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

(10) **DE 698 24 688 T2** 2005.07.14

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 0 919 919 B1**
(21) Deutsches Aktenzeichen: **698 24 688.8**
(96) Europäisches Aktenzeichen: **98 309 634.8**
(96) Europäischer Anmeldetag: **25.11.1998**
(97) Erstveröffentlichung durch das EPA: **02.06.1999**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **23.06.2004**
(47) Veröffentlichungstag im Patentblatt: **14.07.2005**

(51) Int Cl.[7]: **G06F 11/34**

(30) Unionspriorität:
980124 **26.11.1997** **US**

(84) Benannte Vertragsstaaten:
DE, FR, GB

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

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

(72) Erfinder:
Dean, Jeffrey A., Menlo Park, California 94025, US; Chrysos, George Z., Marlboro, Massachussetts 01752, US; Hicks, James E., Newton, Massachussetts 02159, US; Waldspurger, Carl A., Atherton, California 94027, US; Weihl, William E., San Francisco, California 94114, US

(54) Bezeichnung: **System und Verfahren zur Leistungsoptimierung eines Rechnersystems**

Beschreibung

[0001] Diese Erfindung bezieht sich allgemein auf die Optimierung der Leistung von Computersystemen, und insbesondere auf ein Sammeln von Leistungs-Statistiken durch Abtasten, und Analysieren der abgetasteten Statistiken, um eine Systemoptimierung zu leiten.

[0002] Computersysteme werden anspruchsvoller und schneller, allerdings hält die Software-Anwendungs-Funktion nicht damit Schritt. Zum Beispiel wird in einem typischen Vier-Wege-Ausgabe-Prozessor nur ungefähr ein Zwölftel der verfügbaren Ausgabe-Schlitze gut ausgenutzt. Es ist wichtig zu verstehen, warum der Software-Ausführungs-Fluss nicht vollständig von der erhöhten Berechnungsleistung, die zum Verarbeiten von Befehlen verfügbar ist, vorteilhaft Gebrauch machen kann. Ähnliche Probleme entstehen in anderen Vorrichtungen in Computersystemen, die graphische Steuereinheiten, Speichersysteme, Eingabe/Ausgabe-Steuereinheiten und Netzwerk-Schnittstellen umfassen: die tatsächliche Funktionsweise bzw. Leistung ist oftmals geringer als die potentielle Spitzenleistung und es ist wichtig zu verstehen, warum dies der Fall ist.

[0003] Es ist üblich, solche Probleme bei Prozessoren den Speicher-Latenzzeiten zuzuschreiben, wobei, tatsächlich, viele Software-Anwendungen viele Zyklen aufwenden, um darauf zu warten, dass Datenübertragungen abgeschlossen werden. Moderne Speicher sind typischerweise in einer Multi-Level-Hierarchie angeordnet. Dabei ist der Datenfluß komplex und schwierig zu bestimmen, insbesondere dann, wenn mehrere Kontexte gleichzeitig für dieselbe Speicherressource, wie beispielsweise Cache-Blöcke, in Wettbewerb treten. Andere Probleme, wie beispielsweise Verzweigungs-Fehlvorhersagen und Cache-Fehlbeurteilungen, verschwenden auch Prozessor-Zyklen und verbrauchen eine Speicherbandbreite für Daten, auf die nutzlos Bezug genommen ist.

[0004] Eingabe/Ausgabe-Schnittstellen und Netzwerk-Steuereinheiten von Computersystemen werden auch anspruchsvoller. In vielen Ausführungsformen umfassen die Schnittstellen und Steuereinheiten Mikroprozessoren und Pufferspeicher, bei denen es schwieriger wird, das dynamische Verhalten zu messen und zu verstehen, wenn sich die Komplexität erhöht.

[0005] Unabhängig von den allgemeinen Ursachen müssen System-Architekten und Hardware- und Softwareingenieure wissen, welche Transaktionen blockieren, welche Daten zu einer Engstelle werden, und warum dies der Fall ist, um die Leistung von modernen Computersystemen zu verbessern.

[0006] Typischerweise wird dies durch Erzeugen eine „Profils" über die Verhaltensweise eines Computersystems, während es arbeitet, vorgenommen. Ein Profil ist eine Aufzeichnung von Leistungs-Daten. Häufig wird das Profil graphisch oder statistisch so präsentiert, dass Funktions- bzw. Leistungs-Engstellen leicht identifiziert werden können.

[0007] Ein Profilieren kann durch eine Gerätschaft und Simulation vorgenommen werden. Mit einer Gerätschaft bzw. Instrumentierung wird ein zusätzlicher Code zu Ausführungsprogrammen hinzugefügt, um spezifische Ereignisse zu überwachen. Eine Simulation versucht, das Verhalten des gesamten Systems in einer künstlichen Umgebung zu emulieren, im Gegensatz dazu, das Programm in dem realen System auszuführen. Auch kann eine Instrumentierung nur für Prozessor-Pipelines, nicht für andere Vorrichtungen, verwendet werden.

[0008] Jedes dieser zwei Verfahren besitzt seine Nachteile. Eine Instrumentierung stört das wahre Verhalten des Systems aufgrund der hinzugefügten Befehle und der zusätzlichen Daten-Referenzen. Mit anderen Worten schlägt eine Instrumentierung von komplexen Systemen in großem Maßstab in zweierlei Hinsicht fehl. Das System wird verlangsamt und die Funktions- bzw. Leistungs-Daten sind schlecht, oder bestenfalls oberflächlich.

[0009] Eine Simulation vermeidet eine Störung und Overhead. Allerdings arbeitet eine Simulation nur bei kleinen, gut definierten Problemen, die leicht als Modell dargestellt werden können. Es ist extrem schwierig, wenn nicht sogar unmöglich, ein System in großem Maßstab zu simulieren, mit tausenden von Benutzern, die über Faseroptik-Verbindungen mit Netzwerk-Steuereinheiten verbunden sind, die auf Terabytes an Daten über die Verwendung von Dutzenden von Mehrfach-Ausgabe-Prozessoren zugreifen. Es ist an das modellmäßige Darstellen einer Web-Suchmaschine zu denken, wie beispielsweise Digital's Alta Vista, die auf einige Zehnmillionen Treffer jeden Tag von der ganzen Welt anspricht. Jeder Treffer bietet möglicherweise bis zu hunderten von Web-Seiten als Suchergebnisse an.

[0010] Eine in einer Hardware ausgeführte Ereignis-Abtastung ist verwendet worden, um Profil-Informationen für Prozessoren zu liefern. Eine Hardware-Abtastung besitzt eine Anzahl von Vorteilen gegenüber einer Simulation und Instrumentierung: sie erfordert keine modifizierenden Software-Programme, um deren Leistung zu messen. Eine Abtastung arbeitet in Bezug auf vollständige Systeme, mit einem relativ geringen Overhead. Tatsächlich ist in neuerer Zeit gezeigt worden, dass eine auf einer Abtastung basierende Profilierung mit geringem Overhead dazu verwendet werden kann, detaillierte Instruktions-Level-Informationen über Pipeline-Eng-

stellen und deren Ursachen zu erhalten. Allerdings fehlt es vielen Hardware-Abtasttechniken an einer Flexibilität, da sie so ausgelegt sind, um spezifische Ereignisse, in Isolation, zu messen.

[0011] Es ist erwünscht, ein verallgemeinertes Verfahren und ein System zum Optimieren der Leistung, um Computersysteme zu betreiben, zu schaffen. Das Verfahren sollte in der Lage sein, Prozessoren, Speicher-Untersysteme, I/O-Schnittstellen, Graphik-Steuereinheiten, Netzwerk-Steuereinheiten, oder irgendeine andere Komponente, die digitale Signale manipuliert, zu überwachen.

[0012] Die Überwachung sollte in der Lage sein, wahlweise Transaktionen abzutasten und relevante Informationen über jede davon aufzuzeichnen. Im Gegensatz dazu sollte, mit einem auf einem Ereignis basierenden System, eine wahlweise Transaktions-Überwachung ermöglichen, nicht nur diskrete Ereignisse zu überwachen, sondern auch Ereignisse in irgendeiner Kombination. Es sollte auch möglich sein, auf die abgetasteten Ereignisse in Bezug auf individuelle Transaktionen, wie beispielsweise Befehle, oder Speicher-Referenzen, oder Zusammenhänge, in denen die Transaktionen standen, Bezug zu nehmen. Zusätzlich sollte es möglich sein, die abgetasteten Daten zu mehreren, gleichzeitigen Transaktionen in Bezug zu setzen, um ein wahres Verständnis über das System zu erhalten. All dies sollte, ohne Störung der Betriebsweise des Systems, eine andere als die Zeit, die erforderlich ist, um die erwünschien Leistungsdaten zu lesen, möglich sein.

[0013] ROTH C ET AL: „PERFORMANCE MONITORING ON THE POWERPCTM 604 MICROPROCESSOR" INTERNATIONAL CONFERENCE ON COMPUTER DESIGN: VLSI IN COMPUTERS AND PROCESSORS, US, NEW YORK, IEEE, 2. Oktober 1995 (1995-10-2), Seiten 212–215, XP00631915, offenbart ein System zum Optimieren der Leistung eines Computersystems, wobei das Computersystem eine Vielzahl von Funktionseinheiten umfasst, wobei das System zum Optimieren aufweist:
eine Quelle zum Bereitstellen von Transaktionen, die durch das Computersystem abzuarbeiten sind;
eine Einrichtung zum Auswählen von Transaktionen zum Abtasten, die durch eine Vielzahl von Funktionseinheiten des Computersystems abzuarbeiten sind;
einen Prozessor und Software zum Analysieren der Zustandsinformationen.

[0014] Gemäß der vorliegenden Erfindung ist ein solches System dadurch gekennzeichnet, dass die Transaktionen gleichzeitig von den funktionellen Einheiten abgetastet werden; und
durch Pufferspeicher, die Zustandsinformationen speichern, während die ausgewählten Transaktionen durch die Funktionseinheiten abgearbeitet werden, wobei die Zustandsinformationen wenigstens eine Adressen-Information, eine Transaktionsquellen-Information und eine Latenz-Information enthalten,
wobei die Pufferspeicher Zustandsinformationen im Verlauf der Zeit als eine Ansammlung von Zustandsinformationen speichern und die Ansammlung von Zustandsinformationen analysiert wird, um die Leistungsstatistiken des Computersystems zu schätzen, wobei die Leistungsstatistiken verwendet werden, um die Leistung des Computersystems dynamisch zu optimieren, während das Computersystem arbeitet.

[0015] Transaktionen, die durch eine bestimmte Funktionseinheit des Computersystems verarbeitet werden sollen, werden für eine Überwachung ausgewählt. Die Transaktionen können gleichzeitig ausgewählt werden. Zustandsinformationen werden gespeichert, während die ausgewählten Transaktionen durch die Funktionseinheit verarbeitet werden. Die Zustandsinformationen werden analysiert, um eine Optimierung zu leiten.

[0016] Gemäß einem Aspekt können mehrere, unterschiedliche Funktionseinheiten gleichzeitig abgetastet werden.

[0017] Die Erfindung schafft auch ein Verfahren zum Optimieren der Leistung eines Computersystems mit einer Vielzahl von Funktionseinheiten, wobei das Verfahren die folgenden Schritte umfasst:
Bereitstellen von Transaktionen von einer Quelle, die durch das Computersystem abzuarbeiten sind:
Auswählen von durch eine Vielzahl von Funktionseinheiten des Computersystems abzuarbeitenden Transaktionen, wobei die Transaktionen gleichzeitig durch die Funktionseinheiten abgetastet werden;
Speichern von Zustandsinformationen als eine Ansammlung von Zustandsinformationen im Verlauf der Zeit in Pufferspeichern, während die ausgewählten Transaktionen durch die Funktionseinheiten abgearbeitet werden, wobei die Zustandsinformationen wenigstens eine Adressen-Information, eine Transaktionsquellen-Information und eine Latenz-Information enthalten; und
Analysieren der Ansammlung von Zustandsinformationen unter Verwendung eines Prozessors und von Software, um Leistungsstatistiken des Computersystems zu schätzen, wobei die Leistungsstatistiken verwendet werden, um die Leistung des Computersystems dynamisch zu optimieren, während das Computersystem arbeitet.

[0018] In den beigefügten Zeichnungen:

[0019] Fig. 1 zeigt ein Blockdiagramm eines komplexen Computersystems, das abgetastet werden soll;

[0020] Fig. 2 zeigt ein Blockdiagramm eines Mechanismus zum Auswählen von Transaktionen, ver-

arbeitet durch das System der Fig. 1, gemäß einer bevorzugten Ausführungsform;

[0021] Fig. 3 zeigt ein Blockdiagramm eines Abtastpuffers; und

[0022] Fig. 4 zeigt ein Blockdiagramm eines Verfahrens zum Abtasten und Optimieren eines Computersystems.

[0023] Fig. 1 stellt ein Computersystem **100** dar. Das Computersystem **100** umfasst eine oder mehrere funktionale Einheiten) **111**–**114**, zum Beispiel Prozessoren, Speicher, eine Eingabe/Ausgabe-Schnittstelle und Netzwerk-Steuereinheiten. Typischerweise sind die Einheiten **111**–**114** miteinander durch Busse, Datenkanäle und Netzwerke, z. B. **150**, in Abhängigkeit von einer bestimmten Konfiguration, verbunden.

[0024] Der Prozessor **111** kann von einer verschiedenartigen Architektur sein, CISC oder RISC, Einfach- oder Mehrfach-Ausgabe, locker oder eng gekoppelt. Die Speicher **112** sind typischerweise unter hierarchischen Niveaus angeordnet, wobei Speicher näher zu den Prozessoren eine höhere Bandbreite und einen geringeren Speicher haben, und Speicher, die weiter entfernt liegen, eine geringere Bandbreite und mehr Speicher haben. Die Speicher können auf einem Chip oder außerhalb eines Chips befindliche Cache-Speicher, SAM, DRAM, eine Festplatte, oder dergleichen, umfassen. Die I/O-Schnittstelle **113** verbindet sich mit peripheren Vorrichtungen. Einige der Schnittstellen können interne Prozessoren und Speicher umfassen. Die Netzwerk-Steuereinheiten **114** können sich schnittstellenmäßig mit Local- und Wide-Area-Networks verbinden.

[0025] Während eines Betriebs des Systems **100** werden Transaktionen (T) **101** durch eine Quelle **110** als Eingabe für das System erzeugt. Die verschiedenen Funktionseinheiten **111**–**114** des Systems **100** verarbeiten die Transaktionen **101**, um einen Ausgang **102** für ein Drain zu schaffen. Es sollte angemerkt werden, dass die Transaktionen von außerhalb des Systems stammen können, oder von innerhalb des Systems, und zwar aufgrund der Verarbeitung. Zusätzlich erzeugt die Verarbeitung von Transaktionen, zum Beispiel von Befehlen, gewöhnlich viele zusätzliche, interne Transaktionen, wie beispielsweise Speicher-Referenzen, Platten-Transfers, oder Netzwerk-Nachrichten.

[0026] Während die Transaktionen **101** verarbeitet werden, werden Ereignisse **104** signalisiert, z. B. Befehl-Aussonderungen und Traps, Cache-Fehler, Taktzyklen, Ausnahmezustände, usw.. Die tatsächlichen Ereignisse **104** können in Abhängigkeit von der Ausführung der Funktionseinheiten **111**–**114** signalisiert werden.

[0027] Während das System **100** arbeitet, wird ein Zustand **130** beibehalten. Der Zustand **130** wird gewöhnlich über die Funktionseinheiten verteilt. Eine Verarbeitung der Transaktionen verursacht Zustandsübergänge. Einige der Zustandsübergänge können von einem bestimmten Interesse sein, um die Funktionsweise bzw. Leistung des Systems zu verstehen.

[0028] Um die Abtastung von Transaktionen, Ereignissen und dem Zustand vorzunehmen, werden Transaktionen einem Transaktions-Selektor präsentiert. Der Transaktions-Selektor kann für unterschiedliche Funktionseinheiten **111**–**114** unter Verwendung eines üblichen Designs, wie es hier beschrieben ist, repliziert werden.

[0029] Allgemein muss der Selektor eine Einrichtung sein, um Transaktionen basierend auf dem Zustand des Systems und die Transaktion selbst auszuwählen. Obwohl es nicht Teil der Erfindung ist, können Transaktionen zufällig, zum Beispiel durch Initialisieren eines Zählers auf einen zufällig auswählenden Wert, Erhöhen des Zählers für jede Transaktion und Auswählen einer Transaktion, wenn der Zähler überläuft (oder unterläuft), ausgewählt werden. Transaktionen können nur für eine solche Verarbeitung von bestimmten Eigenschaften ausgewählt werden (zum Beispiel durch Erhöhen des Zählers nur für Transaktionen, die einen spezifizierten Satz von Eigenschaften anpassen), oder können nur dann ausgewählt werden, wenn sich das System in bestimmten Zuständen befindet (durch Erhöhen des Zählers nur dann, wenn ein spezifizierter Zustand vorliegt).

[0030] Fig. 2 stellt die herausragenden Merkmale einer möglichen Ausführung eines Selektors **200** dar. Der Selektor **200** umfasst einen Trigger **210**, einen Zähler **220** und einen Markierer **230**.

[0031] Der Zähler **220** wird mit einem Wert (Init-Wert) **221** auf einer Leitung **222** eingestellt und zurückgesetzt (initialisiert). Der Wert **221** kann durch eine Hardware (HW) **223** oder eine Software (SW) **224** erzeugt werden. Wie nachfolgend beschrieben werden wird, bestimmt der Wert **221**, teilweise, eine Abtastrate, und kleine Anfangswerte erhöhen die Abtastrate und große Werte verringern die Abtastrate. Falls der Zähler abwärts zählt, dann erhöhen kleine Anfangswerte die Abtastrate und große Anfangswerte verringern die Rate.

[0032] Als ein Vorteil kann der Wert **221** gleichförmig und zufällig aus einem ganzzahligen Intervall ausgewählt werden. Die durchschnittliche Rate einer Abtastung kann durch Auswählen eines geeigneten, ganzzahligen Intervalls eingestellt werden, von dem der Wert **221** ausgewählt ist.

[0033] Der Zähler **220** wird durch ein Zählerereig-

nissignal, empfangen auf der Leitung **225**, erhöht (oder erniedrigt, in Abhängigkeit von dem Design). Das Zählerereignissignal **225** kann von einem oder von mehreren Ereignissignal(en) (Ereignis1, Ereignis2, Ereignis3) **104** durch ein Zählauswahlsignal auf der Leitung **229** ausgewählt werden. Die Ereignissignale **104** können Taktzyklen, Transaktionen, verfügbar für eine Verarbeitung, Transaktionen, ausgenommen für eine Verarbeitung, usw., umfassen.

[0034] Der Trigger **210** bestimmt, wenn und unter welchen Bedingungen der Zähler **220** aufwärts zählt (oder abwärts zählt), und der Markierer **230** bestimmt, welchen Transaktionen als ausgewählte Transaktionen (T') **103** markiert sind.

[0035] Deshalb empfängt der Trigger **210** die Transaktionen **101**, die Ereignisse **104** und den Zustand **130**, und zwar in Abhängigkeit von der bestimmten Funktionseinheit, die abgetastet wird. Die momentanen Transaktions-Informationen, die Ereignisse und der Zustand werden logisch durch den Trigger **210** unter Verwendung einer Funktion, zum Beispiel einer logischen Funktion, umfassend Bool'sche Operatoren (AND, OR, NOT), kombiniert. Falls das Ergebnis der Kombination wahr ist, dann wird der Zähler **220** zum Zählen freigegeben, wobei ansonsten, falls es falsch ist, dies nicht der Fall ist.

[0036] Einige spezifische Beispiele von nützlichen Triggerfunktionen umfassen, sind allerdings nicht darauf beschränkt, solche, die passen bei: irgendeiner Transaktion, Transaktionen eines bestimmten Typs (Speicher-Referenz-Befehle, Transaktionen, die auf ein bestimmtes Niveau der Speicherhierarchie Bezug nehmen, z. B. ein bestimmter Cache oder Cache-Block), Transaktionen, die einen bestimmten Zustandsübergang verursachen werden, z. B. Dirty-Daten-Ausweisungen, Daten, die auf einen bestimmten Bereich in den Speichern Bezug nehmen, z. B. einen Bereich von Adressen, eine bestimmte Cache-Linie, einen bestimmten Cache-Block innerhalb einer bestimmten Cache-Leitung, einen bestimmten Bereich des Cache, usw., Transaktionen von einer bestimmten Quelle, z. B. von einer Prozessor-Pipeline, von einer I/O-Schnittstelle, z. B. einem Direktspeicherzugriff, Transaktionen, die auf Daten-Kohärenz-Nachrichten in Bezug gesetzt sind, usw.. Es sollte angenommen werden, dass der Trigger **220** durch entweder die Hardware **223** oder die Software **224** programmierbar ist.

[0037] Die Verwendung der Triggerfunktion **250** ermöglicht der Abtast-Hardware, spezifizierte Transaktionen zu überspringen. Zum Beispiel würde dies, in einem Beispiel mit anspruchsvoller Funktion, ermöglichen, drei Zugriffs-Transaktionen zu einem bestimmten Cache-Block zu zählen und dann Speicher-Transaktions-Abtastungen für die nächsten zwei Fehlzugriffe zu diesem selben Block zu sammeln. In einem anderen nützlichen Beispiel kann man eine Markierung nach einer Transaktion, die von einem bestimmten Zusammenhang stammt, wie beispielsweise einem Prozessor, einem Prozess in einem Thread, und einer I/O-Schnittstelle, stammen, triggern, und dann Abtastungen für eine spezifizierte Zahl von darauf folgenden Transaktionen, oder eine spezifizierte Zeitdauer, sammeln.

[0038] Der Markierer **230** identifiziert eine Transaktion als eine ausgewählte Transaktion T' **103**, immer wenn der Zähler überläuft (oder unterläuft). An diesem Punkt kann der Zähler **220** mit einem neuen Wert **221** zurückgesetzt werden. Es sollte angemerkt werden, dass sich der Wert dynamisch in Abhängigkeit davon ändern kann, welcher Typ einer Abtastung erwünscht ist, der detaillierten Abtastung über ein kurzes Zeitintervall mit vielen Abtastungen, oder der allgemeinen Abtastung mit geringem Overhead über ein längeres Zeitintervall unter Verwendung einer zufälligen Abtastung.

[0039] Der Markierer **230** ist auch so konfiguriert, dass er Ereignisse **104** und einen Zustand **130**, ebenso wie die Transaktionen, aufnehmen kann. Die Aufgabe des Markierers **230** ist diejenige, zu bestimmen, ob die momentane Transaktion für eine Abtastung zu dem Zeitpunkt, zu dem die Markierung getriggert ist, aufgrund eines Überlaufens des Zählers **220**, von Interesse ist, und, falls dies nicht der Fall ist, dann kann die Transaktion ignoriert werden und der Zähler kann zurückgesetzt werden.

[0040] Die verschiedenen Funktionen, die tatsächlich bewirken können, dass der Markierer **230** eine Transaktion auswählt, umfassen, sind allerdings nicht darauf beschränkt, Transaktionen, die auf ein bestimmtes Niveau in der Speicherhierarchie Bezug nehmen, Referenzen zu einem bestimmten Bereich eines Speichers innerhalb eines bestimmten Niveaus der Speicherhierarchie, Transaktionen eines bestimmten Typs, z. B. Verzweigungsbefehle, Transaktionen, die ein zugeordnetes Ereignis haben, z. B. ein Fehlschlagen, eine Verzweigungs-Fehlvorhersage, eine Aussonderung, einen bestimmten Zustandsübergang, z. B. Dirty-Aussonderungen, Transaktionen, die von einer bestimmten Quelle stammen, z. B. ein Befehl, der in der Prozessor-Pipeline ausgeführt wird, eine Befehlsausführung von einem bestimmten Zusammenhang, Prozess, Thread, oder Adressenraum, einen Direktspeicherzugriff von einer Eingabe/Ausgabe-Schnittstelle, Cache-Kohärenz-Nachrichten in einem Multiprozessor-Computersystem, usw..

[0041] Die Markierung der ausgewählten Transaktion kann durch Versehen der Transaktion mit einem zusätzlichen „Abtast" („Sample") Bit, z. B. von T zu T', erreicht werden. Zum Beispiel wird, falls die Transaktion ein Befehl ist, das Befehlsfeld mit einem zusätz-

lichen Bit erweitert, das mit der Transaktion mitgeführt wird, bis die Transaktion vollständig verarbeitet ist. Alternativ werden die Transaktionen nummeriert oder in anderer Weise identifiziert. In der vorliegenden Erfindung werden mehrfache Transaktionen gleichzeitig ausgewählt, wobei in einem solchen Fall mehrfache Bits benötigt werden, um sie zu markieren.

[0042] Nachdem eine Transaktion für eine Abtastung ausgewählt worden ist, wird die entsprechende Funktionseinheit, die die markierte Transaktion **103** verarbeiten wird, das Abtast-Bit an jeder entsprechenden Verarbeitungsstufe prüfen, und Zustandsinformationen, die für sie verfügbar sind, sammeln. Die gesammelten Zustandsinformationen werden in einer Mehrzahl von Puffern gespeichert, wie dies im Detail nachfolgend beschrieben ist. Wenn mehr als ein Puffer verwendet wird, können mehrfache Transaktionen gleichzeitig abgetastet werden. Die Stelle des Puffers liegt in der Auswahl des Designers, vorzugsweise nahe zu der Stelle, wo die Abtast-Informationen erzeugt sind.

[0043] Einige Zustandsinformationen können aufgezeichnet werden, bevor die Transaktion durch die Funktionseinheit verarbeitet wird, um einen Anfangszustand zu erfassen, und zusätzliche Informationen können nach dem Abschluss der Transaktion aufgezeichnet werden, um einen Endzustand zu erfassen. Nachdem eine spezifizierte Anzahl von Transaktionen aufgezeichnet worden ist, zum Beispiel dann, wenn der Abtastpuffer voll ist, kann ein Lesesignal erzeugt werden. Das Lesesignal kann in der Form einer Unterbrechung, eines durch eine Software abrufbaren Werts, eingestellt in ein Register, oder eines Ausnahmezustands vorliegen. Das Lesesignal kann eine Software freigeben, um die abgetasteten Daten für eine weitere Verarbeitung zu lesen.

[0044] Es sollte angemerkt werden, dass mehrere Puffer verwendet werden können, um mehrere Abtastungen zusammenzustellen. Ein Erhöhen der Anzahl von Puffern amortisiert die Kosten eines Abtast-Overheads, indem mehr als eine Abtastung pro gelesenem Signal übertragen wird.

[0045] Eine Beendigung einer Verarbeitung einer Abtast-Transaktion kann die Verarbeitung der Transaktion oder die Aussonderung der Verarbeitung aufgrund der Ankunft an einem bestimmten, nicht korrekten oder illegalen Zustand, oder einer Unterbrechung der Verarbeitung aufgrund eines bestimmten, externen Ereignisses, abschließen.

[0046] Fig. 3 stellt die Details dar, wie ein Puffer **300** zum Speichern von Zustandsinformationen zugeordnet werden kann. Der Puffer **300** kann als ein Satz von durch eine Software lesbaren Registern, oder andere Typen von Speichern, ausgeführt werden. Der Puffer umfasst ein Status-Feld **310**, ein Adressen-Feld **320**, ein Kontext-Feld **330**, ein Transaktions-Quellen-Feld **340**, ein Befehls-Feld **350**, ein Latenz-Feld **360** und Felder **370** für andere Zustände und Ereignisse.

[0047] Das Status-Feld **310** speichert Zustandsinformationen, die sich auf die bestimmte Transaktion, die verarbeitet werden soll, beziehen. Zum Beispiel kann, falls die Transaktion eine Speicherreferenz ist, der Zustand anzeigen, ob ein Cache-Block schmutzig (dirty) oder sauber (clean) (modifiziert oder nicht), exklusiv (nicht gemeinsam geteilt), gültig oder ungültig (die Daten sind legitimiert) und ein Cache-Treffer- oder Fehler-Status ist.

[0048] Das Adressen-Feld **320** kann die virtuelle und/oder physikalische Adresse, die der Transaktion zugeordnet ist, speichern.

[0049] Das Kontext-Feld **330** kann die Adressen-Raum-Zahl (Adress Space Number – ASN), den Hardware-Kontext-Identifizierer (HCI) in dem Fall eines Multi-Threaded-Prozessors, einen Prozess-Identifizierer (PID) und/oder einen Thread-Identifizierer (TID) der Quelle der Transaktion speichern. Das Feld kann die Adressen-Raum-Zahl (ASN), auf die durch die Transaktion Bezug genommen ist, speichern.

[0050] Das Quellen-Feld **340** kann dazu verwendet werden, die Quelle der Transaktion, z. B. Lade- oder Speicher-Befehl, eine DMA-Anforderung oder eine Cache-Kohärenz-Protokoll-Operation, ebenso wie zusätzliche Informationen, um die Quelle zu identifizieren, und zwar in Abhängigkeit von der bestimmten Funktionseinheit, zu speichern.

[0051] Falls die Quelle der Transaktion eine Abrufeinheit der Prozessor-Pipeline ist, dann kann der Programm-Zähler (PC) des Befehls in dem Befehls-Feld **350** gespeichert werden. Das Programm-Zähler-Feld **350** kann auch dazu verwendet werden, Informationen über andere Arten von Quellen zu speichern, um ein Register zu sichern. Zum Beispiel kann dann, wenn die Quelle eine Kohärenz-Operation von einem anderen Prozessor in einem Multiprozessor-Computersystem ist, das Feld **350** dazu verwendet werden, die Prozessorzahl des Prozessors, von der DMA-Anforderung ausgehend, die die Kohärenz-Operation verursachte, zu halten. Für einen DMA-Typ von Transaktionen kann die Identität der I/O-Vorrichtung, die die DMA initiierte, gespeichert werden.

[0052] Das Zeitintervall (Latenz) zwischen aufeinanderfolgenden Transaktionen oder aufeinanderfolgenden Stufen einer Transaktions-Verarbeitung kann in dem Latenz-Feld **360** gespeichert werden. Das Intervall kann im Hinblick auf Prozessor-Taktzyklen gemessen werden, oder das Intervall kann in anderen

Einheiten, wie beispielsweise der Zahl von Transaktionen, die verarbeitet werden sollen, gemessen werden. Das Intervall kann auch in die Zeit, erforderlich dazu, die Transaktion an jedem Level einer Speicherhierarchie zu verarbeiten (in dem Fall einer Abtast-Speicher-Funktion), oder Stufen einer Prozessor-Pipeline (in dem Fall eines Prozessors), aufgeteilt werden.

[0053] Zusätzliche Register, wie beispielsweise ein Feld **370**, können zu dieser Struktur hinzugefügt werden, um einen zusätzlichen Systemzustand zu speichern, der zu dem Zeitpunkt erfasst ist, zu dem die abgetastete Transaktion verarbeitet ist. Dieser Zustand kann die Inhalte oder die Zahl von gültigen Eintritten in Speichersystemstrukturen umfassen, wie beispielsweise Schreibpuffer, Victim-Cache, Translations-Lookside-Puffer (TBLs), Fehl-Adressen-Dateien (Miss-Adress-Files – MAFs), und Speicher-Transaktions-Warteschlangen, und zwar in dem Fall einer Funktionseinheit.

[0054] Eine kurze Ausführung über einige beispielhafte Abtaststrategien werden nun angegeben. Die vorliegende Erfindung kann dazu verwendet werden, eine gleichzeitige Abtastung vorzunehmen.

[0055] Eine Art und Weise, in der interessante Informationen zusammengestellt werden können, ist diejenige einer Abtastung an mindestens zwei aufeinanderfolgenden Transaktionen, z. B. ein „Paar"; als Beispiel zwei Speicherreferenzen zu demselben Cache-Block, oder zwei in Bezug stehende Befehle. Ein paarweises Abtasten kann auf eine N-weise Abtastung über ein vorbestimmtes Zeitfenster erweitert werden.

[0056] Deshalb können die Vorrichtung und das Verfahren, die hier offenbart sind, Transaktionen abtasten zu: spezifischen Cache-Blöcken, clean oder dirty, spezifischen Bereichen von Speichern, allen Speicherstellen, allen Speicherstellen, wo Cache-Blöcke „dirty" sind, zu Speicherstellen, wo sich die Daten nicht in dem Cache befinden.

[0057] Die Angabe eines Speicherns von Zustandsinformationen für aufeinanderfolgende Transaktionen können zu sequenziellen Zugriffen verallgemeinert werden, die einfache, durch eine Software spezifizierte Kriterien anpassen, wie beispielsweise aufeinanderfolgende Fehler, aufeinanderfolgende Treffer, aufeinanderfolgende Ungültigkeiten, usw..

[0058] Zusätzlich ermöglichen die Abtast-Techniken, wie sie hier beschrieben sind, eine feinfühligere Überwachung von Funktionseinheiten mit einem geringen Hardware-Overhead. Diese Informationen können auf viele Arten und Weisen verwendet werden. Zum Beispiel können die zusammengestellten Informationen System-Designer dabei unterstützen, besser die Funktionsweise bzw. die Leistung eines Prozessors, und von Speicher-Untersystemen, wie beispielsweise Cache, DRAM, und dergleichen, zu verstehen. Die Funktions-Daten werden dazu verwendet, Optimierungen zu leiten.

[0059] Zum Beispiel kann man, um den Umfang einer gemeinsamen Nutzung eines Speichers abzuschätzen, Paare von Speicher-Transaktionen auswählen, wo die zweite Transaktion in dem Paar ein Cache-Treffer ist. Dies zeigt an, dass dort eine gemeinsame Aufteilung zwischen der ersten und der zweiten Transaktion vorhanden war. Durch Prüfen der den Zusammenhang identifizierenden Informationen, die beiden Abtastungen zugeordnet sind, ist es möglich, zu bestimmen, welche Zusammenhänge nützlicherweise gemeinsam diesen physikalischen Raum während des abgetasteten Zeitintervalls teilen. Durch Sammeln dieser Informationen über viele solche Paare von Abtastungen kann man statistisch Metriken abschätzen, die sich auf eine gemeinsame Teilung von physikalischen Stellen in einem Intra-Kontext und einem Inter-Kontext beziehen.

[0060] Eine nützliche Metrik wird durch Zählen der Anzahl von Paaren bestimmt, wo das erste Paar in der Abtastung einen spezifischen Zusammenhang anpasst und wo das zweite Paar in der Abtastung einen zweiten, spezifischen Zusammenhang anpasst, was effektiv zu einer Matrix von Zählungen führt, die durch die Identifizierer des ersten und des zweiten Kontextes indexiert ist. Ähnlich kann man, durch Analysieren von Paaren, wo die zweite abgetastete Transaktion einen Cache-Fehler erfuhr, statistisch Metriken abschätzen, die sich auf einen Konflikt in Bezug auf physikalische Stellen eines Intra-Kontexts und eines Inter-Kontexts beziehen.

[0061] Eine alternative Verwendung dieser Hardware ist diejenige, einen spezifischen Cache-Bereich dahingehend auszuwählen, um ihn zu überwachen. Der ausgewählte Bereich entspricht dem Raum in dem Cache, der ein bestimmtes Programm, das variabel ist, oder eine Datenstruktur speichert. Durch Zusammenstellen von Abtastungen und Filtern der Abtastungen, um Abtastpaare zu erhalten, wo mindestens eine der Transaktionen die variable oder Datenstruktur, die von Interesse ist, einsetzt, ist es möglich, Cache-Konflikt-Raten abzuschätzen und andere, spezifische Programmvariablen oder Datenstrukturen, die die Quellen von Konflikten sind, zu identifizieren.

[0062] Diese Abschätzung kann dynamisch vorgenommen werden, um ein Online-Programm-Debugging (Fehlersuche) oder eine Optimierung von Funktions- bzw. Leistungsproblemen innerhalb eines laufenden Programms oder Systems zu ermöglichen. Diese Technik ermöglicht Programmierern, interaktiv eine Speichersystemfunktion von Fehlern zu befrei-

en, durch Identifizieren von Konflikten in Ausführprogrammen, die untersucht sind. Ähnliche Techniken können durch eine adaptive Laufzeit-Software verwendet werden, um ernsthafte Cache-Konflikte über eine Dynamik-Daten-Neuzuordnung zu vermeiden.

[0063] Ein Abtasten von individuellen Speichersystem-Transaktionen macht es möglich, eine Vielzahl von statistischen Metriken über Verteilungen von Eigenschaften eines Speichersystemverhaltens zu berechnen. Zum Beispiel ist es möglich, Verteilungen von Latenzen zu schätzen, um Speicheranforderungen zu bedienen, oder Raten von Cache-Treffern unter einem bestimmten Niveau oder einem Bereich in der Speicherhierarchie abzuschätzen. Filtermechanismen können dazu verwendet werden, Untersätze der aufgezeichneten Transaktionen, die von Interesse sind, zu identifizieren, was ermöglicht, Statistiken auf bestimmte Aspekte des Speichersystems, die von Interesse sind, zu fokussieren, wie beispielsweise Transaktionen zu einem bestimmten Bereich oder Level in der Speicherhierarchie, oder eine bestimmte Klasse von Transaktionen, wie beispielsweise Lesevorgänge, Schreibvorgänge oder Ungültigkeiten.

[0064] Nachdem ein Satz von Abtastungen, die von Interesse sind, identifiziert worden ist, können standardmäßige, statistische Techniken verwendet werden, um Durchschnitte, Standardabweichungen, Histogramme und andere Statistiken, über die Abtastungen, die von Interesse sind, abzuleiten. Durchschnitte können dazu verwendet werden, Raten abzuschätzen, für die bestimmte Ereignisse auftreten, wie beispielsweise Cache-Treffer oder -Fehler, oder Aussonderungen.

[0065] Es ist auch möglich, den Anteil von Anforderungen aufgrund von Lesevorgängen, Schreibvorgängen oder Ungültigkeiten abzuschätzen. Diese Raten können auch in Bezug auf einen bestimmten Zusammenhang abgeschätzt werden, wie beispielsweise Abschätzen von Metriken, wie beispielsweise Cache-Treffer-Raten pro Prozess, oder durchschnittliche Speichersystem-Latenzzeit, die durch ein Thread erfahren ist. Es ist auch möglich, den Anteil eines Levels der Speicherhierarchie abzuschätzen, der durch einen bestimmten Kontext verbraucht wird.

[0066] Standardfehler-Abschätztechniken können verwendet werden, um Aussagewahrscheinlichkeits-Intervalle über die Genauigkeit der erwünschten Statistiken zu erhalten. Insbesondere können, für Statistiken, die eine Anzahl von Abtastungen mit einer spezifischen Eigenschaft einsetzen, Fehlerbereichsgrenzen unter Verwendung des Umkehrwerts der Quadratwurzel der Anzahl von Abtastungen mit dieser Eigenschaft abgeschätzt werden. Diese Fehlerbereichsgrenzen können auch dazu verwendet werden, dynamisch die Rate zu steuern, unter der ausgewählte Transaktionen abgetastet werden, um so eine Genauigkeit mit einem Abtast-Overhead abzuwägen.

[0067] Wenn die aufgezeichneten Zustandsinformationen Latenzinformationen umfassen, entweder in der Form der Latenz, die dazu erforderlich ist, die Speicher-Transaktion zu verarbeiten, oder im Hinblick auf die Latenz zwischen zwei aufeinanderfolgenden, abgetasteten Speicher-Transaktionen, können die Informationen dazu verwendet werden, Statistiken, die auf einer Latenz basieren, zu berechnen. Eine Latenz wird typischerweise in Zeiteinheiten gemessen, wie beispielsweise Prozessor-Taktzyklen, können allerdings auch in anderen Einheiten, wie beispielsweise die Zahl von Speicher-Transaktionen, die verarbeitet werden soll, gemessen werden.

[0068] In einem sehr allgemeinen Sinne führen die Prozessoren Befehle aus, die in Bezug auf Daten arbeiten: In vielen modernen Computersystemen werden die Befehle und Daten gewöhnlich als separate Strukturen unter Verwendung von unterschiedlichen Speicherseiten beibehalten, da die Zugriffsmuster für Befehle sehr viel unterschiedlicher als diejenigen für die Daten sind. Eine virtuelle zu physikalische Speicherauflistung für Befehle und Daten wird gewöhnlich durch das Betriebssystem vorgenommen. Alternativ kann eine Neuzuordnung von Strukturen manuell oder durch Compiler, Linker und Loader vorgenommen werden. Einige Systeme können Strukturen dynamisch, wenn die Befehle ausgeführt werden, wieder zuordnen.

[0069] Unter Verwendung der Hardware, die hier beschrieben ist, ist es möglich, ein Feedback in Bezug auf eine Vielfalt von interessanten Teilen einer Software zu liefern. Zum Beispiel können Transaktions-Zustandsinformationen für einen abgetasteten Speicher dazu verwendet werden, zum Beispiel Seiten-Neuauflistungs-Policen zu bewirken, oder eine Selbst-Interferenz zu vermeiden, indem ein Feedback zu Compilern, Linkern oder Loadern vorgesehen wird.

[0070] Zum Beispiel kann eine Software Konfliktadressen auf einem Seitenniveau zusammenstellen, um über dynamische Seiten-Neuauflistungs-Algorithmen, ausgeführt durch das Betriebssystem, zu informieren. Es ist auch möglich, Profilierungs-Tools, die von Interesse sind, die potentielle Funktionsprobleme Programmierern und Benutzern anzeigen, vorzusehen.

[0071] Zum Beispiel ist es nun möglich, abzuschätzen, wie oft Daten „dirty" sind, wenn die Daten ausgesondert werden, und wie oft DMA-Transfers zu Cache-Kohärenz-Protokoll-Transaktionen auftreten, was einen Anhalt dafür gibt, wie effektiv das Speichersystem verwendet wird.

[0072] Bei Multiprozessorsystemen mit nicht gleichförmigem Speicherzugriff (Non-Uniform Memory Access – NUMA) besitzt jeder Prozessor Bereiche des Speichersystems, auf die er schneller (oder mit höherer Bandbreite) zugreifen kann als auf andere Teile des Speichersystems. Um die Funktion bzw. Leistung zu verbessern, können Daten (die entweder Programmdaten oder Befehle sind), auf die häufig durch einen Prozessor zugegriffen wird, zu einem Bereich des Speichersystems bewegt werden, auf den schneller durch den Prozessor zugegriffen werden kann.

[0073] Diese Bewegung kann auf zwei Arten und Weisen ausgeführt werden. Die Daten können durch Markieren von Mehrfach-Kopien der Daten repliziert werden. Idealerweise werden die Daten in vernünftiger Weise „gestreut" („scattered"), und zwar über das Speichersystem. Alternativ können die Daten durch tatsächliches Bewegen der Daten in einen Speicher mit niedrigerer Latenz oder höherer Bandbreite umgestellt werden.

[0074] Informationen über den Typ von Zugriffen (z. B. Lesen, Schreiben und Ungültigkeiten) können weiterhin die Entscheidung leiten, ob die Daten zu replizieren, zu migrieren oder an Ort und Stelle zu belassen sind. Zum Beispiel sollten Daten, die häufig durch mehrere Prozessoren geschrieben werden (z. B. im Schreiben gemeinsam geteilte Seiten), wahrscheinlich nicht repliziert oder migriert werden, während Daten, die häufig gelesen werden, allerdings nur selten geschrieben werden (z. B. in Bezug auf im Lesen gemeinsam geteilter Seiten), gute Kandidaten für eine Replizierung sind. Seiten, auf die selten nur durch einen einzelnen Prozessor zugegriffen wird, sind gute Kandidaten für eine Migration zu einem Speicher, der sich näher zu dem zugreifenden Prozessor befindet. Diese Informationen können durch ein statistisches Abtasten von Speichersystem-Transaktionsinformationen, wie dies hier beschrieben ist, gesammelt werden.

[0075] Die Informationen über Speichersystem-Transaktionen werden dynamisch gesammelt und können in einer On-Line weise verwendet werden, um dynamisch die Replizierungs- und Migrations-Policen des Computersystems zu steuern. Typischerweise werden eine Replizierung und Migration durch das Betriebssystem gehandhabt, allerdings können sie auch durch andere Software- oder Hardware-Ebenen gehandhabt werden.

[0076] Dabei sind mehrere, potenzielle Funktions-Metriken vorhanden, die die Replizierungs- oder Migrations-Policen versuchen können, zu verbessern, einschließlich einer Erhöhung des gesamten Systemdurchsatzes, einer Erhöhung des Durchsatzes für bestimmte Aufgaben mit hoher Priorität, einer Verringerung in dem Verkehr zwischen Prozessoren und Speichern, eine Verringerung der gesamten Speicher-Latenzzeit, oder einer gesamten Erhöhung der Systemleistung.

[0077] Da Cache in einem hierarchischen Speicher Daten speichern, die von verschiedenen Hardware-Kontexten ausgehen, treten Threads, die in unterschiedlichen Hardware-Kontexten ausführen, für Linien (Leines) in einem Cache in Wettbewerb. Deshalb ist es erwünscht, Threads ablaufmäßig so zu planen, dass Ressource-Konflikte minimiert werden. Dies kann durch Abtasten von Speichersystem-Transaktionsinformationen, wie dies hier beschrieben ist, vorgenommen werden.

[0078] Das ablaufmäßige Planen von Entscheidungen kann aus der Betrachtung verschiedener Metriken, einschließlich Erhöhen der Menge eines gemeinsamen Teilens unter Kontexten, die in Bezug auf Speicher-Ressourcen in Wettbewerb treten, oder Verringern von Konflikten unter Kontexten, die in Bezug auf Speicher-Ressourcen in Wettbewerb treten, Gebrauch machen.

[0079] Zum Beispiel macht es Sinn, bevorzugt ein Thread parallel ablaufmäßig zu planen, das einen großen Cache-Footprint besitzt, gleichzeitig mit einem Thread, das nur einen sehr geringen Gebrauch von dem Cache macht, da sich die Speichersystemanforderungen solcher Threads einander ergänzen, um dadurch ein gemeinsames Teilen zu erhöhen. Auch macht es Sinn, nicht überlappende Bereiche des Cache so umfangreich wie möglich zu verwenden.

[0080] Andererseits sollte die Betriebssystem-Software anstreben, eine parallele Ablaufplanung von zwei Threads mit großen Cache-Footprints dort, wo möglich, zu vermeiden, da dies zu viel mehr Konfliktfehlern führen wird, da die Threads nützliche Daten zueinander von dem Cache aussondern, um dadurch Konflikte zu verringern.

[0081] Durch Erfassen von alten und neuen Konflikt-Identifizierern als Teil eines Cache-Monitors kann die Betriebssystem-Software statistisch den Grad abschätzen, den unterschiedliche Kontexte gemeinsam teilen und in dem Cache in Konflikt treten. Diese Abschätzungen können zu einer Ablaufplanungseinrichtung zugeführt werden.

[0082] Eine vernünftige Ablaufplanung ist besonders wichtig für Multithreaded-Prozessoren, wo Speicherreferenzen von unterschiedlichen Hardware-Kontexten unter einem sehr feinkörnigen Niveau verschachtelt werden, und die Relevanz wird erhöht, wenn diese Kontexte gemeinsam Speichersystem-Ressourcen, wie beispielsweise Cache, teilen. Allerdings ist dies auch für Einzel-Threaded-Prozessoren wichtig, wenn die Cache groß genug relativ

zu der Anzahl von Speicher-Transaktionen sind, vorgenommen durch ein Thread während eines Ablaufplanungs-Quantums. Dann ist dabei eine gewisse Hoffnung vorhanden, einige nützliche Cache-Inhalte beizubehalten, wenn das nächste Quantum einem bestimmten Kontext zugeordnet wird. Alle diese Ablaufplanungs-Entscheidungen können sich dynamisch an ein Feedback anpassen, das von einer statistischen Abtastung von Speichersystem-Transaktionen während eines Online-Betriebs gesammelt werden.

[0083] Auf einer Teilung basierende oder auf einer proportionalen Teilung basierende Ablaufplanungs-Policen wollen Idealerweise jedem Kontext eine spezifizierte Teilung bzw. einen Anteil jedes Cache-Speichers in der Speicherhierarchie geben. Mit der vorliegenden Abtasttechnik ist es möglich, statistisch den Bereich eines Cache, der durch jeden Kontext belegt ist, abzuschätzen. Dies ermöglicht der Ablaufplanungs-Einrichtung, ihre Entscheidungen auf Metriken zu stützen, beispielsweise so, dass jedem Prozess ein spezifizierter Anteil der Speichersystem-Ressourcen gegeben wird, was effektiv Speichersystem-Ressourcen unter Kontexten im Verhältnis zu deren Erfordernissen unterteilt.

[0084] Um dies vorzunehmen, können Kontexte, die mehr als deren zugeordneten Anteil belegen, verlangsamt werden oder ausgesetzt werden. Während bestimmte Kontexte ausgesetzt werden, können andere ihren Anteil des Cache erhöhen. Dem ausgesetzten Kontext kann ermöglicht werden, fortzufahren, nachdem die Benutzung des Cache als Folge eines erhöhten Cache-Drucks von anderen, aktiven Kontexten verringert wurde. Dies kann gegenüber bekannten Maßnahmen unterschieden werden, die allgemein nicht zulassen, dass Informationen an der Cache-Leitung bzw. dem Block-Level überwacht werden, andere als durch Simulation.

[0085] Alle diese Ablaufplanungs-Entscheidungen können sich dynamisch an ein Feedback anpassen, gesammelt von einer statistischen Abtastung von Speichersystem-Transaktionen während eines Online-Betriebs.

[0086] Die Profilierungs-Hardware, die vorstehend beschrieben ist, kann in einer Vielfalt von unterschiedlichen Arten und Weisen verwendet werden. Da die vorliegende Technik sehr detaillierte Informationen über die Verarbeitung von individuellen Transaktionen, wie beispielsweise Instruktionen bzw. Befehlen, liefert, könnte eine Anwendung eine große Anzahl von Befehlen profilieren. Die Abtast-Informationen können in einem Speicherpuffer für eine spätere Verarbeitung durch Profilierungs-Tools gespeichert werden, um detaillierte Befehls-Level-Informationen zu erzeugen.

[0087] Die Informationen können dazu verwendet werden, zum Beispiel Histogramme von Lade-Latenzen für jeden Lade-Befehl, Histogramme von Befehlsausführzeiten, und vielleicht sogar eine leicht umfangreichere Analyse des Pipeline-Zustands für jeden Befehl, zu entwickeln. Da die Menge an Informationen, die durch diese Maßnahme geliefert wird, wahrscheinlich ziemlich hoch ist, ist das gesamte Speicher-Overhead der vorliegenden Technik auch wahrscheinlich sehr hoch, da eine wesentliche Menge eines Speicherverkehrs umfasst ist. Zum Beispiel wird, falls eine Billion Befehle pro Sekunde abgerufen werden, und ein Abtasten alle 10.000 abgerufene Befehle durchgeführt, wobei dann die Datenrate für die Profil-Informationen ungefähr 2,4 MB pro Sekunde sein wird.

[0088] Die nachfolgenden Abschnitte beschreiben Verfahren, die durch eine Software ausgeführt sind, zum Verringern einer Bandbreite durch Aggregieren von Profil-Informationen.

[0089] Der Umfang von abgetasteten Daten kann durch Ignorieren einiger Felder der Profil-Aufzeichnung reduziert werden, z. B. die Daten in dem Puffer **300**, mit Ausnahme dann, wenn sie explizit angefordert werden. Ein Benutzer des Systems **100** kann unterschiedliche Level einer Profilierung wünschen. In dem niedrigsten Overhead-Modus kann die Profilierungs-Anwendungs-Software einen Profil-Report für das gesamte oder einen Teil eines Programms erzeugen, unter Verwendung nur der PC- und Retire-Delay-Felder. In Abhängigkeit von der Optimierung, die durchgeführt werden soll, können andere Werte pro PC durch Mittelung, oder andere statistische Metriken, wie beispielsweise Minimum, Maximum, oder Berechnen einer Standardabweichung, zusammengefasst werden. Unter Zuteilung von mehr Zeit, um Daten zu verarbeiten, kann die Profilierungs-Anwendung Histogramme von verschiedenen Befehl-Latenzen erzeugen.

[0090] Die effektive Speicheradresse, die Verzweigungs-Target-Adresse und die Verzweigungs-Historik-Abtastungen werden wahrscheinlich eine kostenintensivere Verarbeitung erfordern als die anderen Felder. Diese Felder können wahrscheinlich ignoriert werden, mit der Ausnahme dann, wenn Daten gesammelt werden, um spezifische Optimierungsaufgaben durchzuführen. Unter Vorgabe eines Inter-Befehl-Abruf-Abstands zwischen Befehlen in Zyklen, kann die Profilierungsanwendung auch Informationen über Level einer Konkurrenz sammeln.

[0091] Ein Filtern der Profilierungs-Informationen kann auch durch Hardwareeinrichtungen vorgenommen werden, zum Beispiel durch ein Maskierungsregister, oder eine programmierbare Logik. Zum Beispiel nur eine Abtastung, wenn dort ein Cache-Fehler vorlag, oder wenn der Befehl ausgesondert wird,

oder andere, Bool'sche Kombinationen von Operationscoden, Operanden, Adressen, Ereignissen und Latenzen.

[0092] Die vorliegende Profilierungstechnik kann dazu verwendet werden, ein präzises Verständnis der internen Operation eines Out-Of-Order-Ausgabe-Prozessor, wie beispielsweise des Prozessors Alpha 21264, zu erhalten. Eines der ersten Dinge, die über diesen Typ einer Maschinen-Organisation anzumerken sind, ist dasjenige, dass dort viele Plätze vorhanden sind, wo ein Befehl in der Pipeline hängen bleiben könnte, und eine große Anzahl von Gründen, warum sie hängen bleiben könnten.

[0093] Zum Beispiel könnte ein Befehl hängen bleiben, da einige seiner Operanden nicht als Daten bereit sind, da einige der Ressourcen, die für die Ausführung des ausgewählten Befehls erforderlich sind, nicht verfügbar sind, oder da andere Befehle ausgewählt wurden, um sie davor auszuführen.

[0094] Eine exakte Bestimmung, wo ein Befehl stecken blieb, warum er stecken blieb und wie lange er stecken blieb, hängt stark von dem präzisen Zustand der Maschine ab, wenn dieser Befehl ausgeführt wird. Da der Prozessor so dynamisch ist, ist es schwierig für Software-Funktions-Tools, diesen Zustand statistisch zu bestimmen.

[0095] Die Profilierungstechnik, die hier beschrieben ist, kann auch dazu verwendet werden, eine N-weise Abtastung durchzuführen. Hierbei kann der dynamische Zustand von Interaktionen zwischen mehreren, gleichzeitig verarbeiteten Transaktionen erfasst werden. Anstelle eines Profilierens einer einzelnen Transaktion werden zwei oder mehr separate Transaktionen gleichzeitig profiliert. Der dynamische „Abstand" zwischen den ausgewählten Transaktionen kann als die Anzahl von Transaktionen gemessen werden, die einer Verarbeitung unterworfen sind, oder als die Anzahl von Taktzyklen, die die gepaarten Transaktionen „separieren".

[0096] Profil-Informationen für N-weise, abgetastete Befehle besitzen viele mögliche Verwendungen. Zuerst können die Informationen analysiert werden, um nützliche Konkurrenz-Level zu messen. Dies macht es möglich, wahre Engstellen zu lokalisieren. Wahre Engstellen sind durch lange Staus, verbunden mit einer niedrigen Konkurrenz, charakterisiert. N-weise Abtastungen können auch eine Pfad-Profilierung erleichtern und können Kandidaten-Ausführungs-Pfade durch Beschränken der Pfade eindeutig machen, um mindestens zwei Punkte entlang des Pfads zu umfassen. Weiterhin kann es von einer N-weisen Abtastung auch möglich sein, statistisch detaillierte Prozessor-Pipeline-Zustände zu rekonstruieren. Hierbei kann die Auswahl der Gruppe von Befehlen auf einer bestimmten Messung einer Ähnlichkeit zwischen den Befehlen basieren, zum Beispiel eine kürzliche Verzweigungs-Historik, Staus, Befehls-Typen, oder eine andere, kürzliche Zustands-Historik.

[0097] Ein genaues Aufzeigen von Funktions-Engstellen in Out-Of-Order-Prozessoren erfordert detaillierte Informationen über sowohl Stauzeit als auch Konkurrenz-Level. Im Gegensatz zu In-Order-Prozessoren ist ein Befehl mit langer Latenz nicht dann problematisch, wenn dabei eine ausreichende Konkurrenz vorhanden ist, um effektiv den Prozessor zu verwenden, während sich ein Befehl mit langer Latenz im Stau befindet.

[0098] Eine Maßnahme, um Konkurrenz-Informationen zu erhalten, ist diejenige, den gesamten Pipeline-Zustand als Speicherauszug zu führen. Dies wird direkt ergeben, wo sich Sätze von gleichzeitig ausführenden Befehlen in den Stufen der Pipeline zu einem gegebenen Zeitpunkt befinden. Allerdings könnte ein „Dumping" („Deponieren") des gesamten Pipeline-Zustands in Register und Puffer hinein extrem kostspielig sein, sowohl in Bezug auf die Zeit als auch in Bezug auf den Raum. Weiterhin können voluminöse Daten, die erzeugt sind, wahrscheinlich nicht effektiv aggregiert werden, um die Kosten einer Abtastung zu amortisieren. Noch schlechter ist dasjenige, dass diese Maßnahme tatsächlich unzureichend ist, da nur solche Befehle, die ausgesondert werden, als „nützlich" gezählt werden, und die Informationen, über die Befehle abgerufen sind, die allerdings nicht ausgesondert sind, sind bis dahin noch nicht bekannt.

[0099] Eine unterschiedliche Art einer Mehrfach-Weg-Abtastung kann auch dazu verwendet werden, die durchschnittliche Anzahl von Befehlen, verarbeitet durch die Pipeline, über eine Anzahl mit festgelegter Größe von Prozessor-Zyklen zu bestimmen.

[0100] Es ist auch möglich, abgetastete Zustands-Informationen zu verwenden, um Fälle, die von Interesse sind, zu identifizieren, während Konkurrenz-Informationen aggregiert werden. Zum Beispiel kann es nützlich sein, den durchschnittlichen Konkurrenz-Level zu berechnen, wenn eine Speicher-Zugriffs-Transaktion in einem der Cache „zutrifft" („hits"), und um dann den durchschnittlichen Konkurrenz-Level mit dem Fall zu vergleichen, wo ein Cache-Fehler aufgetreten ist. Andere interessante Aspekte, um eine Korrelation mit variierenden Konkurrenz-Leveln zu prüfen, umfassen Register abhängige Staus, Cache-Fehler-Staus, Verzweigungs-Fehlvorhersage-Staus, und eine kürzliche Verzweigungs-Historik.

[0101] Ein zusätzlicher Vorteil einer Profilierung eines Clusters von Befehlen ist die Fähigkeit, Pfad-Profile zu erhalten. Pfad-Profile sind für zahlreiche Compiler-Optimierungen, und eine Spur-Ablaufplanung,

nützlich.

[0102] Weiterhin werden, durch Beschränken von Mehrfach-Punkten entlang eines Ausführungs-Pfads eines Programms zusammen mit einer kürzlichen Verzweigung, genommen von einer Historik, Pfad-Profile eindeutig gemacht. Diese Eindeutigkeit verbessert die N-weise Abtastung; d. h. wenn sich N erhöht, verbessert sich eine Eindeutigkeit. Für einen schwer ausgeführten Code kann eine gleichzeitige Profilierung die relative Reihenfolge einer Ausführung von Befehlen an jeder Stufe einer Prozessor-Pipeline für alle ausführenden Befehle ergeben. Demzufolge kann man nun statistisch die tatsächliche Operation der Ausführungs-Pipeline in einem Operationssystem rekonstruieren.

[0103] Die letzte Generation von Mikroprozessoren setzt alle Tricks ein, die Computer-Architekten zulassen, um die höchstmögliche Leistung zu liefern. Diese Mikroprozessoren rufen mehrere Befehle pro Zyklus ab, geben sie aus und schöpfen sie aus. Zusätzlich führen diese Prozessoren Befehle Out-Of-Order aus. Einige davon führen sogar Speicher-Operationen Out-Of-Order aus.

[0104] Allerdings sind Funktions-Charakteristika sehr variabel, und zwar aufgrund der vielen, heuristischen Mechanismen, verwendet durch Prozessoren, die Befehle und Speicher-Operationen Out-Of-Order ausgeben. Als ein Vorteil ermöglichen die Profilierungstechniken, wie sie hier beschrieben sind, dem System, eine Funktion bzw. Leistung eines Programms in ausreichendem Detail so zu messen, dass die Leistung des Systems **100** automatisch verbessert werden kann.

[0105] Die vorliegenden Profilierungstechniken können dazu verwendet werden, eine Optimierung des Systems **100** durchzuführen. Die nachfolgenden Abschnitte sind dazu vorgesehen, Programmierer und durch einen Compiler angewiesene Optimierungen von Software-Programmen zu leiten.

[0106] Da Out-Of-Order-Superskalar-Mikroprozessoren Befehle entsprechend zu einer Daten- und Ressource-Verfügbarkeit umordnen, ist eine Zusammenstellungszeit einer Befehls-Ablaufplanung sehr viel wichtiger als dies für architekturmäßig einfachere Prozessoren der Fall ist. Hierbei liegen Hauptengstellen aufgrund eines Befehls-Abrufens und von Speicher-Operationen vor.

[0107] Genauer gesagt gehen Zyklen nicht in der Prozessor-Pipeline aufgrund einer Verzweigung oder aufgrund von Sprung-Fehlvorhersagen, von On-Chip-Cache-Fehlern und von TLB-Fehlern verloren. Dies sind schwierige, wenn nicht sogar unmögliche, Zustände, um sie statistisch abzuleiten. Zyklen gehen auch für Verzögerungen in Of-Chip-Operationen auf höherem Level, aufgrund von Cache-Fehlern, Ressource-Störstellen und Ordnungs-Störstellen, verloren. Verlorene Zyklen verschwenden Zeit.

[0108] Mit herkömmlichen Ereignis-Fehlern kann man die Aggregat-Zahl dieser Funktions-Entkräftungs-Ereignisse messen, allerdings ist dies extrem schwierig, falls nicht sogar unmöglich, um verlorene Zyklen einem bestimmten Befehl in dem Programm zuzuschreiben.

[0109] Die Profilierungstechnik, wie sie hier beschrieben ist, ermöglicht einem Benutzer, Haupt-Leistungs-Probleme und Korrelations-Probleme in Bezug auf die spezifischen Befehle zu messen.

[0110] Eine Front-End-Optimierung, die eine Leistung unterstützt, ist die Umordnung von Befehlen in Basis-Blöcken und Basis-Blöcken in Prozeduren. Ein Basis-Block ist als ein Satz von Befehlen definiert, die linear als eine Einheit ausgeführt werden, oder nicht auf einmal. Prozeduren sind allgemein ein kohäsiver Satz von Basis-Blöcken, erreicht über Aufruf-Befehle. Prozeduren können mehrere Basis-Blöcke umfassen. Eine Umordnung von Befehlen in Basis-Blöcken und von Basis-Blöcken in Prozeduren kann den Ausführungsfluss und Datenzugriffe ändern, um Seiten- und Cache-Temporär-Lokalitäten zu optimieren, und um die Anzahl von Verzweigungen zu verringern. Verzweigungen verschwenden Zyklen, da sie nur den Ausführungsfluss umleiten, und keine nützliche Arbeit in Bezug auf die Daten vornehmen. Diese Optimierung, als Eingabe, muss Kontroll-Fluss-Graphik-Kanten-Frequenzen kennen.

[0111] Ähnlich muss, um eine Spur-Ablaufplanung von Befehlen durchzuführen, ein Compiler eine Graphik-Kante und Pfad-Häufigkeiten im Fluss kontrollieren. Eine Spur-Ablaufplanungseinrichtung könnte eine noch bessere Arbeit dann vornehmen, wenn sie abschätzen muss, wie lange es benötigt, um jeden Basis-Block oder einen größeren Ausführungs-Pfad auszuführen. Für Betriebssysteme im großen Maßstab, wie beispielsweise die Suchmaschine Alta Vista, ist dies schwierig mit traditionellen Tools in einer Realzeit zu messen.

[0112] Viele Compiler-Optimierungen, wie beispielsweise eine Spur-Ablaufplanung und eine Hot-Cold-Optimierung, beruhen auf einer Erkenntnis, wie Ausführungs-Pfade häufig durch ein Programm genommen werden. Diese werden als „heiße" („hot") Pfade bezeichnet. Bis kürzlich wurden häufig ausgeführte Pfade durch Profilieren des Programms, entweder über eine Instrumentierung oder eine Simulation, geleitet, um Basis-Block- oder Kanten-Zählungen zu sammeln, und dann, unter Verwendung dieser Zählungen, um direkt heiße (hot) oder kalte (cold) Pfade abzuleiten.

[0113] In neuerer Zeit sind Techniken verwendet worden, um Pfad-Informationen direkt zu sammeln. Obwohl diese Techniken exakte Pfad-Informationen liefern, tendieren sie auch dazu, dass sie einen ziemlich hohen Overhead haben, was sie ungeeignet macht, um aktive Computersysteme in großem Maßstab zu messen. Mit der vorliegenden Profilierung können Pfad-Informationen unter Zufall erfasst werden, mit einem minimalen Overhead, und sie geben dennoch eine statistisch korrekte Ansicht über die tatsächlichen Ausführungsabläufe.

[0114] Die meisten modernen Mikroprozessoren protokollieren die Richtungen der letzten N Verzweigungen in einem globalen Verzweigungs-Historik-Register. Das Verzweigungs-Historik-Register kann, als ein sich bewegendes Fenster, dazu verwendet werden, die kürzlichen Verzweigungs-Vorhersagen anzusehen, und können ein zukünftiges Befehls-Abrufen entsprechend beeinflussen. Durch Erfassen der Inhalte dieses Registers zu einer Befehls-Abrufzeit, zusammen mit dem PC des Befehls, der abgetastet werden soll, ist es manchmal möglich, eine statistische Analyse einer Steuer-Ablauf-Graphik zu verwenden, um den exakten Pfad durch die letzten N Verzweigungen zu hypothetisieren, die der Prozessor genommen haben muss.

[0115] Allerdings können, da herkömmliche Historik-Register gewöhnlich nur die Richtungen der Verzweigungen, und nicht die tatsächlichen Ziel-Bestimmungen, enthalten, die Informationen unpräzise sein. Insbesondere können Übergänge in Steuer-Abläufe Unstimmigkeiten beim Identifizieren von tatsächlichen Pfaden, die genommen sind, hervorrufen.

[0116] Auch können asynchrone Ereignisse, die einen verzweigten Code verursachten, um ihn auszuführen, wie beispielsweise Unterbrechungen oder Kontext-Umschaltungen, die Verzweigungs-Historik-Bits belasten. Allerdings sollte dieses Ereignis nicht relativ häufig sein, und dessen Auftreten in einem Betriebssystem sollte zufällig über den Code verteilt sein. Da das Ziel dasjenige ist, Pfade mit einer hohen Häufigkeit zu identifizieren, können Pfade mit einer geringen Häufigkeit, umfassend solche, die durch „mit Rausch behaftete" Verzweigungs-Historik-Bits erzeugt sind, erzeugt durch nicht vorhersagbare, asynchrone Ereignisse, ignoriert werden.

[0117] Andere Informationen über eine kürzliche Ausführungs-Historik des Prozesses können auch dabei unterstützen, den Ausführungs-Pfad, der genommen worden ist, um einen bestimmten Befehl zu erhalten, zu identifizieren. Ein Teil von Informationen, der nützlich ist, ist die Kenntnis eines zweiten PC-Werts eines Befehls, der kürzlich vorher ausgeführt wurde. Unter Verwendung von Mehrfach-PC-Werten, vielleicht mit einer N-weisen Abtastung, können Pfade, die nur einen PC umfassen, eliminiert werden.

[0118] Hohe Fehlraten in Cache oder Translations-Look-Aside-Puffern (TLBs) können wesentlich die Leistung des Systems herabsetzen. Maßnahmen nach dem Stand der Technik haben allgemein auf entweder einer spezifizierten Hardware oder spezialisierten Software-Schemata zum Sammeln von Cache-Fehl-Adressen, wie beispielsweise periodisch Entleeren des TLB, beruht. Diese beobachteten Fehl-Muster vermitteln ein geeignetes Verständnis über die häufig zugegriffenen oder „heißen" (hot) Seiten, die verwendet werden können, um virtuell-zu-physikalische Seitenauflistungspolicen zu beeinflussen. Allerdings können Adressen-Informationen, die dazu notwendig sind, eine Analyse abzuschließen, nicht zu dem Zeitpunkt verfügbar sein, zu dem das Ereignis erfasst ist.

[0119] Eine wichtige Aufgabe, durchgeführt während einer Code-Optimierung, ist eine ideale Befehl-Ablaufplanung. Eine ideale Befehl-Ablaufplanung zeichnet einen Code auf, um Verzögerungen aufgrund von Speicher-Latenzen zu minimieren. Obwohl eine statische Reihenfolge von benachbarten Befehlen in einem Basis-Block weniger wichtig ist, als dies für die vorherige Generation von In-Order-RISC-Prozessoren war, ist eine makroskopische Befehl-Ablaufplanung viel wichtiger in Out-Of-Order-Prozessoren.

[0120] Ein sehr wichtiger Aspekt einer Befehl-Ablaufplanung ist die Ablaufplanung von Belastungen und Speicherungen. Dies ist der Fall, da statische Ablaufplanungseinrichtungen nicht immer exakte Abhängigkeits-Informationen haben, die ihnen ermöglichen würden, eine Ablaufplanung der Speicher-Zugriff-Befehle zu optimieren. Zusätzlich ist es sehr schwierig, exakt die Latenz von Speicher-Zugriff-Befehlen vorherzusagen. Da Befehl-Ablaufplanungseinrichtungen gewöhnlich präzise Informationen über Speicher-Zugriffe fehlen, planen sie allgemein Belastungen und Speicherungen unter Verwendung von D-Chache-Treffern bzw. -Hits. Zum Beispiel versucht eine ausbalancierte Ablaufplanung, eine solche Ablaufplanung zu erzeugen, die eine gleiche Menge einer Latenz pro Last bzw. Beladung umfasst. Dies ist eine Verbesserung gegenüber demjenigen, dass immer angenommen wird, dass Lade/Speicher-Operationen immer in dem Cache ankommen werden.

[0121] Falls man Lade- und Speicher-Latenzen über eine zufällige Abtastung sammelt, kann man jeden Befehl entsprechend seinem Histogramm von Latenzen ablaufmäßig planen. Die vorliegende Technik kann dazu verwendet werden, Optimierungen durch Sammeln von Latenz-Informationen ohne Übernahme der Kosten einer vollständigen Cache-Simulation vorzunehmen.

[0122] Fig. 4 stellt die Schritte **401**–**404** zum Abtasten der Leistung irgendeiner oder aller der Funktionseinheiten **111**–**114** des Computersystems **100** dar. Die Abtastungen werden über die Zeit aggregiert bzw. gesammelt und dann analysiert. Die Analyse wird verwendet, um dynamisch das System zu aktualisieren, während es arbeitet. Die Abtastung kann für das optimierte System über die Zeit so fortgeführt werden, dass das System optimal „abgestimmt" bleibt, falls eine Belastung variiert.

[0123] Die vorstehende Beschreibung ist auf spezifische Ausführungsformen gerichtet worden. Es wird für Fachleute auf dem betreffenden Fachgebiet ersichtlich werden, dass Modifikationen an den beschriebenen Ausführungsformen unter Erreichen aller oder einiger der Vorteile vorgenommen werden können.

Patentansprüche

1. System zum Optimieren der Leistung eines Computersystems (**100**), wobei das Computersystem eine Vielzahl von Funktionseinheiten (**111**–**114**) umfasst und das System zum Optimieren umfasst:
eine Quelle (**110**) zum Bereitstellen von Transaktionen (**111**), die durch das Computersystem abzuarbeiten sind;
eine Einrichtung (**200**) zum Auswählen von Transaktionen zum Abtasten, die durch eine Vielzahl von Funktionseinheiten des Computersystems abzuarbeiten sind;
einen Prozessor (**111**) und Software zum Analysieren der Zustandsinformationen; gekennzeichnet
dadurch, dass die Transaktionen gleichzeitig von den funktionellen Einheiten abgetastet werden; und
durch Pufferspeicher (**300**), die Zustandsinformationen (**130**) speichern, während die ausgewählten Transaktionen (**103**) durch die Funktionseinheiten abgearbeitet werden, wobei die Zustandsinformationen wenigstens eine Adressen-Information (**320**), eine Transaktionsquellen-Information (**340**) und eine Latenz-Information (**360**) enthalten,
wobei die Pufferspeicher (**200**) Zustandsinformationen im Verlauf der Zeit als eine Ansammlung von Zustandsinformationen speichern und die Ansammlung von Zustandsinformationen analysiert wird, um Leistungsstatistiken des Computersystems (**110**) zu schätzen, wobei die Leistungsstatistiken verwendet werden, um die Leistung des Computersystems (**110**) dynamisch zu optimieren, während das Computersystem arbeitet.

2. System nach Anspruch 1, wobei die Auswähleinrichtung (**230**) eine Vielzahl gleichzeitig abgearbeiteter Transaktionen (**101**) auswählt.

3. System nach Anspruch 1 oder Anspruch 2, wobei die Transaktionen (**101**) Befehle sind, die durch den Prozessor (**111**) ausgeführt werden, und der Prozessor eine der Vielzahl von Funktionseinheiten ist.

4. System nach Anspruch 1 oder Anspruch 2, wobei die Transaktionen (**101**) Datenzugriffe sind, die einem Speicher-Teilsystem (**112**) bereitgestellt werden, wobei das Speicher-Teilsystem eine der Vielzahl von Funktionseinheiten ist.

5. System nach Anspruch 1 oder Anspruch 2, wobei die Transaktionen (**101**) Netzwerk-Meldungen sind, die durch eine Netzwerksteuerung (**114**) abgearbeitet werden, wobei die Netzwerksteuerung eine der Vielzahl von Funktionseinheiten ist.

6. System nach einem der vorangehenden Ansprüche, wobei die Zustandsinformationen (**130**) in dem Speicher gespeichert werden, bevor und nachdem die ausgewählten Transaktionen (**103**) analysiert werden.

7. System nach Anspruch 1, wobei die Zustandsinformationen (**130**) gelesen und gespeichert werden, nachdem eine vorgegebene Anzahl ausgewählter Transaktionen (**101**) durch die Auswähleinrichtung (**200**) ausgewählt worden ist.

8. System nach einem der vorangehenden Ansprüche, wobei die Funktionseinheiten (**111**–**114**) Prozessoren (**111**), Speicher (**112**), Ein-/Ausgabe-Schnittstellen (**113**) und Netzwerksteuerungen (**114**) enthalten.

9. System nach einem der vorangehenden Ansprüche, wobei die Einrichtung (**200**) zum Auswählen ein Selektor ist, der eine Einrichtung zum Steuern der Abtastfrequenz des Selektors aufweist, und der Selektor gleichzeitig Transaktionen auswählt, die ausführbare Befehle und Speicherbezugsbefehle einschließen, die auszuführen sind, um die Datenflussrate in den Speicher und aus ihm abzutasten; und wobei die Datenflussraten in den Speicher und aus ihm analysiert werden, um Optimierung des Computersystems zu ermöglichen.

10. Verfahren zum Optimieren der Leistung eines Computersystems (**110**) mit einer Vielzahl von Funktionseinheiten (**111**–**114**), wobei das Verfahren die folgenden Schritte umfasst:
Bereitstellen von Transaktionen (**111**) von einer Quelle (**110**), die durch das Computersystem abzuarbeiten sind;
Auswählen von durch eine Vielzahl von Funktionseinheiten des Computersystems abzuarbeitenden Transaktionen, wobei die Transaktionen gleichzeitig durch die Funktionseinheiten abgetastet werden;
Speichern von Zustandsinformationen (**130**) als eine Ansammlung von Zustandsinformationen im Verlauf der Zeit in Pufferspeichern (**300**), während die ausgewählten Transaktionen durch die Funktionseinheiten abgearbeitet werden, wobei die Zustandsinformatio-

nen wenigstens eine Adressen-Information (**320**), eine Transaktionsquellen-Information (**340**) und eine Latenz-Information (**360**) enthalten; und

Analysieren der Ansammlung von Zustandsinformationen unter Verwendung eines Prozessors (**111**) und von Software, um Leistungsstatistiken des Computersystems zu schätzen, wobei die Leistungsstatistiken verwendet werden, um die Leistung des Computersystem (**110**) dynamisch zu optimieren, während das Computersystem arbeitet.

Es folgen 4 Blatt Zeichnungen

Anhängende Zeichnungen

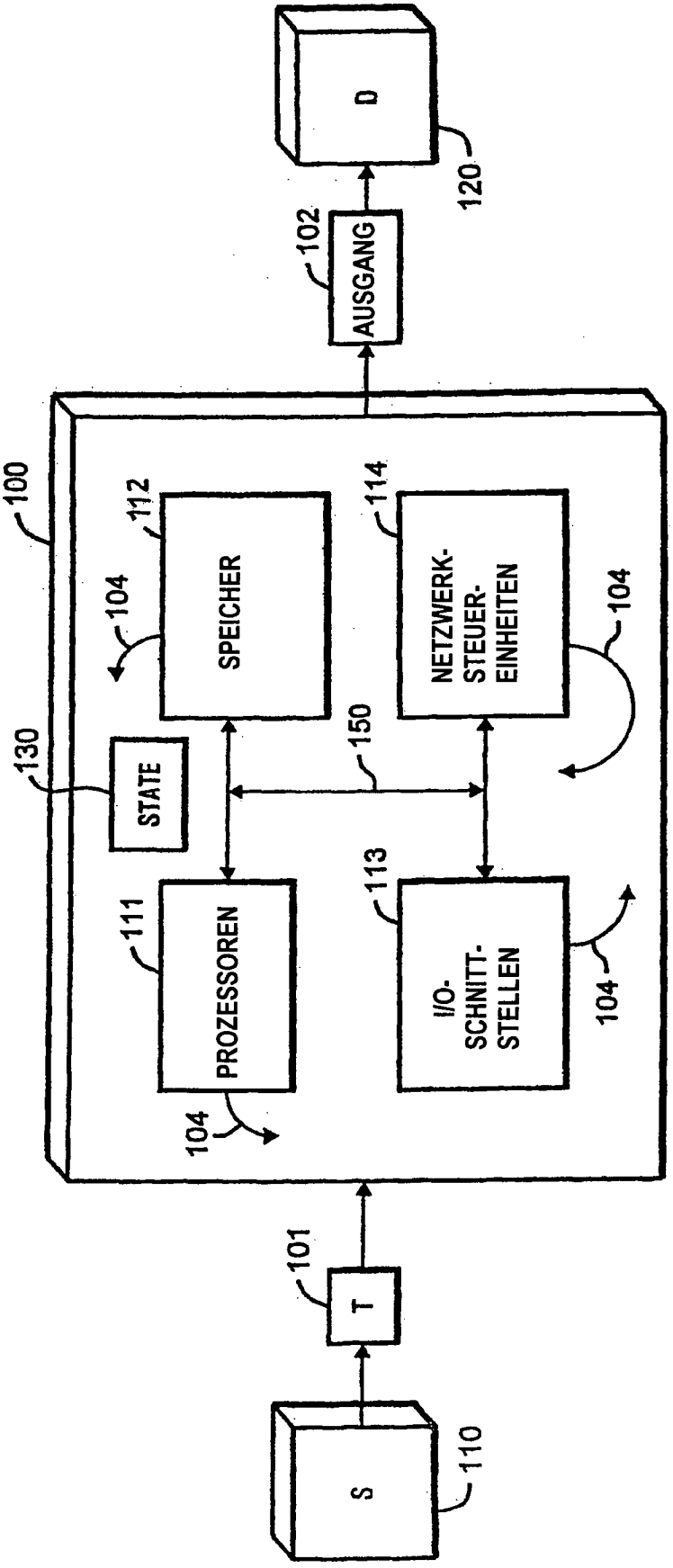

FIG. 1

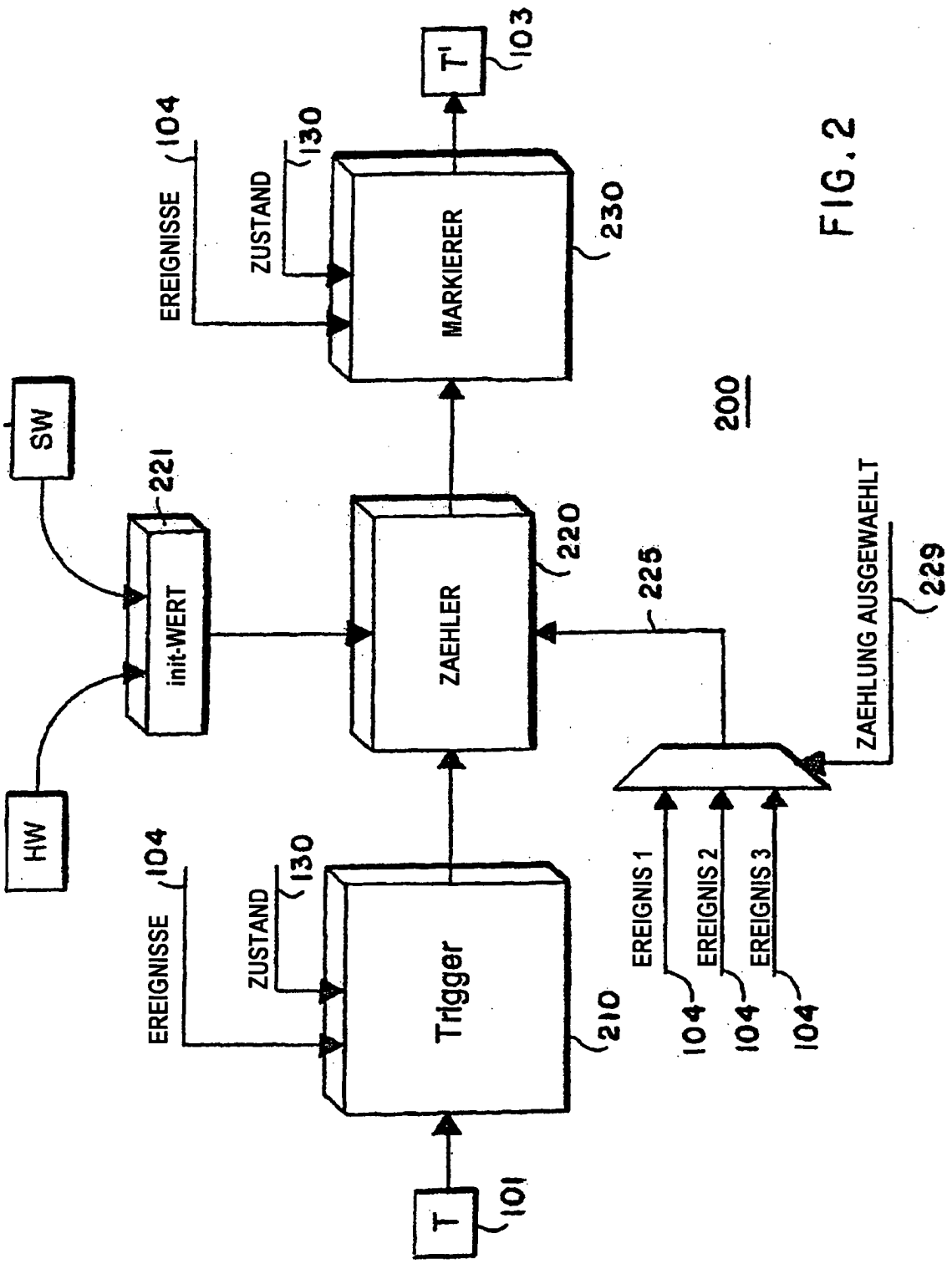
T
101
Trigger
210
EREIGNISSE
104
ZUSTAND
130
HW
SW
init-WERT
221
ZAEHLER
220
MARKIERER
230
T'
103
EREIGNISSE
104
ZUSTAND
130
225
ZAEHLUNG AUSGEWAEHLT
229
EREIGNIS 1
EREIGNIS 2
EREIGNIS 3
104
104
104
200

FIG. 2

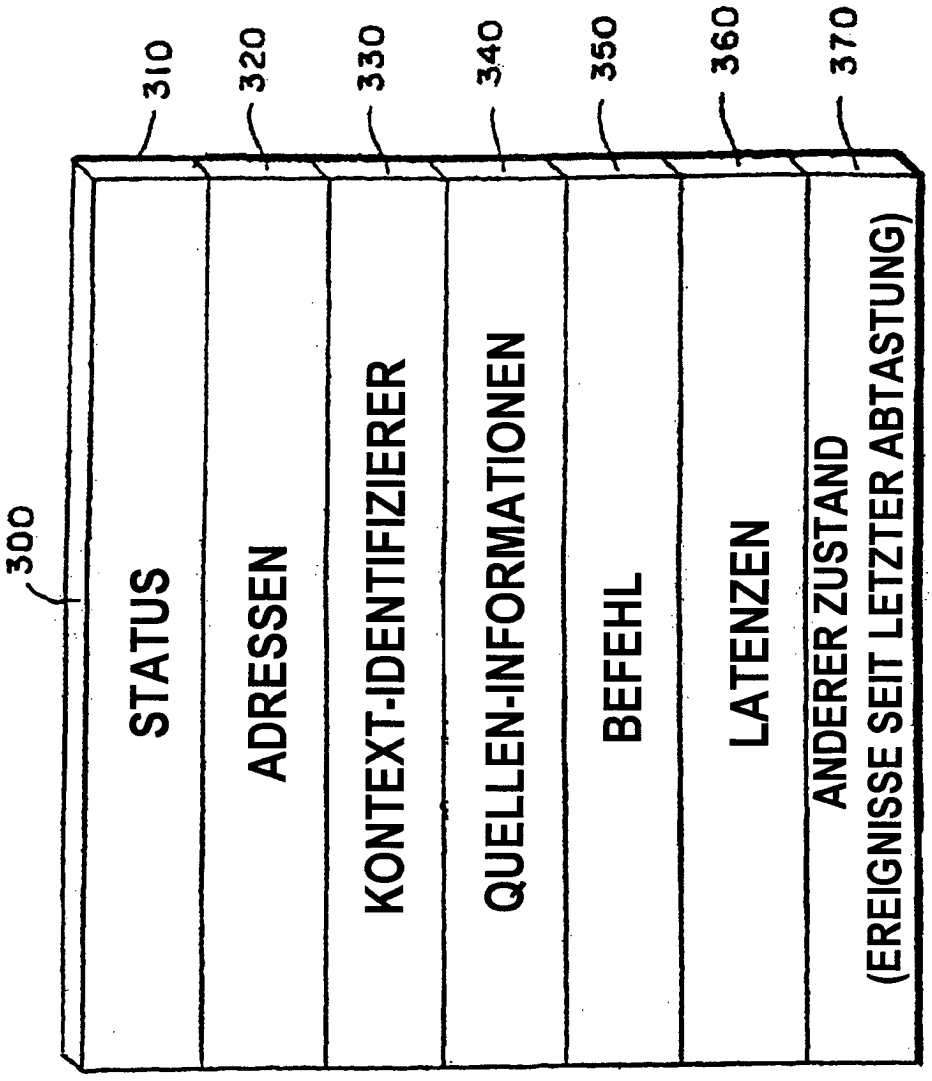
300
310
320
330
340
350
360
370
STATUS
ADRESSEN
KONTEXT-IDENTIFIZIERER
QUELLEN-INFORMATIONEN
BEFEHL
LATENZEN
ANDERER ZUSTAND
(EREIGNISSE SEIT LETZTER ABTASTUNG)

FIG. 3

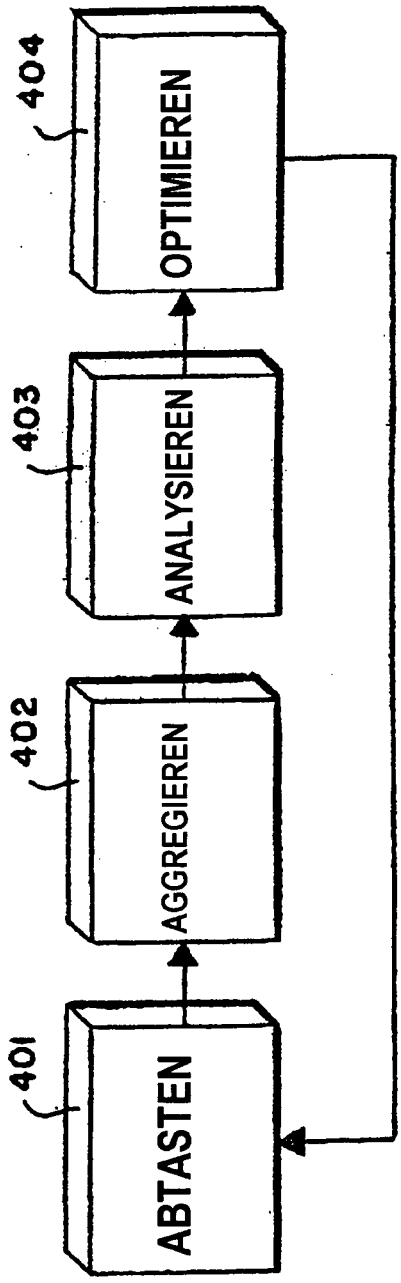
401
ABTASTEN
402
AGGREGIEREN
403
ANALYSIEREN
404
OPTIMIEREN

FIG. 4