(19) Deutsches Patent- und Markenamt

(10) **DE 10 2024 101 981 A1** 2024.10.10

(12) # Offenlegungsschrift

(21) Aktenzeichen: **10 2024 101 981.6**
(22) Anmeldetag: **24.01.2024**
(43) Offenlegungstag: **10.10.2024**

(51) Int Cl.: ***G06F 9/50*** (2006.01)

(30) Unionspriorität:
18/286,197 **05.04.2023** **US**

(71) Anmelder:
Hewlett Packard Enterprise Development LP, Spring, TX, US

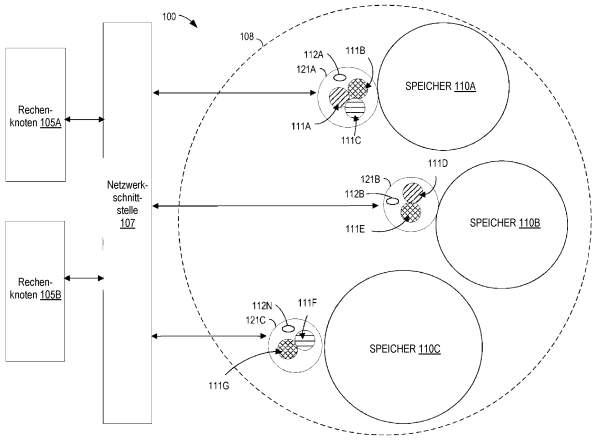
(74) Vertreter:
Fleuchaus & Gallo Partnerschaft mbB - Patent- und Rechtsanwälte, 81369 München, DE

(72) Erfinder:
Bresniker, Kirk M., Spring, TX, US; Milojicic, Dejan S, Spring, TX, US

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen.

(54) Bezeichnung: **OPTIMIERUNG DER ENERGIEEFFIZIENZ DURCH SPEICHERNAHE BERECHNUNGEN IN SKALIERBAREN DISAGGREGIERTEN SPEICHERARCHITEKTUREN**

(57) Zusammenfassung: Die Offenbarung umfasst ein System und Verfahren zur Optimierung der Leistung von disaggregierten Speicherarchitekturen in Bezug auf Zeit und Energie. In Beispielen der hier offenbarten Systeme und Verfahren ist Folgendes vorgesehen: eine speichernahe Recheneinheit in der Nähe eines disaggregierten Speichers, der dazu implementiert sein kann, von einem Rechenknoten eine oder mehrere Anforderungen zu empfangen, um Berechnungsfunktionen an Daten durchzuführen, die in dem disaggregierten Speicher gespeichert sind, und Telemetriedaten für den disaggregierten Speicher, eine speichernahe Recheneinheit in der Nähe des disaggregierten Speichers und den Rechenknoten zu erfassen. Die hier offenbarten Systeme und Verfahren können auch eine Mehrzahl von Konfigurationen zum Ausführen der einen oder mehreren Anforderungen auf Basis der Telemetriedaten modellieren, eine modellierte Konfiguration aus der Mehrzahl der modellierten Konfigurationen zum Ausführen der einen oder mehreren Anforderungen auswählen und einen oder mehrere aus einer Mehrzahl von Datenoperatoren der speichernahen Recheneinheit entsprechend der ausgewählten modellierten Konfiguration zuweisen.



Beschreibung

Hintergrund

[0001] Der herkömmliche Ansatz, Daten von direkt angeschlossenen Speichergeräten zur Berechnung an eine CPU zu übertragen, hat sich für neue datenintensive Scale-out-Anwendungen zu einem erheblichen Leistungsengpass entwickelt, da die Wiederverwendung von Daten begrenzt ist und die Gesamtgröße des Datensatzes die Kapazität der direkt angeschlossenen Geräte übersteigt. Dies hat zum Vorschlag von disaggregierten Speichersystemen geführt, die Speicherressourcen über eine Fabric-Verbindung zu einer oder mehreren CPUs bereitstellen. Gleichzeitig hat der Fortschritt bei den 3D-Integrationstechnologien das jahrzehntealte Konzept der Kopplung von Recheneinheiten in der Nähe des Speichers, das so genannte Near Memory Computing (NMC), realisierbar gemacht. Die Verarbeitung direkt am Speicher kann das Problem der Datenverschiebung bei datenintensiven Anwendungen erheblich mindern.

Kurzbeschreibung der Zeichnungen

[0002] Die vorliegende Offenbarung wird in Übereinstimmung mit einer oder mehreren verschiedenen Ausführungsformen unter Bezugnahme auf die folgenden Figuren im Detail beschrieben. Die Figuren dienen lediglich der Veranschaulichung und stellen lediglich typische oder beispielhafte Ausführungsformen dar.

Fig. 1 zeigt eine Beispielarchitektur, bei der Datenabfrageoperationen an den disaggregierten Speicher auf speichernahe Recheneinheiten ausgelagert werden können.

Fig. 2 zeigt ein weiteres Architekturbeispiel, in dem die hier beschriebenen Systeme und Methoden implementiert werden können.

Abb. 3 zeigt ein Beispiel für einen Datenoperator-Stack, der in einem disaggregierten Speicher implementiert werden kann.

Fig. 4 ist ein schematisches Boxdiagramm eines disaggregierten Speichersystems für telemetriebasierte Optimierung in Übereinstimmung mit den hier offenbarten Implementierungen.

Abb. 5 ist ein Beispiel für eine Rechnerkomponente, die zur Implementierung verschiedener Merkmale der Energieverbrauchsoptimierung in disaggregierten Speichersystemen gemäß den hier offenbarten Implementierungen verwendet werden kann.

Fig. 6 zeigt ein Blockdiagramm eines Beispiel-Computersystems, das zur Implementierung verschiedener Merkmale der Energieverbrauchsoptimierung in disaggregierten Speichersystemen der vorliegenden Offenbarung verwendet werden kann.

[0003] Die Abbildungen sind nicht erschöpfend und beschränken die vorliegende Offenbarung nicht auf die genaue Form, die offenbart wird.

Detaillierte Beschreibung

[0004] Im Laufe der Jahre konnte die Speichertechnologie in Bezug auf Latenzzeiten und Energieverbrauch nicht mit den Fortschritten in der Prozessortechnologie mithalten. Das Ende der Dennard-Skalierung, die Verlangsamung des Mooreschen Gesetzes und die Leistung von Computern mit dunklem Silizium haben ein Plateau erreicht. Gleichzeitig werden in verschiedenen Bereichen (z.B. Materialwissenschaften, Chemie, Gesundheitswissenschaften usw.) enorme Datenmengen erzeugt. Auf herkömmlichen Systemen verursachen diese Anwendungen häufige Datenbewegungen zwischen einem Speicher-Subsystem und einem Prozessor eines Servers. Häufige Datenbewegungen innerhalb eines Servers beeinträchtigen die Leistung und die Energieeffizienz. Systemarchitekten haben versucht, diese Lücke durch die Einführung von Speicherhierarchien zu schließen, die einige der Nachteile von Off-Chip-DRAMs abmildern. Die begrenzte Anzahl von Pins an der CPU, die für die Verbindung mit direkt angeschlossenen Speicherpaketen verwendet werden können, reicht jedoch nicht aus, um die heutigen Bandbreitenanforderungen von Multicore-Prozessoren zu erfüllen. Es wurden disaggregierte Speichersysteme vorgeschlagen, bei denen zusätzliche Speichersubsysteme auf einer Fabric-Schnittstelle platziert werden, auf die eine oder mehrere CPUs zugreifen können, was die Gesamtkapazität erhöht, jedoch auf Kosten zusätzlicher Latenzzeiten und Energie, die für den Zugriff auf diese Speicher in der Fabric erforderlich sind.

[0005] Die herkömmliche Speicherhierarchie besteht in der Regel aus mehreren Cache-Ebenen, einem Hauptspeicher und einem Speicher. Ein herkömmlicher Ansatz besteht beispielsweise darin, Daten zu verar-

beiten, nachdem sie vom Speicher in die Caches verschoben wurden. Im Gegensatz dazu zielt das Near-Memory-Computing (NMC) darauf ab, Daten in der Nähe des Ortes zu verarbeiten, an dem sie sich befinden. Bei diesem datenzentrierten Ansatz werden die Recheneinheiten so nah wie logisch und/oder physisch möglich an den Speicher, auf dem die Daten liegen, verlagert, um die teuren Datenbewegungen zu minimieren. So ermöglicht beispielsweise die dreidimensionale Stapelung die Verarbeitung in der Nähe des Speichers. Durch das Stapeln von Logik und Speicher unter Verwendung von Through-Silicon-Vias (TSVs) kann NMC dazu beitragen, die Speicherzugriffslatenz und den Stromverbrauch zu verringern und eine wesentlich höhere Bandbreite bereitzustellen.

[0006] Die Verlagerung der Berechnungen in die Nähe des Speichers ermöglicht es, Leistungs- und Energieengpässe bei der Datenübertragung zu umgehen, indem die Beschränkungen der Pinanzahl der CPU/Speicher-Gehäuse-Schnittstelle umgangen werden. NMCs beschleunigen den Zugriff auf den Speicher mit Hilfe von Datenoperatoren, die Datenoperationen in der Nähe des Speichers durchführen und dadurch den Overhead mehrerer Verbindungswege zum und vom Speicher eliminieren. NMCs stellen die Verbindung zum Speicher über eine Speicherschnittstelle und nicht über eine Interconnect-Schnittstelle her. Die Speicherschnittstelle kann eine Latenzzeit von einigen Dutzend Nanosekunden bieten, im Gegensatz zu einer Latenzzeit von Hunderten von Nanosekunden (z.B. 300 ns) bei einer CPU-zu-CPU-Verbindungsschnittstelle oder Tausenden von Nanosekunden bei einer Netzwerk-Fabric-Schnittstelle. Durch den Einsatz von Datenoperatoren können NMCs die verfügbare Bandbreite effektiv erhöhen und die Latenzzeit für Lösungen zwischen Speicher und Rechenknoten verringern. Durch die Durchführung von Operationen in der Nähe des Speichers kann die Datenmenge für die Durchführung einer Datenanforderung, die zwischen dem Rechenknoten und dem Speicher übertragen werden muss, reduziert werden, was die Bandbreite der Verbindung effektiv erhöht und die Latenzzeit im Vergleich zur Anforderung aller Daten aus dem Speicher und der Durchführung von Operationen am Rechenknoten verringert. Datenoperatoren können in Form von Adressierung (z.B. Zeigerverfolgung), Filterung (z.B. Datenreduktion) oder komplexeren Berechnungen (z.B. Initialisierung, Verschlüsselung, Komprimierung, Projektionen usw.) erfolgen.

[0007] Ein Datenoperator kann zwischen der Speicher- und der Netzebene angeordnet sein und steuern, wie die Daten vom Speicher zum Netz gelangen, und so kontrollieren, wie die Daten aus einem Speicherstapel abgerufen werden. Der Speicherstapel kann einen disaggregierten Speicher enthalten, der eine Form des Scale-out-Speichers ist und aus einer bestimmten Anzahl von Speichergeräten besteht, die als logischer Speicherpool fungieren, der jedem Server im Netzwerk über eine Netzwerkstruktur zugewiesen werden kann. Rechenknoten (wie z.B. Client-Geräte) können auf den disaggregierten Speicher zugreifen, indem sie eine Verbindung mit einem oder mehreren Datenbetreibern herstellen. Wenn ein Client eine Anfrage stellt, leitet ein Netzwerk-Stack die Anfrage an einen Datenbetreiber weiter, der dem Rechenknoten zugeordnet ist, der die Datenanfrage initiiert hat.

[0008] Datenbanken sind in der einzigartigen Lage, disaggregierten Speicher zu nutzen, um sowohl die Probleme der ineffizienten Datenbewegung als auch der DRAM-Kapazität zu lösen. Die hier vorgestellten NMC-Implementierungen können die physische Trennung der Abfrageverarbeitung von der Speicherpufferverwaltung beinhalten. Traditionell greifen Abfrageverarbeitungs-Threads auf Basistabellen zu, indem sie diese aus einem Pufferpool lesen und die Daten in ihren privaten Arbeitsbereich kopieren. Gemäß den hierin offenbarten Implementierungen kann der Pufferpool auf dem netzwerkgebundenen disaggregierten Speicher platziert werden, wobei Abfrageverarbeitungsknoten bei Bedarf bereitgestellt werden, um eine Abfrage durch Lesen der Daten aus dem netzwerkgebundenen Pufferpool auszuführen. Die hierin offenbarten Implementierungen bieten mehrere Vorteile, wie z.B., aber nicht beschränkt auf (1) Verringerung der Datenbewegung durch Verlagerung von Operatoren in den disaggregierten Speicher, so dass die Verarbeitungsknoten nur die relevanten Daten erhalten; und (2) Verringerung des Speicherbedarfs für Rechenknoten durch Zentralisierung des Puffercaches im disaggregierten Speicher und Beseitigung des unnötigen Kopierens der Daten zu den Rechenknoten.

[0009] Im Gegensatz zum dezentralen Speicher bieten disaggregierte Speichersysteme einen an das Netzwerk angeschlossenen Speicher, der sich vom Speicher in den Rechenknoten unterscheidet, wie z.B., aber nicht nur, Client-Geräte. Rechenknoten können in verschiedenen Implementierungen Geräte sein, die ephemere Speicher-, Netzwerk-, Arbeitsspeicher- und Verarbeitungsressourcen bereitstellen, die disaggregierte Speicherinstanzen verbrauchen können. Bei diesem Ansatz kann der disaggregierte Speicher unabhängig von der Rechen- oder Speicherkapazität des Systems skaliert werden, und es ist nicht mehr erforderlich, eine Ressource zu viel bereitzustellen, um eine andere zu skalieren. In disaggregierten Speichersystemen ist der Netzwerk-Overhead ein Engpass, der die Leistung einschränkt.

[0010] Während die Optimierung der Gesamtleistung ein Motiv für die Nutzung disaggregierter Speichersysteme ist, können die energetischen Auswirkungen einer solchen Nutzung von gleichem oder größerem Nutzen sein, da die Nachfrage nach immer nachhaltigeren IT-Operationen steigt. Aufgrund der Nähe von NMC zu den Speicherknoten bietet die Verfügbarkeit von Schnittstellen mit großer Bandbreite eine potenzielle Reduzierung des Energieverbrauchs pro Datenbit für die Datenübertragung zwischen den Komponenten eines disaggregierten Speichersystems. Mit dem Aufkommen kostengünstiger integrierter photonischer Verbindungen im Datenübertragungspfad mit geringer Energie pro Datenbit und verlustfreier Übertragung über beliebige Entfernungen (z.B. bis zu Tausenden von Metern) werden Energiebewertungen auf Systemebene jedoch komplexer und schwieriger. Die Komplexität nimmt sogar noch zu, wenn photonische Schaltkreise zur Lenkung des Datenverkehrs eingesetzt werden. Bei NMCs, bei denen die Datenoperatoren keine feste Funktion haben, ist die Anzahl der Operatoren in der Regel begrenzt, und die begrenzte Anzahl von Operatoren muss für verschiedene Aufgaben gemeinsam genutzt werden. Die Gesamtenergieeffizienz könnte ein Faktor für die effektive Verwaltung dieser begrenzten Ressource sein.

[0011] So können beispielsweise disaggregierte Speichersysteme über ein Netzwerk als miteinander verbundene Speicherknoten verteilt sein, wobei die Rechenknoten zur Datenverarbeitung auf die Speicherknoten zugreifen. Im gesamten System kann es zahlreiche Variablen dafür geben, wie viel Energie systemweit und von einzelnen Komponenten verbraucht wird, um eine Rechenaufgabe durchzuführen und/oder Daten zwischen Komponenten auszutauschen. Die Bestimmung des Energieverbrauchs ist wesentlich komplizierter als bei einem herkömmlichen Speichersystem einer CPU, das über einen lokalen Speicher oder Cache verfügt, der an die CPU angeschlossen ist und Daten von einem zentralen Ort zur Verarbeitung durch die CPU abruft. Der Energieverbrauch kann in solchen konventionellen Systemen relativ niedrig sein, und die Bestimmung ist einfach, da die Messgrößen wie Latenzzeit, Bandbreite und Stromverbrauch bereits bekannt und festgelegt sind. Folglich hängt die Bestimmung davon ab, ob die Rechenaufgabe innerhalb der vom Betreiber definierten Leistungskriterien ausgeführt werden kann oder nicht, wenn die Metriken festgelegt und bekannt sind. Im Falle eines verteilten Speichersystems, das NMCs und Datenoperatoren nutzen kann, wird die Bestimmung viel komplizierter.

[0012] Implementierungen der vorliegenden Offenbarung sehen Mechanismen vor, mit denen die Auswirkungen des Energieverbrauchs von NMCs sowie von Datenbetreibern gemessen werden können, die sich aus der Ausführung von Berechnungen (z.B. in Form von Energie pro Datenbit zur Durchführung einer Aufgabe) und der Datenbewegung zwischen Komponenten (z.B. in Form von Energie pro übertragenem Datenbit) ergeben. Die Implementierungen der vorliegenden Offenbarung können auch die Auswirkungen des Zeitverbrauchs (oder der Latenz) für die Ausführung einer Aufgabe (z.B. in Form von Zeit) messen. Der Energieverbrauch und die Latenzzeit können als Optimierungsmetriken in disaggregierten Speicherarchitekturen verwendet werden, die zur Optimierung der Leistung von disaggregierten Speicherarchitekturen in Bezug auf Zeit und Energie genutzt werden können.

[0013] Die vorliegende Offenbarung sieht eine Verbrauchs-Engine vor, die Telemetriedaten sammelt, die sich auf den Energieverbrauch in Verbindung mit der Nutzung der NMC-Funktionalität zur Durchführung einer oder mehrerer Datenverarbeitungsaufgaben beziehen. Die Telemetriedaten können eines oder mehrere der folgenden Elemente enthalten: (i) Berechnungsenergie in Form von Energie, die bei der Durchführung einer oder mehrerer Aufgaben verbraucht wird, und Zeit, die für die Durchführung der einen oder mehreren Aufgaben erforderlich ist, und (ii) Kommunikationsenergie in Form von Energie, die für das Senden/Empfangen von Daten verbraucht wird, und Zeit, die dafür benötigt wird. In einer Beispielimplementierung kann die Verbrauchs-Engine als Teil des NMC enthalten sein. Die vorliegende Offenbarung ist jedoch nicht auf diese Implementierung beschränkt. Die Telemetriedaten können für jeden disaggregierten Speicher, die Speicher-zu-NMC-Schnittstelle, das NMC, die NMC-zu-Client-Schnittstelle, den Rechenknoten, die NMC-zu-Peer-NMC-Schnittstelle und das Peer-NMC erfasst werden.

[0014] Implementierungen gemäß der vorliegenden Offenbarung können auch eine Modellierungs-Engine umfassen, die so konfiguriert ist, dass sie Telemetriedaten verwendet, um eine Mehrzahl von Konfigurationen eines disaggregierten Speichersystems zur Durchführung einer oder mehrerer Datenverarbeitungsaufgaben zu modellieren. Beispielsweise kann die Modellierungs-Engine die Mehrzahl von Konfigurationen auf Basis einer Datenverarbeitungsanforderung oder einer Anzahl von Anforderungen modellieren, die von einem Rechenknoten empfangen wurden. Jede modellierte Konfiguration kann die Datenoperator-Ressourcen eines NMC in unterschiedlichem Ausmaß nutzen, wobei jedes Modell eine unterschiedliche Menge an Datenoperator-Ressourcen verwendet. In einem anschaulichen Beispiel umfassen die modellierten Konfigurationen (a) ein Basis-Verbrauchsmodell (manchmal auch als erstes Verbrauchsmodell bezeichnet), bei dem das NMC als Durchgang konfiguriert ist (z.B., keine Datenoperator-Ressourcen verbraucht werden, alle Berechnungen

am Rechenknoten durchgeführt werden) und (b) ein Maximalverbrauchsmodell (manchmal als zweites Verbrauchsmodell bezeichnet), bei dem die clientseitige Berechnung minimal oder null ist und im Wesentlichen alle Berechnungsfunktionen zur Durchführung der einen oder mehreren Aufgaben Datenoperator-Ressourcen des NMC verbrauchen (z.B. ist der Energieverbrauch der Rechenknoten-Ressourcen vernachlässigbar). Die Modellierungs-Engine kann eine Anzahl von Zwischenmodellkonfigurationen zwischen dem Basisverbrauchsmodell und dem Maximalverbrauchsmodell erzeugen, wobei jedes Zwischenmodell eine unterschiedliche Menge an Datenoperatorressourcen verwendet, um die eine oder mehrere Aufgaben auszuführen. Das Modellierungssystem kann so implementiert werden, dass es Telemetrie-Verbrauchsfaktoren für jede modellierte Konfiguration ausgibt. Die Telemetrie-Verbrauchsfaktoren können einen ersten Faktor umfassen, der den Energieverbrauch für die Ausführung der einen oder mehreren Aufgaben pro modellierter Konfiguration vorhersagt, und einen zweiten Faktor, der den Zeitaufwand für die Ausführung der einen oder mehreren Aufgaben pro modelliertem Szenario vorhersagt. In einer Beispielimplementierung kann die Modellierungs-Engine als Teil des NMC enthalten sein. Die vorliegende Offenbarung ist jedoch nicht auf diese Implementierung beschränkt.

[0015] Implementierungen gemäß der vorliegenden Offenbarung können auch eine Optimierungs-Engine umfassen, die so konfiguriert ist, dass sie eine modellierte Konfiguration aus der Mehrzahl der modellierten Konfigurationen zur Ausführung der einen oder mehreren Datenverarbeitungsaufgaben auswählt. Die Optimierungs-Engine kann auch einen oder mehrere der Mehrzahl der Datenoperatoren entsprechend der ausgewählten modellierten Konfiguration zuweisen. Beispielsweise kann die Optimierungs-Engine die Telemetrie-Verbrauchsfaktoren verwenden, um das NMC und die Verwendung der Datenoperatoren entsprechend der modellierten Konfiguration zu optimieren. Ein Kompromiss zwischen Zeit- und Energieverbrauch kann als Teil der Optimierung berücksichtigt werden, z.B. verbraucht die Durchführung von Berechnungen mit einer schnelleren Rate (z.B. weniger Zeit) im Allgemeinen mehr Energie. Dementsprechend kann die Optimierung des disaggregierten Speichersystems ein Abwägen zwischen der Minimierung des zweiten Faktors (z.B. der Zeit) und der Minimierung des ersten Faktors (z.B. des Energieverbrauchs) umfassen. Je nach Energieverbrauch oder der von einem Bediener gewünschten Zeit für die Durchführung von Aufgaben kann die Verwendung des NMC beispielsweise teilweise durch Bezugnahme auf die Telemetrie-Verbrauchsfaktoren und Auswahl einer optimalen modellierten Konfiguration optimiert werden. In einem Beispiel kann die Optimierung auf der Basis von Datenbetreibern durchgeführt werden, z.B. durch Zuweisung bestimmter Funktionen an Datenbetreiber zur lokalen Optimierung des Energieverbrauchs innerhalb des NMC. Die Optimierung kann z.B. die Auswahl einer von der Modellierungs-Engine modellierten Konfiguration umfassen, die eine optimale Leistung bietet, und die Zuweisung von Datenoperatoren entsprechend der ausgewählten Konfiguration. Die Optimierung kann auch global erfolgen, z.B. durch Zuweisung von Funktionen an gleichrangige NMC (und darin enthaltene Datenoperatoren) und/oder an Rechenknoten. In einer Beispielimplementierung kann die Optimierungs-Engine als Teil des NMC enthalten sein. Die vorliegende Offenbarung ist jedoch nicht auf diese Implementierung beschränkt.

[0016] Dementsprechend bieten die hier offenbarten Implementierungen Mechanismen zur Bestimmung des Energieverbrauchs und der Zeit bis zum Abschluss einer oder mehrerer Datenverarbeitungsaufgaben. Wenn sowohl der Energieverbrauch als auch die Zeit bis zur Fertigstellung bekannt sind, können Kunden und Dienstanbieter die Nachfrage und die Gewinnspanne entsprechend gestalten. Beispielsweise können Telemetriedaten, die als Optimierungsfaktor in Übereinstimmung mit den hier offenbarten Implementierungen zur Verfügung stehen, auch in die Ertragsstruktur von Kunden und Dienstleistern einbezogen werden. Dies ermöglicht eine Steigerung der Energieeffizienz und einen nachhaltigen Betrieb, der mit höheren Gewinnspannen angeboten wird. Darüber hinaus kann die Fähigkeit, den Energieverbrauch in disaggregierten Speichersystemen zu messen, Kunden und Dienstleistern eine fundierte Entscheidung beim Abwägen zwischen Leistung, Energie und Infrastrukturkosten ermöglichen. Das heißt, die hierin beschriebenen Implementierungen können implementiert werden, um die Folgen in Bezug auf den Energieverbrauch und die Zeit zu messen, die disaggregierte Speichersysteme benötigen, um eine Lösung zu erreichen, und daher können Kunden die hierin beschriebenen Implementierungen nutzen, um den Betrieb für einen niedrigen Energieverbrauch zu optimieren. In einem anderen Beispiel können Netzbetreiber die hier beschriebenen Implementierungen nutzen, um die Kosten zu minimieren und den Gewinn bei der Bereitstellung von Ressourcen zu maximieren, indem sie einen Kompromiss zwischen der Zeit bis zur Lösung und der verbrauchten Energie im Vergleich zu den Einnahmen aus dem Betrieb der Geräte finden.

[0017] Im Zusammenhang mit den Verbrauchsmodellen (oder der Konfiguration) beziehen sich die Begriffe „Baseline“ und „Maximum“ auf den Umfang der vom NMC durchgeführten Berechnungen und nicht notwendigerweise auf einen vorhergesagten oder beobachteten Energieverbrauch. Das heißt, das Basisverbrauchsmodell ist ein Modell, bei dem das NMC als Durchlauferhitzer arbeitet, und das Maximalverbrauchsmodell ist

ein Modell, bei dem im Wesentlichen alle Berechnungsfunktionen Datenbetreiberressourcen des NMC verbrauchen (z.B. maximale Berechnungen, die vom NMC durchgeführt werden). In einigen Fällen kann das Basisverbrauchsmodell das energieeffizienteste Modell sein, während in anderen Fällen das Modell des maximalen Verbrauchs das energieeffizienteste sein kann. In wieder anderen Fällen kann ein Zwischenverbrauchsmodell am energieeffizientesten sein.

[0018] Es sollte beachtet werden, dass die Begriffe „optimieren“, „optimal“ und dergleichen, wie sie hier verwendet werden, so verwendet werden können, dass sie die Leistung so effektiv oder perfekt wie möglich machen oder erreichen. Wie jedoch ein Fachmann, der dieses Dokument liest, erkennen wird, kann Perfektion nicht immer erreicht werden. Dementsprechend können diese Begriffe auch bedeuten, die Leistung so gut oder effektiv wie unter den gegebenen Umständen möglich oder praktikabel zu machen oder zu erreichen, oder die Leistung besser zu machen oder zu erreichen als die, die mit anderen Einstellungen oder Parametern erreicht werden kann.

[0019] **Fig.** 1 zeigt eine Beispielarchitektur 100, in der Datenabfrageoperationen an einen Speicher auf NMCs ausgelagert werden können. **Fig.** 1 umfasst eine Mehrzahl von Rechenknoten 105 (z.B. einen ersten Rechenknoten 105A und einen zweiten Rechenknoten 105B), die über eine Netzwerkschnittstelle 107 mit einem disaggregierten Speicher 108 verbunden sind. Rechenknoten 105 können im Rahmen von Datenoperationen Datenanfragen an den disaggregierten Speicher 108 stellen. Rechenknoten 105 können als jedes Rechengerät implementiert werden, das über die Netzwerkschnittstelle 107 mit dem disaggregierten Speicher 108 verbunden ist. Ein Rechenknoten 105 kann zum Beispiel ein Client-Gerät sein. Ein weiteres Beispiel: Der Rechenknoten 105 kann ein Server mit CPU, Speicher und Netzwerkschnittstellen sein. In einer Beispielimplementierung kann der Computerknoten 105 als Computersystem 600 von **Fig.** 6 implementiert werden. In einem anderen Beispiel kann eine Anzahl von Rechenknoten in Clustern organisiert werden, wobei jeder Cluster einen Rechenknoten 105 darstellt.

[0020] Der disaggregierte Speicher 108 umfasst einen logischen Pool von Speichern 110 (z.B. Speicher 110A, Speicher 110B, Speicher 110N usw.). Jeder Speicher 110 von **Fig.** 1 kann ein Beispiel für einen disaggregierten Speicherknoten auf einer Verbindungsstruktur sein. Beispielsweise kann eine Mehrzahl von disaggregierten Speicherknoten 110 über eine Mehrzahl von Verbindungsstrukturen verbunden sein. Dementsprechend kann der disaggregierte Speicher 108 in einigen Beispielen dem Zweck eines Speichers dienen, der von den Computerknoten 105 disaggregiert wurde und von einer Mehrzahl von Computerknoten 105 gemeinsam genutzt werden kann. In einigen Fällen kann der disaggregierte Speicher von allen Knoten gemeinsam genutzt werden. Die Netzwerkschnittstelle 107 kann ein Beispiel für eine Fabric-Schnittstelle sein.

[0021] Der disaggregierte Speicher 108 umfasst auch eine Mehrzahl dynamischer Regionen 121 (z.B. eine erste dynamische Region 121A, eine zweite dynamische Region 121B und eine dritte dynamische Region 121N). Jeder dynamische Bereich 121 (hier auch als NMC bezeichnet) umfasst einen oder mehrere Datenoperatoren 111 (z.B. einen ersten Datenoperator 111A, einen zweiten Datenoperator 111B, einen dritten Datenoperator 111C, einen vierten Datenoperator 111D, einen fünften Datenoperator 111E und einen sechsten Datenoperator 111N). Bei den Datenoperatoren 111 kann es sich um Funktionen handeln, die in die Nähe des Speichers verlagert werden, um die Leistung von Rechenoperationen zu verbessern, wie z.B. Adressierung (z.B. Zeigerverfolgung, Hashing usw.), Datenfilterung (z.B. Projektionen und Ansichten) und Datenberechnungen (z.B. Komprimierung, Verschlüsselung usw.). Datenoperatoren ändern möglicherweise nicht die gesamte auszuführende Verarbeitung, können aber verwendet werden, um Operationen in die Nähe eines Speichers 110 des disaggregierten Speichersystems 108 zu verlagern. In einem Beispiel umfassen die mehreren Datenoperatoren 111 Abfrageoperatoren, die Folgendes umfassen: (i) Auswahloperatoren, die so konfiguriert sind, dass sie Daten gemäß einer Sammlung von Prädikaten filtern, (ii) Projektionsoperatoren, die so konfiguriert sind, dass sie die zurückgegebenen Spalten reduzieren und Speicherzugriffe verringern, (iii) Aggregationsoperatoren, (iv) Unterscheidungsoperatoren, (v) Group-by-Operatoren, die so konfiguriert sind, dass sie Tupel kombinieren (z.B., (v) Operatoren, die so konfiguriert sind, dass sie Tupel kombinieren (z.B. distinct, group by aggregation), (vi) Operatoren, die reguläre Ausdrücke abgleichen, (vii) Verschlüsselungsoperatoren und (viii) Systemunterstützungsoperatoren, die so konfiguriert sind, dass sie Daten vor dem Senden von Daten an Ort und Stelle verarbeiten (z.B. Verschlüsselung/Entschlüsselung) und Systemoptimierungsaufgaben wie das Packen von Daten durchführen, um die Gesamtnetznutzung zu verringern. Die Datenoperatoren 111 können mit ASICs, FPGAs, SmartNICs, ARM, anderen Prozessoren usw. implementiert werden. In einer Beispielimplementierung kann ein Datenoperator 111 als eine Instanz eines Computersystems implementiert werden, z.B. das Computersystem 600 von **Fig.** 6.

[0022] In einigen Implementierungen umfasst jede dynamische Region 121 einen oder mehrere Datenspeicher 112 (z.B. Datenspeicher 112A, Datenspeicher 112B ... Datenspeicher 112N). In den Datenspeichern 112 können Metriken gespeichert werden, z.B. Optimierungsmetriken gemäß den hier offenbarten Implementierungen. In den Datenspeichern 112 können auch Richtlinien zum Zuweisen, Laden und Verwalten von Datenoperatoren 111 gespeichert werden, die in den entsprechenden dynamischen Bereich 121 geladen werden. In den Datenspeichern 112 können beispielsweise Metriken für den Energieverbrauch pro Bit und Latenzmetriken gespeichert werden, die von den hier offenbarten Implementierungen genutzt werden können, um Datenoperatoren für die Ausführung angeforderter Aufgaben gemäß den Richtlinien zu optimieren.

[0023] Auf den disaggregierten Speicher 108 kann von einem oder mehreren der Mehrzahl von Rechenknoten 105 zugegriffen werden. Beispielsweise können sowohl der erste Rechenknoten 105A als auch der zweite Rechenknoten 105B auf den disaggregierten Speicher 108 zugreifen und/oder einer von dem ersten Computerknoten 105A und dem zweiten Computerknoten 105B kann auf den disaggregierten Speicher 108 zugreifen. In einem Beispiel können der Rechenknoten 105A und der Computerknoten 105B jeweils eine Datenanforderung in Form einer Projektion und einer Auswahl von zwei Datenabfragen aufrufen, die als Datenoperatoren 111 eines entsprechenden dynamischen Bereichs 121 in den disaggregierten Speicher 108 ausgelagert werden, während eine gemeinsame und endgültige Projektion an den Rechenknoten 105A und 105B ausgeführt wird. In einem Beispiel enthält der disaggregierte Speicher 108 eine Programmierschnittstelle mit einer Daten-API für Pfadoperationen und Verbindungsmanagementoperationen. Durch die Verbindung der Daten-API mit den Speichern 110 können die Datenoperatoren 111 in den disaggregierten Speicher 108 verlagert werden, der dann als Bump-in-the-Wire-Stream-Prozessor fungiert.

[0024] Datenoperator-Pipelines können aus einzelnen Blöcken aufgebaut werden, die einen bestimmten Datenoperator implementieren und Standardschnittstellen zur Kombination der Datenoperatoren in Pipelines bieten. Der modulare Charakter der Pipelines ermöglicht es, die Operatoren leicht auszutauschen und zu erweitern (z.B. Verknüpfungsoperatoren). Ein Beispiel für eine Datenoperator-Pipeline wird unten in Verbindung mit **Fig.** 3 dargestellt.

[0025] **Fig.** 2 ist ein schematisches Blockdiagramm der Architektur 100, in der die hier beschriebenen Systeme und Verfahren implementiert werden können. Die Architektur 100 umfasst einen Speicherstapel 204, einen Datenoperatorstapel 206 und einen Netzwerkstapel 202. Daten werden von einem oder mehreren Rechenknoten (z.B. Rechenknoten 105A, Rechenknoten 105B) über den Netzwerkstapel 202 und den Datenoperatorstapel 206 zum und vom Speicherstapel 204 übertragen. Der Speicherstapel 204 enthält disaggregierten Speicher (z.B. disaggregierter Speicher 108), der einen Speicherpool (z.B. Speicher 110A, Speicher 110B ... Speicher 110N) umfasst, der über eine Mehrzahl von Kanälen (z.B. Kanal 256A, Kanal 256B ... Kanal 256N) mit einer Speicherverwaltungseinheit 208 verbunden ist. Der Datenoperatorstapel 206 umfasst eine Mehrzahl von Datenoperatoren 111 (z.B. Datenoperator 111A, Datenoperator 111B ... Datenoperator 111N) und einen oder mehrere Datenspeicher 112. Die Mehrzahl der Datenoperatoren 111 sind mit der Speicherverwaltungseinheit 208 und der Netzwerkschnittstelle 107 verbunden. Die mehreren Datenoperatoren 111 sind über mehrere Speicherkanäle 236 (z.B. Kanal 236A, Kanal 236B ... Kanal 236N) mit der Speicherverwaltungseinheit 208 verbunden. Die Kanäle 236 können eine Speicherschnittstelle darstellen, über die der Datenoperatorstapel 206 (z.B. ein Beispiel für eine dynamische Region 121) mit dem Speicherstapel 204 verbunden ist. Der Netzwerkstapel 202 ist so konfiguriert, dass er den eingehenden Datenstrom dekodiert und die Daten an den entsprechenden Datenoperator 111 weiterleitet. Der Netzwerkstapel 202 umfasst eine Netzwerkschnittstelle 107. Die Netzwerkschnittstelle 107 ist über eine Mehrzahl von Kanälen 234 (z.B. Kanal 234A, Kanal 234B ... Kanal 234N) mit der Mehrzahl der Datenbetreiber 111 verbunden. In einer Konfiguration ist die Netzwerkschnittstelle 107 ein Beispiel für ein Fabric, das so konfiguriert ist, dass es Daten über eine Mehrzahl von Interconnect Fabrics an jeden Datenbetreiber der Mehrzahl von Datenbetreibern 111 überträgt. In diesem Fall können die Kanäle 234 Beispiele für Fabric-Schnittstellen sein. Die Mehrzahl der Verbindungsstrukturen kann mit einem oder mehreren disaggregierten Speicherknoten verbunden sein. Die Netzwerkschnittstelle 107 gemäß der hier offenbarten Implementierung kann den Datenoperatorstapel 206 mit einem oder mehreren Rechenknoten 105, wie in **Fig.** 1 dargestellt, sowie mit anderen gleichrangigen Datenoperatorstapeln 206 verbinden. So kann die Netzwerkschnittstelle 107 beispielsweise als Client-Fabric fungieren, die eine dynamische Region über Client-Fabric-Schnittstellen mit Rechenknoten verbindet, und gleichzeitig als Peer-Fabric, die eine dynamische Region (z.B. die dynamische Region 121A) mit einer oder mehreren dynamischen Peer-Regionen (z.B. den zweiten dynamischen Regionen 121B, 121N usw.) verbindet.

[0026] Der Netzwerk-Stack 202 ist so konfiguriert, dass er Daten an den Datenbetreiber-Stack 206 sendet oder von ihm empfängt. Beispielsweise empfängt der Netzwerk-Stack 202 eine Datenabfrage (z.B. von

einem Rechenknoten) und leitet sie an den Datenbetreiber-Stack 206 weiter. Der Netzwerk-Stack 202 verwaltet alle Verbindungen zwischen den Rechenknoten und der Mehrzahl der Datenbetreiber 111. Der Netzwerk-Stack 202 empfängt/sendet Daten über die Netzwerkschnittstelle 107. Über die Netzwerkschnittstelle 107 können Rechenknoten in einem Netzwerk Daten im Hauptspeicher unter Umgehung des Betriebssystems und der CPU-Ressourcen austauschen. In einem Beispiel enthält die Netzwerkschnittstelle 107 ein Verbindungsprotokoll, das Zero-Copy-Networking verwendet, um Daten direkt aus dem Hauptspeicher eines Systems zu lesen und Daten direkt in das Hauptnetzwerk eines anderen Systems zu schreiben, wodurch Leistung und Durchsatz verbessert werden, indem Server-Ressourcen in parallelen HPC-Clustern (High-Performance Computing) freigesetzt werden.

[0027] Der Operator-Stack 206 enthält die dynamische Logik, die erforderlich ist, um dem disaggregierten Speicher Datenoperatoren zuzuweisen. Jeder Datenoperator 111 kann so konfiguriert sein, dass er NMC-Operationen (z.B. speichernahe Verarbeitungsoperationen) durchführt. Der Datenoperatorstapel 206 kann Daten für mehrere Rechenknoten gleichzeitig verarbeiten. So können die Datenoperatoren 111 beispielsweise eine Reihe von Datenabfragen verarbeiten. Die Daten können innerhalb der Operator-Pipelines in einem Streaming-Verfahren verarbeitet werden, bei dem jeder Verarbeitungsschritt vollständig in einer Pipeline erfolgt, so dass verschiedene Verarbeitungsschritte innerhalb jedes Datenoperators 111 gleichzeitig stattfinden können. Diese Konfiguration kann verwendet werden, um sowohl die räumliche Parallelisierung (z.B. die gleichzeitige Ausführung von Aufgaben durch eine Mehrzahl von Verarbeitungseinheiten) durch gleichzeitige dynamische Regionen als auch die Pipeline-Parallelisierung zu nutzen. Die Pipeline-Parallelisierung kann sich auf mehrere voneinander abhängige Operatoren beziehen, wobei sich die Ausführung jedes Operators mit anderen Datenoperatoren 111 überschneiden kann. Beides kann zur Gesamtleistung des Systems beitragen. Über den Operatorstapel kann das System Beziehungen zu zuvor identifizierten Strukturen erkennen und Optimierungen für den Datenzugriff vornehmen. Zu den Optimierungen gehören das Prefetching von Daten, auf die zugegriffen werden soll, die Zwischenspeicherung der Daten in einem übergeordneten Cache, der häufig wiederverwendet werden soll, und/oder die Vorberechnung durch spekulative Aktualisierung von Metadaten, um die bevorstehende IO, Pufferzuweisung oder andere Parameter im Zusammenhang mit der Anwendung und den Daten zu berücksichtigen.

[0028] Die Mehrzahl der Datenoperatoren 111 kann über einen Management-Operator gemäß den in den Datenspeichern 112 gespeicherten Richtlinien und Indikatoren verwaltet werden. In einem Beispiel kann ein Management-Operator einer der Operatoren 111 sein, der zuerst geladen wird und auf Basis des ersten Ladens so konfiguriert ist, dass er andere, später geladene Datenoperatoren 111 verwaltet. Der Verwaltungsoperator überwacht die Verarbeitungslast der anderen Datenoperatoren und reagiert auf Basis von Richtlinien. Der Management-Operator kann auch eine Programm- und Benutzerschnittstelle zu anderen Tools und Systemverwaltungen bereitstellen. In einem anderen Beispiel kann ein Management-Operator ein Controller sein, der z.B. als Rechengerät 600 von **Fig.** 6 implementiert werden kann.

[0029] In einem Beispiel ist der Verwaltungsoperator der erste Datenoperator, der hochfährt (z.B. sich einschaltet, startet, online geht usw.). In Konfigurationen, in denen der erste Datenoperator, der hochfährt, der Managementoperator ist, kann der erste Operator als „Lead Operator“ bezeichnet werden. Der Management-Operator bleibt während des gesamten Datenbetriebs eingeschaltet. Wenn ein Management-Operator ausfällt, kann jeder beliebige Operator aus einer Mehrzahl von Operatoren den Management-Operator übernehmen. Wenn z.B. ein Management-Operator ausfällt, kann ein Nicht-Management-Operator zum Management-Operator werden.

[0030] Der Datenoperatorstapel 206 sendet/empfängt Daten an/von dem Speicherstapel 204 über eine Mehrzahl von Verbindungen 236. Der Speicherstapel 204 implementiert den Speicherpool 110 und kann als regulärer Speicher verwendet werden, wobei die Daten nach Bedarf aus dem Speicher geladen werden. Der Speicherstapel 204 ist so konfiguriert, dass er Speicherzuweisungen, Adressübersetzungen und gleichzeitige Zugriffe handhabt. Der Speicherstapel 204 umfasst eine Speicherverwaltungseinheit 208 zur Übersetzung von Adressen in den Speicher. Die Speicherverwaltungseinheit 208 ist mit dem Speicher 110 innerhalb des Speicherstapels 204 über eine Mehrzahl von Kanälen verbunden (z.B. Kanal 256A, Kanal 256B ... Kanal 256N usw.). Beispielsweise kann die Mehrzahl von Kanälen 256 einen Speicher 110 zum Zugriff auf die Speicherverwaltungseinheit 208 leiten. Jedem Kanal kann Speicher in einem Streifenmuster zugewiesen werden, um die verfügbare Bandbreite zu optimieren. Der Operator-Stack 206 kann eine Schnittstelle zum Speicher-Stack 204 bilden, um Daten und Metadaten vom Speicher-Stack 204 zu empfangen.

[0031] Rechenknoten können auf den disaggregierten Speicher zugreifen, indem sie eine Verbindung zu einer der dynamischen Regionen 121 öffnen, die einen oder mehrere Datenoperatoren 111 enthalten, die

jeweils eine der möglichen Operator-Pipelines enthalten können. Wenn ein Rechenknoten eine Leseanforderung stellt, leitet der Netzwerk-Stack 202 die Anforderung an eine dynamische Region im Operator-Stack 206 weiter, die Datenoperatoren 111 enthält, die dem Rechenknoten zugewiesen sind, der die Anforderung initiiert hat. Die Leseanforderung wird an den Speicherstapel 204 weitergeleitet, der eine virtuelle Adresse in eine physische Adresse in einem Speicher 110 übersetzt und dann die eigentliche Datenanforderung an den Speicher 110 ausgibt. Die Rechenknoten können über lokale Kataloginformationen verfügen, die zur Bestimmung der virtuellen Adressen der Speicher verwendet werden können, auf die zugegriffen werden soll. Die zurückgegebenen Daten werden zurück in den dynamischen Bereich 121 geleitet, wo geladene Datenoperatoren 111 auf die Daten angewendet werden. Schließlich werden die resultierenden Daten an den Netzwerkstapel 202 weitergeleitet und dann direkt an den Rechenknoten gesendet.

[0032] **Fig.** 3 zeigt eine beispielhafte dynamische Region 121a des Datenoperatorstapels 206. **Fig.** 3 zeigt eine generische Operator-Pipeline 330, die in einem dynamischen Bereich 121 enthalten ist und eine Mehrzahl von Datenoperatoren 332A-332N (hier zusammenfassend als Datenoperatoren 332 bezeichnet) enthält, die ein Datenobjekt verarbeiten können. In einem Beispiel kann die Mehrzahl der Operatoren 332 Projektion, Auswahl (z.B. Prädikatsauswahl, Abgleich mit regulären Ausdrücken), Gruppierung (z.B. unterscheidbar, gruppiert nach und Aggregation) und Systemunterstützung (z.B. Verschlüsselung/Entschlüsselung) umfassen. Im anschaulichen Beispiel von **Fig.** 3 umfasst die Mehrzahl der Operatoren einen Entschlüsselungsoperator 332A, einen Projektionsoperator 332B, einen Auswahloperator 332C, einen Gruppierungsoperator 332D und einen Systemoperator 332N. Die Operatoren 332 innerhalb der Operator-Pipeline 330 können in Abhängigkeit von der angeforderten Menge der auszuführenden Abfragen variieren. Ein oder mehrere Operatoren 332 können feste Funktionen oder programmierbare Funktionen haben, die im Fall von programmierbaren Funktionen neu zugewiesen werden können.

[0033] Jede Operator-Pipeline ist einem entsprechenden Rechenknoten zugewiesen (z.B. einem der entsprechenden Rechenknoten 105). Die Operator-Pipeline 330 umfasst einen oder mehrere Datenoperatoren 332, von denen jeder eine teilweise Abfrageverarbeitung (z.B. Verarbeitung von Datenleseoperationen) von Datenoperationen im disaggregierten Speicher bereitstellt. Die Abfrage kann über den Query Request Handler 302 an den Query Request Handler im Operator Stack weitergeleitet werden. Abfrageanforderungs-Handler 302 kann das mit der Abfrage verbundene Datenobjekt aus dem Speicherstapel (z.B. Speicherstapel 204) anfordern. Dies kann durch Adressierung erreicht werden, die es ermöglicht, Daten sowohl in Zeilenspeicher- als auch in Spaltenspeicherformaten basierend auf der gegebenen Anfrage zu lesen. In einigen Ausführungsformen können die Daten auch in Diagrammformaten oder anderen Darstellungen gelesen werden. Die Datenoperatoren 332, die für die Verarbeitung des Datenobjekts verwendet werden, können vorkompiliert und zur Laufzeit in der dynamischen Region 121 bereitgestellt (z.B. geladen) werden. Der Query Request Handler 302 kann das Datenobjekt je nach Abfrage an einen oder mehrere Datenoperatoren 332 übergeben.

[0034] Wenn eine Abfrage eintrifft, wird sie zunächst an den Query Request Handler 302 weitergeleitet, der die Daten aus dem Speicherstapel anfordert. Gleichzeitig werden alle erforderlichen Parameter für die weitere Verarbeitung an die Datenoperatoren 332 in der Pipeline weitergeleitet. Die aus dem Speicher ankommenden Daten werden von diesen Datenoperatoren in einem Streaming-Verfahren verarbeitet. Sobald die Verarbeitung abgeschlossen ist, werden die resultierenden Daten über den Netzstapel an den Client zurückgesendet.

[0035] In einer beispielhaften Ausführungsform kann der Entschlüsselungsoperator 332A die vom Speicherstapel 204 eingehenden Daten entschlüsseln, wenn die Daten verschlüsselt sind. Für diesen Vorgang müssen die Rechenknoten möglicherweise bestimmte Schlüssel bereitstellen, die für die Entschlüsselung von verschlüsseltem in Klartext verwendet werden. Diese Schlüssel können mit der Abfrageanforderung bereitgestellt werden. Auf diese Weise kann das System zusätzliche Sicherheitsebenen bereitstellen, was in modernen disaggregierten Umgebungen wichtig ist.

[0036] Der Projektionsoperator 332B kann eine Teilmenge des Datenobjekts zurückgeben. Der Projektionsoperator 332B kann eine Tabelle aus dem disaggregierten Speicher lesen, den eingehenden Datenstrom auf Basis von Abfrageparametern, die Tupel und deren Größe beschreiben, analysieren und nur die erforderlichen Spalten in die Pipeline zur weiteren Verarbeitung mit Hilfe von Anmerkungen projizieren. Auf diese Weise wird die Datenmenge, die den nachfolgenden Stufen zur Verfügung gestellt wird, verringert und letztlich die Gesamtmenge der Datenbewegungen reduziert. Der Projektionsoperator 332B kann die Tupel mit Parametern aus der angeforderten Abfrage beschriften. Die Parameter können angeben, welche Spalten Teil der Projektions-, Auswahl- und Gruppierungsphasen sind. Der Projektionsoperator 332B kann die Teilmenge des Datenobjekts mit Anmerkungen versehen und sie an den Auswahloperator 332C weiterleiten.

[0037] Der Selektionsoperator 332C kann die Daten weiter filtern, um die letztlich über das Netz gesendete Datenmenge zu verringern und so den E/A-Overhead zu reduzieren. Der Auswahloperator 332C kann zum Filtern der Daten die Prädikatsauswahl, den Abgleich mit regulären Ausdrücken, die Vektorisierung oder andere Methoden verwenden. Die Prädikatsauswahl kann durch den Vergleich des Wertes eines Attributs des Datenobjekts mit einer in der Abfrage angegebenen Konstante erfolgen. Die Anmerkungen des Projektionsoperators 332B können dabei helfen, zu bestimmen, was in der Phase des Prädikatsabgleichs ausgewertet wird. Der Abgleich mit regulären Ausdrücken kann den Abgleich von Zeichenketten über mehrere parallele Engines im Operatorstapel 206 verwenden. Die Leistung dieses Operators kann durch die Länge der Zeichenkette bestimmt werden und hängt nicht von der Komplexität des verwendeten regulären Ausdrucks ab. Die Vektorisierung kann durch paralleles Lesen von Daten aus mehreren Kanälen im Operatorstapel 206 erfolgen. Einzelne Tupel können an eine Reihe von parallel arbeitenden Auswahloperatoren ausgegeben werden. Die Anzahl der parallelen Operatoren kann auf Basis der Anzahl der Speicherkanäle und der Tupelbreite gewählt werden. Für einfachere Abfragen, die ohne Datenabhängigkeiten parallelisiert werden können, kann eine Vektorisierung implementiert werden.

[0038] Der Gruppierungsoperator 332D kann eine Aggregation durchführen und wiederholte Spalteneinträge eliminieren, bevor sie an die Client-Anwendung und das Gerät gesendet werden. Der Gruppierungsoperator 332D kann die Werte mit einem Hash-Verfahren versehen und die Einträge in einer Hash-Tabelle im Speicherstapel speichern. In einigen Beispielen kann die Hash-Tabelle Cuckoo-Hashing mit mehreren Hash-Tabellen verwenden, die parallel überprüft werden können. Beim Cuckoo-Hashing können Einträge aus einer Hash-Tabelle entfernt und in eine separate Hash-Tabelle mit einer anderen Funktion eingefügt werden, um Kollisionen zu vermeiden. Der Gruppierungsoperator 332D kann Daten durch Aggregation gruppieren und Dateneinträge auf Basis der angeforderten Aggregationsergebnisse aus der Hashtabelle löschen. Die Aggregation kann zu spezifischen Gruppierungen führen, die einen großen Teil der Verarbeitung bereits im disaggregierten Knoten durchführen können (der auf Basis der jeweiligen Anwendung bestimmt werden kann), ohne dass alle Daten zunächst zum Client übertragen und dann vollständig verarbeitet werden müssen.

[0039] Der Systemoperator 332N kann die durch den Projektionsoperator, den Auswahloperator und den Gruppierungsoperator gefilterten Daten so aufbereiten, dass sie an den Rechenknoten gesendet werden können. Der Systemoperator 332N kann die Daten auch nach Bedarf oder auf Anforderung des Rechenknotens verschlüsseln. Der Systemoperator 332N kann die Daten auch auf Basis der vom Projektionsoperator 332B mit Anmerkungen versehenen Spalten packen, was zu einer effizienteren Nutzung der verfügbaren Netzwerkbandbreite führt. Beim Packen kann ein Überlaufpuffer verwendet werden, um die Leitungsrate aufrechtzuerhalten. Im Falle der Vektorisierung können die Tupel aus jeder der parallelen Pipelines mit einem Round-Robin-Arbiter kombiniert werden. Der Systemoperator 332N kann auch eine Absendereinheit umfassen, um die korrekten Header-Informationen im Netzwerkstapel zu erzeugen. Die Absendereinheit kann RDMA-Befehle ohne Informationen über die endgültige Datengröße erstellen, was eine Filterung in den Operatoren ermöglichen kann, wenn die endgültige Datengröße zum Zeitpunkt der Ausgabe der Anforderung nicht bekannt ist. Die Absendereinheit kann die Daten auch an den Netzwerkstapel weiterleiten, um sie an die Client-Anwendung und das Gerät zu senden.

[0040] **Fig.** 4 ist ein schematisches Boxdiagramm eines disaggregierten Speichersystems 400 zur Optimierung auf Basis von Telemetriedaten gemäß den hierin offenbarten Implementierungen. Das disaggregierte Speichersystem 400 umfasst einen NMC 421, der sich über einen Speicherkanal 401 in der Nähe eines Speicherarrays 410 befindet. Der NMC 421 kann als eine in einen Speicherchip integrierte Funktion implementiert werden (z.B. am unteren Ende eines Stapels von Speicherchips in einer einzelnen verpackten Komponente). Ein weiteres Beispiel ist, dass der NMC 421 als Chiplet auf einem Substrat neben einem Speicherchip implementiert sein kann. In einem anderen Beispiel kann die NMC 421 eine verpackte Komponente auf einer Speicherplatine in der Nähe von Speicherkomponenten (z.B. DIMM oder SSD) sein. In einem weiteren Beispiel kann die NMC 431 als Universal-CPU implementiert werden, die mit ihrem eigenen Hauptspeicher gekoppelt ist, in dem Anweisungen gespeichert sind, die, wenn sie von der CPU ausgeführt werden, die hierin offenbarten Prozesse und Funktionen ausführen. In einem Beispiel kann die NMC 431 als Computersystem 600 implementiert werden, das unten in Verbindung mit **Fig.** 6 beschrieben wird. Die NMC 421 kann in einigen Implementierungen als speichernaher Computerknoten bezeichnet werden.

[0041] Die Speicheranordnung 410 und NMC 421 können Beispiele für einen der Speicher 110 bzw. den dynamischen Bereich 121 sein. Der Speicherkanal 401 kann eine Beispielimplementierung der Speicherkanäle 236 sein. Der NMC 421 ist über eine Client-Fabric 406 mit einem oder mehreren Rechenknoten 405 und über eine Peer-Fabric 432 mit einem oder mehreren Peer-NMCs 431 verbunden. Die Client-Fabric 406 kann ein Beispiel für eine Netzwerkschnittstelle 107 sein, und ein oder mehrere Rechenknoten 405 können

Beispiele für Rechenknoten 105 von **Fig.** 1 sein. Peer NMC 431 können Beispiele für Instanzen der dynamischen Region 121 sein und die Peer Fabric 432 kann als Netzwerkschnittstelle 107 implementiert werden. Die Peer Fabric 432 und die Client Fabric 406 können beispielsweise als ein einziges Netzwerk vereinigt werden, wie in **Fig.** 1 dargestellt. In einem anderen Beispiel können die Peer Fabric 432 und die Client Fabric 406 getrennt und somit als separate Netzwerke implementiert sein.

[0042] Jede Komponente des disaggregierten Speichersystems 400 verbraucht Energie, um datenbezogene Funktionen in einer jeweiligen Komponente auszuführen. Beispielsweise kann die Speichermatrix 410 pro Datenbit Energie verbrauchen, um einen Lese- oder Schreibvorgang auszuführen, sowie statische Energie, statische Energie aufgrund von Leckagen oder laufende Funktionen wie die Auffrischung verbrauchen. Der Speicherkanal 401 verbraucht Energie pro Datenbit, das über die Schnittstelle mit einer bestimmten Bandbreite übertragen wird. In ähnlicher Weise verbrauchen die Client Fabric 406 und die Peer Fabric 432 Energie pro Datenbit, das über die Schnittstelle mit einer bestimmten Bandbreite übertragen wird. Der eine oder die mehreren Rechenknoten 405 und der eine oder die mehreren Peer-NMCs 431 verbrauchen ebenfalls Energie, um Lese- oder Schreibvorgänge durchzuführen, und verbrauchen statische Energie aufgrund von Leckagen.

[0043] Der NMC 421 verbraucht auch während des Betriebs und/oder im Leerlauf Energie. Beispielsweise umfasst das NMC 421 eine Speicherschnittstelle 424, die das NMC 421 mit dem Speicherkanal 401 verbindet, und eine Fabric-Schnittstelle 425, die das NMC 421 mit der Client-Fabric 406 und/oder der Peer-Fabric 432 verbindet. Ein oder mehrere Datenoperatoren 411 werden als verbrauchbare Ressourcen bereitgestellt, die kommunikativ zwischen der Speicherschnittstelle 424 und der Fabric-Schnittstelle 425 verbunden sind, um die oben beschriebenen Funktionen auszuführen. Die ein oder mehreren Datenoperatoren 411 können Beispielimplementierungen der Datenoperatoren 111 von **Fig.** 1 sein. Insgesamt verbrauchen die Datenoperatoren 411 Energie, um zugewiesene Funktionen auszuführen. Die Datenoperatoren 411 können feste Funktionen oder programmierbare Funktionen haben, die im Falle von programmierbaren Funktionen neu zugewiesen werden können. Der Energieverbrauch der einzelnen Datenoperatoren 411 hängt davon ab, dass der jeweilige Datenoperator 411 zur Ausführung der zugewiesenen Funktion aufgerufen wird. Der NMC 421 verbraucht auch statische Energie aufgrund von Leckagen. Die Speicherschnittstelle 424 und die Fabric-Schnittstelle 425 verbrauchen jeweils Energie pro Datenbit, das über die jeweilige Schnittstelle übertragen wird, sowie statische Energie aufgrund von Leckagen.

[0044] Das NMC 421 ist in **Fig.** 4 so dargestellt, dass es Entscheidungspunkte 429a und 249b (hier gemeinsam als Entscheidungspunkte 429 bezeichnet) für einen Vorgangsablauf enthält. Wenn die Datenoperatoren 411 für die Durchführung eines Vorgangs nicht erforderlich sind oder keine Ressourcen für die Datenoperatoren 411 zur Verfügung stehen, wird der Vorgangsfluss über den Pfeil 426 durch die NMC 421 geleitet. Andernfalls ruft der Betriebsablauf einen oder mehrere Datenoperatoren 411 über die Pfeile 427a und 427b auf. Dieser Vorgang wird weiter unten ausführlicher beschrieben,

[0045] Die nachstehende Tabelle 1 enthält eine nicht erschöpfende Auflistung einer Reihe von Energieverbrauchskennzahlen im Zusammenhang mit der Durchführung von Berechnungsaufgaben (im Folgenden manchmal als Berechnungskennzahlen bezeichnet).

Tabelle 1

Berechnungsmetriken		
Metrisch	Beschreibung	Messbare Einheit
Ea_w	Energieverbrauch des Speicherarrays pro Schreibvorgang	(J/Bit)
Ea_r	Energieverbrauch des Speicherarrays pro Lesevorgang	(J/Bit)
Pa_s	Statische Leistungsaufnahme des Speicherarrays	(W)
Bm_r	# Anzahl der Bits, die aus dem Speicherfeld in den NMC gelesen werden	(Bits)
Bm_w	# Anzahl der vom NMC in das Speicherfeld geschriebenen Bits	(Bits)
Emc_T	Energieverbrauch des Speicherkanals pro übertragenem Bit	(J/Bit)
BWmc	Speicher-Kanal-Bandbreite	(bit/s)

Berechnungsmetriken		
Metrisch	Beschreibung	Messbare Einheit
Em_T	Pro übertragenem Bit verbrauchte Energie der Speicherschnittstelle	(J/bits)
Pm_s	Statische Leistungsaufnahme der Speicherschnittstelle	(W)
Eo[n] [i]	NMC-Energieverbrauch pro verarbeitetem Bit [i] pro Datenoperator [n]	(J/bits)
Po_s	Statische Leistungsaufnahme des NMC	(W)
Bo_r [n][i]	NMC-Bits, die für eine Leseoperation pro Datenoperator [n] pro Bit [i] verarbeitet werden	(bit)
Bo_w [n][i]	NMC-Bits, die für den Schreibvorgang pro Datenoperator [n] pro Bit [i] verarbeitet werden	(bit)
Ef_T	Energieverbrauch der Fabric-Schnittstelle pro übertragenem Bit	(J/bits)
Pf_s	Statische Leistungsaufnahme der Fabric-Schnittstelle	(W)
Ecf_T [n][i]	Energieverbrauch der Clientstruktur pro übertragenem Bit [i] pro Rechenknoten [n]	(J/Bit)
BWcf[n][i]	Client-Fabric-Bandbreite pro Rechenknoten [n] pro Bit [i]	(bit/s)
Bc_r [n][i]	# Anzahl der Bits [i], die bei einem Lesevorgang pro Rechenknoten [n] übertragen werden	(bit)
Bc_w [n][i]	# Anzahl der für einen Schreibvorgang übertragenen Bits [i] pro Rechenknoten [n]	(bit)
Epf_T [m][i]	Peer-Fabric-Energieverbrauch pro übertragenem Bit [i] pro Peer-NMC [m]	(J/Bit)
BWpf[m][i]	Peer-Fabric-Bandbreite pro Peer NMC [m] pro Bit [i]	(bit/s)
Bp_r	# Anzahl der übertragenen Bits bei Leseoperationen pro Peer NMC	(bit)
Bp_w	# Anzahl der übertragenen Bits bei Schreibvorgängen pro Peer NMC	(bit)

[0046] Jede Komponente des disaggregierten Speichersystems 400 benötigt Zeit, um Funktionen auszuführen, bevor sie Daten an eine nachfolgende Komponente überträgt (z.B. Latenzzeit der einzelnen Komponenten). Beispielsweise kann die Speichermatrix 410 eine Speichermatrix-Lese- oder Schreiblatenz für die Zeit haben, die für die Ausführung eines Lese- bzw. Schreibvorgangs benötigt wird. Der Speicherkanal 401 kann eine Speicherkanal-Latenzzeit für die Zeitdauer der Datenübertragung zwischen dem Speicherfeld 410 und dem NMC 421 aufweisen. Die Speicherschnittstelle 424 kann eine Speicherschnittstellenlatenz für eine bestimmte Zeit haben, um Daten zwischen dem Speicherkanal 401 und dem NMC 421 zu übertragen. Der NMC 421 kann eine NMC-Latenzzeit für die Zeit haben, die für die Ausführung von Funktionen auf aufgerufenen Datenoperatoren 411 (falls vorhanden) benötigt wird. Die Fabric-Schnittstelle 425 kann eine Fabric-Schnittstellenlatenz für eine bestimmte Zeit haben, um Daten zwischen NMC 421 und Peer Fabric 432 und/oder Client Fabric 406 zu übertragen. Die Client-Fabric 406 und die Peer-Fabric 432 können jeweils eine Latenzzeit für eine Zeitspanne haben, um eine Datenübertragung zwischen dem NMC 421 und einem oder mehreren Rechenknoten 405 bzw. einem oder mehreren Peer-NMCs 431 durchzuführen.

[0047] Die nachstehende Tabelle 2 enthält eine nicht erschöpfende Auflistung einer Reihe von Latenzmetriken.

Tabelle 2

Latenz-Metriken		
Metrisch	Beschreibung	Messbare Einheit
Lo_s	NMC-Ausführungslatenz	(s)

Latenz-Metriken		
Metrisch	Beschreibung	Messbare Einheit
La_r	Latenzzeit für Lesevorgänge im Speicherbereich	(s)
La_w	Latenzzeit für Schreibvorgänge im Speicherbereich	(s)
Lmc	Latenzzeit des Speicherkanals	(s)
Lm	Latenzzeit der Speicherschnittstelle	(s)
Lf	Latenzzeit der Fabric-Schnittstelle	(s)
Lpf	NMC-Latenzzeit pro Peer	(s)
Lcf	Latenzzeit pro Rechenknoten	(s)

[0048] Im anschaulichen Beispiel von **Fig.** 4 beherbergt das NMC 421 eine Energieverbrauchs-Engine 422, eine Energiemodellierungs-Engine 423 und eine Optimierungs-Engine 428. In diesem Fall können Metriken, wie die in den Tabellen 1 und 2 dargestellten, in Datenspeichern (z.B. in Datenspeichern 112) gespeichert werden. Die hier offenbarten Implementierungen sind jedoch nicht auf diese Konfiguration beschränkt. In einigen Implementierungen können das Energieverbrauchsmodul 422, das Energiemodellierungsmodul 423 und/oder das Optimierungsmodul 428 in einem anderen Gerät des disaggregierten Speichersystems 400 untergebracht sein. Beispielsweise können das Energieverbrauchsmodul 422, das Energiemodellierungsmodul 423 und/oder das Optimierungsmodul 428 von der Speichermatrix 410 oder einer externen Rechenvorrichtung, wie einem Server oder 600 von **Fig.** 6, gehostet werden.

[0049] Gemäß verschiedenen hier offenbarten Implementierungen kann die Energieverbrauchs-Engine 422 zur Messung der Energieverbrauchskennzahlen des disaggregierten Speichersystems 400 verwendet werden. Beispielsweise kann das Energieverbrauchsmodul 422 die Energie messen, die bei der Durchführung von Rechenaufgaben verbraucht wird, sowie die Energie, die während der Kommunikation (z.B. Datenbewegung zwischen Komponenten des disaggregierten Speichersystems 400) verbraucht wird. Tabelle 1 zeigt Beispiele für Berechnungsenergiemetriken (z.B. Metriken, die mit der Ausführung von Berechnungsaufgaben verbunden sind) und Kommunikationsenergiemetriken (z.B. Energie, die während der Datenbewegung verbraucht wird), die alle von der Energieverbrauchs-Engine 422 gemessen werden können. Beispielsweise können Metriken, die mit der Speicheranordnung 410, dem NMC 421, den Rechenknoten 405 und/oder einem oder mehreren Peer-NMCs 431 bei der Ausführung von Funktionen verbunden sind, als Berechnungsenergiemetriken bezeichnet werden. Metriken, die mit dem Speicherkanal 401, der Client-Fabric 406 und der Peer-Fabric 432 beim Austausch von Daten verbunden sind, können als Kommunikationsenergiemetriken bezeichnet werden. Das Energieverbrauchsmodul 422 kann die Energieverbrauchsmetriken als Telemetriedaten sammeln. In einer Beispielimplementierung kann das NMC 421 einen oder mehrere Leistungssensoren 419 umfassen, die so konfiguriert sind, dass sie Leistungs-Telemetriedaten (J/s) erfassen und die erfasste Leistung über die Zeit integrieren, um Energie-Telemetriedaten (J) zu erzeugen, die dann mit der Energie pro Transaktion eines Datenbetreibers verknüpft werden können. Ein anschauliches Beispiel für einen Leistungssensor 419 ist ein On-Die-Leistungssensor, der Leistung in Watt (J/s) liefert und diese über die Zeit summiert, um Energie in Joule zu erhalten.

[0050] In einigen Fällen können die Energieverbrauchskennzahlen beispielsweise in den Datenspeichern 112 vorgespeichert werden. Bei Datenoperatoren 411 mit festen Funktionen (wie z.B. memset(), memchr(), crypt(), popcount()) können beispielsweise die statische Leistungsaufnahme (W) und die Leistung pro Transaktion (J/Bit) in den Datenspeichern 112 gespeichert und der Energieverbrauchs-Engine 422 zur Verfügung gestellt werden. Bei festen Funktionen können die statische Leistungsaufnahme und die Leistung pro Bit während des Entwurfs und der Charakterisierung bestimmt werden. Die Energieverbrauchskennzahlen können in den Datenspeichern 112 vorab gespeichert werden.

[0051] Bei programmierbaren Datenbetreibern 411 können Schätzungen des statischen Overheads (W) und des prozentualen Anteils der Kapazität pro Transaktion (J/Bit) dem NMC 421 zunächst zur Verfügung stehen (z.B. in den Datenspeichern 112 gespeichert) und dann anhand von Messungen aktualisiert werden, sobald sie beispielsweise von der Energieverbrauchs-Engine 422 erfasst werden. Beispielsweise können Schätzungen des statischen Leistungsoverheads und der Transaktionsleistung pro Bit aus dem gesamten maximalen und durchschnittlichen Energieverbrauch geschätzt werden, der während des Entwurfs und der Charakterisierung gemessen wurde, und dann entsprechend einem Prozentsatz der verwendeten Ressourcen anteilig

berechnet werden (z.B. wenn eine programmierbare Funktion 50 % der Ressourcen im NMC verwendet, würde eine Schätzung der statischen Leistung, die mit der Funktion verbunden ist, 50 % der durchschnittlichen oder maximalen Energienutzung betragen, die während des Entwurfs und der Charakterisierung gemessen wurde, um eine konservativere Schätzung durchzuführen). Wenn ein neuer programmierbarer Datenoperator konfiguriert wird, kann eine anfängliche Schätzung auf Basis des Prozentsatzes der genutzten Ressourcen gespeichert werden. Während der Nutzung des Datenoperators kann die tatsächlich verbrauchte Energie über die Betriebszeit gesammelt werden, um die Schätzung zu verfeinern. Zusätzliche Stichproben können im Laufe der Zeit gesammelt werden, um die Schätzung weiter zu verbessern und eine mögliche Datenabhängigkeit zu berücksichtigen. Die Schätzungen können in Datenspeichern 112 gespeichert und von der Energieverbrauchs-Engine 422 aktualisiert werden.

[0052] Das Energieverbrauchsmodul 422 kann zur Messung von Latenzmetriken des disaggregierten Speichersystems 400 verwendet werden. Beispielsweise kann das Energieverbrauchsmodul 422 die Zeit messen, die die Komponenten des disaggregierten Speichersystems 400 zur Ausführung von Funktionen benötigen. In Tabelle 2 sind Beispiele für diese Latenzmetriken dargestellt, die jeweils von der Energieverbrauchs-Engine 422 gemessen werden können. Ähnlich wie die Energieverbrauchsmetriken können die Latenzmetriken geschätzt und zunächst über die Datenspeicher 112 verfügbar sein und auf Basis von Messungen durch das Energieverbrauchsmodul 422 aktualisiert werden.

[0053] Die Energieverbrauchs-Engine 422 kann Kommunikations-Energiemetriken anhand eines Fingerabdrucks des Kommunikationspfads kennzeichnen. Beispielsweise können Metadaten an eine Kommunikationsenergiemetrik angehängt werden, die Daten eines Fingerabdrucks enthalten, der eine Quelle der Kommunikationsenergiemetrik angibt. Im Falle einer Datenanforderung von einem oder mehreren Rechenknoten 405 kann der Fingerabdruck Informationen zur Identifizierung der Kommunikation enthalten. In einem Beispiel kann der Fingerabdruck einen Identifikator des einen oder der mehreren Rechenknoten 405 (z.B. ein beliebiges Identifizierungsattribut, das bis zu einer Zuweisung auf niedriger Ebene zurückverfolgt werden kann, wie z.B. eine MAC-Adresse, Komponenten-IDs oder Ähnliches angesichts der Konstruktionsmerkmale der verwendeten Struktur) enthalten, der eine Anforderung ausgegeben hat, eine lokale Adresse des einen oder der mehreren Rechenknoten 405 (z. B, eine lokale IP-Adresse, Komponenten-ID oder ähnliches) und alle entfernten Peer-Adressen (z.B. IP-Adressen, Komponenten-IDs oder ähnliches) von einem oder mehreren Peer-Rechenknoten 405, die mit dem anfordernden einen oder mehreren Rechenknoten 405 kombiniert werden können (falls vorhanden). Im Falle einer Kommunikation vom Speicherarray 410 kann der Fingerabdruck eine Kennung des Speicherarrays 410 und eine lokale Adresse enthalten. Im Falle einer Kommunikation vom NMC 421 kann der Fingerabdruck eine Kennung des NMC 421 und eine lokale Adresse enthalten. Auf diese Weise können die Energiemetriken der Kommunikation mit einer Quelle in Verbindung gebracht werden. Im Falle eines oder mehrerer Peer-NMCs 431 kann der Fingerabdruck ein Token umfassen, das mit einem bestimmten oder mehreren Peer-NMCs 431 verbunden ist.

[0054] In einigen Implementierungen ermöglicht die Kennzeichnung von Kommunikations-Energiemetriken mit einem Fingerabdruck die Bestimmung einer kommunikationsbasierten Energie-Basislinie, wenn das NMC 421 im Pass-Through-Modus konfiguriert ist. Ein Vergleich der mit der Kommunikation verbundenen dynamischen Energie mit der vom NMC 421 verbrauchten Gesamtenergie, wenn keine Datenoperatoren aktiviert sind (z.B. im Durchreichmodus), kann eine statische Verlustleistungsmetrik liefern. Für Datenoperatoren mit fester Funktion können die Daten der Entwurfscharakterisierung Metriken für Zeit und Energie pro Bit für die Datenoperatoren liefern. Ein Vergleich der statischen Verlustleistungsmetrik mit dem Betrieb des Datenoperators kann dynamische Datenoperator-Energieschätzungen liefern, die zur Ableitung von Energieverbrauchsvorhersagen für programmierte Datenoperator-Energiemetriken verwendet werden können. Beispielsweise kann der Vergleich der statischen Verlustmetrik mit der Energie, die bei einem aufgerufenen Datenoperator verbraucht wird, auf die Energie hinweisen, die von diesem Datenoperator für die Ausführung einer programmierten Funktion verbraucht wird. Zur Veranschaulichung ein Beispiel: Wenn nur die statische Leistung (Po_s) angegeben wird, ist nur die durchschnittliche Leistung oder die Leerlaufleistung bekannt. Wenn sowohl die Leistung im Leerlauf als auch unter Last gemessen werden kann, lässt sich ein bestimmter Datenoperator mit dem Energieverbrauch eines bestimmten Kunden bei der Nutzung dieses Datenoperators in Verbindung bringen (und möglicherweise über einen Fingerabdruck). Wenn die Datenbetreiber ausgetauscht werden, kann die Metrik für jeden Datenbetreiber durch einen ähnlichen Vergleich wie die statische Verlustleistung geschätzt werden. Auf diese Weise kann die NMC-Metrik für die statische Verlustleistung (Po_s) sowie der NMC-Energieverbrauch pro Datenbetreiber und pro verarbeitetem Bit (Eo) ermittelt werden.

[0055] Die Energiemodellierungs-Engine 423 kann so konfiguriert sein, dass sie die Telemetriedaten verwendet, um eine Mehrzahl von Berechnungsszenarien für die Ausführung einer Datenverarbeitungsanforde-

rung durch das disaggregierte Speichersystem 400 zu modellieren. Die Energiemodellierungs-Engine 423 kann statische Gemeinkosten (in Form von W oder J/s) und pro Transaktion (J/Bit) über die Betriebslebensdauer aufgrund von Bandbreiten- (Bit/s) und Latenzzeit- (s) Verbrauchsmetriken von der Energieverbrauchs-Engine 422 erhalten, so dass der zeitbasierte Energieverbrauch berechnet werden kann. Beispielsweise generiert die Energiemodellierungs-Engine 423 Modelle für eine Mehrzahl von Konfigurationen des disaggregierten Speichersystems 400 zur Ausführung einer oder mehrerer Datenverarbeitungsanforderungen (z.B. Aufgaben) von einem Rechenknoten. Die Mehrzahl der modellierten Konfigurationen kann ein Basisverbrauchsmodell, ein Maximalverbrauchsmodell und ein oder mehrere Zwischenverbrauchsmodelle umfassen. Das Basis-Verbrauchsmodell kann eine Konfiguration sein, in der NMC 421 als Passthrough arbeitet, so dass keine Datenoperator-Ressourcen verbraucht werden (z.B. keine Datenoperatoren aufgerufen werden) und alle Berechnungsfunktionen am Rechenknoten ausgeführt werden. In dieser Konfiguration werden Daten vom Speicherarray 410 über die Speicherschnittstelle 424 und die Fabric-Schnittstelle 425 über den Pfeil 426 an einen oder mehrere Rechenknoten 405 übertragen, ohne dass Datenoperator-Ressourcen in Anspruch genommen werden. Im Modell des maximalen Verbrauchs kann die rechenknotenseitige Berechnung minimal oder gleich Null sein, und im Wesentlichen werden alle Berechnungsfunktionen von aufgerufenen Datenoperatoren 411 des NMC 421 ausgeführt (z.B. wie durch die Pfeile 427a und 427b gezeigt), so dass der Ressourcenverbrauch des Rechenknotens vernachlässigbar ist. Zwischenverbrauchskonfigurationen können unterschiedliche Grade von Berechnungsfunktionen sein, die von Datenoperatoren 411 und vom Rechenknoten 405 (oder einer oder mehreren Peer-NMCs 431) ausgeführt werden. Beispielsweise kann eine erste Zwischenverbrauchskonfiguration einen ersten Datenoperator 411 zur Durchführung von Aufgaben aufrufen und die verbleibenden Funktionen an den Rechenknoten 405 weiterleiten. Die zweite Zwischenverbrauchskonfiguration kann die ersten Datenoperatoren 411 und einen zweiten Datenoperator 411 aufrufen. Somit kann jede n-te Zwischenverbrauchskonfiguration n-Datenoperatoren 411 aufrufen. Wenn n gleich der Anzahl der Ressourcen der Datenoperatoren 411 ist, wäre die Modellkonfiguration die maximale Verbrauchskonfiguration.

[0056] Die Energiemodellierungs-Engine 423 simuliert jedes der Modelle unter Verwendung der von der Energieverbrauchs-Engine 422 erhaltenen Berechnungs- und Kommunikationsenergiemetriken sowie der Latenzmetriken und gibt Telemetrieverbrauchsfaktoren für die modellierte Konfiguration aus. Die Telemetrie-Verbrauchsfaktoren können Vorhersagen über den Energie- und Zeitverbrauch für die Ausführung der angeforderten Aufgabe auf der Basis der einzelnen modellierten Konfigurationen treffen. Beispielsweise kann jede modellierte Konfiguration mit einem Paar von Telemetrie-Verbrauchsfaktoren verknüpft werden, die zur Optimierung des disaggregierten Speichersystems 400 in Bezug auf Energie und Zeit verwendet werden können. Die Telemetrie-Verbrauchsfaktoren können beispielsweise einen ersten Faktor umfassen, der den Energieverbrauch für die Ausführung der angeforderten Aufgaben pro modellierter Konfiguration vorhersagt, und einen zweiten Faktor, der den Zeitaufwand für die Ausführung der einen oder mehreren Aufgaben pro modelliertem Szenario vorhersagt.

[0057] Zum Beispiel kann der Basis-Energieverbrauch von der Energiemodellierungs-Engine 423 als die Energie modelliert werden, die für einen Lese-/Schreibvorgang im lokalen Speicher des Speicherfelds 410 und für die Kommunikation erforderlich ist, um Daten für den Vorgang zu bewegen. In diesem Fall werden die Datenoperatoren 411 des NMC 421 umgangen und ihre Energie ist Null. Der Basis-Energieverbrauch kann mit Hilfe von Berechnungs- und Kommunikations-Energiemetriken wie folgt modelliert werden:

$$\text{BaselineEnergyClient}_n = \left(\text{Ef}_T + \text{Ecf}_T[n][i] + \text{Pf}_s / \text{BWcf}[n][i]\right) * \left(\text{Bc}_r[n][i] + \text{Bc}_w[n][i]\right) \quad \text{Gleichung 1}$$

$$\text{BaselineNMCEnergy} = 0 \quad \text{Gleichung 2}$$

$$\text{BaselineEnergyMemoryArray} = \text{Ea}_r * \text{Bm}_r + \text{Ea}_w * \text{Bm}_w + \left(\text{Emc}_T + \text{Em}_T + \left(\text{Pa}_s + \text{Pm}_s\right) / \text{BWmc}\right) * \left(\text{Bm}_r + \text{Bm}_w\right), \text{ wobei } Bm_{r|w} = Bm_{r|w}[n][i] \quad \text{Gleichung 3}$$

[0058] Wobei, wie in Tabelle 1 oben dargelegt, Ef_T die verbrauchte Energie der Fabric-Schnittstelle pro übertragenem Bit ist; Ecf_T [n][i] die verbrauchte Energie der Client-Fabric pro übertragenem Bit [i] pro Rechenknoten [n] ist; Pf_s die verbrauchte statische Leistung der Fabric-Schnittstelle ist; BWcf[n][i]) ist die Client-Fabric-Bandbreite pro Rechenknoten [n] pro Bit [i]; Bc_r [n][i] ist die Anzahl der Bits [i], die für Lesevorgänge pro Rechenknoten [n] übertragen werden; Bc_w [n][i] ist die Anzahl der Bits [i], die für Schreibvorgänge pro Rechenknoten [n] übertragen werden; Ea_r ist die pro Lesevorgang verbrauchte Energie des Speicherarrays; Bm_r ist die Anzahl der aus dem Speicherarray in den NMC gelesenen Bits; Ea_w ist die pro Schreibvorgang verbrauchte Energie des Speicherarrays; Bm_w ist die Anzahl der vom NMC in das Speicherarray geschriebe-

nen Bits; Emc_T ist die verbrauchte Energie des Speicherkanals pro übertragenem Bit; Em_T ist die verbrauchte Energie der Speicherschnittstelle pro übertragenem Bit; Pa_s ist die verbrauchte statische Leistung des Speicherarrays; Pm_s ist die verbrauchte statische Leistung der Speicherschnittstelle; und BWmc ist die Bandbreite der Clientstruktur pro Rechenknoten pro Bit.

[0059] $BaselineEnergyClient_n$ steht für den voraussichtlichen Gesamtenergieverbrauch des n-ten Rechenknotens bei der Ausführung der Aufgabe und dem Datenaustausch mit dem NMC 421. BaselineNMCEnergy steht für den voraussichtlichen Gesamtenergieverbrauch des NMC 421, der in diesem Fall mit 0 angesetzt wird. BaselineEnergyMemoryArray stellt die voraussichtliche Gesamtenergie dar, die vom Speicher-Array 410 bei der Ausführung der Aufgabe und dem Datenaustausch mit dem NMC 421 verbraucht wird. Die Summe der Gleichungen 1-3 kann einen ersten Faktor darstellen, der mit der Basisverbrauchskonfiguration verbunden ist (hier auch als erste Verbrauchskonfiguration bezeichnet). In diesem Beispiel können die Metriken des n-ten Rechenknotens mit einem Fingerabdruck für den Rechenknoten versehen werden, wie oben beschrieben, so dass die Aufgabe durch das System verfolgt werden kann.

[0060] In einem Beispiel für ein maximales Energieverbrauchsmodell für eine reine Kommunikationsdatenanforderung kann der client-seitige Energieverbrauch als vernachlässigbar angenommen und durch ein NMC-Modell ersetzt werden, das von einem bit- und dauerbasierten Modell abhängt. In diesem Szenario kann der Betrieb auf der Speichermatrix 410 der Gleichung 3 entsprechen, da auf der Speichermatrix 410 immer noch die gleiche Aktivität stattfindet. Der maximale Energieverbrauch in diesem Beispiel kann wie folgt modelliert werden:

$$MaxEnergyClient_n = 0 \qquad \text{Gleichung 4}$$

$$MaxNMCEnergy = Eo^{*}\left(Bo_r + Bo_w\right) + Po_s * Lo_s, \ \textit{wobei}\ Bo_{r|w} = Bo_{r|w}[n][i] \qquad \text{Gleichung 5}$$

$$MaxEnergyMemoryArray = Ea_r * Bo_r + Ea_w * Bo_w + \left(Emc_T + Em_T + \left(Pa_s + Pm_s\right)/BWmc\right) * \left(Bo_r + Bo\right)_w \qquad \text{Gleichung 6}$$

[0061] Wobei, unter Bezugnahme auf die Tabellen 1 und 2, Eo die NMC-Energie ist, die pro verarbeitetem Bit pro Datenoperator verbraucht wird; Po_s die verbrauchte statische NMC-Leistung ist; Lo_s die NMC-Ausführungslatenz ist; Ea_r die pro Lesevorgang verbrauchte Speicherfeld-Energie ist; Bo_r die pro Datenoperator pro Bit für den Lesevorgang verarbeiteten NMC-Bits ist; Ea_w die pro Schreibvorgang verbrauchte Speicherfeld-Energie ist; Bo_w ist die für Schreibvorgänge verarbeiteten NMC-Bits pro Datenoperator pro Bit; Emc_T ist die pro übertragenem Bit verbrauchte Energie des Speicherkanals; Em_T ist die pro übertragenem Bit verbrauchte Energie der Speicherschnittstelle; Pa_s ist die verbrauchte statische Leistung des Speicherarrays; Pm_s ist die verbrauchte statische Leistung der Speicherschnittstelle; und BWmc ist die Client-Fabric-Bandbreite pro Rechenknoten pro Bit.

[0062] $MaxEnergyClient_n$ steht für den voraussichtlichen Gesamtenergieverbrauch des n-ten Rechenknotens bei der Ausführung der Aufgabe und dem Datenaustausch mit dem NMC 421, der in diesem Fall als 0 angenommen wird. MaxNMCEnergy steht für den voraussichtlichen Gesamtenergieverbrauch des NMC 421, z.B. aufgrund des Aufrufs von 41// zur Ausführung aller Berechnungsfunktionen der angeforderten Aufgabe. MaxEnergyMemoryArray stellt die voraussichtliche Gesamtenergie dar, die vom Speicher-Array 410 bei der Ausführung der Aufgabe und dem Datenaustausch mit dem NMC 421 verbraucht wird. Die Summe der Gleichungen 4-6 kann einen ersten Faktor darstellen, der mit der Konfiguration des maximalen Verbrauchs verbunden ist. Metriken vom n-ten Rechenknoten können wie oben beschrieben mit einem Fingerabdruck für den Rechenknoten versehen werden, so dass die Aufgabe im System verfolgt werden kann.

[0063] Wie oben gezeigt, kann die NMC-Energie von Gleichung 5 zunächst als Null angenommen werden, da sie einen potenziell unbekannten Energiebeitrag auf der Seite des Rechenknotens ausgleicht. Ausgehend von diesem Ausgangszustand können die statischen und bitweisen Beiträge durch lokale Energietelemetriemetriken abgeleitet werden. Dies ermöglicht zwar keinen vollständigen End-to-End-Vergleich des Energieverbrauchs auf einem oder mehreren Rechenknoten 405 und dem NMC 421, aber einen relativen Vergleich der Energiekosten im NMC 421 zur Unterstützung der Ressourcenoptimierung.

[0064] Für komplexere Datenoperatoren, die eine Peer-NMC-Kommunikation beinhalten, kann diese Energie anhand der Peer-Fabric-Metriken modelliert werden (z.B. wie in Tabelle 1 und Tabelle 2 dargestellt). Beispielsweise kann die Berechnungsenergie für jedes Peer-NMC 431 mit einer ähnlichen Gleichung wie für

NMC 421 (z.B. Gleichung 5) modelliert werden, basierend auf einer Anzahl von aufgerufenen Datenoperatoren und unter Verwendung der Peer-Fabric-432-Metriken für die Kommunikationsenergiemetriken. Einige modellierte Zwischenkonfigurationen können die Verwendung einer oder mehrerer Peer-NMCs 431 beinhalten, um weitere Konfigurationsoptionen zu erkunden. Das Modell kann im Wesentlichen den Client-Fabric-Metriken im Falle einer Unified Fabric entsprechen.

[0065] Die Optimierungs-Engine 428 kann eine modellierte Konfiguration aus der Mehrzahl der modellierten Konfigurationen für die Ausführung der angeforderten Aufgabe(n) auswählen. Beispielsweise kann die Optimierungs-Engine 428 die Telemetrie-Verbrauchsfaktoren und die zugehörigen modellierten Konfigurationen von der Energiemodellierungs-Engine 423 erhalten und die Telemetrie-Verbrauchsfaktoren verwenden, um eine modellierte Konfiguration zur Ausführung der angeforderten Aufgabe (oder Aufgaben) auszuwählen. Das Optimierungsmodul 428 kann eine modellierte Konfiguration auswählen, die das disaggregierte Speichersystem 400 für die Ausführung der angeforderten Aufgabe(n) in Bezug auf Energie und Zeit bis zum Abschluss optimiert. Dies kann ein Gleichgewicht zwischen Zeit und verbrauchter Energie beinhalten, um beide Faktoren zu minimieren.

[0066] Das Optimierungssystem 428 kann jede modellierte Konfiguration nach einem oder mehreren Telemetrie-Verbrauchsfaktoren ordnen. Beispielsweise kann das Optimierungssystem 428 die modellierten Konfigurationen nach dem ersten zugehörigen Faktor ordnen, wobei die modellierten Konfigurationen vom niedrigsten vorhergesagten Energieverbrauch (z.B. kleinster erster Faktor) zum größten geordnet sind. Als weiteres Beispiel kann die Optimierungs-Engine 428 die modellierten Konfigurationen nach dem zugehörigen zweiten Faktor ordnen, wobei die modellierten Konfigurationen von der kleinsten Zeitspanne bis zur Fertigstellung (z.B. kleinster zweiter Faktor) bis zur größten Zeitspanne geordnet sind. Das heißt, von der schnellsten Fertigstellungszeit bis zur langsamsten Fertigstellungszeit. In einem anderen Beispiel können der erste und der zweite Faktor gemäß einem Kompromiss ausgeglichen werden, um das disaggregierte Speichersystem 400 durch Minimierung sowohl des ersten als auch des zweiten Faktors zu optimieren.

[0067] Der Kompromiss zwischen Zeit- und Energieverbrauch kann als Teil der Auswahl einer modellierten Konfiguration berücksichtigt werden. Beispielsweise kann die Optimierungs-Engine 428 in Abhängigkeit von den innerhalb des disaggregierten Speichersystems 400 festgelegten Beschränkungen für den Energieverbrauch oder die Zeit bis zur Fertigstellung Telemetrie-Verbrauchsfaktoren auswählen, die mit den Beschränkungen übereinstimmen, um das disaggregierte Speichersystem 400 gemäß den festgelegten Beschränkungen zu optimieren. Beispielsweise können Energieverbrauchs- und/oder Zeitbeschränkungen von einem Benutzer festgelegt werden, die als Kriterien bei der Auswahl einer modellierten Konfiguration berücksichtigt werden können. Die Telemetrieverbrauchsfaktoren für jede modellierte Konfiguration können von der Optimierungs-Engine 428 referenziert und verwendet werden, um eine oder mehrere modellierte Konfigurationen zu finden, die die Kriterien erfüllen. Beispielsweise kann ein Benutzer eine gewünschte Zeit bis zum Abschluss der Ausführung einer angeforderten Aufgabe eingeben, und die Optimierungs-Engine 428 kann eine oder mehrere modellierte Konfigurationen mit einem zweiten Faktor finden, der die gewünschte Ausführungszeit erfüllt. Aus der einen oder mehreren modellierten Konfigurationen kann eine optimale modellierte Konfiguration ausgewählt werden, die den niedrigsten Energieverbrauch aufweist und gleichzeitig die Zeitkriterien erfüllt. Ein weiteres Beispiel ist die Eingabe des gewünschten Energieverbrauchs durch den Benutzer, die dazu verwendet werden kann, die modellierten Konfigurationen auf eine oder mehrere modellierte Konfigurationen zu filtern, die den gewünschten Energieverbrauch erfüllen. Von dort aus kann eine modellierte Konfiguration mit der schnellsten Ausführungszeit (z.B. mit dem niedrigsten zweiten Faktor) ausgewählt werden. Im Hinblick auf unterschiedliche Kombinationen von Energie- und Zeitvorgaben sind auch andere Implementierungen möglich.

[0068] In einer Beispielimplementierung kann die Optimierungs-Engine 428 einen oder mehrere der Datenoperatoren 411 entsprechend der ausgewählten modellierten Konfiguration zuweisen. Beispielsweise kann das Optimierungsmodul 428 die Telemetrieverbrauchsfaktoren ermitteln und die Faktoren verwenden, um eine Konfiguration für das disaggregierte Speichersystem 400 auszuwählen, wie oben beschrieben. Die Verwendung des Datenoperators 411 kann entsprechend der ausgewählten Modellkonfiguration aufgerufen werden. Die Datenoperatoren können zugewiesen oder neu zugewiesen werden, um das NMC 421 entsprechend der ausgewählten Modellkonfiguration zu konfigurieren.

[0069] Gemäß einigen Implementierungen kann die Optimierung durch die Optimierungs-Engine 428 auf einer Operator-Basis durchgeführt werden, zum Beispiel durch Zuweisung von Aufgaben an Datenoperatoren 411, um den Energieverbrauch lokal innerhalb des NMC 421 zu optimieren. Die Optimierung kann beispielsweise die Zuweisung von Datenoperatoren 411 innerhalb des NMC 421 in Übereinstimmung mit der ausge-

wählten modellierten Konfiguration umfassen. Die Optimierung auf Datenbetreiberbasis des NMC 421 kann als lokalisiertes Management der Datenbetreiber 411 bezeichnet werden. Bei diesem Ansatz kann die NMC 421 durch Minimierung des Energieverbrauchs der NMC 421 optimiert werden, indem beispielsweise Berechnungsfunktionen zu den Rechenknoten 405 zurückverlagert werden und somit eine lokale Optimierung auf Kosten einer globalen Optimierung erreicht wird. In diesem Fall kann die NMC 421 Ausnahmebedingungen an die anfragenden Rechenknoten 405 zurückgeben, die Funktionen an einen oder mehrere Rechenknoten 405 zurückschieben. Ein solcher Vorgang kann über eine beliebige Sammlung von Fingerabdrücken erfolgen, z.B. über den anfragenden Client, den lokalen Zieladressbereich und den entfernten Peer-Adressbereich oder über einen Token, der bis zu einer Föderation von Peer-NMCs 431 bereitgestellt wird. Das heißt, NMC 421 kann Berechnungen auf Basis des mit der Anforderung verbundenen Fingerabdrucks an eine beliebige Quelle weiterleiten.

[0070] Die Optimierung kann auch global erfolgen, z.B. durch Zuweisung von Aufgaben an gleichrangige NMC 431 (und darin enthaltene Datenoperatoren). Dieser Vorgang kann als globale Verwaltung von NMC-Operatoren bezeichnet werden. Wenn beispielsweise die knotenseitige Berechnungsenergie bekannt ist, können End-to-End-Energieprognosen verwendet werden, um den Gesamtenergieverbrauch im disaggregierten Speichersystem 400 zu optimieren, indem beispielsweise Berechnungsfunktionen an eine oder mehrere Peer-NMCs 431 übertragen werden, um ein gewünschtes Ergebnis zu erzielen. Wenn zusätzlich zu den Bandbreitenmetriken und der Gesamtdatengröße in Bits auch die durchschnittlichen Latenzmetriken für den Speicherkanal 401 und das Speicherfeld 410 bekannt sind, kann auch die Zeit bis zur Fertigstellung modelliert werden.

[0071] Wenn sowohl die Energie als auch die Zeit bis zur Fertigstellung bekannt sind, können Kunden und Dienstleister die Nachfrage und die Gewinnspanne entsprechend gestalten. Beispielsweise können Telemetriedaten, die gemäß den hier beschriebenen Implementierungen als Optimierungsfaktor zur Verfügung stehen, auch in die Ertragsstruktur von Kunden und Dienstleistern einbezogen werden. Dies ermöglicht eine Steigerung der Energieeffizienz und einen nachhaltigen Betrieb, der mit höheren Gewinnspannen angeboten wird. Darüber hinaus kann die Fähigkeit, den Energieverbrauch des NMC 421 zu messen, den Kunden und Dienstleistern eine fundierte Entscheidung bei der Abwägung zwischen Leistung, Energie und Infrastrukturkosten ermöglichen, wie oben beschrieben. Das heißt, das disaggregierte Speichersystem 400 kann so implementiert werden, dass die Folgen von Datenoperationen im disaggregierten Speichersystem gemessen werden können, und daher können Kunden das disaggregierte Speichersystem 400 nutzen, um den Betrieb für den geringsten Energieverbrauch zu optimieren. In einem anderen Beispiel können Netzbetreiber das disaggregierte Speichersystem 400 nutzen, um die Kosten für die Bereitstellung von Ressourcen durch einen Kompromiss zwischen der Zeit bis zur Lösung und der Energie bis zur Lösung zu minimieren.

[0072] **Fig.** 5 zeigt ein Beispiel einer Rechnerkomponente, die zur Optimierung des Energieverbrauchs in disaggregierten Speichersystemen gemäß verschiedenen Ausführungsformen verwendet werden kann. Wie in **Fig.** 5 dargestellt, kann die Rechnerkomponente 500 beispielsweise ein Servercomputer, ein Controller oder eine andere ähnliche Rechnerkomponente sein, die Daten verarbeiten kann. In der Beispielimplementierung von **Fig.** 5 umfasst die Rechnerkomponente 500 einen Hardwareprozessor 502 und ein maschinenlesbares Speichermedium 504.

[0073] Bei dem Hardware-Prozessor 502 kann es sich um eine oder mehrere Zentraleinheiten (CPUs), halbleiterbasierte Mikroprozessoren und/oder andere Hardwarevorrichtungen handeln, die zum Abrufen und Ausführen von Befehlen geeignet sind, die im maschinenlesbaren Speichermedium 504 gespeichert sind. Der Hardware-Prozessor 502 kann Befehle, wie die Befehle 506-514, abrufen, dekodieren und ausführen, um Prozesse oder Vorgänge zur Optimierung des Energieverbrauchs in einem disaggregierten Speichersystem zu steuern. Alternativ oder zusätzlich zum Abrufen und Ausführen von Befehlen kann der Hardware-Prozessor 502 einen oder mehrere elektronische Schaltkreise enthalten, die elektronische Komponenten zur Ausführung der Funktionalität eines oder mehrerer Befehle umfassen, wie z.B. ein Field Programmable Gate Array (FPGA), einen anwendungsspezifischen integrierten Schaltkreis (ASIC) oder andere elektronische Schaltkreise.

[0074] Ein maschinenlesbares Speichermedium, wie das maschinenlesbare Speichermedium 504, kann ein beliebiges elektronisches, magnetisches, optisches oder anderes physikalisches Speichergerät sein, das ausführbare Anweisungen enthält oder speichert. Bei dem maschinenlesbaren Speichermedium 504 kann es sich beispielsweise um einen Direktzugriffsspeicher (RAM), einen nichtflüchtigen RAM (NVRAM), einen elektrisch löschbaren, programmierbaren Festspeicher (EEPROM), ein Speichergerät, eine optische Platte oder Ähnliches handeln. In einigen Ausführungsformen kann das maschinenlesbare Speichermedium 504

ein nichttransitorisches Speichermedium sein, wobei der Begriff „nicht-transitorisch“ nicht die transitorischen Übertragungssignale umfasst. Wie nachstehend im Detail beschrieben, kann das maschinenlesbare Speichermedium 504 mit ausführbaren Befehlen kodiert sein, z.B. mit den Befehlen 506-514.

[0075] Der Hardware-Prozessor 502 kann die Anweisung 506 ausführen, um eine oder mehrere Anfragen zur Durchführung von Berechnungsfunktionen an Daten zu empfangen, die im disaggregierten Speicher gespeichert sind. Beispielsweise kann ein Rechenknoten eine Anforderung zur Durchführung einer oder mehrerer Aufgaben an einen Speicherknoten (oder ein Speicherarray) eines disaggregierten Speichersystems (z.B. das disaggregierte Speichersystem 400) senden. Die angeforderte Aufgabe kann an den im Speicherknoten gespeicherten Daten durch eine oder mehrere Berechnungsfunktionen ausgeführt werden.

[0076] Der Hardware-Prozessor 502 kann die Anweisung 508 ausführen, um Telemetriedaten für den disaggregierten Speicher, einen speichernahen Rechenknoten in der Nähe des disaggregierten Speichers und den Rechenknoten zu sammeln. Zum Beispiel können, wie oben erläutert, Telemetriedaten erhalten werden, die Metriken für den Berechnungs- und Kommunikationsenergieverbrauch für jede Komponente des disaggregierten Speichersystems enthalten. Die speichernahe Berechnung kann eine Mehrzahl von Datenoperatoren umfassen, von denen jeder zur Ausführung von Berechnungsfunktionen aufgerufen werden kann und daher Energie verbraucht.

[0077] Der Hardware-Prozessor 502 kann die Anweisung 510 ausführen, um eine Mehrzahl von Konfigurationen für die Ausführung der einen oder mehreren Anforderungen auf Basis der Telemetriedaten zu modellieren. Zum Beispiel kann die Anweisung 510 das Erzeugen einer Mehrzahl von modellierten Konfigurationen des disaggregierten Speichersystems zur Durchführung der vom Rechenknoten angeforderten Aufgabe umfassen. Die mehreren modellierten Konfigurationen können ein erstes Verbrauchsmodell (z.B. ein Basisverbrauchsmodell), bei dem keine Datenoperatoren aufgerufen werden, ein zweites Verbrauchsmodell (z.B. ein Maximalverbrauchsmodell), bei dem im Wesentlichen alle Berechnungsfunktionen von Datenoperatoren ausgeführt werden, und ein oder mehrere Zwischenverbrauchsmodelle umfassen, von denen jedes einen unterschiedlichen Grad an Berechnungsfunktionen unter Verwendung einer unterschiedlichen Anzahl von Datenoperatoren ausführt. In einigen Ausführungsformen kann die Anweisung 510 als Reaktion auf den Empfang der Anforderung von dem Rechenknoten ausgeführt werden.

[0078] Die Anweisung 510 kann auch die Erzeugung einer Mehrzahl von Telemetrie-Verbrauchsfaktoren für jede modellierte Konfiguration umfassen. Die Telemetrie-Verbrauchsfaktoren können für jede modellierte Konfiguration einen ersten Faktor, der den Energieverbrauch einer jeweiligen modellierten Konfiguration bei der Ausführung der einen oder mehreren Anforderungen angibt, und einen zweiten Faktor, der die von der jeweiligen modellierten Konfiguration bei der Ausführung der einen oder mehreren Anforderungen verbrauchte Zeit angibt, enthalten.

[0079] Der Hardware-Prozessor 502 kann den Befehl 512 ausführen, um eine modellierte Konfiguration aus der Mehrzahl der modellierten Konfigurationen für die Ausführung der einen oder mehreren Anforderungen auszuwählen. Zum Beispiel können die Telemetrie-Verbrauchsfaktoren referenziert und verwendet werden, um eine modellierte Konfiguration auszuwählen, die mit Telemetrie-Verbrauchsfaktoren verbunden ist, die das disaggregierte Speichersystem in Bezug auf Energie und Zeit bis zur Fertigstellung für die Ausführung der angeforderten Aufgabe optimieren. Wie oben erläutert, kann die Auswahl auf der Minimierung sowohl des ersten als auch des zweiten Faktors sowie auf Basis eines Schwellenwertkriteriums für den Energieverbrauch oder die Zeit bis zur Fertigstellung beruhen. Das Schwellenwertkriterium kann auf Basis einer Benutzereingabe festgelegt werden.

[0080] Der Hardware-Prozessor 502 kann die Anweisung 514 ausführen, um einen oder mehrere der Mehrzahl der Datenoperatoren entsprechend der ausgewählten modellierten Konfiguration zuzuweisen.

[0081] Fig. 6 zeigt ein Blockdiagramm eines Beispiel-Computersystems 600, in dem verschiedene der hier beschriebenen Ausführungsformen implementiert werden können. Das Computersystem 600 kann eine Beispielimplementierung der Architektur 100 oder einer beliebigen Komponente davon sein. Das Computersystem 600 umfasst einen Bus 602 oder einen anderen Kommunikationsmechanismus zur Übermittlung von Informationen sowie einen oder mehrere Hardware-Prozessoren 604, die mit dem Bus 602 verbunden sind, um Informationen zu verarbeiten. Bei dem/den Hardware-Prozessor(en) 604 kann es sich zum Beispiel um einen oder mehrere Allzweck-Mikroprozessoren handeln.

[0082] Das Computersystem 600 umfasst auch einen Hauptspeicher 606, wie z.B. einen Speicher mit wahlfreiem Zugriff (RAM), einen Cache und/oder andere dynamische Speichergeräte, die mit dem Bus 602 verbunden sind, um Informationen und Anweisungen zu speichern, die vom Prozessor 604 ausgeführt werden sollen. Der Hauptspeicher 606 kann auch zum Speichern von temporären Variablen oder anderen Zwischeninformationen während der Ausführung von Befehlen verwendet werden, die vom Prozessor 604 ausgeführt werden sollen. Wenn solche Befehle in Speichermedien gespeichert werden, auf die der Prozessor 604 zugreifen kann, wird das Computersystem 600 zu einer Spezialmaschine, die so angepasst ist, dass sie die in den Befehlen angegebenen Operationen ausführt.

[0083] Das Computersystem 600 umfasst außerdem einen Festwertspeicher (ROM) 608 oder ein anderes statisches Speichergerät, das mit dem Bus 602 verbunden ist, um statische Informationen und Anweisungen für den Prozessor 604 zu speichern. Ein Speichergerät 610, z.B. eine Magnetplatte, eine optische Platte oder ein USB-Stick (Flash-Laufwerk) usw., ist vorgesehen und mit dem Bus 602 verbunden, um Informationen und Anweisungen zu speichern.

[0084] Das Computersystem 600 kann über den Bus 602 mit einer Anzeige 612, z.B. einer Flüssigkristallanzeige (LCD) (oder einem Berührungsbildschirm), verbunden sein, um einem Computerbenutzer Informationen anzuzeigen. Ein Eingabegerät 614, einschließlich alphanumerischer und anderer Tasten, ist mit dem Bus 602 gekoppelt, um Informationen und Befehlsauswahlen an den Prozessor 604 zu übermitteln. Eine andere Art von Benutzereingabegerät ist die Cursorsteuerung 616, wie z.B. eine Maus, ein Trackball oder Cursorrichtungstasten zur Übermittlung von Richtungsinformationen und Befehlsauswahlen an den Prozessor 604 und zur Steuerung der Cursorbewegung auf dem Display 612. In einigen Ausführungsformen können die gleichen Richtungsinformationen und Befehlsauswahlen wie bei der Cursorsteuerung über den Empfang von Berührungen auf einem Touchscreen ohne Cursor implementiert werden.

[0085] Das Computersystem 600 kann ein Benutzerschnittstellenmodul zur Implementierung einer grafischen Benutzeroberfläche enthalten, das in einem Massenspeichergerät als ausführbare Softwarecodes gespeichert werden kann, die von dem/den Computergerät(en) ausgeführt werden. Dieses und andere Module können beispielsweise Komponenten wie Softwarekomponenten, objektorientierte Softwarekomponenten, Klassenkomponenten und Aufgabenkomponenten, Prozesse, Funktionen, Attribute, Prozeduren, Unterprogramme, Segmente von Programmcode, Treiber, Firmware, Mikrocode, Schaltkreise, Daten, Datenbanken, Datenstrukturen, Tabellen, Arrays und Variablen umfassen.

[0086] Im Allgemeinen kann sich das Wort „Komponente“, „Engine“, „System“, „Datenbank“, „Datenspeicher“ und dergleichen, wie es hier verwendet wird, auf eine in Hardware oder Firmware verkörperte Logik oder auf eine Sammlung von Softwareanweisungen beziehen, die möglicherweise Ein- und Ausstiegspunkte haben und in einer Programmiersprache wie z.B. Java, C oder C++ geschrieben sind. Eine Softwarekomponente kann kompiliert und zu einem ausführbaren Programm verknüpft werden, in einer dynamischen Link-Bibliothek installiert werden oder in einer interpretierten Programmiersprache wie BASIC, Perl oder Python geschrieben sein. Es ist klar, dass Softwarekomponenten von anderen Komponenten oder von sich selbst aus aufgerufen werden können und/oder als Reaktion auf erkannte Ereignisse oder Unterbrechungen aufgerufen werden können. Softwarekomponenten, die für die Ausführung auf Computergeräten konfiguriert sind, können auf einem computerlesbaren Medium, wie z.B. einer Compact Disc, einer digitalen Videodisc, einem Flash-Laufwerk, einer Magnetplatte oder einem anderen greifbaren Medium, oder als digitaler Download bereitgestellt werden (und können ursprünglich in einem komprimierten oder installierbaren Format gespeichert sein, das vor der Ausführung eine Installation, Dekomprimierung oder Entschlüsselung erfordert). Ein solcher Softwarecode kann teilweise oder vollständig in einem Speicher des ausführenden Computergeräts gespeichert werden, damit er von dem Computergerät ausgeführt werden kann. Softwareanweisungen können in Firmware, wie z.B. einem EPROM, eingebettet sein. Darüber hinaus können die Hardwarekomponenten aus verbundenen Logikeinheiten wie Gattern und Flipflops und/oder aus programmierbaren Einheiten wie programmierbaren Gatteranordnungen oder Prozessoren bestehen.

[0087] Das Computersystem 600 kann die hierin beschriebenen Techniken unter Verwendung von kundenspezifischer festverdrahteter Logik, einem oder mehreren ASICs oder FPGAs, Firmware und/oder Programmlogik implementieren, die in Kombination mit dem Computersystem bewirkt oder programmiert, dass das Computersystem 600 eine Spezialmaschine ist. Gemäß einer Ausführungsform werden die hierin beschriebenen Techniken vom Computersystem 600 als Reaktion auf den/die Prozessor(en) 604 ausgeführt, der/die eine oder mehrere Sequenzen von einem oder mehreren im Hauptspeicher 606 enthaltenen Anweisungen ausführt/ausführen. Solche Anweisungen können in den Hauptspeicher 606 von einem anderen Speichermedium, wie z.B. dem Speichergerät 610, eingelesen werden. Die Ausführung der im Hauptspeicher

606 enthaltenen Befehlssequenzen veranlasst den/die Prozessor(en) 604, die hier beschriebenen Prozessschritte durchzuführen. In alternativen Ausführungsformen können fest verdrahtete Schaltungen anstelle von oder in Kombination mit Softwareanweisungen verwendet werden.

[0088] Der Begriff „nichtflüchtige Medien“ und ähnliche Begriffe, wie sie hier verwendet werden, beziehen sich auf alle Medien, die Daten und/oder Befehle speichern, die eine Maschine in einer bestimmten Weise arbeiten lassen. Solche nichtflüchtigen Medien können nichtflüchtige Medien und/oder flüchtige Medien umfassen. Zu den nichtflüchtigen Medien gehören beispielsweise optische oder magnetische Festplatten, wie das Speichergerät 610. Zu den flüchtigen Medien gehören dynamische Speicher, wie der Hauptspeicher 606. Zu den gängigen Formen nichtflüchtiger Medien gehören beispielsweise Disketten, flexible Platten, Festplatten, Solid-State-Laufwerke, Magnetbänder oder andere magnetische Datenspeichermedien, CD-ROMs, andere optische Datenspeichermedien, physische Medien mit Lochmustern, RAM, PROM und EPROM, FLASH-EPROM, NVRAM, andere Speicherchips oder -kassetten sowie deren vernetzte Versionen.

[0089] Nicht-transitorische Medien unterscheiden sich von Übertragungsmedien, können aber in Verbindung mit ihnen verwendet werden. Übertragungsmedien sind an der Übertragung von Informationen zwischen nichttransitorischen Medien beteiligt. Zu den Übertragungsmedien gehören beispielsweise Koaxialkabel, Kupferdraht und Glasfaserkabel, einschließlich der Drähte, die den Bus 602 bilden. Übertragungsmedien können auch in Form von Schall- oder Lichtwellen auftreten, wie sie bei der Datenkommunikation über Funk und Infrarot erzeugt werden.

[0090] Das Computersystem 600 umfasst auch eine Kommunikationsschnittstelle 618, die mit dem Bus 602 verbunden ist. Die Kommunikationsschnittstelle 618 stellt eine bidirektionale Datenkommunikationsverbindung zu einem oder mehreren Netzwerkverbindungen her, die mit einem oder mehreren lokalen Netzwerken verbunden sind. Bei der Kommunikationsschnittstelle 618 kann es sich beispielsweise um eine ISDN-Karte (Integrated Services Digital Network), ein Kabelmodem, ein Satellitenmodem oder ein Modem handeln, um eine Datenkommunikationsverbindung zu einer entsprechenden Art von Telefonleitung herzustellen. Als weiteres Beispiel kann die Kommunikationsschnittstelle 618 eine LAN-Karte sein, die eine Datenkommunikationsverbindung zu einem kompatiblen LAN (oder einer WAN-Komponente zur Kommunikation mit einem WAN) herstellt. Es können auch drahtlose Verbindungen implementiert werden. In jeder dieser Implementierungen sendet und empfängt die Kommunikationsschnittstelle 618 elektrische, elektromagnetische oder optische Signale, die digitale Datenströme mit verschiedenen Informationstypen übertragen.

[0091] Eine Netzverbindung ermöglicht in der Regel die Datenkommunikation über ein oder mehrere Netze zu anderen Datengeräten. Eine Netzverbindung kann beispielsweise eine Verbindung über ein lokales Netz zu einem Host-Computer oder zu Datengeräten herstellen, die von einem Internetdienstanbieter (ISP) betrieben werden. Der ISP wiederum bietet Datenkommunikationsdienste über das weltweite Paketdatenkommunikationsnetz an, das heute gemeinhin als „Internet“ bezeichnet wird. Sowohl das lokale Netz als auch das Internet verwenden elektrische, elektromagnetische oder optische Signale, die digitale Datenströme übertragen. Die Signale über die verschiedenen Netze und die Signale auf der Netzverbindung und über die Kommunikationsschnittstelle 618, die die digitalen Daten zum und vom Computersystem 600 übertragen, sind Beispiele für Übertragungsmedien.

[0092] Das Computersystem 600 kann Nachrichten senden und Daten, einschließlich Programmcode, über das/die Netzwerk(e), die Netzwerkverbindung und die Kommunikationsschnittstelle 618 empfangen. Im Internet-Beispiel könnte ein Server einen angeforderten Code für ein Anwendungsprogramm über das Internet, den ISP, das lokale Netzwerk und die Kommunikationsschnittstelle 618 übertragen. Der empfangene Code kann vom Prozessor 604 ausgeführt werden, wenn er empfangen wird, und/oder in der Speichervorrichtung 610 oder einem anderen nichtflüchtigen Speicher zur späteren Ausführung gespeichert werden.

[0093] Jeder der in den vorangegangenen Abschnitten beschriebenen Prozesse, Methoden und Algorithmen kann in Code-Komponenten verkörpert und vollständig oder teilweise durch diese automatisiert werden, die von einem oder mehreren Computersystemen oder Computerprozessoren mit Computerhardware ausgeführt werden. Das eine oder die mehreren Computersysteme oder Computerprozessoren können auch so betrieben werden, dass sie die Ausführung der entsprechenden Vorgänge in einer „Cloud Computing“-Umgebung oder als „Software as a Service“ (SaaS) unterstützen. Die Prozesse und Algorithmen können teilweise oder vollständig in anwendungsspezifischen Schaltkreisen implementiert sein. Die verschiedenen oben beschriebenen Merkmale und Verfahren können unabhängig voneinander verwendet oder auf verschiedene Weise kombiniert werden. Verschiedene Kombinationen und Unterkombinationen sollen in den Anwendungsbereich dieser Offenbarung fallen, und bestimmte Verfahrens- oder Prozessblöcke können in einigen Implementie-

rungen weggelassen werden. Die hier beschriebenen Methoden und Prozesse sind auch nicht auf eine bestimmte Reihenfolge beschränkt, und die damit verbundenen Blöcke oder Zustände können in anderen geeigneten Reihenfolgen, parallel oder auf andere Weise ausgeführt werden. Blöcke oder Zustände können zu den offenbarten Beispielen hinzugefügt oder aus ihnen entfernt werden. Die Ausführung bestimmter Operationen oder Prozesse kann auf Computersysteme oder Computerprozessoren verteilt werden, die sich nicht nur in einer einzigen Maschine befinden, sondern über eine Reihe von Maschinen verteilt sind.

[0094] Eine Schaltung kann in jeder Form von Hardware, Software oder einer Kombination davon implementiert werden. Beispielsweise können ein oder mehrere Prozessoren, Controller, ASICs, PLAs, PALs, CPLDs, FPGAs, logische Komponenten, Software-Routinen oder andere Mechanismen implementiert werden, um eine Schaltung zu bilden. Bei der Implementierung können die verschiedenen hier beschriebenen Schaltungen als diskrete Schaltungen implementiert werden, oder die beschriebenen Funktionen und Merkmale können teilweise oder insgesamt auf eine oder mehrere Schaltungen aufgeteilt werden. Auch wenn verschiedene Merkmale oder Funktionselemente einzeln als separate Schaltungen beschrieben oder beansprucht werden, können diese Merkmale und Funktionen von einer oder mehreren gemeinsamen Schaltungen gemeinsam genutzt werden, und eine solche Beschreibung soll nicht voraussetzen oder implizieren, dass separate Schaltungen erforderlich sind, um diese Merkmale oder Funktionen zu implementieren. Wenn eine Schaltung ganz oder teilweise mit Software implementiert ist, kann diese Software so implementiert werden, dass sie mit einem Computer- oder Verarbeitungssystem arbeitet, das in der Lage ist, die in Bezug auf sie beschriebene Funktionalität auszuführen, wie z.B. das Computersystem 600.

[0095] Wie hierin verwendet, kann der Begriff „oder“ sowohl im einschließenden als auch im ausschließenden Sinne verstanden werden. Darüber hinaus ist die Beschreibung von Ressourcen, Vorgängen oder Strukturen im Singular nicht so zu verstehen, dass der Plural ausgeschlossen wird. Bedingte Ausdrücke wie z.B. „kann“, „könnte“, „könnte“ oder „kann“, sofern nicht ausdrücklich anders angegeben oder im Kontext anders verstanden, sollen im Allgemeinen zum Ausdruck bringen, dass bestimmte Ausführungsformen bestimmte Merkmale, Elemente und/oder Schritte enthalten, während andere Ausführungsformen diese nicht enthalten.

[0096] Die in diesem Dokument verwendeten Begriffe und Ausdrücke sowie deren Abwandlungen sind, sofern nicht ausdrücklich etwas anderes angegeben ist, nicht als einschränkend, sondern als offen zu verstehen. Adjektive wie „konventionell“, „traditionell“, „normal“, „Standard“, „bekannt“ und Begriffe mit ähnlicher Bedeutung sind nicht so zu verstehen, dass sie den beschriebenen Gegenstand auf einen bestimmten Zeitraum oder auf einen zu einem bestimmten Zeitpunkt verfügbaren Gegenstand beschränken, sondern sollten so verstanden werden, dass sie konventionelle, traditionelle, normale oder Standardtechnologien umfassen, die jetzt oder zu einem beliebigen Zeitpunkt in der Zukunft verfügbar oder bekannt sein können. Das Vorhandensein erweiternder Wörter und Formulierungen wie „eine oder mehrere“, „mindestens“, „aber nicht beschränkt auf“ oder ähnlicher Formulierungen in einigen Fällen ist nicht so zu verstehen, dass der engere Fall beabsichtigt oder erforderlich ist, wenn solche erweiternden Formulierungen nicht vorhanden sind.

Patentansprüche

1. Ein speichernaher Rechenknoten in der Nähe eines disaggregierten Speichers, wobei die speichernahe Recheneinheit Folgendes umfasst:
eine Mehrzahl von Datenoperatoren, die so konfiguriert sind, dass sie als Reaktion auf eine oder mehrere Anforderungen von einem Rechenknoten Berechnungsfunktionen an in dem disaggregierten Speicher gespeicherten Daten durchführen;
eine Verbrauchs-Engine, die so konfiguriert ist, dass sie Telemetriedaten für den disaggregierten Speicher, die speichernahe Recheneinheit und den Rechenknoten sammelt;
eine Modellierungs-Engine, die so konfiguriert ist, dass sie eine Mehrzahl von Konfigurationen zur Ausführung der einen oder mehreren Anforderungen auf Basis der Telemetriedaten modelliert; und
eine Optimierungs-Engine, die so konfiguriert ist, dass sie eine modellierte Konfiguration aus der Mehrzahl der modellierten Konfigurationen auswählt, um die eine oder die mehreren Anforderungen von dem Rechenknoten auszuführen, und einen oder mehrere aus der Mehrzahl der Datenoperatoren entsprechend der ausgewählten modellierten Konfiguration zuweist.

2. Der speichernahe Rechenknoten nach Anspruch 1, wobei die Telemetriedaten Berechnungsenergiemetriken umfassen, die den Energieverbrauch des disaggregierten Speichers, der speichernahen Recheneinheit und des Rechenknotens anzeigen.

3. Der speichernahe Rechenknoten nach Anspruch 2, wobei die Berechnungsenergiemetrik die pro Bit verarbeiteter Daten verbrauchte Energie und die für den disaggregierten Speicher, den speichernahen Rechenknoten und den Rechenknoten jeweils verbrauchte statische Leistung umfasst.

4. Der speichernahe Rechenknoten nach Anspruch 1, wobei die Telemetriedaten Kommunikationsenergiemetriken umfassen, die auf die Energie hinweisen, die beim Austausch von Daten zwischen dem disaggregierten Speicher und der speichernahen Recheneinheit und zwischen der speichernahen Recheneinheit und dem Rechenknoten verbraucht wird.

5. Der speichernahe Rechenknoten nach Anspruch 4, wobei die Kommunikationsenergiemetrik die pro Datenbit verbrauchte Energie umfasst, die über eine Speicherschnittstelle zwischen dem disaggregierten Speicher und dem speichernahen Rechenknoten und über eine Verbindungsschnittstelle zwischen dem speichernahen Rechenknoten und dem Rechenknoten übertragen wird.

6. Der speichernahe Rechenknoten nach Anspruch 1, wobei die Mehrzahl der modellierten Konfigurationen umfasst:
eine erste Verbrauchskonfiguration, in der die speichernahe Recheneinheit im Durchreichmodus für die Ausführung der einen oder mehreren Anforderungen konfiguriert ist;
eine zweite Verbrauchskonfiguration, in der die speichernahe Recheneinheit so konfiguriert ist, dass sie alle Berechnungsfunktionen zum Ausführen der einen oder mehreren Anforderungen unter Verwendung der Mehrzahl der Datenoperatoren ausführt; und
eine oder mehrere Zwischenverbrauchskonfigurationen, in denen die speichernahe Recheneinheit so konfiguriert ist, dass sie eine oder mehrere Rechenfunktionen zur Ausführung der einen oder mehreren Anforderungen unter Verwendung der Mehrzahl der Datenoperatoren ausführt und für die übrigen Rechenfunktionen im Durchreichmodus konfiguriert ist.

7. Der speichernahe Rechenknoten nach Anspruch 1, wobei die Modellierungs-Engine ferner so konfiguriert ist, dass sie Telemetrie-Verbrauchsfaktoren für jede modellierte Konfiguration aus der Mehrzahl der modellierten Konfigurationen erzeugt, wobei die Telemetrie-Verbrauchsfaktoren einen ersten Faktor, der den Energieverbrauch durch eine jeweilige modellierte Konfiguration bei der Ausführung der einen oder mehreren Anforderungen anzeigt, und einen zweiten Faktor umfassen, der einen Zeitbetrag anzeigt, der durch die jeweilige modellierte Konfiguration bei der Ausführung der einen oder mehreren Anforderungen verbraucht wird.

8. Der speichernahe Rechenknoten nach Anspruch 7, wobei die Optimierungs-Engine ferner so konfiguriert ist, dass sie eine modellierte Konfiguration aus der Mehrzahl der modellierten Konfigurationen auf Basis der Minimierung des ersten Faktors und des zweiten Faktors der Telemetrieverbrauchsfaktoren auswählt.

9. Der speichernahe Rechenknoten nach Anspruch 7, wobei die Optimierungs-Engine ferner so konfiguriert ist, dass sie Optimierungskriterien erhält, die einen Schwellenwert für mindestens entweder den ersten Faktor oder den zweiten Faktor definieren, wobei die Optimierungs-Engine ferner so konfiguriert ist, dass sie eine modellierte Konfiguration aus der Mehrzahl der modellierten Konfigurationen auswählt, die den Schwellenwert für mindestens entweder den ersten Faktor oder den zweiten Faktor erfüllt.

10. Verfahren zur Optimierung des Energieverbrauchs in einem disaggregierten Speichersystem, wobei das Verfahren umfasst:
Empfangen einer oder mehrerer Anforderungen von einem Rechenknoten zur Durchführung von Berechnungsfunktionen an Daten, die im disaggregierten Speicher gespeichert sind;
Sammeln von Telemetriedaten für den disaggregierten Speicher, eine speichernahe Recheneinheit in der Nähe des disaggregierten Speichers und den Rechenknoten;
Modellieren einer Mehrzahl von Konfigurationen zur Ausführung der einen oder mehreren Anforderungen auf Basis der Telemetriedaten;
Auswählen einer modellierten Konfiguration aus der Mehrzahl der modellierten Konfigurationen zur Ausführung der einen oder mehreren Anforderungen; und
Zuweisen eines oder mehrerer aus einer Mehrzahl von Datenoperatoren der speichernahen Recheneinheit entsprechend der ausgewählten modellierten Konfiguration.

11. Verfahren nach Anspruch 10, wobei die Telemetriedaten Berechnungsenergiemetriken umfassen, die den Energieverbrauch des disaggregierten Speichers, der speichernahen Recheneinheit und des Rechenknotens anzeigen.

12. Verfahren nach Anspruch 11, wobei die Berechnungsenergiemetrik die verbrauchte Energie pro verarbeitetem Datenbit und die für den disaggregierten Speicher, die speichernahe Recheneinheit und den Rechenknoten jeweils verbrauchte statische Leistung umfasst.

13. Verfahren nach Anspruch 10, wobei die Telemetriedaten Kommunikationsenergiemetriken umfassen, die auf die Energie hinweisen, die beim Austausch von Daten zwischen dem disaggregierten Speicher und der speichernahen Recheneinheit sowie zwischen der speichernahen Recheneinheit und dem Rechenknoten verbraucht wird.

14. Verfahren nach Anspruch 13, wobei die Kommunikationsenergiemetrik die verbrauchte Energie pro Datenbit umfasst, das über eine Speicherschnittstelle zwischen dem disaggregierten Speicher und der speichernahen Recheneinheit und über eine Verbindungsschnittstelle zwischen der speichernahen Recheneinheit und dem Rechenknoten übertragen wird.

15. Verfahren nach Anspruch 10, wobei die Mehrzahl der modellierten Konfigurationen umfasst:
eine erste Verbrauchskonfiguration, in der die speichernahe Recheneinheit im Durchreichmodus für die Ausführung der einen oder mehreren Anforderungen konfiguriert ist;
eine zweite Verbrauchskonfiguration, in der die speichernahe Recheneinheit so konfiguriert ist, dass sie alle Berechnungsfunktionen zum Ausführen der einen oder mehreren Anforderungen unter Verwendung der Mehrzahl der Datenoperatoren ausführt; und
eine oder mehrere Zwischenverbrauchskonfigurationen, in denen die speichernahe Recheneinheit so konfiguriert ist, dass sie eine oder mehrere Rechenfunktionen zur Ausführung der einen oder mehreren Anforderungen unter Verwendung der Mehrzahl der Datenoperatoren ausführt und für die übrigen Rechenfunktionen im Durchreichmodus konfiguriert ist.

16. Verfahren nach Anspruch 10, das ferner Folgendes umfasst:
Erzeugen von Telemetrie-Verbrauchsfaktoren für jede modellierte Konfiguration aus der Mehrzahl der modellierten Konfigurationen, wobei die Telemetrie-Verbrauchsfaktoren einen ersten Faktor, der den Energieverbrauch einer jeweiligen modellierten Konfiguration bei der Ausführung der einen oder mehreren Anforderungen angibt, und einen zweiten Faktor umfassen, der einen Zeitbetrag angibt, der von der jeweiligen modellierten Konfiguration bei der Ausführung der einen oder mehreren Anforderungen verbraucht wird.

17. Verfahren nach Anspruch 16, das ferner Folgendes umfasst:
Auswählen einer modellierten Konfiguration aus der Mehrzahl der modellierten Konfigurationen basierend auf der Minimierung des ersten Faktors und des zweiten Faktors der Telemetrieverbrauchsfaktoren.

18. Verfahren nach Anspruch 16, das ferner Folgendes umfasst:
Erhalten von Optimierungskriterien, die einen Schwellenwert für mindestens einen von dem ersten Faktor und dem zweiten Faktor definieren, wobei das Auswählen der modellierten Konfiguration aus der Mehrzahl von modellierten Konfigurationen das Auswählen einer modellierten Konfiguration umfasst, die den Schwellenwert für mindestens einen von dem ersten Faktor und dem zweiten Faktor erfüllt.

19. Ein disaggregiertes Speichersystem, wobei das System Folgendes umfasst:
einen oder mehrere Speicherknoten, die Daten speichern;
mindestens eine speichernahe Rechenfunktion, die über eine Speicherschnittstelle mit einem ersten Speicherknoten des einen oder der mehreren Speicherknoten kommunikativ verbunden ist, wobei die mindestens eine speichernahe Rechenfunktion einen oder mehrere Datenoperatoren umfasst;
mindestens einen Rechenknoten, der über eine Netzwerkschnittstelle mit der mindestens einen speichernahen Rechenfunktion verbunden ist; und
mindestens einen Prozessor, der so konfiguriert ist, dass er in einem Speicher gespeicherte Anweisungen ausführt zum:
Sammeln einer Mehrzahl von Energieverbrauchsmetriken, die den Energieverbrauch durch das disaggregierte Speichersystem für die Ausführung von Datenverarbeitungsanforderungen anzeigen;
eine Datenverarbeitungsanforderung von dem Rechenknoten empfangen;
Erzeugen einer Mehrzahl von modellierten Konfigurationen des disaggregierten Speichersystems zum Aus-

führen der Datenverarbeitungsanforderung von dem Rechenknoten auf Basis der Energieverbrauchsmetriken;
Erzeugen von Telemetrie-Verbrauchsfaktoren für jede der modellierten Konfigurationen aus der Mehrzahl der modellierten Konfigurationen; und
Ausführen der Datenverarbeitungsanforderung vom Rechenknoten unter Verwendung einer modellierten Konfiguration, die aus der Mehrzahl der modellierten Konfigurationen auf Basis der Telemetrieverbrauchsfaktoren ausgewählt wurde.

20. Das System nach Anspruch 19, wobei die Mehrzahl der modellierten Konfigurationen umfasst:
eine erste Verbrauchskonfiguration, in der die mindestens eine speichernahe Rechenfunktion im Durchreichmodus zur Ausführung der Datenverarbeitungsanforderung konfiguriert ist;
eine zweite Verbrauchskonfiguration, in der die mindestens eine speichernahe Recheneinheit so konfiguriert ist, dass sie alle Berechnungsfunktionen zur Ausführung von Datenverarbeitungsanforderungen unter Verwendung des einen oder der Mehrzahl der Datenoperatoren ausführt; und
eine oder mehrere Zwischenverbrauchskonfigurationen, in denen die mindestens eine speichernahe Recheneinheit so konfiguriert ist, dass er eine oder mehrere Berechnungsfunktionen zur Ausführung der Datenverarbeitungsanforderung unter Verwendung des einen oder der Mehrzahl der Datenoperatoren ausführt und für die übrigen Berechnungsfunktionen im Durchreichmodus konfiguriert ist.

Es folgen 6 Seiten Zeichnungen

Anhängende Zeichnungen

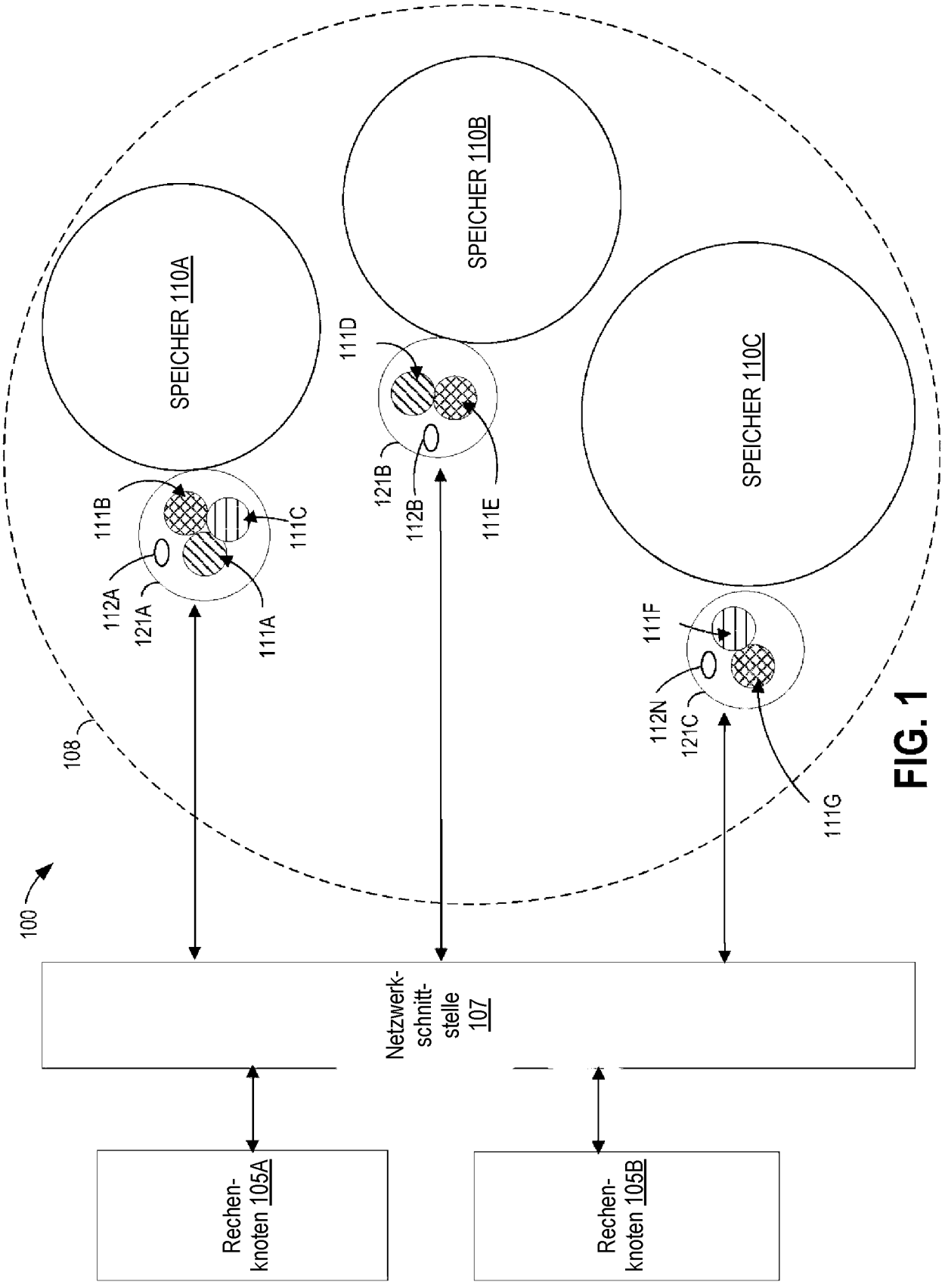


FIG. 1

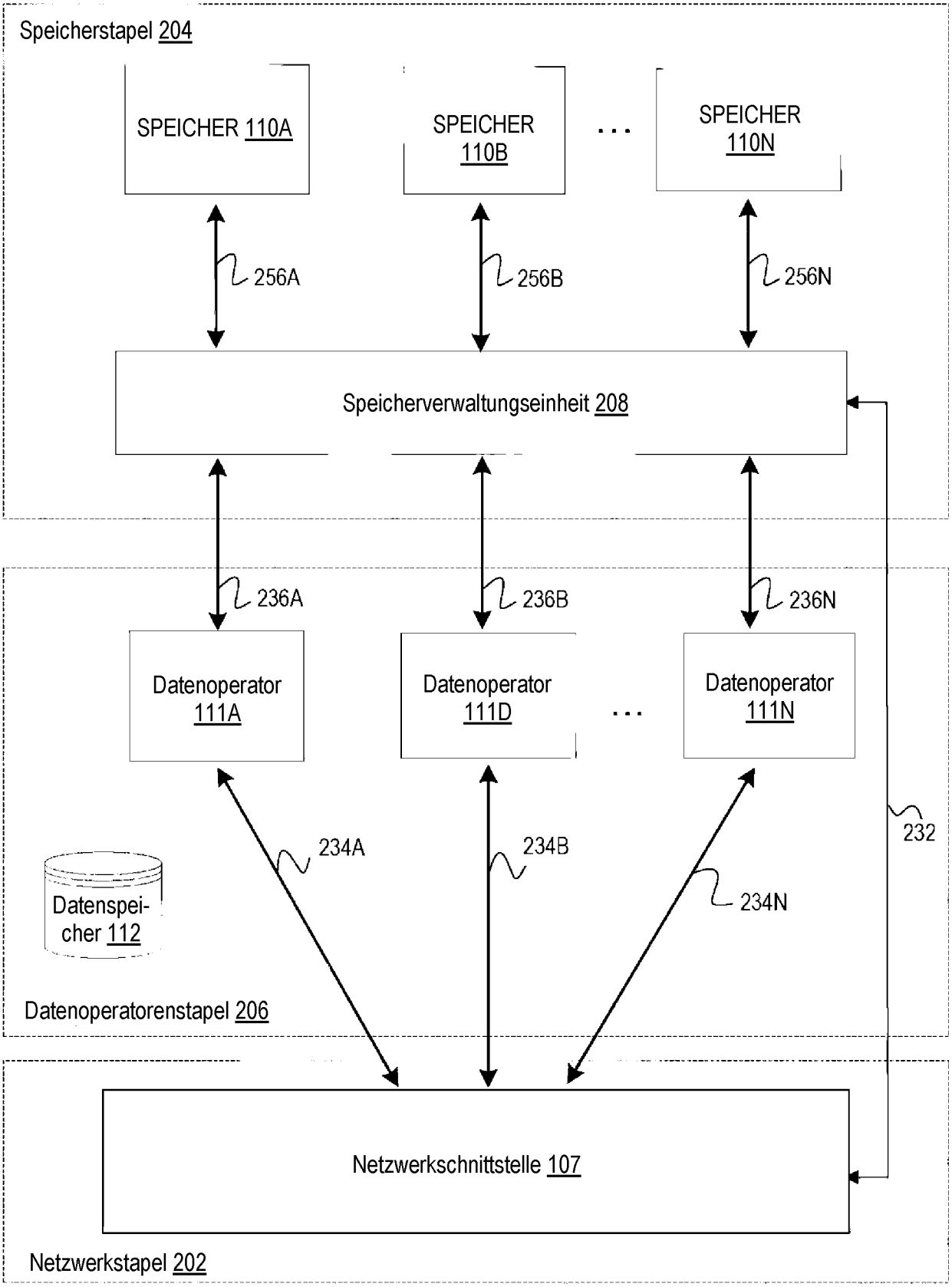

Speicherstapel 204
SPEICHER 110A
SPEICHER 110B
SPEICHER 110N
256A
256B
256N
Speicherverwaltungseinheit 208
236A
236B
236N
Datenoperator 111A
Datenoperator 111D
Datenoperator 111N
234A
234B
234N
232
Datenspeicher 112
Datenoperatorenstapel 206
Netzwerkschnittstelle 107
Netzwerkstapel 202

FIG. 2

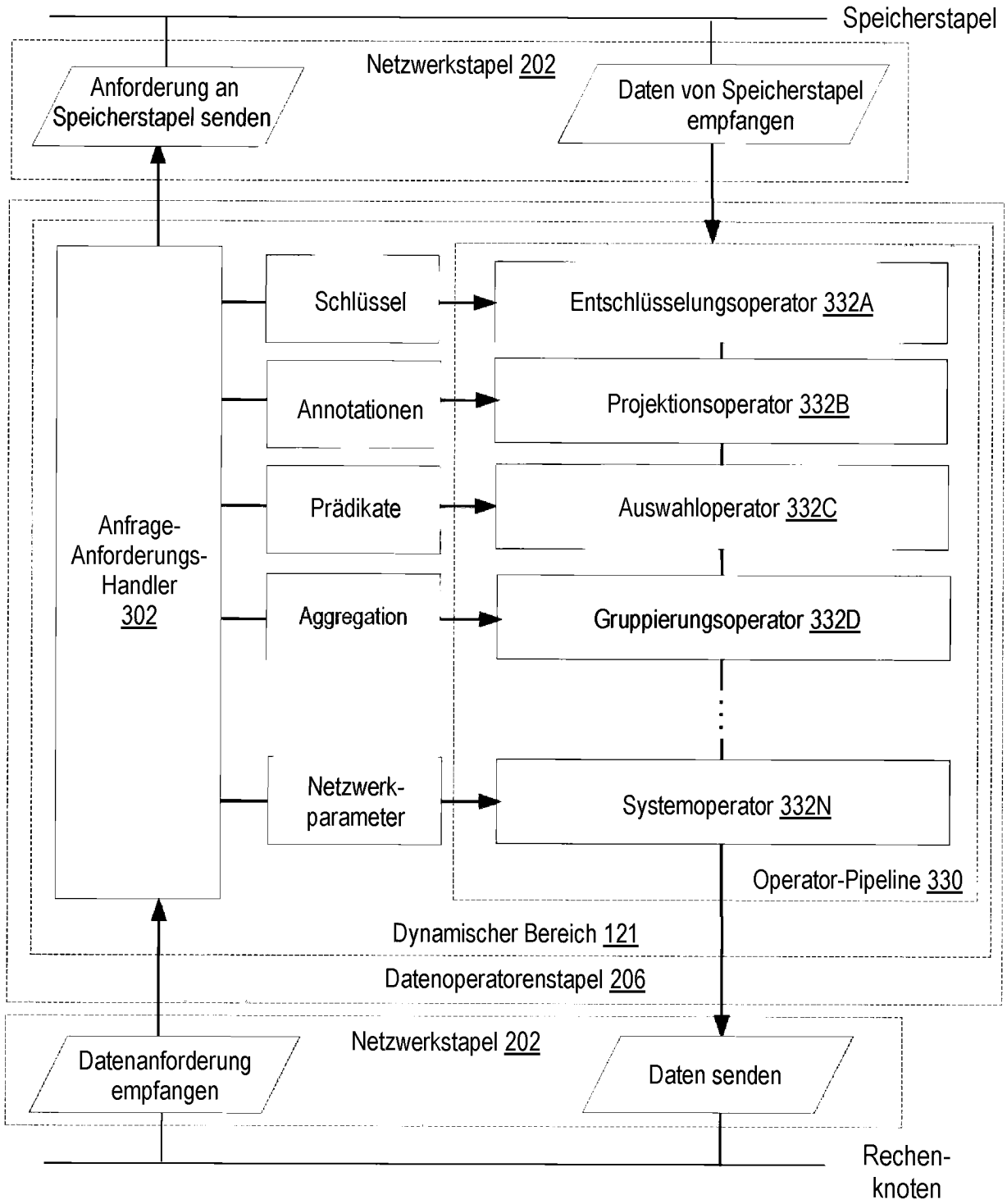

Speicherstapel
Netzwerkstapel 202
Anforderung an Speicherstapel senden
Daten von Speicherstapel empfangen
Anfrage-Anforderungs-Handler 302
Schlüssel
Annotationen
Prädikate
Aggregation
Netzwerk-parameter
Entschlüsselungsoperator 332A
Projektionsoperator 332B
Auswahloperator 332C
Gruppierungsoperator 332D
Systemoperator 332N
Operator-Pipeline 330
Dynamischer Bereich 121
Datenoperatorenstapel 206
Netzwerkstapel 202
Datenanforderung empfangen
Daten senden
Rechen-knoten

FIG. 3

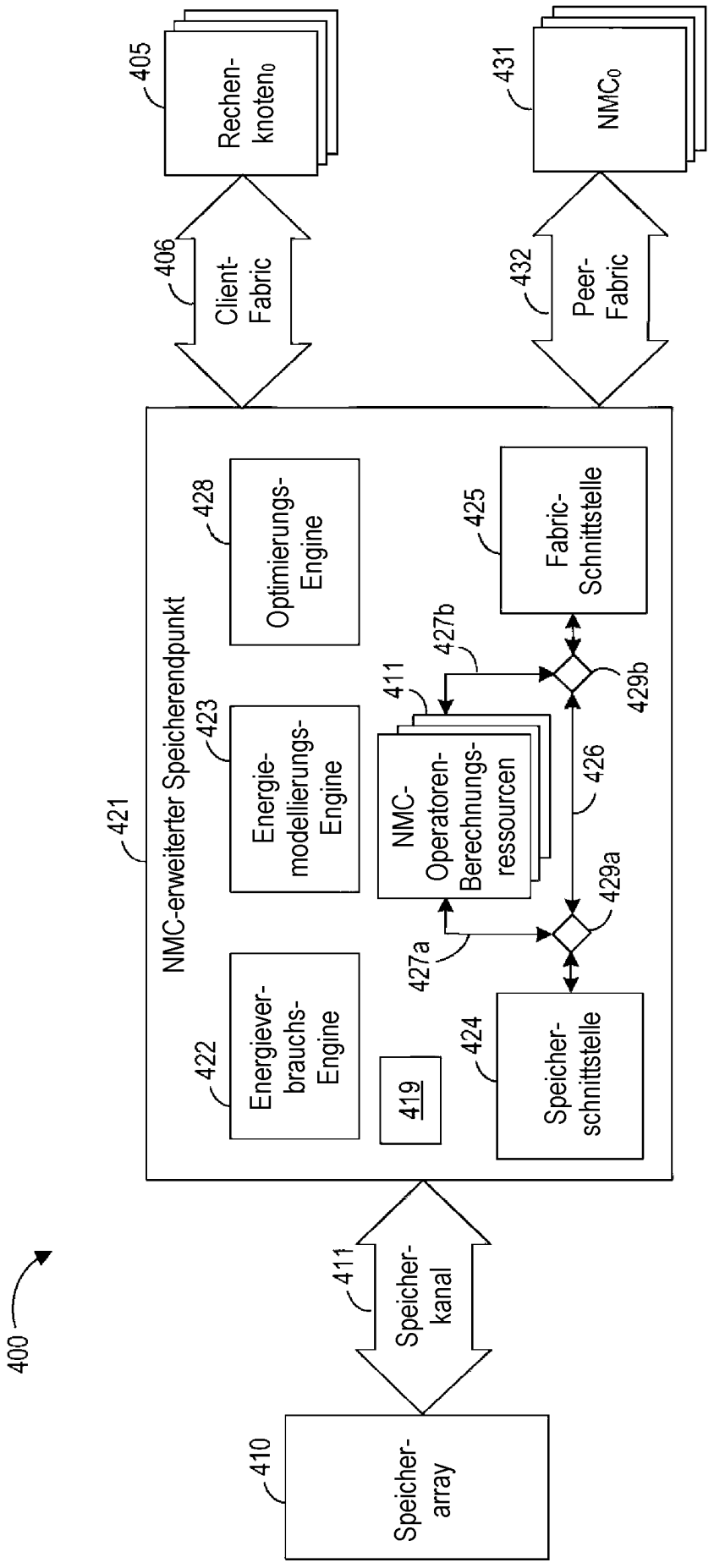

400
405
Rechen-knoten$_0$
406
Client-Fabric
431
NMC$_0$
432
Peer-Fabric
421
NMC-erweiterter Speicherendpunkt
428
Optimierungs-Engine
425
Fabric-Schnittstelle
423
Energie-modellierungs-Engine
411
427b
429b
NMC-Operatoren-Berechnungs-ressourcen
426
429a
427a
422
Energiever-brauchs-Engine
419
424
Speicher-schnittstelle
411
Speicher-kanal
410
Speicher-array

FIG. 4

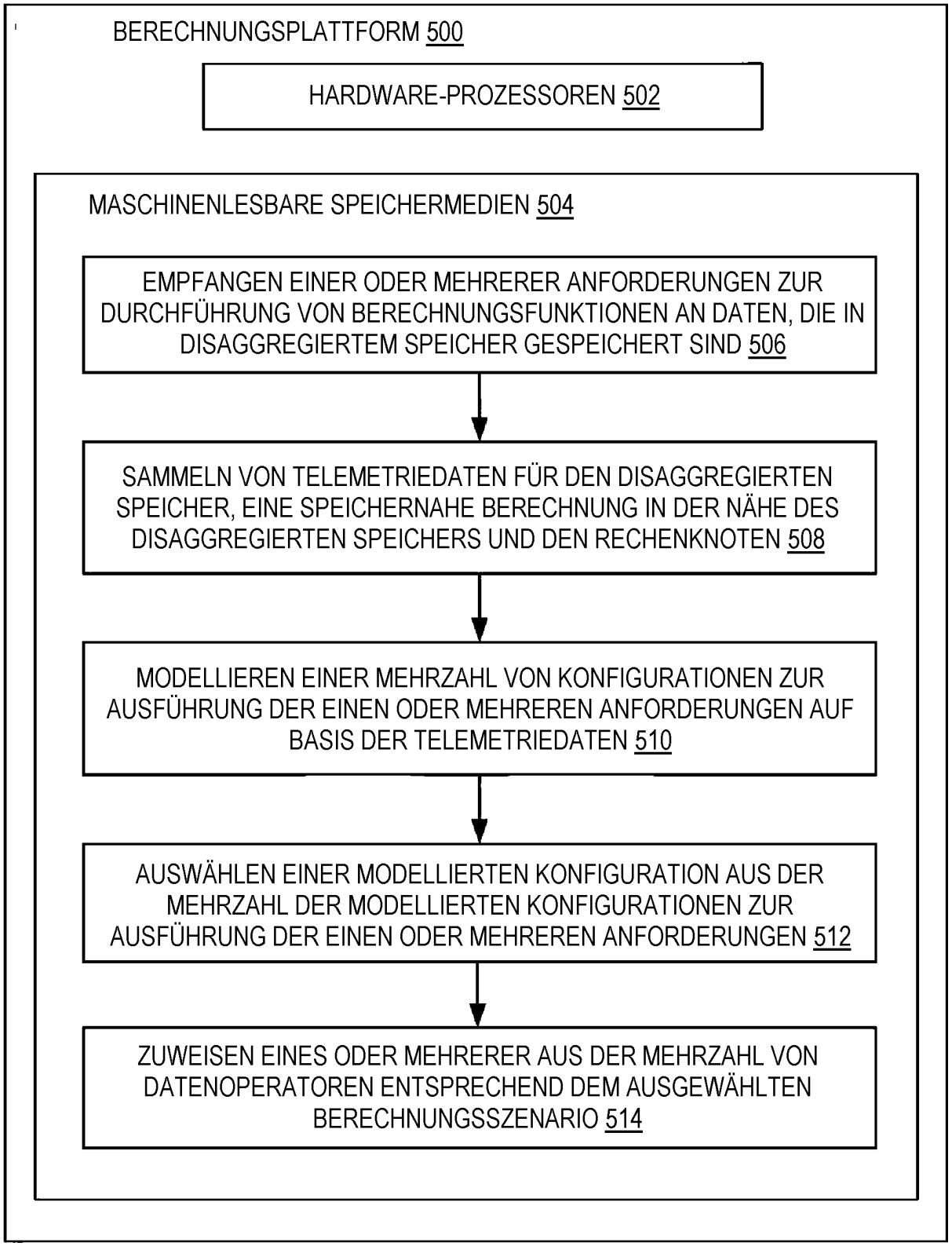

BERECHNUNGSPLATTFORM 500
HARDWARE-PROZESSOREN 502
MASCHINENLESBARE SPEICHERMEDIEN 504
EMPFANGEN EINER ODER MEHRERER ANFORDERUNGEN ZUR DURCHFÜHRUNG VON BERECHNUNGSFUNKTIONEN AN DATEN, DIE IN DISAGGREGIERTEM SPEICHER GESPEICHERT SIND 506
SAMMELN VON TELEMETRIEDATEN FÜR DEN DISAGGREGIERTEN SPEICHER, EINE SPEICHERNAHE BERECHNUNG IN DER NÄHE DES DISAGGREGIERTEN SPEICHERS UND DEN RECHENKNOTEN 508
MODELLIEREN EINER MEHRZAHL VON KONFIGURATIONEN ZUR AUSFÜHRUNG DER EINEN ODER MEHREREN ANFORDERUNGEN AUF BASIS DER TELEMETRIEDATEN 510
AUSWÄHLEN EINER MODELLIERTEN KONFIGURATION AUS DER MEHRZAHL DER MODELLIERTEN KONFIGURATIONEN ZUR AUSFÜHRUNG DER EINEN ODER MEHREREN ANFORDERUNGEN 512
ZUWEISEN EINES ODER MEHRERER AUS DER MEHRZAHL VON DATENOPERATOREN ENTSPRECHEND DEM AUSGEWÄHLTEN BERECHNUNGSSZENARIO 514

FIG. 5

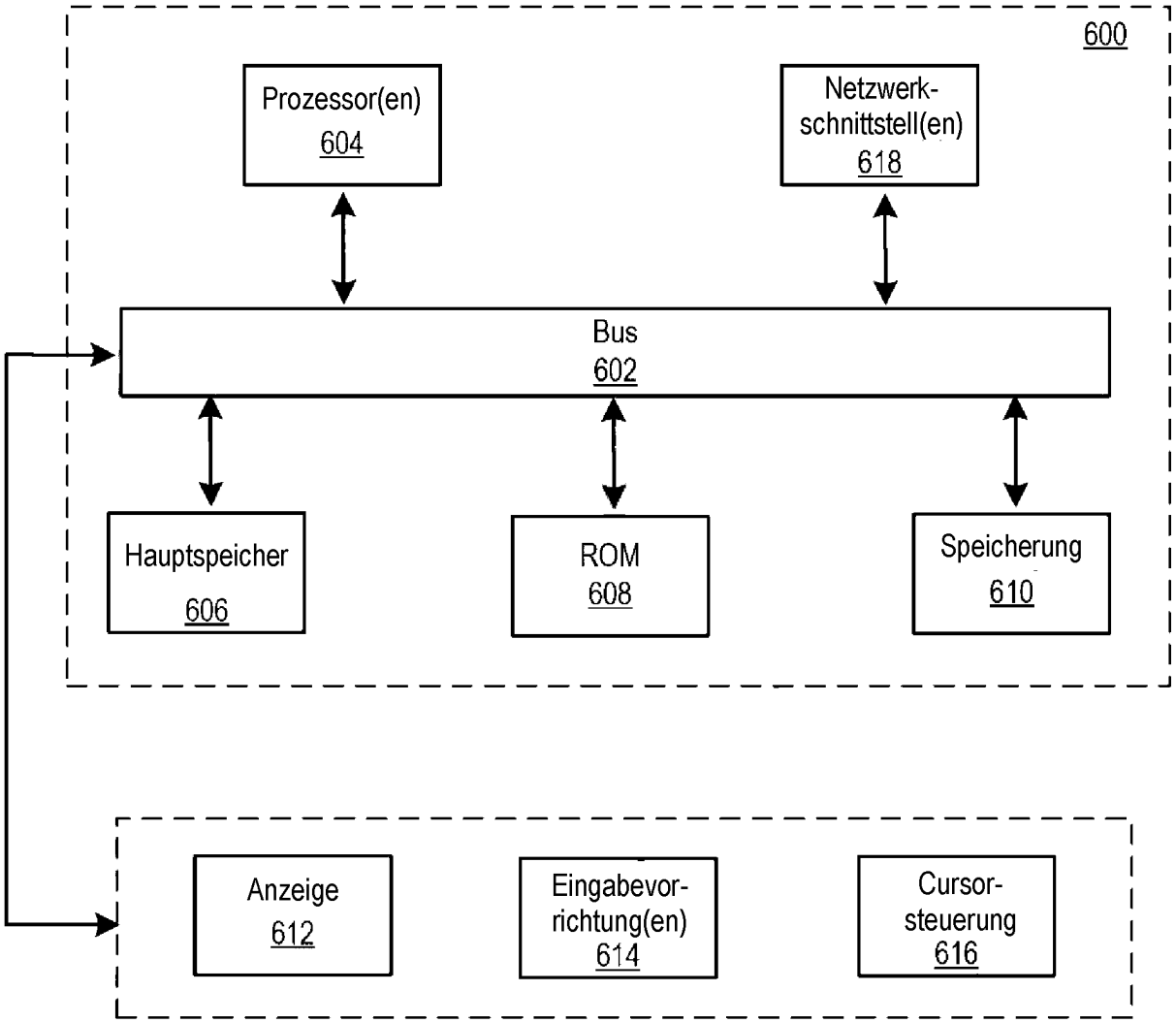


FIG. 6

