennzeichnung des Beschleunigers angegeben wird.

[0076] Das SLMPM (**646**) in dem Beispiel von Fig. 6 stellt in Abhängigkeit von der überwachten Leistung (**674**) auch fest, ob die Daten (**676**) entsprechend dem angeforderten Datenübertragungsmodus übertragen werden sollen. Die Feststellung, ob die Daten (**676**) entsprechend dem angeforderten Datenübertragungsmodus übertragen werden sollen, kann die Feststellung beinhalten, ob die Daten von einem angeforderten Netzwerk-Typ übertragen werden sollen, ob die Daten über eine angeforderte Datenübertragungs-Verbindungsleitung übertragen werden sollen oder ob die Daten entsprechend einem angeforderten Protokoll übertragen werden sollen.

[0077] In hybriden Datenverarbeitungsumgebungen gemäß Ausführungsformen der vorliegenden Erfindung, in denen die über alle Datenübertragungsmodi hinweg stattfindende Überwachung der Datenübertragungsleistung die Überwachung von mehreren Anforderungen in einer Nachrichten-Sendeanforderungswarteschlange

(**662** bis **165**) für eine Datenübertragungs-Verbindungsleitung beinhaltet, kann die Feststellung, ob die Daten (**676**) entsprechend dem angeforderten Datenübertragungsmodus übertragen werden sollen, getroffen werden, indem festgestellt wird, ob die Anzahl der Anforderungen in der Nachrichten-Sendeanforderungswarteschlange einen vorher festgelegten Schwellwert überschreitet. In hybriden Datenverarbeitungsumgebungen gemäß Ausführungsformen der vorliegenden Erfindung, in denen die Überwachung der Datenübertragungsleistung für eine Vielzahl von Datenübertragungsmodi die Überwachung der Auslastung eines gemeinsam genutzten Speicherbereichs beinhaltet, kann die Feststellung, ob die Daten (**676**) entsprechend dem angeforderten Datenübertragungsmodus übertragen werden sollen, getroffen werden, indem festgestellt wird, ob die Auslastung des gemeinsam genutzten Speicherbereichs einen vorher festgelegten Schwellwert überschreitet.

[0078] Wenn die Daten nicht entsprechend dem angeforderten Datenübertragungsmodus übertragen werden sollen, wählt das SLMPM (**646**) in Abhängigkeit von der überwachten Leistung einen anderen Datenübertragungsmodus für die Übertragung der Daten aus und überträgt die Daten (**676**) entsprechend dem ausgewählten Datenübertragungsmodus. Die Auswahl eines anderen Datenübertragungsmodus für die Übertragung der Daten kann die in Abhängigkeit von der überwachten Leistung erfolgende Auswahl eines anderen Typs eines Datenübertragungsnetzwerks, durch den die Daten übertragen werden sollen, die Auswahl einer Datenübertragungs-Verbindungsleitung, über die die Daten übertragen werden sollen, und die Auswahl eines anderen Datenübertragungsprotokolls beinhalten. Betrachten wir als ein Beispiel, dass der angeforderte Datenübertragungsmodus eine DMA-Übertragung unter Verwendung einer PUT-Operation über die Verbindungsleitung (**638**) des PCIe-Netzwerks (**630**) an den Beschleuniger (**604**) ist. Wenn die überwachte Datenübertragungsleistung (**674**) anzeigt, dass die Anzahl der Anforderungen in der Nachrichten-Sendeanforderungswarteschlange (**662**), die zu der Verbindungsleitung (**638**) gehört, einen vorher festgelegten Schwellwert überschreitet, kann das SLMPM einen anderen Netzwerk-Typ, das Ethernet-Netzwerk (**628**) und die Verbindungsleitung (**631**, **632**), über die die Daten (**676**) übertragen werden sollen, auswählen. Ebenfalls zu berücksichtigen ist, dass die überwachte Leistung (**676**) anzeigt, dass die aktuelle Auslastung des gemeinsam genutzten Speicherbereichs (**658**) einen vorher festgelegten Schwellwert unterschreitet, während die Anzahl der ausstehenden DMA-Übertragungen in der Warteschlange (**662**) einen vorher festgelegten Schwellwert überschreitet. In solch einem Fall kann das SLMPM (**646**) auch ein anderes Protokoll auswählen, wie zum Beispiel das Shared-Memory-Transfer-Protokoll, mittels dem die Daten (**674**) übertragen werden sollen.

[0079] Die Auswahl eines anderen Datenübertragungsmodus durch das SLMPM für die Übertragung der Daten (**672**) kann auch die Auswahl eines Datenübertragungsprotokolls (**678**) in Abhängigkeit von der Größe (**672**) der Datenübertragungsnachricht beinhalten. Die Auswahl eines Datenübertragungsprotokolls (**678**) in Abhängigkeit von der Größe (**672**) der Datenübertragungsnachricht kann erfolgen, indem festgestellt wird, ob die Größe einer Nachricht einen vorher festgelegten Schwellwert überschreitet. Bei größeren Nachrichten (**670**) kann das DMA-Protokoll ein bevorzugtes Protokoll sein, da die Auslastung des Prozessors bei der Durchführung einer DMA-Übertragung einer größeren Nachricht (**670**) gewöhnlich geringer ist als bei der Durchführung einer Übertragung einer Nachricht derselben Größe, bei der ein gemeinsamer Speicher genutzt wird.

[0080] Wie vorstehend erwähnt wurde, kann das SLMPM die Daten auch entsprechend dem ausgewählten Datenübertragungsmodus übertragen. Das Übertragen der Daten entsprechend dem ausgewählten Datenübertragungsmodus kann das Übertragen der Daten durch den ausgewählten Typ des Datenübertragungsnetzwerks, das Übertragen der Daten über die ausgewählte Datenübertragungs-Verbindungsleitung oder das Übertragen der Daten gemäß dem ausgewählten Protokoll beinhalten. Das SLMPM (**646**) kann eine Übertragung der Daten entsprechend dem ausgewählten Datenübertragungsmodus durchführen, indem es über einen Einheitentreiber den Übertragungsadapter für den Typ des Datenübertragungs-Netzwerks des ausgewählten Datenübertragungsmodus anweist, die Nachricht (**670**) gemäß einem Protokoll des ausgewählten Datenübertragungsmodus zu übertragen, wobei die Nachricht in einem Kopfbereich der Nachricht eine Kennzeichnung des Beschleunigers und in den Nutzdaten der Nachricht die zu übertragenden Daten (**676**) enthält.

[0081] In dem Beispiel von Fig. 6 ist sowohl auf dem Hostcomputer (**610**) als auch auf dem Beschleuniger (**604**) von einem der Computerknoten ein hierarchischer verteilter Compiler (**155**) installiert. Der hierarchische verteilte Compiler (**155**) ist der Vollständigkeit halber sowohl im Hostcomputer als auch im Beschleuniger gezeigt. Tatsächlich kann der hierarchische verteilte Compiler (**155**) gemäß Ausführungsformen der vorliegenden Erfindung entweder auf einem Hostcomputer oder auf einem oder mehreren Beschleunigern oder sowohl auf einem Hostcomputer als auch auf einem oder mehreren Beschleunigern installiert sein, wie für den Fachmann zu erkennen ist. Der hierarchische verteilte Compiler (**155**) von Fig. 6 ist ein Modul einer automatisierten Datenverarbeitungsmaschine, der Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung kompilieren kann. Der hierarchische verteilte Compiler (**155**) enthält einen Befehl eines Computerprogramms, um die Software durch den Kompilierungsknoten zu kompilieren;

um kompilierte Software, die auf dem Kompilierungsknoten ausgeführt werden soll, durch den Kompilierungsknoten zu verwalten; um einen oder mehrere Knoten in einer nächsten Ebene der Hierarchie des verteilten Verarbeitungssystems durch den Kompilierungsknoten in Abhängigkeit davon, ob kompilierte Software für den ausgewählten Knoten oder für die Nachkommen des ausgewählten Knotens bestimmt ist, auszuwählen; um nur die kompilierte Software, die von dem ausgewählten Knoten oder von dem Nachkommen des ausgewählten Knotens ausgeführt werden soll, an den ausgewählten Knoten zu senden. Jeder der anderen Computerknoten (**602**) von **Fig.** 16 kann auch kompilierte Software empfangen; Feststellen, ob die kompilierte Software für diesen Knoten oder für einen seiner Nachkommen bestimmt ist; die Software zur Ausführung verwalten, wenn die kompilierte Software für diesen Knoten bestimmt ist; und einen anderen Knoten in einer nächsten Ebene des hierarchischen verteilten Verarbeitungssystems in Abhängigkeit von einem Nachkommen für die kompilierte Software auswählen, wenn die kompilierte Software für einen der Nachkommen bestimmt ist, und die kompilierte Software an den ausgewählten anderen Knoten senden.

[0082] Zur näheren Erklärung zeigt Fig. 7 ein beispielhaftes Verfahren zum Kompilieren von Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung. Unter Kompilieren versteht man den Vorgang der Umwandlung von Quellcode, der in einer Computersprache, oftmals in einer höheren Programmiersprache, geschrieben ist, in eine andere, üblicherweise ausführbare Computersprache, die gewöhnlich in Binärform dargestellt ist und manchmal als Objektcode bezeichnet wird. Das Kompilieren gemäß Ausführungsformen der vorliegenden Erfindung erfolgt üblicherweise mit einem hierarchischen verteilten Compiler, der auf Kompilierknoten gemäß der vorliegenden Erfindung installiert ist. Solch ein hierarchischer verteilter Compiler kann Software gewöhnlich zur Verwendung auf mehreren unterschiedlichen Arten von Zielcomputern kompilieren. Als solches kann ein hierarchischer verteilter Compiler Teile von nicht kompilierter Quell-Software in Ziel-Objektcode zur Verwendung auf Computern unterschiedlichster Art kompilieren.

[0083] Ein hierarchisches verteiltes Verarbeitungssystem kann auf mehrere Arten, zum Beispiel in einer Baumstruktur, realisiert werden. Eine solche Baumstruktur kann kär, das heißt eine Baumstruktur beliebiger Ordnung, oder binär sein oder irgendeine andere Form haben, wie für den Fachmann zu erkennen ist. Alternativ können hierarchische verteilte Verarbeitungssysteme gemäß der vorliegenden Erfindung in anderen Formen, die nicht als Baumstrukturen betrachtet werden, realisiert werden, wie für den Fachmann zu erkennen ist. Das Verfahren von Fig. 7 kann in einem verteilten Datenverarbeitungssystem durchgeführt werden, das den vorstehend beschriebenen beispielhaften verteilten Datenverarbeitungssystemen ähnlich ist: den als Beispiel dienenden parallelen Computern der Fig. 1 bis Fig. 5, der als Beispiel dienenden hybriden Datenverarbeitungsumgebung von Fig. 6 und anderen, wie für den Fachmann zu erkennen ist.

[0084] Das Verfahren von Fig. 7 beinhaltet das Kennzeichnen (**802**) des einen oder der mehreren Kompilierungsknoten. Wie vorstehend erwähnt wurde, wandeln der eine oder die mehreren Kompilierungsknoten Quellcode, der in einer Computersprache, üblicherweise in einer höheren Programmiersprache, geschrieben ist, in eine andere, häufig ausführbare Computersprache um, die gewöhnlich in Binärform dargestellt ist und manchmal als Objektcode bezeichnet wird. Die Kompilierungsknoten kompilieren Teile der Software, die auf den Kompilierungsknoten selbst ausgeführt werden sollen, sowie andere Teile der Software, die auf einem anderen Knoten in dem hierarchischen verteilten Netzwerk ausgeführt werden sollen.

[0085] Das Kennzeichnen (**802**) des einen oder der mehreren Kompilierungsknoten gemäß dem Verfahren von Fig. 7 kann durchgeführt werden, indem ein oder mehrere Knoten ausgewählt werden, die für den Kompiliervorgang rechnerisch optimiert werden. Das Auswählen von einem oder mehreren Knoten, die für den Kompiliervorgang rechnerisch optimiert werden, kann das Kennzeichnen von Knoten in Abhängigkeit von deren E/A-Funktionen, Verarbeitungsfunktionen und Speicherfunktionen beinhalten. Oftmals ist ein bestimmtes Gleichgewicht dieser Funktionen optimal für den Kompiliervorgang. Ein solches optimales Gleichgewicht kann beim Kompilieren von verschiedenen Arten von Software-Programmen unterschiedlich sein, wobei das Auswählen von einem oder mehreren Knoten, die für den Kompiliervorgang rechnerisch optimiert werden, das Auswählen von einem oder mehreren Knoten in Abhängigkeit von der jeweiligen Software, die kompiliert werden soll, beinhalten kann.

[0086] Das Kennzeichnen (**802**) des einen oder der mehreren Kompilierungsknoten gemäß dem Verfahren von Fig. 7 kann auch durchgeführt werden, indem ein oder mehrere Knoten ausgewählt werden, die aufgrund von ihrem Standort in der Topologie des hierarchischen verteilten Verarbeitungssystems für den Kompiliervorgang optimiert werden. In einem hierarchischen Datenverarbeitungssystem mit Baumstruktur zum Beispiel kann sich ein Knoten, der ein Wurzelknoten ist, oder ein Knoten, der viele Nachkommen hat, so in der Topologie angeordnet sein, dass dieser Knoten zum Kompilieren von Software für seine Nachkommen optimiert wird. Ein

Nachkomme, so wie der Begriff in dieser Beschreibung verwendet wird, ist ein Knoten, der sich in Ebenen des hierarchischen Datenverarbeitungssystems befindet, welche unterhalb des Kompilierungsknotens und auf einem Zweig des hierarchischen Verarbeitungssystems, das den Kompilierungsknoten enthält, angeordnet sind.

[0087] Das Verfahren von Fig. 7 beinhaltet auch das Bereitstellen (**804**) von zu kompilierender Software für einen oder mehrere Kompilierungsknoten, wobei mindestens ein Teil der zu kompilierenden Software von einem oder mehreren anderen Knoten ausgeführt werden soll. Das Bereitstellen (**804**) von zu kompilierender Software für einen oder mehrere Kompilierungsknoten kann durchgeführt werden, indem die zu kompilierende Software in einer Nachricht an den Kompilierungsknoten gesendet wird, die Software auf den Kompilierungsknoten heruntergeladen wird, die Software von einem Systemadministrator auf dem Kompilierungsknoten installiert wird, oder die zu kompilierende Software in irgendeiner anderen Weise einem oder mehreren Kompilierungsknoten bereitgestellt wird, wie für den Fachmann zu erkennen ist.

[0088] Das Verfahren von Fig. 7 beinhaltet auch das Kompilieren (**806**) der Software durch den Kompilierungsknoten. Das Kompilieren (**806**) der Software durch den Kompilierungsknoten kann durchgeführt werden, indem Teile der Software, die auf dem Kompilierungsknoten ausgeführt werden sollen, gekennzeichnet werden und der in einer Computersprache der nichtkompilierten Software geschriebene Code in eine ausführbare Computersprache zur Ausführung auf dem Kompilierungsknoten umgewandelt wird. Das Kompilieren (**806**) der Software durch den Kompilierungsknoten kann durchgeführt werden, indem Teile der Software, die auf einem anderen Knoten ausgeführt werden sollen, gekennzeichnet werden, die Ziel-Ausführungsumgebung auf dem anderen Knoten gekennzeichnet wird und die in der Computersprache der nichtkompilierten Software geschriebene Software in eine ausführbare Computersprache zur Ausführung auf dem anderen Knoten umgewandelt wird.

[0089] Das Verfahren von Fig. 7 beinhaltet auch das Verwalten (**808**) von kompilierter Software, die auf dem Kompilierungsknoten ausgeführt werden soll, durch den Kompilierungsknoten. Das Verwalten (**808**) von kompilierter Software, die auf dem Kompilierungsknoten ausgeführt werden soll, durch den Kompilierungsknoten kann durchgeführt werden, indem kompilierte Software, die auf dem Kompilierungsknoten ausgeführt werden soll, zur Ausführung gespeichert wird.

[0090] Das Verfahren von Fig. 7 beinhaltet auch das Auswählen (**810**) von einem oder mehreren Knoten in einer nächsten Ebene der Hierarchie des verteilten Verarbeitungssystems durch den Kompilierungsknoten in Abhängigkeit davon, ob kompilierte Software für den ausgewählten Knoten oder für die Nachkommen des ausgewählten Knotens bestimmt ist, und das Senden (**812**) von nur der kompilierten Software, die von dem ausgewählten Knoten oder den Nachkommen des ausgewählten Knotens ausgeführt werden soll, an den ausgewählten Knoten. Das Auswählen (**810**) von einem oder mehreren Knoten in einer nächsten Ebene der Hierarchie des verteilten Verarbeitungssystems durch den Kompilierungsknoten in Abhängigkeit davon, ob kompilierte Software für den ausgewählten Knoten oder für die Nachkommen des ausgewählten Knotens bestimmt ist, kann durchgeführt werden, indem eine Darstellung der Topologie des hierarchischen verteilten Verarbeitungssystems durchlaufen wird, um den Standort von Knoten, auf denen kompilierte Software ausgeführt werden soll, zu kennzeichnen, ein Zweig des hierarchischen Datenverarbeitungssystems ermittelt wird, auf dem sich diejenigen Knoten, die die kompilierte Software ausführen, befinden, und indem ermittelt wird, welcher Kindknoten des Kompilierungsknotens sich ebenfalls auf diesem Zweig des hierarchischen Datenverarbeitungssystems befindet.

[0091] Das Senden (**812**) von nur der kompilierten Software, die von dem ausgewählten Knoten oder den Nachkommen des ausgewählten Knotens ausgeführt werden soll, an den ausgewählten Knoten kann durchgeführt werden, indem eine Nachricht erzeugt wird, welche die Teile der kompilierten Software enthält, die von dem ausgewählten Knoten oder den Nachkommen des ausgewählten Knotens ausgeführt werden sollen, und die Nachricht an den ausgewählten Knoten gesendet wird. Das Senden (**812**) von nur der kompilierten Software, die von dem ausgewählten Knoten oder den Nachkommen des ausgewählten Knotens ausgeführt werden soll, an den ausgewählten Knoten kann auch durchgeführt werden, indem der ausgewählte Knoten über den Standort der kompilierten Software zum Herunterladen auf den Knoten, welcher die kompilierte Software ausführt, benachrichtigt wird oder indem nur die kompilierte Software, die von dem ausgewählten Knoten oder den Nachkommen des ausgewählten Knotens ausgeführt werden soll, auf irgendeine andere Art und Weise, welche für den Fachmann zu erkennen ist, an den ausgewählten Knoten gesendet (**812**) wird.

[0092] Die kompilierte Software kann von dem ausgewählten Knoten ausgeführt oder auch nicht ausgeführt werden. Das heißt, die kompilierte Software kann von einem Knoten in einer Ebene unterhalb des ausgewählten Knotens ausgeführt werden. Zur näheren Erklärung zeigt Fig. 8 daher einen Ablaufplan, der ein weiteres

beispielhaftes Verfahren zum Kompilieren von Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung veranschaulicht. Das Verfahren von Fig. 8 ist dem Verfahren in Fig. 7 insofern ähnlich, als das Verfahren von Fig. 8 das Kennzeichnen (**802**) des einen oder der mehreren Kompilierungsknoten; das Bereitstellen (**804**) von zu kompilierender Software für einen oder mehrere Kompilierungsknoten, wobei mindestens ein Teil der zu kompilierenden Software von einem oder mehreren anderen Knoten ausgeführt werden soll; das Kompilieren (**806**) der Software durch den Kompilierungsknoten; das Verwalten (**808**) von kompilierter Software, die auf dem Kompilierungsknoten ausgeführt werden soll, durch den Kompilierungsknoten; das Auswählen (**810**) von einem oder mehreren Knoten in einer nächsten Ebene der Hierarchie des verteilten Verarbeitungssystems durch den Kompilierungsknoten in Abhängigkeit davon, ob kompilierte Software für den ausgewählten Knoten oder für die Nachkommen des ausgewählten Knotens bestimmt ist; und das Senden (**812**) von nur der kompilierten Software, die von dem ausgewählten Knoten oder von den Nachkommen des ausgewählten Knotens ausgeführt werden soll, an den ausgewählten Knoten, beinhaltet.

[0093] Das Verfahren von Fig. 8 beinhaltet auch zusätzliche Schritte, die von Knoten unterhalb des Kompilierungsknotens in dem hierarchischen verteilten Verarbeitungssystem durchgeführt werden. Das Verfahren von Fig. 8 beinhaltet das Empfangen (**814**) von kompilierter Software durch einen ausgewählten Knoten. Die kompilierte Software kann in einer Nachricht empfangen werden, die zum Herunterladen durch den Kompilierungsknoten gekennzeichnet ist, oder sie kann auf andere Arten, die für den Fachmann zu erkennen sind, empfangen werden.

[0094] Das Verfahren von Fig. 8 beinhaltet auch das Feststellen (**816**), ob die kompilierte Software für den ausgewählten Knoten oder für einen seiner Nachkommen bestimmt ist. Der Vorgang des Feststellens (**816**), ob die kompilierte Software für den ausgewählten Knoten oder für einen seiner Nachkommen bestimmt ist, kann durchgeführt werden, indem von dem Kompilierungsknoten eine Kennzeichnung der Knoten, die bestimmte Teile der kompilierten Software ausführen, empfangen wird und indem festgestellt wird, ob der gekennzeichnete Knoten der ausgewählte Knoten ist oder ob der gekennzeichnete Knoten einer seiner Nachkommen ist.

[0095] Wenn die kompilierte Software für den ausgewählten Knoten bestimmt ist, beinhaltet das Verfahren von Fig. 8 das Verwalten (**818**) der Software durch den ausgewählten Knoten zur Ausführung. Das Verwalten (**818**) der Software durch den ausgewählten Knoten zur Ausführung kann durchgeführt werden, indem kompilierte Software, die auf dem ausgewählten Knoten ausgeführt werden soll, zur Ausführung gespeichert wird.

[0096] Wenn die kompilierte Software für einen der Nachkommen bestimmt ist, beinhaltet das Verfahren von Fig. 8 das Auswählen (**820**) eines anderen Knotens in einer nächsten Ebene des hierarchischen verteilten Verarbeitungssystems in Abhängigkeit von einem Nachkommen für die kompilierte Software und das Senden (**822**) der kompilierten Software an den ausgewählten anderen Knoten. Das Auswählen (**820**) von einem anderen Knoten in einer nächsten Ebene des hierarchischen verteilten Verarbeitungssystems in Abhängigkeit von einem Nachkommen für die kompilierte Software kann durchgeführt werden, indem eine Darstellung der Topologie des hierarchischen verteilten Verarbeitungssystems durchlaufen wird, um den Standort von Knoten, auf denen kompilierte Software ausgeführt werden soll, zu kennzeichnen, ein Zweig des hierarchischen Datenverarbeitungssystems ermittelt wird, auf dem sich diejenigen Knoten, die die kompilierte Software ausführen, befinden, und indem ermittelt wird, welcher Kindknoten des ausgewählten Knotens sich ebenfalls auf diesem Zweig des hierarchischen Datenverarbeitungssystems befindet.

[0097] Das Senden (**822**) der kompilierten Software an den ausgewählten anderen Knoten kann durchgeführt werden, indem eine Nachricht erzeugt wird, welche die Teile der kompilierten Software enthält, die von dem ausgewählten anderen Knoten oder den Nachkommen des ausgewählten anderen Knotens ausgeführt werden sollen, und die Nachricht an den ausgewählten anderen Knoten gesendet wird. Das Senden (**822**) der kompilierten Software an den ausgewählten anderen Knoten kann auch durchgeführt werden, indem der ausgewählte andere Knoten über den Standort der kompilierten Software zum Herunterladen auf den Knoten, welcher die kompilierte Software ausführt, benachrichtigt wird oder indem die kompilierte Software auf irgendeine andere Art und Weise, welche für den Fachmann zu erkennen ist, an den ausgewählten Knoten gesendet (**822**) wird.

[0098] Zur näheren Erklärung zeigt Fig. 9 ein Schaubild eines beispielhaften Anwendungsfalls eines Systems zum Kompilieren von Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß Ausführungsformen der vorliegenden Erfindung. In dem Beispiel von Fig. 9 verfügt ein Kompilierungs-Laptopcomputer (**702**) über nichtkompilierte Software (**722**). Die nichtkompilierte Software (**722**) verfügt über Teile von Software (**724**) zur Ausführung durch den Computer (**704**), Teile von Software (**726** und **728**) zur Ausführung durch einen oder mehrere Grafikprozessoren (Graphics Processing Units (GPUs) (**704**), Teile von Software (**728**, **730**, **732**, **734**)

zur Ausführung durch Computerknoten in dem parallelen Computer (**712**), Teile von Software (**736**) zur Ausführung durch den Front-End-Knoten (**714**) der hybriden Datenverarbeitungsumgebung, Teile von Software (**738**, **740** und **742**) zur Ausführung durch Hostcomputer (**718**) in dem hybriden Datenverarbeitungssystem und Teile von Software (**744**, **746**, **748** und **750**) zur Ausführung durch Beschleuniger (**720**) des hybriden Datenverarbeitungssystems.

[0099] In dem Beispiel von Fig. 9 kompiliert der Kompilierungs-Laptopcomputer (**702**) den Teil der Software (**724**) für den Computer (**704**) und sendet diesen Teil der kompilierten Software an den Computer (**704**) zur Ausführung. In dem Beispiel von Fig. 9 kompiliert der Kompilierungs-Laptopcomputer (**702**) auch die Teile der Software (**726** und **728**) für bestimmte GPUs (**708**) und sendet diese Teile der kompilierten Software an den Computer (**704**), der den Teil der kompilierten Software wiederum an die bestimmte GPU sendet, die den Teil der kompilierten Software ausführt.

[0100] In dem Beispiel von Fig. 9 kompiliert der Kompilierungs-Laptopcomputer (**702**) den Teil der Software (**728**, **730**, **732** und **734**) für bestimmte Computerknoten (**712**) des parallelen Computers und sendet diese Teile der kompilierten Software an den E/A-Knoten (**710**) des parallelen Computers, der die Teile der kompilierten Software wiederum an den Wurzelknoten der Computerknoten (**712**) sendet. Der Wurzelknoten der Computerknoten stellt dann fest, welcher Kindknoten Nachkommen hat, die die Teile der kompilierten Software ausführen, und sendet nur die Teile an jedes Kind, die von diesem Kind oder seinen Nachkommen ausgeführt werden. Jedes Kind, das diese Teile empfängt, stellt fest, ob es den Teil ausführen wird oder ob einer seiner Nachkommen den Teil ausführen wird, und sendet nur diejenigen Teile, die für seine Nachkommen bestimmt sind, an ein Kind auf demselben Zweig wie der Nachkomme. Auf diese Weise werden die Teile der kompilierten Software Ebene um Ebene an den bestimmten Computerknoten gesendet, der diese kompilierte Software ausführt.

[0101] In dem Beispiel von Fig. 9 kompiliert der Kompilierungscomputer (**702**) den Teil der Software (**736**) für den Front-End-Knoten (**714**) der hybriden Datenverarbeitungsumgebung und sendet diesen Teil der kompilierten Software an den Front-End-Knoten (**714**) der hybriden Datenverarbeitungsumgebung zur Ausführung. In dem Beispiel von Fig. 9 kompiliert der Kompilierungscomputer (**702**) auch die Teile der Software (**726**, **738**, **740** und **742**) für bestimmte Hostcomputer (**718**) und sendet diese Teile der kompilierten Software an den Front-End-Knoten (**714**) der hybriden Datenverarbeitungsumgebung, der den Teil der kompilierten Software wiederum an die bestimmten Hostcomputer sendet, die den Teil der kompilierten Software ausführen. In dem Beispiel von Fig. 9 kompiliert der Kompilierungs-Laptop-Computer (**702**) auch die Teile der Software (**744**, **746**, **748** und **750**) für bestimmte Beschleuniger (**720**) von Hostcomputern (**718**) und sendet diese Teile der kompilierten Software an den Front-End-Knoten (**714**) der hybriden Datenverarbeitungsumgebung, welcher den Teil der kompilierten Software wiederum an die jeweiligen Hostcomputer für diese Beschleuniger sendet, die den Teil der kompilierten Software wiederum an den bestimmten Beschleuniger senden, der den Teil der kompilierten Software ausführt.

[0102] In den vorstehenden Beispielen wurde das Kompilieren von Software für ein hierarchisches verteiltes Verarbeitungssystem im Allgemeinen mit einem einzigen Kompilierungsknoten erörtert. Dies geschah zum Zweck der Erklärung und ist nicht als Einschränkung zu verstehen. Tatsächlich kann in vielen Ausführungsformen der vorliegenden Erfindung mehr als ein Knoten Software für ein hierarchisches verteiltes Verarbeitungssystem gemäß der vorliegenden Erfindung kompilieren.

[0103] Der Fachmann wird als vorteilhaft erkennen, dass Erscheinungsformen der vorliegenden Erfindung als ein System, ein Verfahren oder ein Computerprogrammprodukt realisiert werden können. Folglich können Erscheinungsformen der vorliegenden Erfindung die Form einer ganz in Hardware realisierten Ausführung, einer ganz in Software realisierten Ausführung (darunter Firmware, residente Software, Mikrocode usw.) oder einer Ausführung annehmen, die Software- und Hardware-Erscheinungsformen kombiniert, die hier alle allgemein als eine ”Schaltung”, ein ”Modul” oder ein ”System” bezeichnet werden können. Überdies können Erscheinungsformen der vorliegenden Erfindung die Form eines Computerprogrammprodukts annehmen, das sich auf einem oder mehreren computerlesbaren Datenträgern) befindet, auf dem beziehungsweise denen sich computerlesbarer Programmcode befindet.

[0104] Jede beliebige Kombination aus einem oder mehreren computerlesbaren Datenträgern kann verwendet werden. Der computerlesbare Datenträger kann ein computerlesbarer Signaldatenträger oder ein computerlesbares Speichermedium sein. Ein computerlesbares Speichermedium kann zum Beispiel, ohne auf diese beschränkt zu sein, ein(e) elektronische(s), magnetische(s), optische(s), elektromagnetische(s), Infrarot- oder Halbleitersystem, -vorrichtung, -einheit oder eine beliebige geeignete Kombination des Vorstehenden sein.

Zu konkreteren Beispielen (wobei die Liste keinen Anspruch auf Vollständigkeit erhebt) für das computerlesbare Speichermedium würden folgende gehören: eine elektrische Verbindung mit einer oder mehreren Leitungen, eine Diskette eines tragbaren Computers, eine Festplatte, ein Direktzugriffsspeicher (RAM), ein Nur-Lese-Speicher (ROM), ein löschbarer programmierbarer Nur-Lese-Speicher (EPROM oder Flash-Speicher), ein Lichtwellenleiter, ein tragbarer Compact-Disk-Nur-Lese-Speicher (CD-ROM), eine optische Speichereinheit, eine magnetische Speichereinheit oder jede beliebige geeignete Kombination des Vorstehenden. Im Rahmen dieses Schriftstücks kann ein computerlesbares Speichermedium jedes physisch greifbare Medium sein, das ein Programm zur Verwendung durch ein Befehlsausführungssystem, eine Befehlsausführungsvorrichtung oder -einheit oder zur Verwendung in Verbindung mit einem Befehlsausführungssystem, einer Befehlsausführungsvorrichtung oder -einheit enthalten oder speichern kann.

[0105] Ein computerlesbarer Signaldatenträger kann ein übertragenes Datensignal mit einem darin enthaltenen computerlesbaren Programmcode, beispielsweise in einem Basisband oder als Teil einer Trägerwelle, enthalten. Solch ein übertragenes Signal kann eine beliebige einer Vielzahl von Formen einschließlich elektromagnetischer, optischer Formen oder jede beliebige geeignete Kombination dieser Formen, ohne auf diese beschränkt zu sein, annehmen. Bei einem computerlesbaren Signaldatenträger kann es sich um jeden beliebigen computerlesbaren Datenträger handeln, der kein computerlesbares Speichermedium ist und der ein Programm zur Verwendung durch oder zur Verwendung in Verbindung mit einem Befehlsausführungssystem, einer Befehlsausführungsvorrichtung oder -einheit übertragen, weiterleiten oder transportieren kann.

[0106] Auf einem computerlesbaren Datenträger enthaltener Programmcode kann mittels eines geeigneten Mediums, darunter einschließlich ein drahtloses Medium, ein drahtgebundenes Mediums, ein Lichtwellenleiterkabel, mittels Hochfrequenz (HF) usw., ohne auf diese beschränkt zu sein, oder mittels jeder beliebigen geeigneten Kombination des Vorstehenden übertragen werden.

[0107] Computer-Programmcode zur Durchführung von Operationen für Erscheinungsformen der vorliegenden Erfindung kann in einer beliebigen Kombination aus einer oder mehreren Programmiersprachen, darunter eine objektorientierte Programmiersprache wie beispielsweise Java, Smalltalk, C++ oder dergleichen, sowie in herkömmlichen prozeduralen Programmiersprachen wie beispielsweise der Programmiersprache ”C” oder in ähnlichen Programmiersprachen, geschrieben sein. Die Ausführung des Programmcodes kann vollständig auf dem Computer des Benutzers, teilweise auf dem Computer des Benutzers, als eigenständiges Software-Paket, teilweise auf dem Computer des Benutzers und teilweise auf einem fernen Computer oder vollständig auf dem fernen Computer oder Server erfolgen. Im letzteren Szenario kann der ferne Computer mit dem Computer des Benutzers über jede beliebige Art eines Netzwerks einschließlich eines lokalen Netzwerks (LAN) oder eines Weitverkehrsnetzes (WAN) verbunden werden oder die Verbindung kann zu einem externen Computer (zum Beispiel über das Internet mittels eines Internet-Diensteanbieters) hergestellt werden.

[0108] Erscheinungsformen der vorliegenden Erfindung wurden vorstehend mit Bezug auf Darstellungen in Ablaufplänen und/oder Blockschaltbilder von Verfahren, Vorrichtungen (Systemen) und Computerprogrammprodukten gemäß Ausführungsformen der Erfindung beschrieben. Es versteht sich, dass jeder Block der Darstellungen in den Ablaufplänen und/oder der Blockschaltbilder sowie Kombinationen aus Blöcken in den Darstellungen der Ablaufpläne und/oder den Blockschaltbildern mittels Computerprogrammbefehlen realisiert werden können. Diese Computerprogrammbefehle können einem Prozessor eines Universalcomputers, eines Computers für spezielle Anwendungen oder einer anderen programmierbaren Datenverarbeitungsvorrichtung bereitgestellt werden, um eine Maschine zu erzeugen, so dass die Befehle, die über den Prozessor des Computers oder einer anderen programmierbaren Datenverarbeitungsvorrichtung ausgeführt werden, ein Mittel zur Ausführung der Funktionen/Vorgänge erzeugen, die in dem Ablaufplan und/oder dem Block oder den Blöcken der Blockschaltbilder angegeben sind.

[0109] Diese Computerprogrammbefehle können auch auf einem computerlesbaren Datenträger gespeichert werden, der einen Computer, eine andere programmierbare Datenverarbeitungsvorrichtung oder andere Einheiten anweisen kann, auf eine bestimmte Art und Weise zu funktionieren, so dass die auf dem computerlesbaren Datenträger gespeicherten Befehle einen Herstellungsgegenstand erzeugen, der Befehle enthält, die die Funktion/den Vorgang ausführen, welche beziehungsweise welcher in dem Ablaufplan und/oder dem Block oder den Blöcken des Blockschaltbilds angegeben ist.

[0110] Die Computerprogrammbefehle können auch auf einen Computer, eine andere programmierbare Datenverarbeitungsvorrichtung oder auf andere Einheiten geladen werden, um die Durchführung einer Reihe von Betriebsschritten auf dem Computer, einer anderen programmierbaren Vorrichtung oder auf anderen Einheiten zu bewirken, um einen von einem Computer ausgeführten Prozess zu erzeugen, so dass die Befehle, die auf

dem Computer oder einer anderen programmierbaren Vorrichtung ausgeführt werden, Prozesse zur Ausführung der Funktionen/Vorgänge ermöglichen, die in dem Ablaufplan und/oder dem Block oder den Blöcken des Blockschaltbilds angegeben sind.

[0111] Der Ablaufplan und die Blockschaltbilder in den Figuren zeigen die Architektur, die Funktionalität und die Betriebsweise von möglichen Ausführungsarten von Systemen, Verfahren und Computerprogrammprodukten gemäß verschiedenen Ausführungsformen der vorliegenden Erfindung. In dieser Hinsicht kann jeder Block in dem Ablaufplan oder den Blockschaltbildern ein Modul, ein Segment oder einen Teil von Code darstellen, das beziehungsweise der einen oder mehrere ausführbare Befehle zur Ausführung der angegebenen logischen Funktion(en) umfasst. Es sei auch angemerkt, dass die in dem Block angegebenen Funktionen in manchen alternativen Ausführungsarten nicht in der in den Figuren angegebenen Reihenfolge auftreten können. In Abhängigkeit von der mit ihnen verbundenen Funktionalität können beispielsweise zwei Blöcke, die als aufeinanderfolgende Blöcke dargestellt sind, tatsächlich weitgehend gleichzeitig ausgeführt werden oder die Blöcke können manchmal in der umgekehrten Reihenfolge ausgeführt werden. Man wird auch feststellen, dass jeder Block der Blockschaltbilder und/oder der Darstellung in dem Ablaufplan sowie Kombinationen aus Blöcken in den Blockschaltbildern und/oder der Darstellung in dem Ablaufplan von Systemen, die auf Hardware für spezielle Anwendungen beruhen und die angegebenen Funktionen oder Vorgänge durchführen, oder von Kombinationen aus Hardware für spezielle Anwendungen und Computerbefehlen ausgeführt werden können.

[0112] Die Beschreibungen in dieser Darlegung dienen lediglich der Veranschaulichung und sind nicht als Einschränkung zu verstehen. Der Umfang der vorliegenden Erfindung ist nur durch die Sprache der folgenden Ansprüche beschränkt.

ZITATE ENTHALTEN IN DER BESCHREIBUNG

Diese Liste der vom Anmelder aufgeführten Dokumente wurde automatisiert erzeugt und ist ausschließlich zur besseren Information des Lesers aufgenommen. Die Liste ist nicht Bestandteil der deutschen Patent- bzw. Gebrauchsmusteranmeldung. Das DPMA übernimmt keinerlei Haftung für etwaige Fehler oder Auslassungen.

Zitierte Nicht-Patentliteratur

- Standard IEEE 802.3 [**0040**]
- Standard IEEE 1149.1 mit dem Titel ”Standard Test Access Port and Boundary-Scan Architecture for test access ports used for testing printed circuit boards using boundary scan” [**0041**]

Patentansprüche

1. Verfahren zum Kompilieren von Software für ein hierarchisches verteiltes Verarbeitungssystem, wobei das Verfahren Folgendes umfasst:
Bereitstellen von zu kompilierender Software für einen oder mehrere Kompilierungsknoten, wobei mindestens ein Teil der zu kompilierenden Software von einem oder mehreren anderen Knoten ausgeführt werden soll;
Kompilieren der Software durch den Kompilierungsknoten;
Verwalten von kompilierter Software, die auf dem Kompilierungsknoten ausgeführt werden soll, durch den Kompilierungsknoten; und
Auswählen von einem oder mehreren Knoten in einer nächsten Ebene der Hierarchie des verteilten Verarbeitungssystems durch den Kompilierungsknoten in Abhängigkeit davon, ob kompilierte Software für den ausgewählten Knoten oder für die Nachkommen des ausgewählten Knotens bestimmt ist; und
Senden von nur der kompilierten Software, die von dem ausgewählten Knoten oder von den Nachkommen des ausgewählten Knotens ausgeführt werden soll, an den ausgewählten Knoten.

2. Verfahren nach Anspruch 1, das des Weiteren Folgendes umfasst:
Empfangen von kompilierter Software durch einen ausgewählten Knoten;
Feststellen, ob die kompilierte Software für den ausgewählten Knoten oder für einen seiner Nachkommen bestimmt ist;
wenn die kompilierte Software für den ausgewählten Knoten bestimmt ist, Verwalten der Software durch den ausgewählten Knoten zur Ausführung; und
wenn die kompilierte Software für einen der Nachkommen bestimmt ist, Auswählen eines anderen Knotens in einer nächsten Ebene des hierarchischen verteilten Verarbeitungssystems in Abhängigkeit von einem Nachkommen für die kompilierte Software und Senden der kompilierten Software an den ausgewählten anderen Knoten.

3. Verfahren nach Anspruch 1 oder 2, das des Weiteren das Kennzeichnen des einen oder der mehreren Kompilierungsknoten umfasst.

4. Verfahren nach Anspruch 3, wobei das Kennzeichnen des einen oder der mehreren Kompilierungsknoten des Weiteren das Auswählen von einem oder mehreren Knoten umfasst, die für den Kompiliervorgang rechnerisch optimiert werden.

5. Verfahren nach Anspruch 3 oder 4, wobei das Kennzeichnen des einen oder der mehreren Kompilierungsknoten des Weiteren das Auswählen von einem oder mehreren Knoten umfasst, die aufgrund von ihrem Standort in der Topologie des hierarchischen verteilten Verarbeitungssystems für den Kompiliervorgang optimiert werden.

6. Verfahren nach einem der vorhergehenden Ansprüche, wobei das hierarchische verteilte Verarbeitungssystem des Weiteren einen parallelen Computer umfasst, der Folgendes enthält:
eine Vielzahl von Computerknoten;
ein erstes Datenübertragungsnetzwerk, das die Computerknoten für Datenübertragungen verbindet und für Punkt-zu-Punkt-Datenübertragungen optimiert ist; und
ein zweites Datenübertragungsnetzwerk, das Datenübertragungs-Verbindungsleitungen enthält, welche die Computerknoten verbinden, um die Computerknoten als einen Baum aufzubauen, wobei jeder Computerknoten über ein gesondertes Rechenwerk (arithmetic logic unit (”ALU”) verfügt, das für parallele Operationen bestimmt ist.

7. Verfahren nach einem der vorhergehenden Ansprüche, wobei das hierarchische verteilte Verarbeitungssystem des Weiteren eine hybride Datenverarbeitungsumgebung umfasst, wobei die hybride Datenverarbeitungsumgebung eine Vielzahl von Computerknoten umfasst, wobei jeder Computerknoten Folgendes umfasst:
einen Hostcomputer mit einer Hostcomputer-Architektur; und
einen Beschleuniger mit einer Beschleuniger-Architektur, wobei die Beschleuniger-Architektur in Bezug auf die Architektur des Hostcomputers hinsichtlich der Ausführungsgeschwindigkeit einer bestimmten Klasse von Berechnungsfunktionen optimiert ist, wobei der Hostcomputer und der Beschleuniger für Datenübertragungen durch ein Nachrichtenübermittlungsmodul auf Systemebene aufeinander abgestimmt sind.

8. Vorrichtung zum Kompilieren von Software für ein hierarchisches verteiltes Verarbeitungssystem, wobei die Vorrichtung einen Computerprozessor und einen Computerspeicher umfasst, der mit dem Computerpro-

zessor betriebsfähig verbunden ist, wobei sich in dem Computerspeicher Computerprogrammbefehle befinden, um
zu kompilierende Software für einen oder mehrere Kompilierungsknoten bereitzustellen, wobei mindestens ein Teil der zu kompilierenden Software von einem oder mehreren anderen Knoten ausgeführt werden soll;
die Software durch den Kompilierungsknoten zu kompilieren;
kompilierte Software, die auf dem Kompilierungsknoten ausgeführt werden soll, durch den Kompilierungsknoten zu verwalten; und
einen oder mehrere Knoten in einer nächsten Ebene der Hierarchie des verteilten Verarbeitungssystems durch den Kompilierungsknoten in Abhängigkeit davon, ob kompilierte Software für den ausgewählten Knoten oder für die Nachkommen des ausgewählten Knotens bestimmt ist, auszuwählen; und
nur die kompilierte Software, die von dem ausgewählten Knoten oder von den Nachkommen des ausgewählten Knotens ausgeführt werden soll, an den ausgewählten Knoten zu senden.

9. Vorrichtung nach Anspruch 8, wobei sich in dem Computerspeicher auch Computerprogrammbefehle befinden, um
kompilierte Software durch einen ausgewählten Knoten zu empfangen;
festzustellen, ob die kompilierte Software für den ausgewählten Knoten oder für einen seiner Nachkommen bestimmt ist;
die Software durch den ausgewählten Knoten zur Ausführung zu verwalten, wenn die kompilierte Software für den ausgewählten Knoten bestimmt ist; und
einen anderen Knoten in einer nächsten Ebene des hierarchischen verteilten Verarbeitungssystems in Abhängigkeit von einem Nachkommen für die kompilierte Software auszuwählen, wenn die kompilierte Software für einen der Nachkommen bestimmt ist, und die kompilierte Software an den ausgewählten anderen Knoten zu senden.

10. Vorrichtung nach Anspruch 8 oder 9, wobei sich in dem Computerspeicher auch Computerprogrammbefehle befinden, um den einen oder die mehreren Kompilierungsknoten zu kennzeichnen.

11. Vorrichtung nach Anspruch 10, wobei Computerprogrammbefehle zum Kennzeichnen des einen oder der mehreren Kompilierungsknoten des Weiteren Computerprogrammbefehle umfassen, um einen oder mehrere Knoten auszuwählen, die für den Kompiliervorgang rechnerisch optimiert werden.

12. Vorrichtung nach Anspruch 10 oder 11, wobei Computerprogrammbefehle zum Kennzeichnen des einen oder der mehreren Kompilierungsknoten des Weiteren Computerprogrammbefehle umfassen, um einen oder mehrere Knoten auszuwählen, die aufgrund von ihrem Standort in der Topologie des hierarchischen verteilten Verarbeitungssystems für den Kompiliervorgang optimiert werden.

13. Vorrichtung nach einem der Ansprüche 8 bis 12, wobei das hierarchische verteilte Verarbeitungssystem des Weiteren einen parallelen Computer umfasst, der Folgendes enthält:
eine Vielzahl von Computerknoten;
ein erstes Datenübertragungsnetzwerk, das die Computerknoten für Datenübertragungen verbindet und für Punkt-zu-Punkt-Datenübertragungen optimiert ist; und
ein zweites Datenübertragungsnetzwerk, das Datenübertragungs-Verbindungsleitungen enthält, welche die Computerknoten verbinden, um die Computerknoten als einen Baum aufzubauen, wobei jeder Computerknoten über ein gesondertes Rechenwerk (ALU) verfügt, das für parallele Operationen bestimmt ist.

14. Vorrichtung nach einem der Ansprüche 8 bis 13, wobei das hierarchische verteilte Verarbeitungssystem des Weiteren eine hybride Datenverarbeitungsumgebung umfasst, wobei die hybride Datenverarbeitungsumgebung eine Vielzahl von Computerknoten umfasst, wobei jeder Computerknoten Folgendes umfasst:
einen Hostcomputer mit einer Hostcomputer-Architektur; und
einen Beschleuniger mit einer Beschleuniger-Architektur, wobei die Beschleuniger-Architektur in Bezug auf die Architektur des Hostcomputers hinsichtlich der Ausführungsgeschwindigkeit einer bestimmten Klasse von Berechnungsfunktionen optimiert ist, wobei der Hostcomputer und der Beschleuniger für Datenübertragungen durch ein Nachrichtenübermittlungsmodul auf Systemebene aufeinander abgestimmt sind.

15. Computerprogrammprodukt zum Kompilieren von Software für ein hierarchisches verteiltes Verarbeitungssystem, wobei das Computerprogrammprodukt in einem Computerlesbaren Speichermedium angeordnet ist, wobei das Computerprogrammprodukt Computerprogrammbefehle umfasst, um
zu kompilierende Software für einen oder mehrere Kompilierungsknoten bereitzustellen, wobei mindestens ein Teil der zu kompilierenden Software von einem oder mehreren anderen Knoten ausgeführt werden soll;

die Software durch den Kompilierungsknoten zu kompilieren;
kompilierte Software, die auf dem Kompilierungsknoten ausgeführt werden soll, durch den Kompilierungsknoten zu verwalten; und
einen oder mehrere Knoten in einer nächsten Ebene der Hierarchie des verteilten Verarbeitungssystems durch den Kompilierungsknoten in Abhängigkeit davon, ob kompilierte Software für den ausgewählten Knoten oder für die Nachkommen des ausgewählten Knotens bestimmt ist, auszuwählen; und
nur die kompilierte Software, die von dem ausgewählten Knoten oder von den Nachkommen des ausgewählten Knotens ausgeführt werden soll, an den ausgewählten Knoten zu senden.

16. Computerprogrammprodukt nach Anspruch 15, das des Weiteren Computerprogrammbefehle umfasst, um
kompilierte Software durch einen ausgewählten Knoten zu empfangen;
festzustellen, ob die kompilierte Software für den ausgewählten Knoten oder für einen seiner Nachkommen bestimmt ist;
die Software durch den ausgewählten Knoten zur Ausführung zu verwalten, wenn die kompilierte Software für den ausgewählten Knoten bestimmt ist; und
einen anderen Knoten in einer nächsten Ebene des hierarchischen verteilten Verarbeitungssystems in Abhängigkeit von einem Nachkommen für die kompilierte Software auszuwählen, wenn die kompilierte Software für einen der Nachkommen bestimmt ist, und die kompilierte Software an den ausgewählten anderen Knoten zu senden.

17. Computerprogrammprodukt nach Anspruch 15 oder Anspruch 16, das des Weiteren Computerprogrammbefehle umfasst, um den einen oder die mehreren Kompilierungsknoten zu kennzeichnen.

18. Computerprogrammprodukt nach Anspruch 17, wobei Computerprogrammbefehle zum Kennzeichnen des einen oder der mehreren Kompilierungsknoten des Weiteren Computerprogrammbefehle umfassen, um einen oder mehrere Knoten auszuwählen, die für den Kompiliervorgang rechnerisch optimiert werden.

19. Computerprogrammprodukt nach Anspruch 17 oder 18, wobei Computerprogrammbefehle zum Kennzeichnen des einen oder der mehreren Kompilierungsknoten des Weiteren Computerprogrammbefehle umfassen, um einen oder mehrere Knoten auszuwählen, die aufgrund von ihrem Standort in der Topologie des hierarchischen verteilten Verarbeitungssystems für den Kompiliervorgang optimiert werden.

20. Computerprogrammprodukt nach einem der Ansprüche 15 bis 19, wobei das hierarchische verteilte Verarbeitungssystem des Weiteren einen parallelen Computer umfasst, der Folgendes enthält:
eine Vielzahl von Computerknoten;
ein erstes Datenübertragungsnetzwerk, das die Computerknoten für Datenübertragungen verbindet und für Punkt-zu-Punkt-Datenübertragungen optimiert ist; und
ein zweites Datenübertragungsnetzwerk, das Datenübertragungs-Verbindungsleitungen enthält, welche die Computerknoten verbinden, um die Computerknoten als einen Baum aufzubauen, wobei jeder Computerknoten über ein gesondertes Rechenwerk (ALU) verfügt, das für parallele Operationen bestimmt ist.

Es folgen 9 Blatt Zeichnungen

Anhängende Zeichnungen

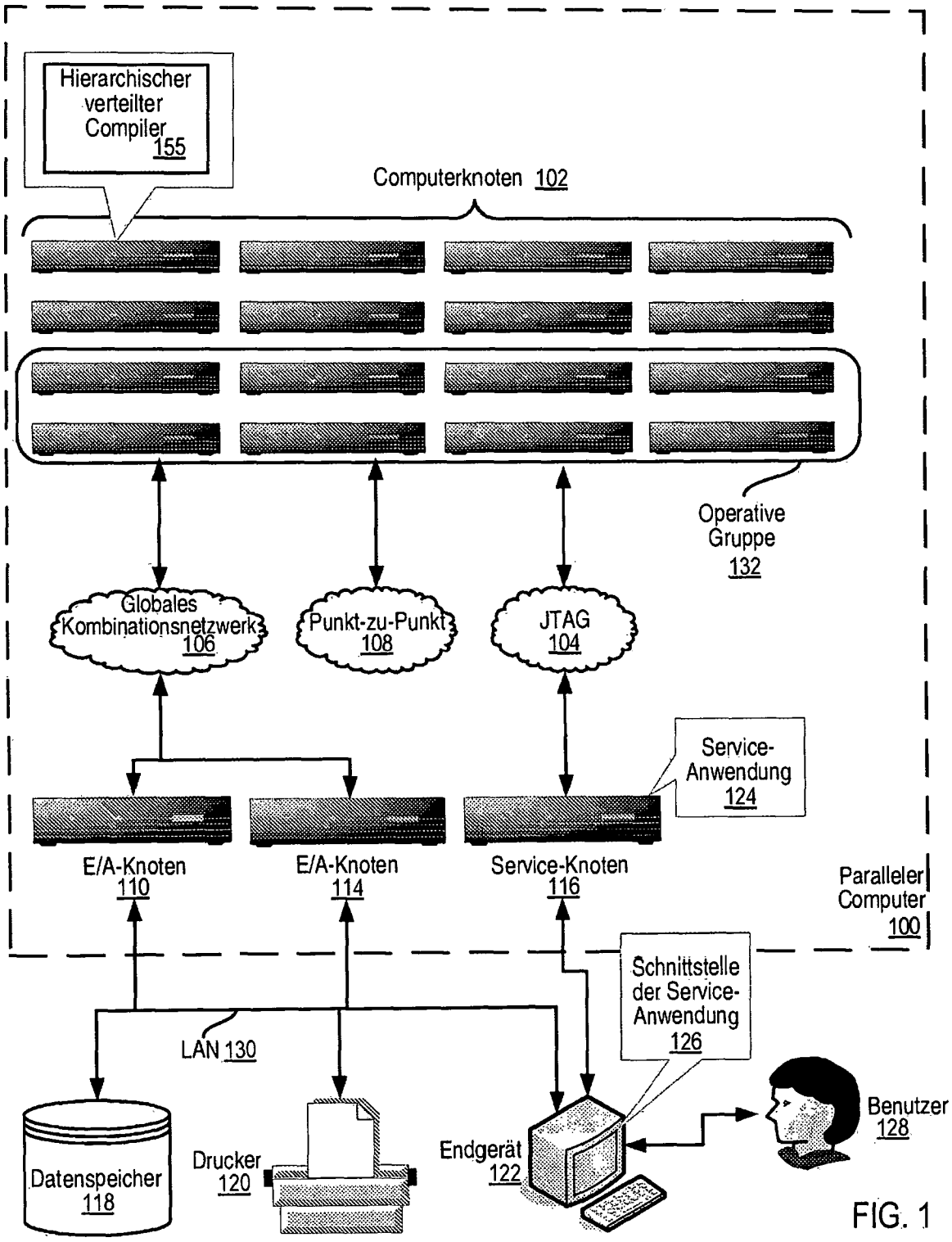


FIG. 1

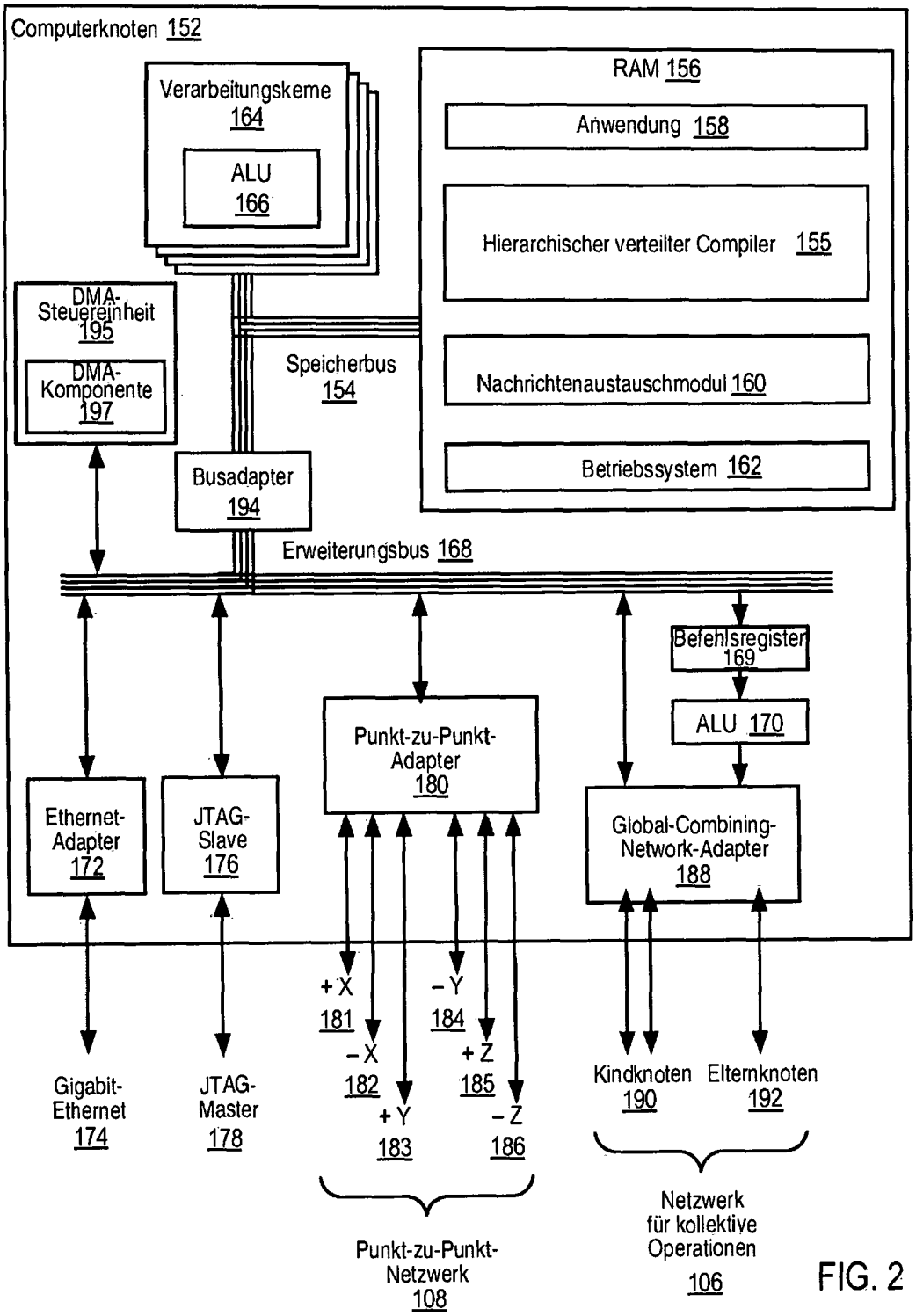

Computerknoten 152
Verarbeitungskerne 164
ALU 166
RAM 156
Anwendung 158
Hierarchischer verteilter Compiler 155
Nachrichtenaustauschmodul 160
Betriebssystem 162
DMA-Steuereinheit 195
DMA-Komponente 197
Speicherbus 154
Busadapter 194
Erweiterungsbus 168
Befehlsregister 169
ALU 170
Punkt-zu-Punkt-Adapter 180
Global-Combining-Network-Adapter 188
Ethernet-Adapter 172
JTAG-Slave 176
+X 181
-X 182
+Y 183
-Y 184
+Z 185
-Z 186
Gigabit-Ethernet 174
JTAG-Master 178
Kindknoten 190
Elternknoten 192
Punkt-zu-Punkt-Netzwerk 108
Netzwerk für kollektive Operationen 106

FIG. 2

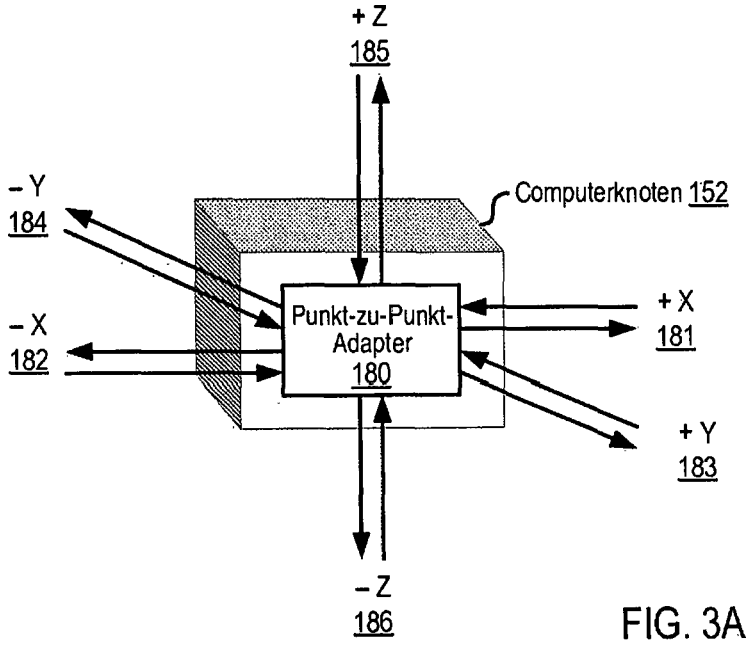

+Z
185
-Y
184
Computerknoten 152
-X
182
Punkt-zu-Punkt-
Adapter
180
+X
181
+Y
183
-Z
186

FIG. 3A

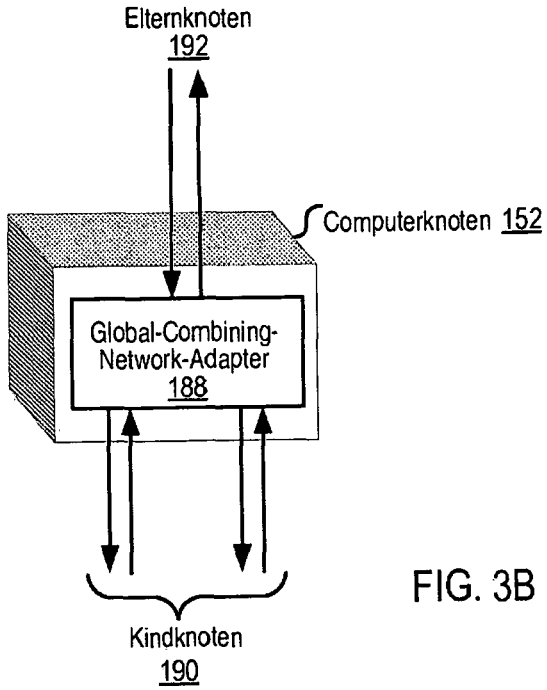

Elternknoten
192
Computerknoten 152
Global-Combining-
Network-Adapter
188
Kindknoten
190

FIG. 3B

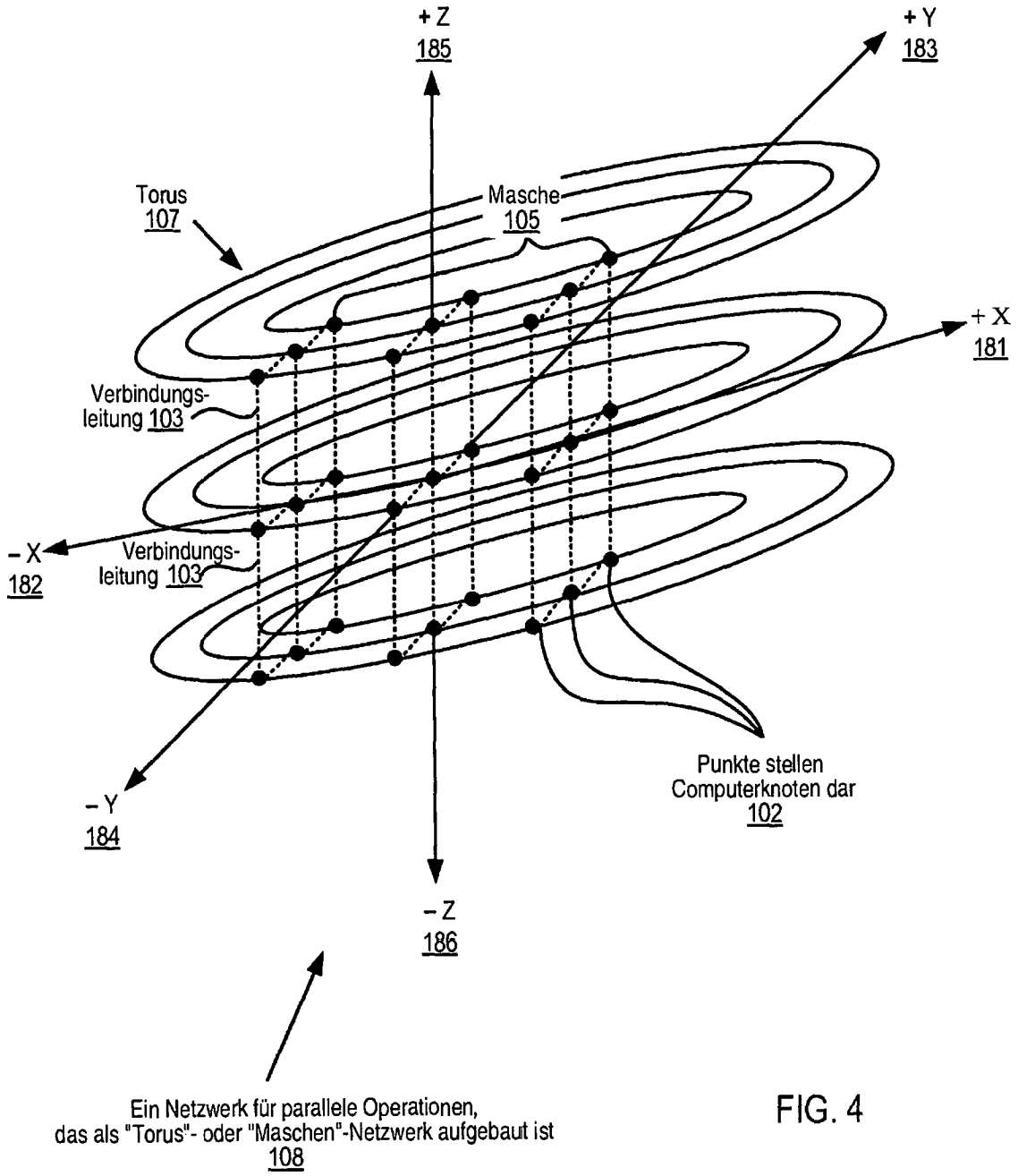

+ Z
185
+ Y
183
Torus
107
Masche
105
+ X
181
Verbindungs-
leitung 103
- X
182
Verbindungs-
leitung 103
- Y
184
Punkte stellen
Computerknoten dar
102
- Z
186
Ein Netzwerk für parallele Operationen,
das als "Torus"- oder "Maschen"-Netzwerk aufgebaut ist
108

FIG. 4

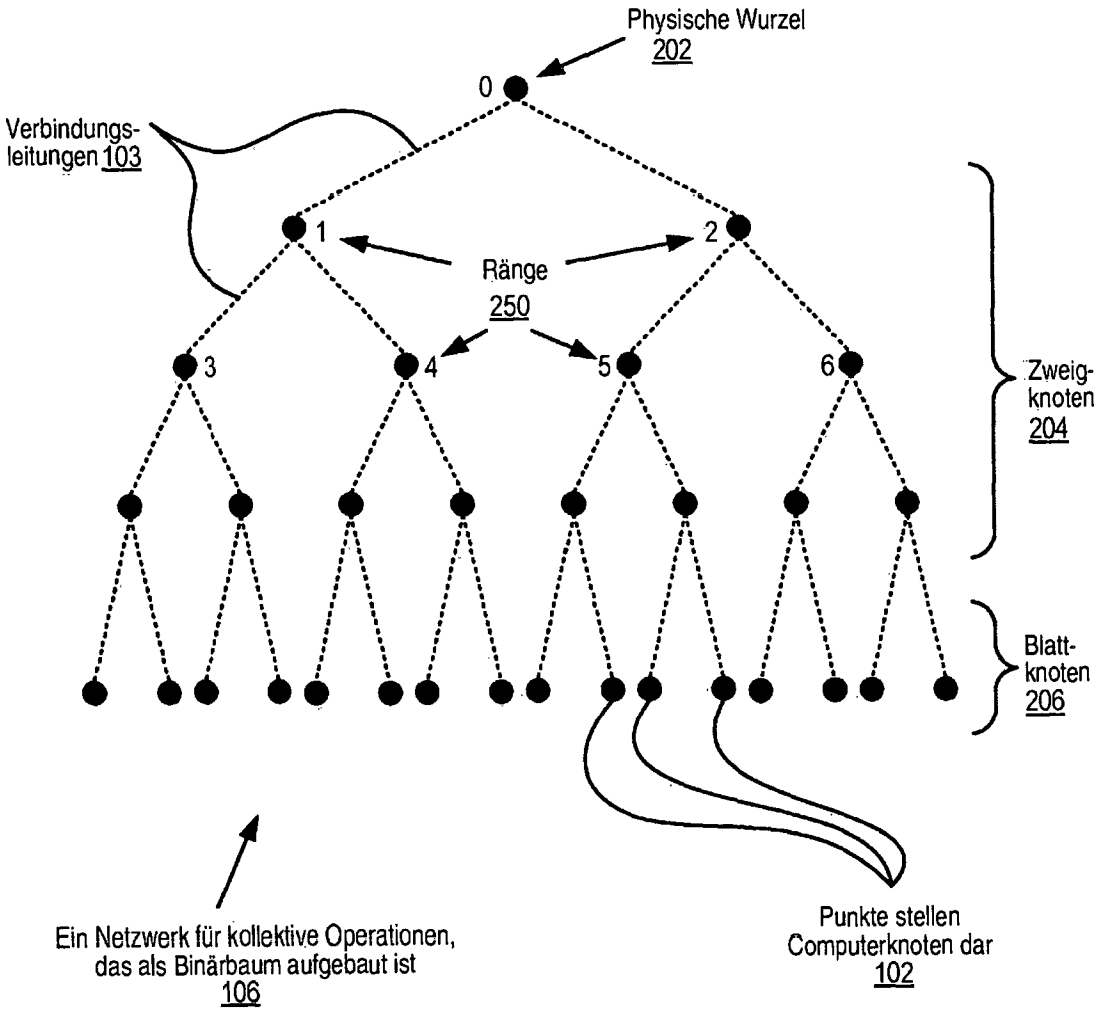

Physische Wurzel
202
0
Verbindungs-
leitungen 103
1
2
Ränge
250
3
4
5
6
Zweig-
knoten
204
Blatt-
knoten
206
Ein Netzwerk für kollektive Operationen,
das als Binärbaum aufgebaut ist
106
Punkte stellen
Computerknoten dar
102

FIG. 5

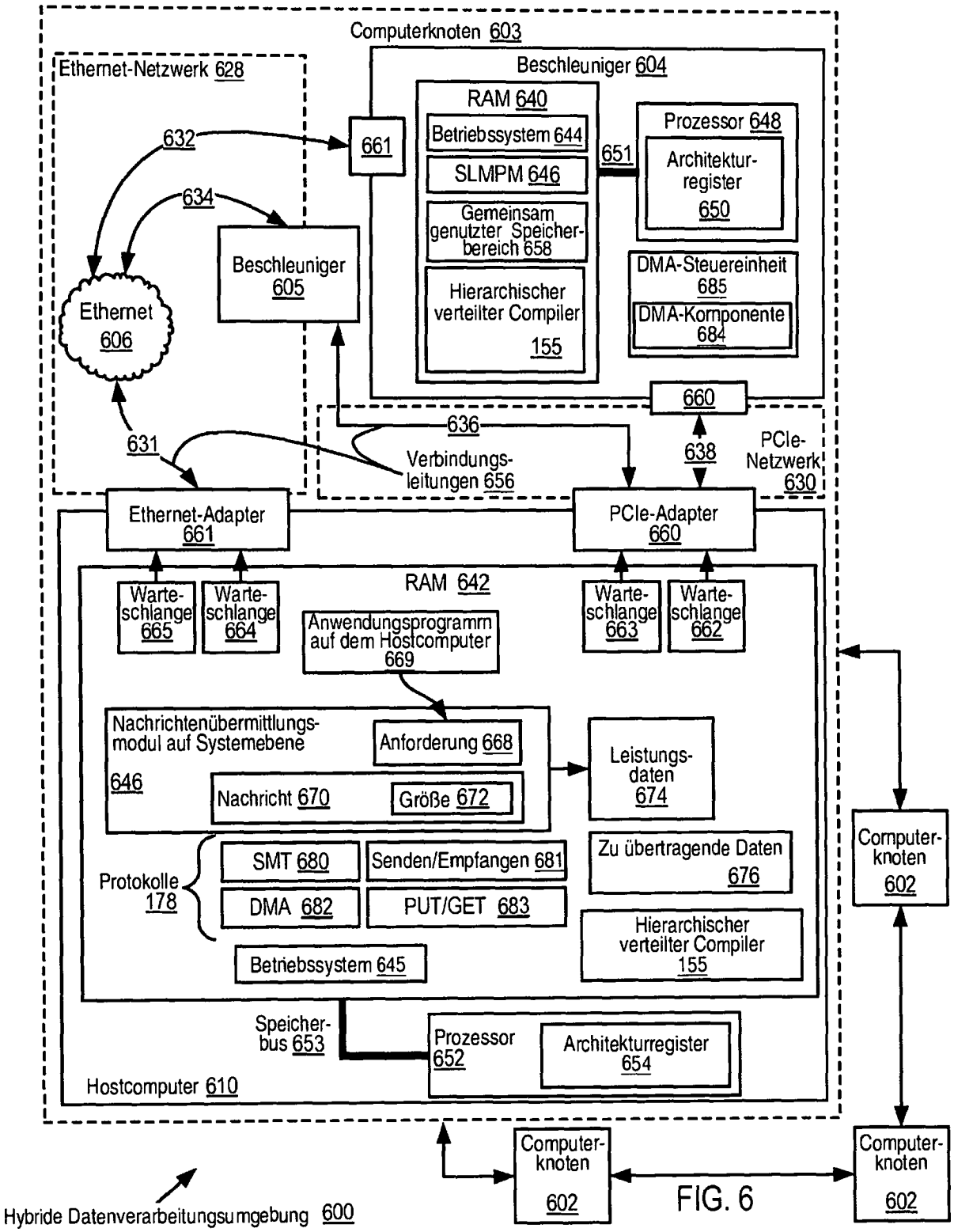

Computerknoten 603
Ethernet-Netzwerk 628
Beschleuniger 604
RAM 640
Betriebssystem 644
SLMPM 646
Gemeinsam genutzter Speicherbereich 658
Hierarchischer verteilter Compiler 155
Prozessor 648
Architekturregister 650
651
DMA-Steuereinheit 685
DMA-Komponente 684
661
632
634
Beschleuniger 605
Ethernet 606
660
636
631
Verbindungsleitungen 656
638
PCIe-Netzwerk 630
Ethernet-Adapter 661
PCIe-Adapter 660
RAM 642
Warteschlange 665
Warteschlange 664
Warteschlange 663
Warteschlange 662
Anwendungsprogramm auf dem Hostcomputer 669
Nachrichtenübermittlungsmodul auf Systemebene 646
Anforderung 668
Nachricht 670
Größe 672
Leistungsdaten 674
Protokolle 178
SMT 680
Senden/Empfangen 681
DMA 682
PUT/GET 683
Zu übertragende Daten 676
Hierarchischer verteilter Compiler 155
Betriebssystem 645
Speicherbus 653
Prozessor 652
Architekturregister 654
Hostcomputer 610
Computerknoten 602
Computerknoten 602
Computerknoten 602
Hybride Datenverarbeitungsumgebung 600

FIG. 6

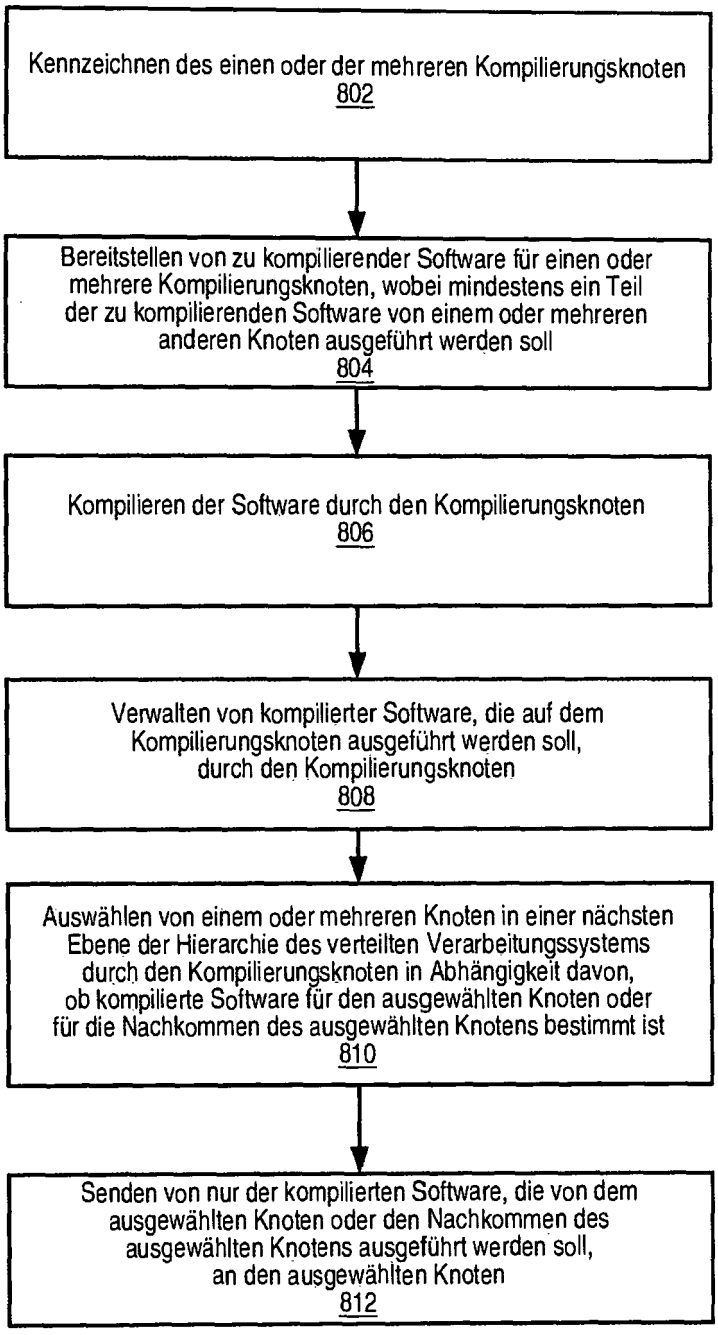

Kennzeichnen des einen oder der mehreren Kompilierungsknoten
802
Bereitstellen von zu kompilierender Software für einen oder mehrere Kompilierungsknoten, wobei mindestens ein Teil der zu kompilierenden Software von einem oder mehreren anderen Knoten ausgeführt werden soll
804
Kompilieren der Software durch den Kompilierungsknoten
806
Verwalten von kompilierter Software, die auf dem Kompilierungsknoten ausgeführt werden soll, durch den Kompilierungsknoten
808
Auswählen von einem oder mehreren Knoten in einer nächsten Ebene der Hierarchie des verteilten Verarbeitungssystems durch den Kompilierungsknoten in Abhängigkeit davon, ob kompilierte Software für den ausgewählten Knoten oder für die Nachkommen des ausgewählten Knotens bestimmt ist
810
Senden von nur der kompilierten Software, die von dem ausgewählten Knoten oder den Nachkommen des ausgewählten Knotens ausgeführt werden soll, an den ausgewählten Knoten
812

FIG. 7

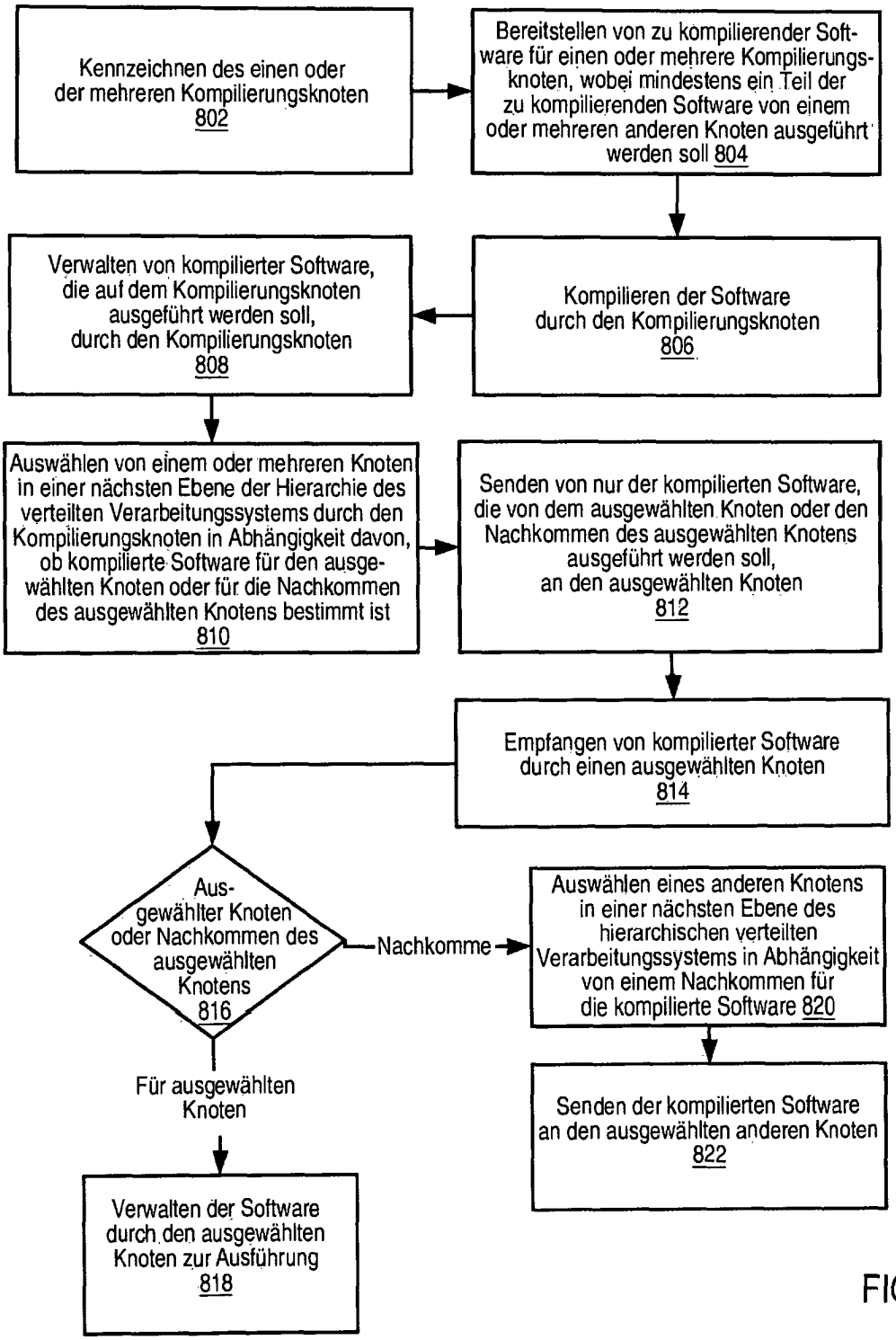

Kennzeichnen des einen oder der mehreren Kompilierungsknoten 802
Bereitstellen von zu kompilierender Software für einen oder mehrere Kompilierungsknoten, wobei mindestens ein Teil der zu kompilierenden Software von einem oder mehreren anderen Knoten ausgeführt werden soll 804
Kompilieren der Software durch den Kompilierungsknoten 806
Verwalten von kompilierter Software, die auf dem Kompilierungsknoten ausgeführt werden soll, durch den Kompilierungsknoten 808
Auswählen von einem oder mehreren Knoten in einer nächsten Ebene der Hierarchie des verteilten Verarbeitungssystems durch den Kompilierungsknoten in Abhängigkeit davon, ob kompilierte Software für den ausgewählten Knoten oder für die Nachkommen des ausgewählten Knotens bestimmt ist 810
Senden von nur der kompilierten Software, die von dem ausgewählten Knoten oder den Nachkommen des ausgewählten Knotens ausgeführt werden soll, an den ausgewählten Knoten 812
Empfangen von kompilierter Software durch einen ausgewählten Knoten 814
Ausgewählter Knoten oder Nachkommen des ausgewählten Knotens 816
Nachkomme
Auswählen eines anderen Knotens in einer nächsten Ebene des hierarchischen verteilten Verarbeitungssystems in Abhängigkeit von einem Nachkommen für die kompilierte Software 820
Senden der kompilierten Software an den ausgewählten anderen Knoten 822
Für ausgewählten Knoten
Verwalten der Software durch den ausgewählten Knoten zur Ausführung 818

FIG. 8

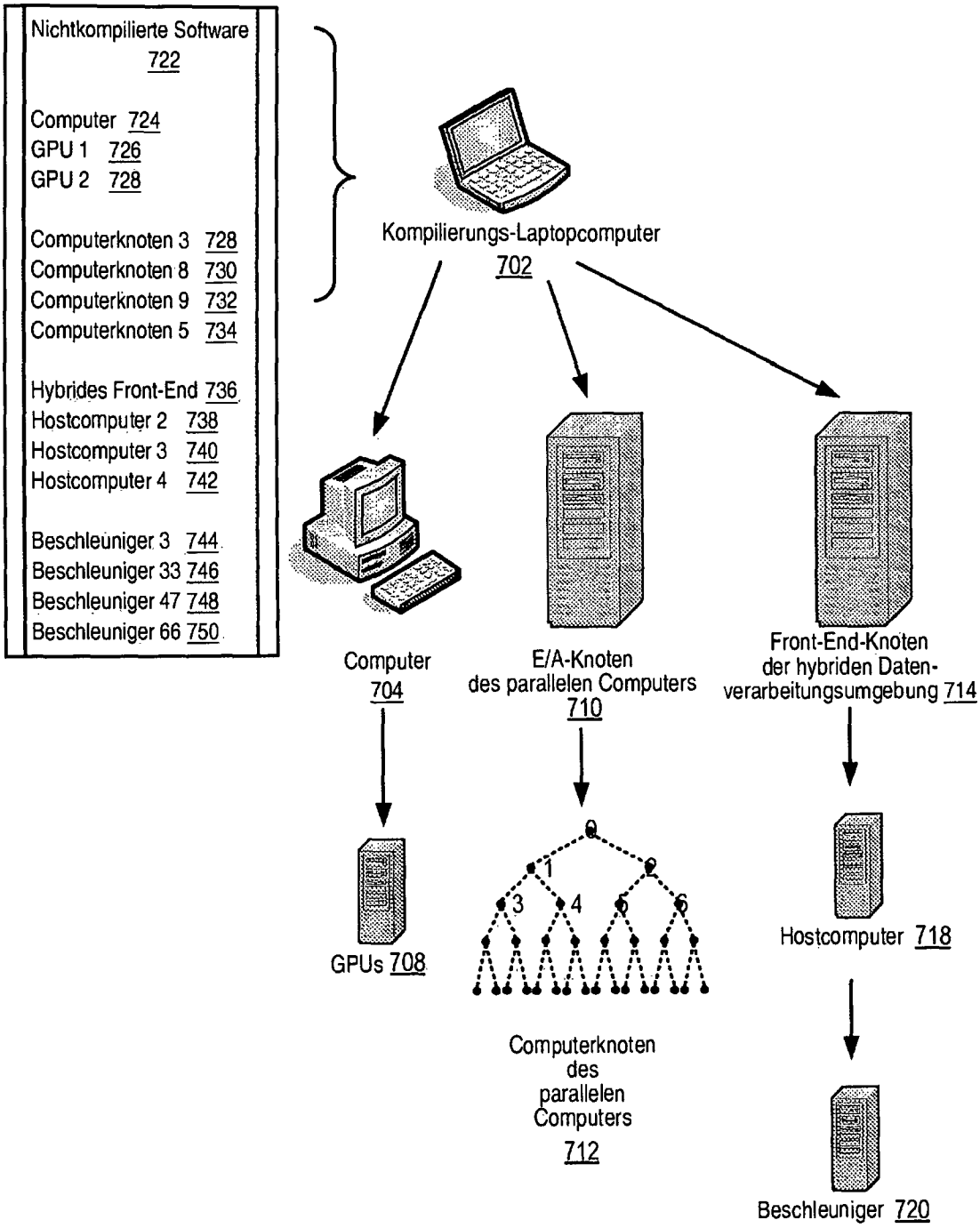

Nichtkompilierte Software 722
Computer 724
GPU 1 726
GPU 2 728
Computerknoten 3 728
Computerknoten 8 730
Computerknoten 9 732
Computerknoten 5 734
Hybrides Front-End 736
Hostcomputer 2 738
Hostcomputer 3 740
Hostcomputer 4 742
Beschleuniger 3 744
Beschleuniger 33 746
Beschleuniger 47 748
Beschleuniger 66 750
Kompilierungs-Laptopcomputer 702
Computer 704
E/A-Knoten des parallelen Computers 710
Front-End-Knoten der hybriden Datenverarbeitungsumgebung 714
GPUs 708
Computerknoten des parallelen Computers 712
Hostcomputer 718
Beschleuniger 720

FIG. 9