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

(10) **DE 601 32 132 T2** 2009.01.02

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 162 542 B1**
(21) Deutsches Aktenzeichen: **601 32 132.4**
(96) Europäisches Aktenzeichen: **01 304 340.1**
(96) Europäischer Anmeldetag: **16.05.2001**
(97) Erstveröffentlichung durch das EPA: **12.12.2001**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **02.01.2008**
(47) Veröffentlichungstag im Patentblatt: **02.01.2009**

(51) Int Cl.[8]: ***G06F 12/08*** (2006.01)

(30) Unionspriorität:
591918 **09.06.2000** **US**

(84) Benannte Vertragsstaaten:
DE, FR, GB

(73) Patentinhaber:
Agere Systems Guardian Corp., Orlando, Fla., US

(74) Vertreter:
Klunker, Schmitt-Nilson, Hirsch, 80797 München

(72) Erfinder:
Kaxiras, Stefanos, Jersey City, New Jersey 07310, US; Young, Clifford Reginald, New York, NY 10023, US

(54) Bezeichnung: **Verzeichnis-basiertes Vorhersageverfahren und -einrichtung für Multiprozessorsysteme mit gemeinsamem Speicher**

Beschreibung

GEBIET DER ERFINDUNG

[0001] Die vorliegende Erfindung bezieht sich allgemein auf Multiprozessorcomputer und andere Typen von Verarbeitungssystemen, die mehrere Prozessoren aufweisen, und insbesondere auf Speicherprognosetechniken, die für die Verwendung in solchen Systemen geeignet sind.

HINTERGRUND DER ERFINDUNG

[0002] In einem Multiprozessorsystem mit gemeinsam genutztem Speicher erscheint es einem Benutzer, dass alle Prozessoren den Status in einem einzelnen gemeinsam genutzten Speicher einer Speichereinrichtung lesen und modifizieren. Eine wesentliche Schwierigkeit beim Implementieren eines solchen Systems und insbesondere einer verteilten Version eines solchen Systems ist die Verbreitung von Werten von einem Prozessor zu einem anderen, da die eigentlichen Werte in der Nähe eines Prozessors geschaffen werden, aber von vielen anderen Prozessoren in dem System verwendet werden können. Wenn die Implementierung die Muster gemeinsamer Nutzung eines gegebenen Programms genau prognostizieren könnte, könnten die Prozessorknoten eines verteilten Multiprozessorsystems mehr ihrer Zeit zum Rechnen und weniger ihrer Zeit zum Warten auf das Abrufen von Werten von entfernten Standorten verbringen. Trotz der Entwicklung von Prozessormerkmalen wie z. B. nicht-blockierender Caches und der Ausführung von Out-of-Order-Anweisungen bleibt die relativ lange Zugriffslatenz in einem verteilten System mit gemeinsam genutztem Speicher eine ernsthafte Beeinträchtigung der Leistungsfähigkeit.

[0003] Es wurden Prognosetechniken verwendet, um durch den Versuch, Daten so früh wie möglich von ihrem Schaffungspunkt zu ihren erwarteten Verwendungspunkten zu bewegen, Zugriffslatenz in verteilten Systemen mit gemeinsam genutztem Speicher zu reduzieren. Diese Prognosetechniken ergänzen typischerweise das Standardkohärenzprotokoll bei gemeinsam genutztem Speicher, das in erster Linie mit korrektem Betrieb und in zweiter Linie mit Leistungsfähigkeit befasst ist. In einem verteilten System mit gemeinsam genutztem Speicher hält das typischerweise auf einem Verzeichnis basierende Kohärenzprotokoll Prozessor-Caches kohärent und überträgt Daten zwischen den Prozessorknoten. Im Wesentlichen führt das Kohärenzprotokoll die gesamte Kommunikation im System aus. Kohärenzprotokolle können gemeinsam genutzte Kopien eines Datenblocks bei jedem Schreiben des Datenblocks entweder für ungültig erklären oder aktualisieren. Die Aktualisierung beinhaltet das Weiterleiten von Daten von Erzeugerknoten an Verbraucherknoten, aber schafft keinen Rückkopplungsmechanismus, um den Nutzen des Weiterleitens von Daten zu bestimmen. Die Ungültigkeitserklärung schafft dadurch einen natürlichen Rückkopplungsmechanismus, dass für ungültig erklärte Leser die Daten verwendet haben müssen, aber die Ungültigkeitserklärung schafft kein Mittel, um Daten an ihren Zielort weiterzuleiten.

[0004] Eine in S. S. Mukherjee und M. D. Hill, „Using Prediction to Accelerate Coherence Protocols", Proceedings of the 25th Annual International Symposium an Computer Architecture (ISCA) Juni-Juli 1998, beschriebene herkömmliche Herangehensweise an die Prognose verwendet adressenbasierte 2-Ebenen-Prädiktoren an den Verzeichnissen und Caches der Prozessorknoten eines Multiprozessorsystems, um Kohärenznachrichten zu verfolgen und zu prognostizieren. A. Lai und B. Falsafi, "Memory Sharing Predictor: The Key to a Speculative Coherent DSM", Proceedings of the 26th Annual ISCA, Mai 1999, beschreiben, wie diese 2-Ebenen-Prädiktoren so modifiziert werden können, dass sie durch das Zusammenführen von Nachrichten von verschiedenen Knoten in Bitmaps weniger Platz verwenden, und zeigen, wie die modifizierten Prädiktoren verwendet werden können, um das Lesen von Daten zu beschleunigen. Eine andere Reihe von bekannten Prognosetechniken, beschrieben in S. Kaxiras und J. R. Goodman, "Improving CC-NUMA Performance Using Instruction-Based Prediction", Proceedings of the 5th Annual IEEE Symposium an High-Performance Computer Architecture (HPCA), Januar 1999, schafft eine anweisungsbasierte Prognose für migratorische gemeinsame Nutzung, breite gemeinsame Nutzung und gemeinsame Erzeuger-Verbraucher-Nutzung. Da es weitaus weniger statische Anweisungen als Datenblocks gibt, erfordern anweisungsbasierte Prädiktoren weniger Platz zum Erfassen von Mustern gemeinsamer Nutzung.

[0005] Trotz der durch die oben identifizierten Prognosetechniken geschaffenen Fortschritte bleibt ein Bedarf an zusätzlichen Verbesserungen bestehen, um Zugriffslatenz weiter zu reduzieren und die Implementierung von Multiprozessorsystemen mit gemeinsam genutztem Speicher dadurch zu erleichtern.

ZUSAMMENFASSUNG DER ERFINDUNG

[0006] Die Erfindung schafft verbesserte Techniken zum Bestimmen eines Satzes von prognostizierten Lesern eines Datenblocks, der Gegenstand einer Schreibanforderung ist, in einem Multiprozessorsystem mit gemeinsam genutztem Speicher. Gemäß einem Aspekt der Erfindung wird ein momentaner Satz von Lesern des Datenblocks bestimmt und wird der Satz von prognostizierten Lesern dann auf der Basis des momentanen Satzes von Lesern und mindestens eines zusätzlichen Satzes von Lesern erzeugt, der für wenigstens einen Teil einer globalen Vorgeschichte eines mit dem Datenblock assoziierten Verzeich-

nisses repräsentativ ist. In einer möglichen Implementierung wird der Satz von prognostizierten Lesern durch das Anwenden einer Funktion auf den momentanen Satz von Lesern und auf einen oder mehrere zusätzliche Sätze von Lesern erzeugt. Die Funktion kann zum Beispiel eine Vereinigungsfunktion, eine Schnittmengenfunktion oder eine musterbasierte Funktion sein und das Verzeichnis und der Datenblock können Elemente eines mit einem bestimmten Prozessorknoten des Multiprozessorsystems assoziierten Speichers sein.

[0007] Die globale Vorgeschichte des Verzeichnisses weist mehrere Sätze von vorhergehenden Lesern auf, die von dem Verzeichnis verarbeitet wurden, wobei die Gesamtanzahl von Sätzen von prognostizierten Lesern einer designierten Vorgeschichtstiefe entspricht, die mit der Erzeugung des Satzes von prognostizierten Lesern assoziiert ist. Die globale Vorgeschichte kann zum Beispiel in einem Schieberegister mit einer Anzahl von Speicherorten geführt werden, die der bestimmten Vorgeschichtstiefe entspricht. Die Vorgeschichtstiefe wird vorzugsweise als ein Wert ausgewählt, der größer ist als zwei, wie z. B. vier.

[0008] Im Betrieb sendet das Verzeichnis oder ein anderes Prozessorknotenelement, das mit dem Datenblock assoziiert ist, der Gegenstand der Schreibanforderung ist, eine Ungültigkeitserklärungsanforderung an jeden der Leser in dem momentanen Satz von Lesern und sendet auf den Empfang einer Ungültigkeitserklärungsbestätigung von jedem der Leser in dem momentanen Satz von Lesern hin eine gültige Kopie des Datenblocks an einen Schreiber, der die Schreibanforderung erzeugt hat. Jeder Leser in dem System kann ein Bit, auf das zugegriffen wurde, für jeden einer Anzahl von Datenblöcken aufrecht erhalten, wobei das Bit, auf das zugegriffen wurde, eines bestimmten Lesers für einen gegebenen Datenblock anzeigt, ob der bestimmte Leser den gegebenen Datenblock tatsächlich gelesen hat. Die Information des Bits, auf das zugegriffen wurde, kann von dem bestimmten Leser in Verbindung mit einer Ungültigkeitserklärungsbestätigung an das Verzeichnis gesendet werden. Nachdem das angeforderte Schreiben auf den Datenblock vervollständigt ist, wird der resultierende Datenblock an jeden der Leser in dem Satz von prognostizierten Lesern gesendet.

[0009] Gemäß einem anderen Aspekt der Erfindung kann die oben genannte Funktion dynamisch ausgewählt werden. Zum Beispiel kann die Funktion auf einer Pro-Programm-Basis ausgewählt werden, so dass jedes einer Anzahl von Programmen, die in dem Multiprozessorsystem laufen, unabhängig die Funktion bestimmt, die anzuwenden ist, um den Satz von prognostizierten Lesern zu bestimmen. Als ein anderes Beispiel kann die Funktion unter Programmsteuerung zur Laufzeit von einem gegebenen Programm ausgewählt werden, das auf dem Multiprozessorsystem läuft. Als ein weiteres Beispiel kann die Funktion auf einer Pro-Seite-Basis ausgewählt werden, so dass die angewendete Funktion für jede einer Anzahl von Speicherseiten, von denen jede mehrere Datenblöcke enthalten kann, unabhängig bestimmt werden kann. Als noch ein anderes Beispiel kann die Funktion wenigstens teilweise auf der Basis von Information bezüglich der Netzwerknutzung ausgewählt werden. Verschiedene Kombinationen dieser und anderer Typen von Information können auch bei der dynamischen Auswahl der oben genannten Funktion verwendet werden.

[0010] Der Prognoseprozess gemäß der vorliegenden Erfindung kann in Verbindung mit der oben beschriebenen Verzeichnisinformation zusätzliche Information, wie z. B. einen designierten Teilsatz der Cache-Adressinformation, Prozessorknotenidentifikationsinformation oder Programmzählerinformation, verwenden.

[0011] Vorteilhafterweise schaffen die Prognosetechniken der vorliegenden Erfindung im Vergleich zu herkömmlichen Techniken eine verbesserte Prognosegenauigkeit bezüglich sowohl weniger falscher Positive als auch weniger falscher Negative.

[0012] Diese und andere Merkmale und Vorteile der vorliegenden Erfindung werden aus den begleitenden Zeichnungen und der folgenden detaillierten Beschreibung offensichtlicher.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

[0013] Fig. 1 und Fig. 2 stellen die Arbeitsweise eines verteilten Multiprozessorsystems mit gemeinsam genutztem Speicher dar, in dem ein verzeichnisbasierter Prädiktor gemäß der vorliegenden Erfindung implementiert werden kann.

[0014] Fig. 3 zeigt ein Beispiel für eine Abfolge von Ereignissen und eine Aggregierung von Lesern.

[0015] Fig. 4 zeigt ein Beispiel für eine verzeichnisbasierte Prognose gemäß der Erfindung.

[0016] Fig. 5 ist ein Ablaufdiagramm eines verzeichnisbasierten Prognoseprozesses gemäß der Erfindung.

[0017] Fig. 6 zeigt einen Satz von Tabellen, die Beispiele für Prädiktoren gemäß der Erfindung auflisten.

DETAILLIERTE BESCHREIBUNG DER ERFINDUNG

[0018] Die Erfindung wird hierin in Verbindung mit beispielhaften verteilten Multiprozessorsystemen mit gemeinsam genutztem Speicher erläutert. Es sollte

jedoch zu verstehen sein, dass die Erfindung allgemeiner auf jegliches Multiprozessorsystem mit gemeinsam genutztem Speicher anwendbar ist, in dem es wünschenswert ist, durch die Verwendung von verzeichnisbasierter Prognose eine verbesserte Leistungsfähigkeit zu liefern. Der Begriff "Multiprozessorsystem", wie hierin verwendet, soll jegliche Vorrichtung beinhalten, in der abgerufene Anweisungen unter Verwendung von einem oder mehreren Prozessoren ausgeführt werden. Beispielhafte Prozessoren gemäß der Erfindung können zum Beispiel Mikroprozessoren, zentrale Verarbeitungseinheiten (CPUs), Prozessoren mit sehr langem Befehlswort (VLIW), Single-Issue-Prozessoren, Multi-Issue-Prozessoren, Digitalsignalprozessoren, anwendungsspezifische integrierte Schaltungen (ASICs), Personalcomputer, Mainframecomputer, Netzwerkcomputer, Arbeitsplatzrechner und Server und andere Typen von Datenverarbeitungsvorrichtungen sowie auch Teile und Kombinationen dieser und anderer Vorrichtungen umfassen.

[0019] Fig. 1 und Fig. 2 stellen die Handhabung beispielhafter Lese- bzw. Schreibanforderungen in einem verteilten Multiprozessorsystem mit gemeinsam genutztem Speicher **100** dar. Das System **100** ist ein Beispiel für einen Typ von System, in dem die verzeichnisbasierte Prognose der vorliegenden Erfindung implementiert werden kann. Das System **100** weist Knoten A, B und C auf, die über entsprechende Netzwerkschnittstellen (NIs) **104A**, **104B** bzw. **104C** mit einem Verbindungsnetzwerk **102** verbunden sind. Die Knoten A, B und C weisen entsprechende, wie gezeigt angeordnete Prozessoren **106A**, **106B** und **106C**, Speicher **108A**, **108B** und **108C** und Busse **110A**, **110B** und **110C** auf. Innerhalb eines gegebenen Knotens i des Systems **100**, wobei i = A, B, C, sind der Prozessor **106i**, der Speicher **108i** und die Netzwerkschnittstelle **104i** jeweils mit dem entsprechenden Bus **110i** gekoppelt und kommunizieren über ihn.

[0020] Mit jedem der Prozessoren **106i** in dem System **100** ist ein Satz von Caches L1 und L2 assoziiert und mit jedem der Speicher **108i** sind ein Verzeichnis und ein Cache L3 assoziiert. Jeder der Speicher **108i** wird von seinem jeweiligen einzigen Verzeichnis verwaltet. Die Speicher **108i** oder Teile davon werden hierin als Datenblöcke oder einfach als Blöcke bezeichnet. Obwohl es mehrere Verzeichnisse in dem System **100** gibt, wird in dieser erläuternden Ausführungsform jeder Block von nur einem von ihnen verwaltet. Wenn ein vermeintlicher Leser oder ein vermeintlicher Schreiber keine aktuelle Kopie eines Blocks hat, fordert er das entsprechende Verzeichnis auf, die neueste Version des Blocks zu finden. Das Verzeichnis kann eine oder mehrere momentane Kopien des Blocks für ungültig erklären müssen, um einer Anforderung nachzukommen.

[0021] Auch in Fig. 1 dargestellt ist ein Beispiel für eine Leseoperation, in der der Prozessor **106A** des Knotens A Daten aus dem Speicher **108B** des Knotens B liest. Als ein Teil dieser Operation geht die Leseanforderung (1) von Knoten A zu Knoten B und kehrt eine Antwort (2) von Knoten B zu Knoten A zurück. Knoten A speichert die Daten in seiner lokalen Cache-Hierarchie, d. h. den Caches L1, L2 und L3, zwischen. Das Verzeichnis in Knoten B speichert einen Hinweis darauf, dass Knoten A eine Kopie der Daten hat. Andere Knoten lesen auf die gleiche Weise Daten aus Knoten B.

[0022] Man beachte, dass die Begriffe "Leser" und "Schreiber", wie hierin verwendet, ohne Einschränkung sowohl einen gegebenen Prozessorknoten oder dessen assoziierten Prozessor als auch Elemente oder Teile eines Prozessorknotens oder dessen assoziierten Prozessors beinhalten sollen.

[0023] Fig. 2 zeigt eine Schreiboperation, in der der Prozessor **106C** des Knotens C die gleichen Daten schreibt, die sich in Speicher **108B** des Knotens B befinden. Als ein Teil dieser Operation geht die Schreibanforderung (1) von Knoten C zu Knoten B. Da das Verzeichnis in Knoten B weiß, dass Knoten A eine Kopie der Daten hat, sendet es eine Ungültigkeitserklärungsanforderung (2) an Knoten A. Knoten A sendet eine Bestätigung (3) der Ungültigkeitserklärung seiner Kopie der Daten. Knoten B sendet dann die Daten (4) zum Schreiben an Knoten C, da es keine weiteren Kopien im System gibt.

[0024] Ein gegebener Speicherblock in dem System **100** kann folglich betrachtet werden als zwischen Phasen wechselnd, in denen er von einem einzelnen Prozessor geschrieben wird und in denen er von einem oder mehreren Prozessoren gelesen wird. Das mit dem gegebenen Block assoziierte Verzeichnis verwaltet diese abwechselnden Phasen, wobei eine konsistente Version des Blocks zu jeder Zeit aufrecht erhalten wird.

[0025] Man beachte, dass die in Fig. 1 und Fig. 2 gezeigten speziellen Anordnungen von Caches und Cache-Ebenen nur Beispiele sind und nicht als den Umfang der vorliegenden Erfindung auf irgendeine Weise einschränkend ausgelegt werden sollten. Die Erfindung kann unter Verwendung einer breiten Vielfalt verschiedener Cache-Architekturen oder Multiprozessorsystemkonfigurationen implementiert werden.

[0026] Fig. 3 stellt ein Beispiel für diese abwechselnden Phasen für einen einzelnen Block in einem System mit fünf Knoten, bezeichnet mit 1, 2, 3, 4 und 5, dar. Die linke Seite der Figur zeigt die rohe Abfolge von Lese- und Schreibereignissen, die sich auf den einzelnen Block beziehen, und die rechte Seite der Figur zeigt eine Zusammenfassung der oben ge-

nannten Phasen. Wie aus diesem Beispiel offensichtlich wird, ist es allgemein für mehrere Leser sicher, die vor kürzester Zeit erzeugte Version eines Blocks zu prüfen.

[0027] Die vorliegende Erfindung schafft in einer erläuternden Ausführungsform einen verzeichnisbasierten Prognosemechanismus, der den nächsten Satz von Lesern eines Blocks prognostiziert, wenn eine Schreibanforderung von dem Schreiber zu dem mit dem Block assoziierten Verzeichnis geht. Der Mechanismus prognostiziert einen wahrscheinlichen Satz von Lesern des von dem Schreiber produzierten Werts und nachdem der Schreiber mit dem Schreiben fertig ist, wird diese Prognose verwendet, um die Daten an alle prognostizierten Leser weiterzuleiten. Im Gegensatz zu herkömmlichen Prädiktoren, die zwischen Blöcken oder zwischen Anweisungen unterscheiden, um separate Vorgeschichten für Blöcke in dem System zu führen, führt der Prognosemechanismus der vorliegenden Erfindung mehrere Sätze von Lesern für mehrere Blöcke, die von dem Verzeichnis bedient werden, zusammen. Diese Information wird hierin als die globale Vorgeschichte des Verzeichnisses bezeichnet.

[0028] In der beispielhaften Implementierung der unten in Verbindung mit Fig. 4 beschriebenen erläuternden Ausführungsform wird eine Vorgeschichtstiefe von vier verwendet, d. h. der prognostizierte Satz von Lesern, der für eine momentane Schreiboperation auf einen gegebenen Block erzeugt wurde, wird als eine Funktion des momentanen Satzes von Lesern dieses Blocks und der in einem Prädiktorenschieberegister gespeicherten drei anderen aktuellsten Sätzen von Lesern bestimmt.

[0029] Fig. 4 zeigt ein Beispiel für die Arbeitsweise eines verzeichnisbasierten Prädiktors in der erläuternden Ausführungsform der Erfindung. In diesem Beispiel wird eine Schreibanforderung für einen Datenblock X empfangen, der mit einem Speicher und einem Verzeichnis **120** assoziiert ist. Die momentanen Leser des Datenblocks X sind ein Satz von Knoten {a, b, c} eines Multiprozessorsystems, das mit a, b, c, d, e, f, g, h, i, j, k, l, m usw. bezeichnete Knoten aufweist. Jeder der Knoten kann einen Knoten eines Multiprozessorsystems, wie z. B. des in Verbindung mit Fig. 1 und Fig. 2 erläuterten, darstellen. Der Prädiktor in diesem Beispiel verwendet auf eine unten beschriebene Weise ein Schieberegister **122**.

[0030] Fig. 5 zeigt ein Ablaufdiagramm der allgemeinen Verarbeitungsoperationen des verzeichnisbasierten Prädiktors aus dem Beispiel in Fig. 4. Die allgemeinen Operationen werden zuerst mit Bezug auf Fig. 5 beschrieben und dann wird die Anwendung der allgemeinen Operationen auf das Beispiel aus Fig. 4 im Detail beschrieben.

[0031] In Schritt **200** in Fig. 5 sendet ein Schreiber eine Schreibanforderungsnachricht an das Verzeichnis des zu schreibenden Blocks. Das Verzeichnis erklärt in Schritt **202** die momentanen Leser für ungültig. Schritte **204** und **206** werden dann von jedem der Leser ausgeführt. In Schritt **204** empfängt ein gegebener Knoten, der einem potentiellen Leser entspricht, die Ungültigkeitserklärung von dem Verzeichnis. In Schritt **206** schickt der Knoten ein "Bit, auf das zugegriffen wurde" mit einer Bestätigung der Ungültigkeitserklärung zurück.

[0032] Wie in Schritt **208** aufgezeigt, wartet das Verzeichnis auf Ungültigkeitserklärungsbestätigungen von den Lesern. Ein Satz von echten Lesern wird als der Satz von für ungültig erklärten Knoten bestimmt, für die das zurückgeschickte Bit, auf das zugegriffen wurde, festgelegt ist. Das Verzeichnis liefert dem Prädiktor in Schritt **210** Information, die den Satz von echten Lesern identifiziert.

[0033] Der Prädiktor fügt dann seinem Schieberegister (Schritt **212**) den Satz von echten Lesern hinzu, verwirft den ältesten Satz von vorhergehenden Lesern in dem Schieberegister (Schritt **214**), prognostiziert unter Verwendung einer Schnittmengen- oder Vereinigungsoperation (Schritt **216**) eine Funktion der Sätze und sendet die Prognose dann an den Schreiber (Schritt **218**).

[0034] Das Verzeichnis sendet dem Schreiber in Schritt **220** eine gültige Kopie des Blocks. Diese Kopie kann dem Schreiber zusammen mit der Prognose aus Schritt **218** gesendet werden. In Schritt **222** vergeht Zeit, bis der Schreiber die Schreiboperation abschließt. Nachdem die Schreiboperation vervollständigt ist, verwendet der Schreiber die Information in der Prognose, um den neuen Block an jeden prognostizierten Leser weiterzuleiten, wie in Schritt **224** gezeigt.

[0035] Geeignete Techniken zum Bestimmen einer geeigneten Zeit zum Weiterleiten eines neuen Blocks an jeden prognostizierten Leser sind z. B. in S. Kaxiras, "Identification and Optimization of Sharing Patterns für Scalable Shared-Memory Multiprocessors", PhD. Thesis, Computer Sciences, University of Wisconsin-Madison, 1998, und in der oben angeführten Referenz A. Lai und B. Falsafi, "Memory Sharing Predictor: The Key to a Speculative Coherent DSM", Proceedings of the 26th Annual ISCA, Mai 1999, beschrieben, die hierin beide durch Bezugnahme aufgenommen sind.

[0036] Die Wahl der Vereinigungs- oder Schnittmengenfunktion in Schritt **216** aus Fig. 5 hängt allgemein von dem erwünschtem Aggressivitätsgrad bei der Datenweiterleitung ab. Zum Beispiel kann in Systemen mit hoher Bandbreite die mit der Vereinigungsfunktion assoziierte aggressivere Datenweiterleitung

geeigneter sein, während für Systeme mit niedriger Bandbreite die Schnittmengenfunktion geeigneter sein kann. Man beachte, dass diese Funktionen nur beispielhaft gegeben werden und die Erfindung unter Verwendung von anderen Typen von Funktionen implementiert werden kann. Als ein anderes Beispiel können musterbasierte Funktionen in Verbindung mit der vorliegenden Erfindung verwendet werden. Solche Funktionen sind detaillierter z. B. in T. Yeh und Y. Patt, "Two-Level Adaptive Branch Prediction", Proceedings of the 24th Annual ACM/IEEE International Symposium and Workshop an Microarchitecture, Los Alamitos, CA, November 1991, beschrieben, das hierin durch Bezugsnahme aufgenommen ist.

[0037] Die Wahl zwischen Vereinigungs-, Schnittmengen- oder anderen Funktionen in Schritt **216** kann auf einer dynamischen Basis getroffen werden. Zum Beispiel kann die Wahl auf einer Pro-Programm-Basis getroffen werden, so dass jedes Programm seinen eigenen Betriebsmodus einstellen kann. Als ein anderes Beispiel kann die Auswahl der Funktion unter Programmsteuerung implementiert werden, so dass Programme den Betriebsmodus zur Laufzeit entsprechend ihrem Bedarf ändern können. Die Wahl könnte alternativ auf einer Pro-Seite-Basis getroffen werden, wobei jede Speicherseite, die mehrere Datenblöcke aufweisen kann, ihren eigenen Modus hat. In diesem Fall kann ein Betriebssystem den Prädiktor über den Betriebsmodus verschiedener Seiten benachrichtigen. Als noch ein anderes Beispiel könnte die Wahl der Funktion entsprechend der Netzwerknutzung getroffen werden, wobei z. B. eine niedrige Netzwerknutzung die Vereinigungsfunktion erfordert und eine hohe Netzwerknutzung die Schnittmengenfunktion erfordert. In diesem Fall kann eine Netzwerküberwachungsvorrichtung verwendet werden, um dem Prognosemechanismus eine Rückkopplung zu liefern.

[0038] Wieder mit Bezug auf das Beispiel aus Fig. 4 sind, wenn die Schreibanforderung für den Datenblock X empfangen wird, die momentanen Leser die Prozessoren in dem Satz {a, b, c}. Das Prädiktorenschieberegister **122** wird wie gezeigt um eins verschoben und der Satz {a, b, c} wird in dem obersten mit Slot 0 bezeichneten Slot installiert. In Folge der Änderung enthalten die Slots 1, 2 und 3 die Sätze {a, c, e, f, g}, {a, c, d} bzw. {a, h, i, c} und wird der Satz {k, l, m} aus dem Schieberegister fallen gelassen. Der momentane Inhalt des Schieberegisters **122** zu einem gegebenen Zeitpunkt stellt die globale Vorgeschichte des entsprechenden Verzeichnisses dar. Das Verzeichnis erklärt die momentanen Leser durch das Schicken von Ungültigkeitserklärungsanforderungen an die Knoten a, b und c für ungültig, wartet auf Bestätigung der Ungültigkeitserklärung und sendet später eine gültige Kopie des Datenblocks X an den anfordernden Schreiber.

[0039] Der Prädiktor bestimmt die Vereinigung oder die Schnittmengenbildung der Sätze in dem Schieberegister **122** gemäß Schritt **216** aus Fig. 5, wobei die Wahl von Vereinigung oder Schnittmengenbildung auf der Basis eines oder mehrerer der oben beschriebenen Faktoren getroffen wird. Die Vereinigung der Sätze in dem Schieberegister ist der Satz {a, b, c, d, e, f, g, h, i}, während die Schnittmenge der Sätze in dem Schieberegister der Satz {a, c} ist. In beiden Fällen ist das Ergebnis ein Satz von prognostizierten Lesern, der dem Schreiber von dem Prädiktor zugeschickt wird. Nachdem die Schreiboperation auf dem Datenblock X vervollständigt ist, leitet der Schreiber den neuen Block an jeden der prognostizierten Leser weiter. Das Auslösen der Datenweiterleitung kann auf einem Zeitgeber oder auf dem nächsten Schreiben in das Verzeichnis, ungeachtet dessen, welcher Datenblock geschrieben wird, oder auf dem nächsten Lesen des Datenblocks X oder auf anderen geeigneten Techniken basieren. Die Datenweiterleitung kann dadurch durchgeführt werden, dass das Verzeichnis eine Kopie der Daten von dem Schreiber abruft und sie an die prognostizierten Leserknoten sendet.

[0040] Man beachte, dass, obwohl der Prädiktor in dem Beispiel aus Fig. 4 eine Vorgeschichtstiefe von vier verwendet, d. h. das Schieberegister **122** die vier aktuellsten Sätze von Lesern für einen gegebenen Datenblock speichert, die vorliegende Erfindung unter Verwendung von anderen Vorgeschichtstiefen, einschließlich Vorgeschichtstiefen, die mehr oder weniger als vier sind, implementiert werden kann. Herkömmliche Prädiktoren nutzen allgemein eine Vorgeschichtstiefe von nicht mehr als zwei.

[0041] Um für den oben beschriebenen Prognosemechanismus eine genaue Rückkopplung zu schaffen, muss jeder Leser allgemein in der Lage sein, zwischen einem prognostizierten Lesen und einem tatsächlichen Lesen zu unterscheiden. Wenn ein Schreiber exklusiven Zugriff auf einen Cache-Block gewinnt, prognostiziert ein Multiprozessorsystem gemäß der Erfindung den zukünftigen Satz von Lesern des Blocks und stellt dann sicher, dass Kopien des Blocks an diese prognostizierten Leser weitergeleitet werden, nachdem das Schreiben vervollständigt ist. Um die Rückkopplungsschleife zu schließen, muss das System herausfinden, wie viele dieser prognostizierten Leser den Block tatsächlich verwendet haben. Um zu sagen, ob dies der Fall ist, kann jeder Leser in dem System das oben genannte "Bit, auf das zugegriffen wurde," für jede lokale Cache-Linie aufrecht erhalten. Dieses Bit, auf das zugegriffen wurde, ist dem so genannten "schmutzigen Bit" ähnlich, das zur Seitenverwaltung in einem virtuellen Speichersystems aufrecht erhalten wird, abgesehen davon, dass das Bit, auf das zugegriffen wurde, eingestellt wird, wenn ein Cache-Block gelesen wird, anstatt wenn er geschrieben wird. Auch sollte das Bit, auf das zugegriffen wurde, auf Cache-Block-Granularität

aufrecht erhalten werden, während schmutzige Bits typischerweise auf Seitengranularität aufrecht erhalten werden. Bei der nächsten Ungültigkeitserklärung nimmt jeder Leser die Information des Bits, auf das zugegriffen wurde, auf die Ungültigkeitserklärungsbestätigung „Huckepack". Das System kann dann die Bits, auf die zugegriffen wurde, verwenden, um seinen Status für die nächste Prognose zu aktualisieren.

[0042] Man beachte, dass Verarbeitungsoperationen, die hierin als von einem Verzeichnis ausgeführt oder anderweitig implementiert beschrieben werden, von einem assoziierten Element eines Prozessorknotens, wie z. B. von einem Prozessor unter Programmsteuerung, ausgeführt oder anderweitig implementiert werden können.

[0043] In alternativen Ausführungsformen der Erfindung kann die Verzeichnisinformation durch andere Information ergänzt werden, um weitere Verbesserungen der Leistungsfähigkeit zu schaffen. Zum Beispiel kann die Verzeichnisinformation durch einen designierten Teilsatz von Cache-Block-Adressinformation ergänzt werden. Vorteilhafterweise verwendet eine solche Anordnung weniger Information als herkömmliche adressenbasierte Prognosetechniken, während sie auch eine höhere Prognosegenauigkeit erreicht. In anderen Ausführungsformen der Erfindung kann die Verzeichnisinformation mit dem oder ohne den Teilsatz der Cache-Adressinformation mit anderen Typen von Information, wie z. B. Prozessorknoten- und Programmzählerinformation, kombiniert werden. Zum Beispiel kann die Erfindung in dem Prozess des Bestimmens eines Satzes von prognostizierten Lesern für eine gegebene Schreibanforderung verschiedene Kombinationen von Verzeichnis-, Adress-, Prozessorknoten- und Programmzählerinformation nutzen.

[0044] Wie der Begriff hierin verwendet wird, soll die "globale Vorgeschichte" eines Verzeichnisses nicht nur eine auf Verzeichnisinformation allein basierende Vorgeschichte, sondern auch eine Vorgeschichte aufweisen, die sowohl Verzeichnisinformation als auch eine Menge an zusätzlicher Information, wie z. B. Adress-, Prozessorknoten- oder Programmzählerinformation aufweist, was weniger ist als das volle verfügbare Ausmaß an solcher zusätzlicher Information. Zum Beispiel kann eine globale Vorgeschichte Verzeichnisinformation aufweisen, die durch eine kleine Anzahl von Adressbits, d. h. eine Menge an Adressbits, die kleiner ist als ein voller Satz von verfügbaren Adressbits, ergänzt ist.

[0045] Fig. 6 zeigt einen Satz von sechs Tabellen, die Beispiele für Prädiktoren gemäß der Erfindung und entsprechende Leistungsfähigkeitssimulationsergebnisse auflisten. Die gezeigten Prädiktoren basieren auf verschiedenen Kombinationen aus einer oder mehrerer Informationen von Verzeichnis (dir), Adresse (add), Prozessorknoten (pid) und Programmzähler (pc). Die Prädiktorennamen haben die Form von prediction-function(index)^{depth}, wobei prediction-function die Funktion anzeigt, die für die Aktualisierung des Prädiktors verwendet wird, Index die spezielle von dem Prädiktor verwendete Kombination von Verzeichnis-, Adress-, Prozessorknoten- und Programmzählerinformation anzeigt und depth die Vorgeschichtstiefe ist. Im Fall von Adress(add)- oder Programmzähler(pc)-Information weist der jeweilige Identifikator eine tiefgestellte Zahl auf, die die entsprechende Anzahl von Informationsbits anzeigt.

[0046] Die in Fig. 6 gezeigten Prädiktoren werden auch als entweder direkt oder weitergeleitet klassifiziert, um den speziellen verwendeten Aktualisierungsmechanismus anzuzeigen. Bei einem Mechanismus direkter Aktualisierung wird der Satz von für ungültig erklärten echten Lesern jedes Mal, wenn ein Datenblock geschrieben wird, als Vorgeschichte verwendet, um die neue Prognose zu erzeugen. Bei einem Mechanismus weitergeleiteter Aktualisierung leitet ein Schreiber, wenn er einen Satz von mit einem anderen Knoten assoziierten Lesern für ungültig erklärt, diese Vorgeschichte an den entsprechenden Prädiktoreneintrag weiter, so dass sie von dem korrekten Schreiber verwendet werden kann. Eine weitergeleitete Aktualisierung erfordert folglich Information vom letzten Schreiber für jeden Datenblock, so dass für ungültig erklärte Leser mit einem bestimmten Schreiber assoziiert werden können. Tabellen 1, 2 und 3 aus Fig. 6 listen Prädiktoren auf, die einen Mechanismus direkter Aktualisierung nutzen, während Tabellen 4, 5 und 6 Prädiktoren auflisten, die einen Mechanismus weitergeleiteter Aktualisierung verwenden.

[0047] Beispielsweise stellt der Prädiktor union(pid + di r +add_4)^4 in Tabelle 6 ein Prognoseschema unter Verwendung von direkter Aktualisierung dar, wobei sein Prognosestatus unter Verwendung der Prozessornummer, des Verzeichnisknotens und vier Datenblockadressen-Bits indiziert wird und die letzten zwei Bitmaps gemeinsamer Nutzung vereinigt, um die nächste für jeden Index zu prognostizieren. Als ein anderes Beispiel kann ein von Verzeichnisknoten und acht Adresseninformations-Bits indizierter Prädiktor der letzten Bitmap als union(dir + add_8)^1 oder inter(dir + add_8)^1 bezeichnet werden, abhängig von der speziellen verwendeten Funktion. Man beachte, dass ein auf Vereinigung basierender oder auf Schnittmengenbildung basierender Prädiktor mit einer Vorgeschichtstiefe von eins der gleiche ist wie ein Prädiktor der letzten Bitmap.

[0048] Zusätzliche Details bezüglich dieser und anderer Aspekte von Prädiktoren sind in S. Kaxiras und C. Young, "Coherence Communication Prediction in Shared Memory Multiprocessors", Proceedings of the 6th Annual IEEE Symposium an High-Perfor-

mance Computer Architecture (HPCA), Januar 2000, beschrieben, das hierin durch Bezugnahme aufgenommen ist.

[0049] Für jeden der in Fig. 6 gezeigten beispielhaften Prädiktoren ist eine Anzahl von Leistungsparametern aufgelistet. Diese umfassen Prädiktorgröße, Sensivität, Spezifität, Prognosewert eines positiven Tests (PVP) und Prognosewert eines negativen Tests (PVN).

[0050] Die Prädiktorgröße wird als $\log_t$ der Anzahl von Bits gemessen, die von dem Prädiktor genutzt werden.

[0051] Sensitivität ist das Verhältnis korrekter Prognosen zu der Summe aus korrekten Prognosen und weggelassenen Prognosen und zeigt an, wie gut der Prädiktor die gemeinsame Nutzung prognostiziert, wenn die gemeinsame Nutzung tatsächlich stattfindet. Ein sensitiver Prädiktor ist gut im Finden und Nutzen von Möglichkeiten gemeinsamer Nutzung, während ein insensitiver Prädiktor viele Gelegenheiten verpasst.

[0052] Spezifität ist das Verhältnis vermiedener Prognosen zu der Summe aus vermiedenen Prognosen und zusätzlichen Prognosen und zeigt die Wahrscheinlichkeit an, dass keine Ressourcen an nicht gemeinsam genutzte Daten verschwendet werden.

[0053] PVP ist das Verhältnis korrekter Prognosen zu der Summe aus korrekten und zusätzlichen Prognosen und schafft einen Hinweis auf den Prozentsatz von nützlichem Datenweiterleitungsverkehr unter allem Datenweiterleitungsverkehr.

[0054] PVN ist das Verhältnis vermiedener Prognosen zu der Summe aus vermiedenen Prognosen und weggelassenen Prognosen und schafft einen Hinweis, mit welcher Wahrscheinlichkeit bezüglich eines nicht gemeinsam genutzten Blocks korrekt prognostiziert wird, dass er nicht gemeinsam genutzt wird.

[0055] Tabellen 1 und 4 zeigen die zehn Prädiktoren mit den höchsten PVPs unter direkter Aktualisierung bzw. weitergeleiteter Aktualisierung eines Satzes von möglichen Prädiktoren, für die Leistungsfähigkeit simuliert wurde. Alle Prädiktoren in dieser Gruppe sind Schnittmengenprädiktoren mit tiefer Vorgeschichte, die den PVP durch Spekulieren auf nur sehr stabile Beziehungen gemeinsamer Nutzung maximieren. Zwei der Top-Ten-Schemen sind den zwei Tabellen gemeinsam. Es ist zu sehen, dass direkte Aktualisierung und weitergeleitete Aktualisierung sehr weinig Einfluss auf den PVP haben. Jedoch sind die weitergeleiteten Schemen im Durchschnitt sensitiver. Keines der Schemen mit hohem PVP ist im Vergleich zu einem Schema mit letzter Bitmap oder einem Vereinigungsprädiktorschema sensitiv. Dies bedeutet, dass sie sehr produktiven Verkehr erzeugen, aber viele Gelegenheiten zur gemeinsamen Nutzung verpassen.

[0056] Tabelle 2 zeigt die zehn sensitivsten Schemen in dem Satz möglicher Prädiktoren unter Verwendung von direkter Aktualisierung. Alle sind Vereinigungsschemen mit der in diesem Beispiel verwendeten maximalen Vorgeschichtstiefe, d. h. einer Vorgeschichtstiefe von 4. Alle Schemen sind in ihrer Sensivität grob vergleichbar, aber haben verschiedene PVP-Werte. Es ist interessant, festzustellen, dass das weitaus kostengünstigste Schema (union(dir + $add_2)^4$) bezüglich der Sensivität das fünftbeste insgesamt ist.

[0057] Tabelle 3 zeigt die zehn sensitivsten Schemen in dem Satz möglicher Prädiktoren unter Verwendung von weitergeleiteter Aktualisierung. Es gibt einen sehr kleinen Unterschied zwischen den Schemen mit direkter und mit weitergeleiteter Aktualisierung. Sechs der obersten zehn Schemen sind den zwei Listen gemeinsam und die Statistiken unterscheiden sich von Spalte zu Spalte wenig.

[0058] Tabellen 5 und 6 zeigen die Top-Ten-Prädiktoren in dem Satz möglicher Prädik toren mit weitergeleiteter Aktualisierung bezüglich Spezifität bzw. Sensivität.

[0059] Es sollte nochmals betont werden, dass die in Fig. 6 gezeigten Prädiktoren nur Beispiele sind und die Erfindung unter Verwendung anderer Typen von Prädiktoren implementiert werden kann. Zum Beispiel können, obwohl die maximale Vorgeschichtstiefe in dem Beispiel aus Fig. 6 vier ist, andere Prädiktoren größere Vorgeschichtstiefen verwenden.

[0060] Die vorliegende Erfindung kann so konfiguriert werden, dass sie die Anforderungen an eine Vielfalt verschiedener Verarbeitungsanwendungen und -umgebungen unter Verwendung jeglicher erwünschter Typen und Anordnungen von Prozessoren erfüllt. Die oben beschriebenen Ausführungsformen der Erfindung sollen deshalb nur erläuternd sein. Für den Fachmann werden zahlreiche alternative Ausführungsformen innerhalb des Umfangs der folgenden Ansprüche offensichtlich sein.

Patentansprüche

1. Verfahren zur Bestimmung eines Satzes von prognostizierten Lesern eines Datenblocks in einem Multiprozessorsystem, wobei das Verfahren die Schritte aufweist:
Bestimmung (**208**) eines momentanen Satzes von Lesern eines Datenblocks, welcher Gegenstand einer Schreibanforderung ist; und
Generieren (**216**) des Satzes von prognostizierten

Lesern basierend auf dem momentanen Satz von Lesern und wenigstens einem zusätzlichen Satz von Lesern, der repräsentativ ist für wenigstens einen Teil einer globalen Vorgeschichte eines Verzeichnisses, das mit dem Datenblock assoziiert ist.

2. Verfahren nach Anspruch 1, wobei der Generierungsschritt weiterhin den Schritt des Anwendens einer Funktion auf den momentanen Satz von Lesern und wenigstens einen zusätzlichen Satz von Lesern aufweist.

3. Verfahren nach Anspruch 1 oder Anspruch 2, wobei die globale Vorgeschichte des Verzeichnisses eine Mehrzahl von Sätzen von durch das Verzeichnis verarbeiteten vorhergehenden Lesern aufweist, wobei die gesamte Anzahl der Mehrzahl von Sätzen von vorhergehenden Lesern einer designierten Vorgeschichtentiefe entspricht, die mit der Generierung des Satzes von prognostizierten Lesern assoziiert ist.

4. Verfahren nach einem der Ansprüche 1 bis 3, wobei jeder Leser in dem System ein Bit, auf das zugegriffen wurde, für jeden einer Mehrzahl von Datenblöcken hält, wobei das Bit, auf das zugegriffen wurde, eines bestimmten Lesers für einen gegebenen Datenblock anzeigt, ob der bestimmte Leser den gegebenen Datenblock tatsächlich gelesen hat.

5. Verfahren nach einem der Ansprüche 1 bis 4, wobei nachdem das angeforderte Schreiben auf den Datenblock vervollständigt ist, der resultierende Datenblock zu jedem der Leser in dem Satz von prognostizierten Lesern gesendet wird.

6. Verfahren nach einem der Ansprüche 1 bis 5, wobei der Generierungsschritt weiterhin ein Benutzen von Information betreffend die globale Vorgeschichte des Verzeichnisses in Verbindung mit wenigstens einem Teilsatz von Cache-Adressinformation beinhaltet, die mit einem oder mehreren der Leser assoziiert ist, um den Satz von prognostizierten Lesern zu bestimmen.

7. Verfahren nach einem der Ansprüche 1 bis 6, wobei der Generierungsschritt weiterhin ein Benutzen von Information betreffend die globale Vorgeschichte des Verzeichnisses in Verbindung mit Prozessorknoteninformation beinhaltet, die assoziiert ist mit einem oder mehreren der Leser, um den Satz von prognostizierten Lesern zu bestimmen.

8. Verfahren nach einem der Ansprüche 1 bis 7, wobei der Generierungsschritt weiterhin ein Benutzen von Information betreffend die globale Vorgeschichte des Verzeichnisses in Verbindung mit Programmzählerinformation beinhaltet, die assoziiert ist mit einem oder mehreren der Leser, um den Satz von prognostizierten Lesern zu bestimmen.

9. Vorrichtung zur Bestimmung eines Satzes von prognostizierten Lesern eines Datenblocks in einem Multiprozessorsystem, wobei die Vorrichtung aufweist:
einen Prozessorknoten, der betreibbar ist, einen momentanen Satz von Lesern eines Datenblocks zu bestimmen, welcher Gegenstand einer Schreibanforderung ist, und einen Prognostizierungsmechanismus zu implementieren, welcher den Satz von prognostizierten Lesern basierend auf dem momentanen Satz von Lesern und wenigstens einem zusätzlichen Satz von Lesern generiert, der repräsentativ ist für wenigstens einen Teil einer globalen Vorgeschichte eines Verzeichnisses, das mit dem Datenblock assoziiert ist.

10. Multiprozessorsystem, aufweisend:
eine Mehrzahl von Prozessorknoten, wobei wenigstens ein gegebener der Prozessorknoten betreibbar ist, einen momentanen Satz von Lesern eines Datenblocks zu bestimmen, welcher Gegenstand einer Schreibanforderung ist, wobei der gegebene Prozessorknoten einen Prognostizierungsmechanismus implementiert, welcher einen Satz von prognostizierten Lesern des Datenblocks basierend auf dem momentanen Satz von Lesern und wenigstens einem zusätzlichen Satz von Lesern generiert, der repräsentativ ist für wenigstens einen Teil einer globalen Vorgeschichte eines Verzeichnisses, das mit dem Datenblock assoziiert ist.

Es folgen 6 Blatt Zeichnungen

Anhängende Zeichnungen

FIG. 1

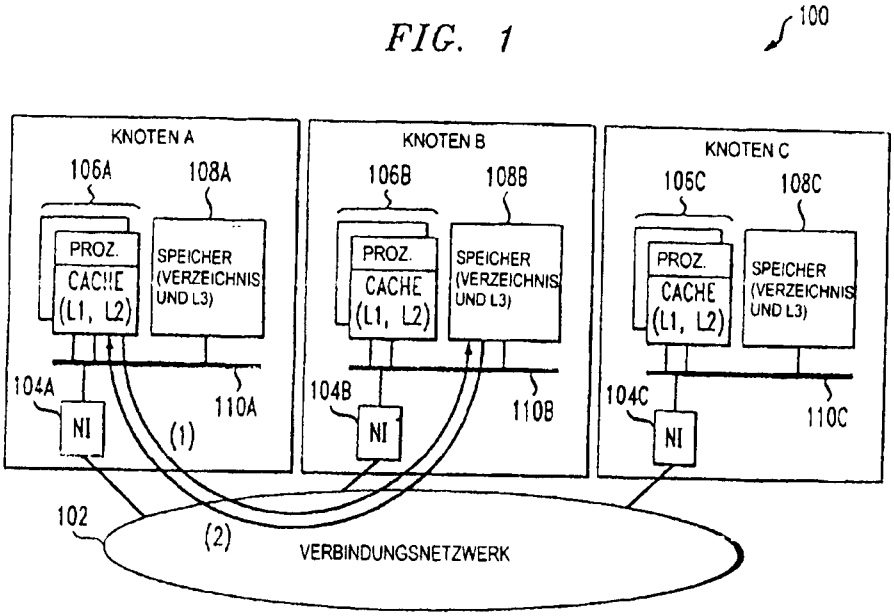

FIG. 2

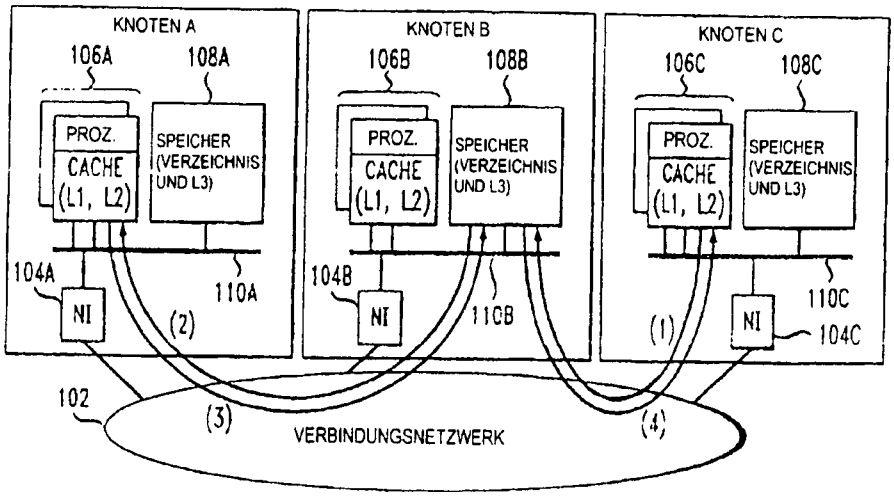

FIG. 3

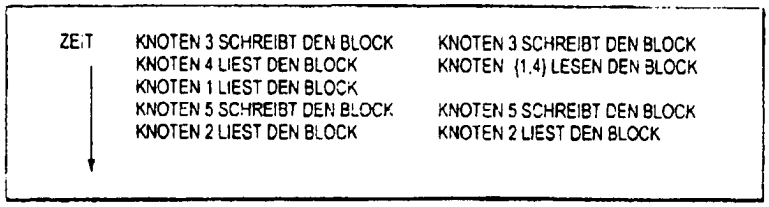
ZEIT
KNOTEN 3 SCHREIBT DEN BLOCK
KNOTEN 4 LIEST DEN BLOCK
KNOTEN 1 LIEST DEN BLOCK
KNOTEN 5 SCHREIBT DEN BLOCK
KNOTEN 2 LIEST DEN BLOCK
KNOTEN 3 SCHREIBT DEN BLOCK
KNOTEN {1,4} LESEN DEN BLOCK
KNOTEN 5 SCHREIBT DEN BLOCK
KNOTEN 2 LIEST DEN BLOCK

FIG. 4

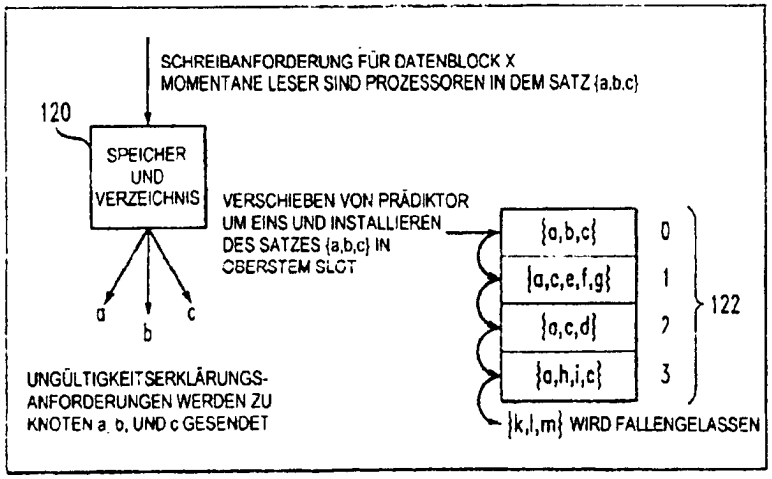
SCHREIBANFORDERUNG FÜR DATENBLOCK X
MOMENTANE LESER SIND PROZESSOREN IN DEM SATZ {a,b,c}
120
SPEICHER UND VERZEICHNIS
a
b
c
VERSCHIEBEN VON PRÄDIKTOR UM EINS UND INSTALLIEREN DES SATZES {a,b,c} IN OBERSTEM SLOT
{a,b,c}
{a,c,e,f,g}
{a,c,d}
{a,h,i,c}
0
1
2
3
122
{k,l,m} WIRD FALLENGELASSEN
UNGÜLTIGKEITSERKLÄRUNGS-ANFORDERUNGEN WERDEN ZU KNOTEN a, b, UND c GESENDET

FIG. 5

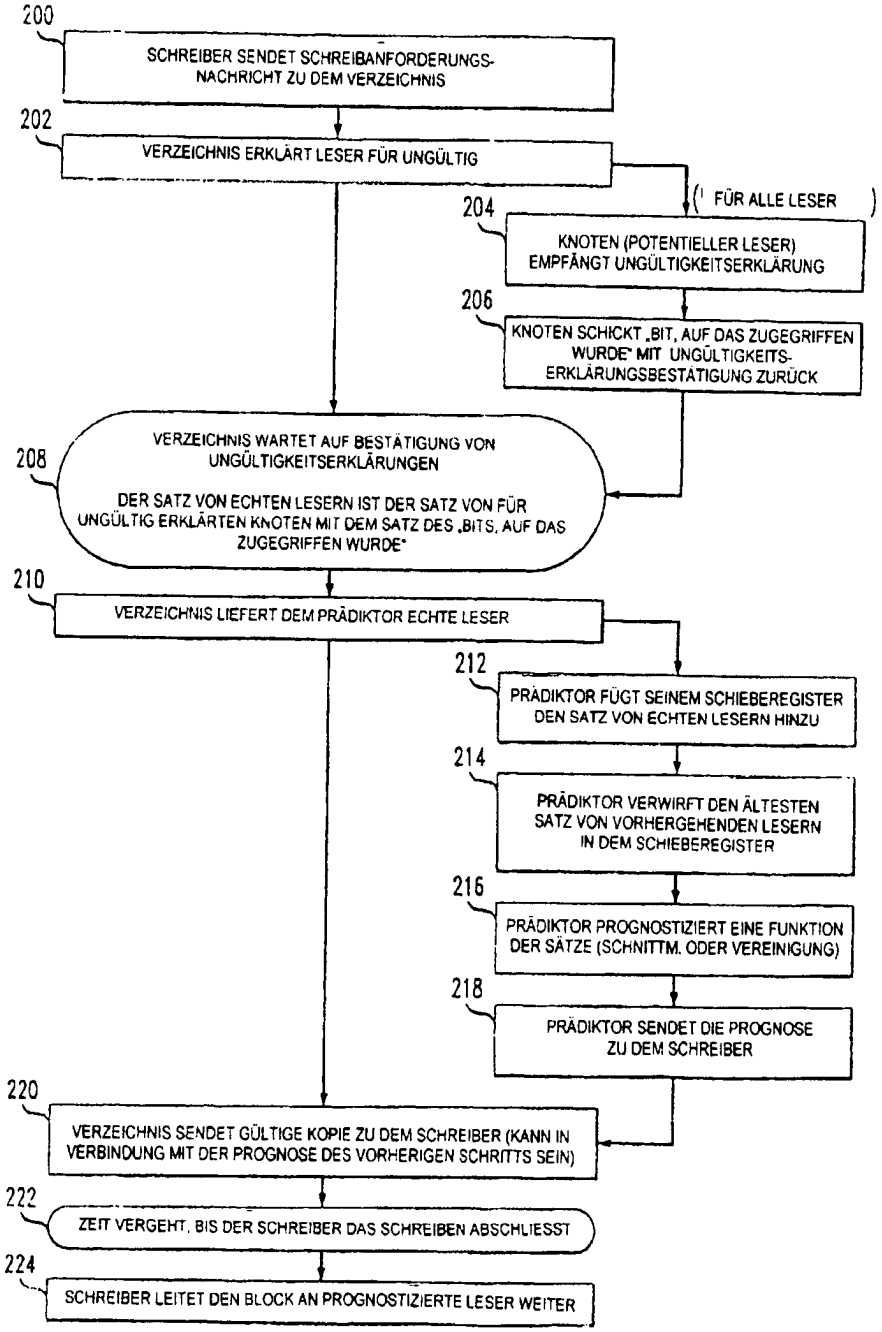
200
SCHREIBER SENDET SCHREIBANFORDERUNGS-NACHRICHT ZU DEM VERZEICHNIS
202
VERZEICHNIS ERKLÄRT LESER FÜR UNGÜLTIG
(FÜR ALLE LESER)
204
KNOTEN (POTENTIELLER LESER) EMPFÄNGT UNGÜLTIGKEITSERKLÄRUNG
206
KNOTEN SCHICKT „BIT, AUF DAS ZUGEGRIFFEN WURDE" MIT UNGÜLTIGKEITS-ERKLÄRUNGSBESTÄTIGUNG ZURÜCK
208
VERZEICHNIS WARTET AUF BESTÄTIGUNG VON UNGÜLTIGKEITSERKLÄRUNGEN
DER SATZ VON ECHTEN LESERN IST DER SATZ VON FÜR UNGÜLTIG ERKLÄRTEN KNOTEN MIT DEM SATZ DES „BITS, AUF DAS ZUGEGRIFFEN WURDE"
210
VERZEICHNIS LIEFERT DEM PRÄDIKTOR ECHTE LESER
212
PRÄDIKTOR FÜGT SEINEM SCHIEBEREGISTER DEN SATZ VON ECHTEN LESERN HINZU
214
PRÄDIKTOR VERWIRFT DEN ÄLTESTEN SATZ VON VORHERGEHENDEN LESERN IN DEM SCHIEBEREGISTER
216
PRÄDIKTOR PROGNOSTIZIERT EINE FUNKTION DER SÄTZE (SCHNITTM. ODER VEREINIGUNG)
218
PRÄDIKTOR SENDET DIE PROGNOSE ZU DEM SCHREIBER
220
VERZEICHNIS SENDET GÜLTIGE KOPIE ZU DEM SCHREIBER (KANN IN VERBINDUNG MIT DER PROGNOSE DES VORHERIGEN SCHRITTS SEIN)
222
ZEIT VERGEHT, BIS DER SCHREIBER DAS SCHREIBEN ABSCHLIESST
224
SCHREIBER LEITET DEN BLOCK AN PROGNOSTIZIERTE LESER WEITER

FIG. 6

TABLLE 1: TOP 10 PVP DIREKTE AKTUALISIERUNG

SCHEMA	GRÖSSE	SENS.	SPEZ.	PVP	PVN
$inter(pid+add_4)^4$	14	0.30	1.00	0.92	0.93
$inter(pid+cdd_8)^4$	18	0.31	1.00	0.92	0.93
$inter(pid+add_8)^3$	18	0.35	1.00	0.91	0.93
$inter(pid+pc_4+add_4)^4$	18	0.35	1.00	0.90	0.93
$inter(pid+add_4)^3$	14	0.34	1.00	0.89	0.93
$inter(pid+dir+add_4)^4$	18	0.32	1.00	0.88	0.93
$inter(pid+pc_4+add_4)^3$	18	0.37	1.00	0.88	0.93
$inter(pid)^4$	10	0.29	1.00	0.88	0.92
$inter(pid+pc_4)^4$	14	0.37	0.99	0.87	0.93
$inter(pid+pc_8)^4$	18	0.39	0.99	0.86	0.93

TABLLE 2: TOP 10 SPEZ. DIREKTE AKTUALISIERUNG

SCHEMA	GRÖSSE	SENS.	SPEZ.	PVP	PVN
$inter(pc_4+add_8)^4$	18	0.12	1.00	0.72	0.91
$inter(add_4)^4$	10	0.07	1.00	0.78	0.91
$inter(pid+add_8)^4$	18	0.31	1.00	0.92	0.93
$inter()^4$	8	0.03	1.00	0.73	0.90
$inter(add_8)^4$	14	0.11	1.00	0.70	0.91
$inter(pc_4+add_4)^4$	14	0.08	1.00	0.74	0.91
$inter(pc_8+add_4)^4$	18	0.09	1.00	0.74	0.91
$inter(pc_4)^4$	10	0.04	1.00	0.71	0.90
$inter(pc_8)^4$	14	0.04	1.00	0.71	0.90
$inter(pc_4+dir+add_4)^4$	18	0.20	1.00	0.73	0.92

FIG. 6 (FORTSETZUNG)

TABLLE 3: TOP 10 SENS DIREKTE AKTUALISIERUNG					
SCHEMA	GRÖSSE	SENS.	SPEZ.	PVP	PVN
union(dir+add$_4$)^4	14	0.67	0.86	0.40	0.95
union(dir)^4	10	0.67	0.85	0.39	0.95
union(pc$_4$+dir)^4	14	0.67	0.85	0.40	0.95
union(pc$_4$+dir+add$_4$)^4	18	0.67	0.85	0.40	0.95
union(dir+add$_8$)^4	18	0.67	0.87	0.42	0.95
union(pc$_8$+dir)^4	18	0.66	0.86	0.42	0.95
union(pid+add$_4$)^4	14	0.66	0.87	0.41	0.95
union(pid+add$_8$)^4	18	0.66	0.87	0.42	0.95
union(add$_{12}$)^4	18	0.66	0.86	0.40	0.95
union(add$_4$)^4	10	0.66	0.80	0.29	0.95

TABLLE 4: TOP 10 PVP WEITERGELEITETE AKTUALISIERUNG					
SCHEMA	GRÖSSE	SENS.	SPEZ.	PVP	PVN
inter(pid+pc$_4$+add$_4$)^4	18	0.34	1.00	0.92	0.93
inter(pid+add$_4$)^4	14	0.33	1.00	0.91	0.93
inter(pid+add$_8$)^4	18	0.33	1.00	0.91	0.93
inter(pid+dir+add$_4$)^4	18	0.34	1.00	0.91	0.93
inter(pid+add$_8$)^3	18	0.37	1.00	0.90	0.93
inter(pid+add$_4$)^3	14	0.36	1.00	0.88	0.93
inter(pid+pc$_4$+add$_4$)^3	18	0.37	1.00	0.88	0.93
inter(pid+dir)^4	14	0.38	0.99	0.88	0.93
inter(pid+dir+add$_4$)^3	18	0.37	1.00	0.88	0.93
inter(pid)^4	10	0.33	1.00	0.87	0.93

FIG. 6 (FORTSETZUNG)

TABLLE 5. TOP 10 SPEZ. WEITERGELEITETE AKTUALISIERUNG

SCHEMA	GRÖSSE	SENS	SPEZ	PVP	PVN
$inter(pc_4+add_8)^4$	18	0.11	1.00	0.69	0.91
$inter(add_4)^4$	10	0.07	1.00	0.78	0.91
$inter()^4$	6	0.03	1.00	0.73	0.90
$inter(add_8)^4$	14	0.11	1.00	0.70	0.91
$inter(pid+add_8)^4$	18	0.33	1.00	0.91	0.93
$inter(pc_8+add_4)^4$	18	0.09	1.00	0.74	0.91
$inter(pc_4+add_4)^4$	14	0.07	1.00	0.69	0.91
$inter(pid+pc_4+add_4)^4$	18	0.34	1.00	0.92	0.93
$inter(pc_4+dir+add_4)^4$	18	0.18	1.00	0.73	0.91
$inter(pc_4)^4$	20	0.03	1.00	0.61	0.90

TABLLE 6: TOP 10 SENS. WEITERGELEITETE AKTUALISIERUNG

SCHEMA	GRÖSSE	SENS.	SPEZ.	PVP	PVN
$union(dir+add_4)^4$	14	0.67	0.86	0.40	0.95
$union(dir)^4$	10	0.67	0.85	0.39	0.95
$union(dir+add_8)^4$	18	0.67	0.87	0.42	0.95
$union(pc+dir+add_4)^4$	18	0.66	0.89	0.45	0.95
$union(add_{12})^4$	18	0.66	0.86	0.40	0.95
$union(pid+dir)^4$	14	0.66	0.89	0.45	0.95
$union(add_4)^4$	10	0.66	0.80	0.29	0.95
$union()^4$	6	0.65	0.77	0.25	0.95
$union(add_8)^4$	14	0.65	0.84	0.35	0.95
$union(pid+add_8)^4$	18	0.65	0.88	0.44	0.95