TECHNISCHE UNIVERSITÄT DRESDEN

Density-Aware Linear Algebra in a Column-Oriented In-Memory Database System

Kurzfassung der Dissertation

zur Erlangung des akademischen Grades
Doktoringenieur (Dr.-Ing.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von
Dipl.-Phys. David Kernert
geboren am 02. April 1987 in Karlsruhe

Betreuender Hochschullehrer: **Prof. Dr. Wolfgang Lehner**
Technische Universität Dresden
Fakultät Informatik, Institut für Systemarchitektur
Lehrstuhl für Datenbanken
01062 Dresden

Dresden, im Mai 2016

1 EINLEITUNG

Der beträchtliche Zuwachs an elektronisch gespeicherten Datenmengen in Wirtschaft und Wissenschaft hat eine neue Ära der Datenauswertung eingeläutet. So sprach Jim Gray (2007) von einem „vierten Paradigma“ der wissenschaftlichen Exploration, und meint damit die wachsende Herausforderung der maschinellen Auswertung der immer größeren Datenmengen, die von wissenschaftlichen Experimenten und Simulationen erzeugt werden. Auch in der Wirtschaft werden Analysen von Daten (z.B. Verkaufsdaten) immer wichtiger, da sie Einblicke in das Kundenverhalten geben und wesentlichen Einfluss auf geschäftskritische Entscheidungen haben können (McGuire et al., 2012).

Auf diesem Gebiet der maschinellen Datenanalyse, dem maschinellen Lernen, sowie der zahlreichen wissenschaftlichen Berechnungen verschiedenster Domänen, sind mathematische Ausdrücke der linearen Algebra Teil von nahezu jeder Anwendung. Bis heute tendieren Wissenschaftler und Analysten dazu, solche komplexe Berechnungen basierend auf linearer Algebra mittels Desktop-Software wie MATLAB oder R, sowie mit handgeschriebenen Programmen auf dem lokalen Arbeitsrechner durchzuführen (Hey et al., 2009). Durch die rasante Zunahme der Datenmengen können die Analyseprogramme jedoch oft nicht mit den großen Datenvolumen und den erforderlichen Anpassungen der Software auf Parallelrechnern Schritt halten. Im Rahmen der *Big-Data*-Entwicklung gibt es mittlerweile zwar skalierbare Systeme (z.B. Ghoting et al., 2011) und Hochleistungsalgorithmen (z.B. Dongarra et al., 2014), welche auch Funktionalität aus dem Bereich der linearen Algebra anbieten. Diese Lösungen basieren allerdings meist auf statischen Datenstrukturen und haben bedingt durch die verteilte Rechner-Landschaft eine hohe Anfrage-Latenz. Sie sind daher nicht geeignet für Anwendungen, die auf Ad-hoc-Analysen und veränderbaren Datensätzen basieren. Des Weiteren steigt die Nachfrage von Wissenschaftlern nach Fähigkeiten zur Datenverwaltung und -orchestrierung. Diese Eigenschaften zählen zu den Schlüsselcharakteristiken von relationalen Datenbankmanagementsystemen (DBMS), von denen einige Systeme die Fähigkeit, große Datenmengen zu verwalten, in den letzten Jahrzehnten unter Beweis gestellt haben. Allerdings ist das Fehlen von effizienter Linearer-Algebra-Funktionalität der Hauptgrund dafür, dass ausgereifte DBMS nicht für die Kernberechnungen in komplexeren Analyseanwendungen verwendet werden können (Stonebraker et al., 2007, 2013).

Durch das Aufkommen von Hauptspeicherdatenbanken (Garcia-Molina und Salem, 1992) sind heute Ad-Hoc-Lese- und Schreib-Operationen bei geringer Latenz auf großen Datenmengen möglich geworden. Zudem hat sich ein spaltenorientiertes Datenlayout als besonders effizient für Analyse-Anfragen herausgestellt (Abadi et al., 2005). Unter dem Einfluss dieses Technologiewandels gab es in den letzten Jahren ein Trend hin zu umfangreichen Datenmanagement- und Analyse-Plattformen, die sowohl die Rolle des DBMS übernehmen, aber auch einige Anwendungsalgorithmen direkt integrieren (Färber et al., 2015). Unter diesem Aspekt ist es insbesondere interessant zu untersuchen, ob diese Systeme in Zukunft auch als Plattform für komplexe Analysen basierend auf linearer Algebra genutzt werden können.

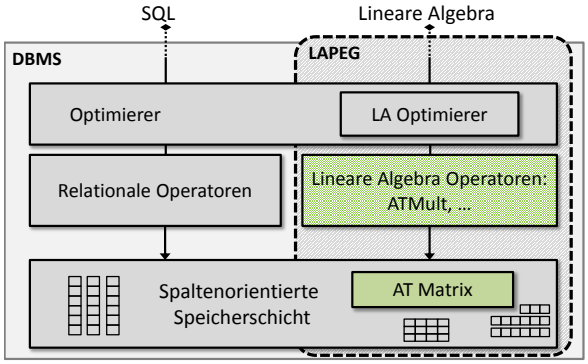


Abbildung 1.1: Architektur der Linearen-Algebra Ausführungseinheit LAPEG, eingebettet in eine spaltenorientierte Hauptspeicherdatenbank.

2 FORSCHUNGSBEITRÄGE

Diese Dissertation präsentiert die tiefe Integration von wesentlichen Strukturen und Operationen der linearen Algebra, im Speziellen Matrizen, Vektoren und Multiplikationen derer, in eine spaltenorientierte Hauptspeicherdatenbank. Basierend auf einigen konkreten Anwendungsfällen, welche in dieser Arbeit vorgestellt werden, leiten wir die fundamentalen Voraussetzungen ab, die an ein solches System mit Unterstützung für lineare Algebra gestellt werden:

- Einige komplexe Ausdrücke der linearen Algebra können auf verschiedenen Wegen ausgewertet werden, von denen einige sehr ineffizient sind. Durch **Optimierung der Ausdrücke** kann das System die Anfragelaufzeiten minimieren.

- Die in der Realität vorkommenden Matrizen besitzen häufig eine hohe Anzahl von Null-Elementen (dünnbesetzt), weshalb es je nach Anzahl und Verteilung der Nicht-Null-Elemente unterschiedliche physische Speicherstrukturen gibt. Für den Nutzer sind daher **transparente Matrixoperationen** wichtig, unabhängig von den physischen Beschaffenheit der Matrix.

- Da Matrixoperationen, wie z.B. Multiplikationen, oft den Hauptanteil der Berechnungszeit in Anspruch nehmen, sind geeignete **Partitionierung** der Datenstrukturen und **Skalierbarkeit** der Operationen fundamental.

- Einige Anwendungen beruhen auf veränderbaren Datensätzen und Ad-Hoc-Manipulationen. Es gibt daher eine steigende Nachfrage der Nutzer nach ausgeprägten **Manipulationsfähigkeiten für Matrixdaten**.

Diese Anforderungen werden umgesetzt von der im Rahmen dieser Arbeit entwickelten, DBMS-integrierten *Linear Algebra Processing Engine* (engl. für Lineare-Algebra-Ausführungseinheit), kurz LAPEG. Die Architektur von LAPEG wird in Abbildung 1.1 skizziert und besteht aus mehreren Komponenten, welche die Sprachschnittstelle und Ausdrucksoptimierung auf der logischen Ebene, sowie verschiedene Operatoren und Datenstrukturen auf der physischen Ebene umfassen. Die einzelnen Komponenten werden im Folgenden weiter erläutert.

2.1 Anwendungen und Systeme: Eine Übersicht

Die oben genannten Anforderungen lassen sich aus verschiedenen Anwendungen ableiten, von denen wir drei beispielhaft skizzieren. Der erste Anwendungsfall kommt aus dem Bereich der Nuklearphysikforschung und hat als Ziel, die Energiezustände von Atomkernen zu berechnen. Dabei wird in einer vorausgehenden Simulation eine sogenannte *Hamilton*-Matrix generiert, von welcher anschließend die Eigenwerte mittels eines numerischen Algorithmus bestimmt werden. Das besondere an diesem Beispiel ist nicht nur, dass die Matrix dünnbesetzt ist und beliebige Größen annehmen kann, sondern auch, dass in einem erweiterten Verfahren die Matrix iterativ manipuliert wird, d.h. Bereiche hinzugefügt oder entfernt werden (Roth, 2009). Ein weiteres Anwendungsbeispiel ist das Dokumenten-Clustering, bei der eine Singulärwertzerlegung der Term-Dokument-Matrix vorgenommen wird. Letztere verbindet die Dokumente mit der Anzahl der darin enthaltenen Terme, und ist üblicherweise auch hochdimensional und dünnbesetzt. Ist die Dimensionalität besonders hoch, werden für die effiziente Berechnung der Singulärwertzerlegung von solchen Matrizen üblicherweise randomisierte Algorithmen eingesetzt. Diese enthalten oft Kettenmultiplikationen der dünnbesetzten Matrix mit kleineren, vollbesetzten Matrizen (Martinsson et al., 2011). In beiden genannten Anwendungsfällen liegt der Hauptaufwand der Berechnung in den Matrix-Vektor bzw. Matrix-Matrix Multiplikationen, weshalb sich ein großer Teil dieser Dissertation mit diesen Operationen beschäftigt.

Computerprogramme und Desktop-Systeme für die eigentliche Berechnung von Operationen der linearen Algebra wurden schon vor Jahrzehnten erstmals entwickelt. Neben den BLAS-Bibliotheken (Basic Linear Algebra Subprograms, Dongarra et al., 2001), den Industriestandard für voll besetzte Matrixoperationen, und der verteilten Version SCALAPACK (Scalable Linear Algebra Package, Blackford et al., 1997), gibt es mittlerweile auch einige Hochleistungsalgorithmen für dünnbesetzte Matrizen (z.B. Kreutzer et al., 2015). Endanwenderprogramme wie R oder MATLAB bieten dem Benutzer viele Algorithmen mit einer natürlichen Sprachschnittstelle für lineare Algebra. Daneben gibt es in der Datenbankwelt einige Ansätze, die Datenbank-Funktionalität mit linearer Algebra zu erweitern (z.B. Brown, 2010). Nichtsdestotrotz zeigt ein abschließender Vergleich, dass keiner dieser Ansätze alle der zuvor identifizierten Anforderungen gleichermaßen erfüllt. Diese Lücke wird von LAPEG geschlossen.

2.2 Matrizen in einem spaltenorientierten DBMS

Ein Kernpunkt dieser Arbeit ist die Integration der primären Datenobjekte der linearen Algebra (im Wesentlichen Matrizen und Vektoren) in die spaltenorientierte Datenablage einer Hauptspeicherdatenbank. Bei der Wahl der Speicherstruktur für Matrizen unterscheidet man zunächst zwischen dünn- und vollbesetzten, bzw. mit Nicht-Null-Elementen besiedelten Matrizen. Für dünnbesetzte Matrizen kann man weiter zwischen verschiedenen Kompressionsarten unterscheiden. Wir zeigen, wie man vollbesetzte Matrizen, sowie die Tripel-Darstellung und das *Compressed Sparse Row* (CSR)-Format für dünnbesetzte Matrizen nahtlos in die spaltenförmige Speicherablage integrieren kann. Ausgehend von der vorgestellten Speicherstruktur diskutieren wir verschiedene Algorithmen zur Multiplikation von Matrizen mit Vektoren und Matrizen. Bedingt durch das gleichzeitige Vorliegen von voll- und dünn besetzten Matrixstrukturen gibt es mehrere Kernel-Algorithmen für die Operation $\boldsymbol{C} = \boldsymbol{AB}$, welche jeweils für ihre bestimmte Datenstruktur optimiert sind.

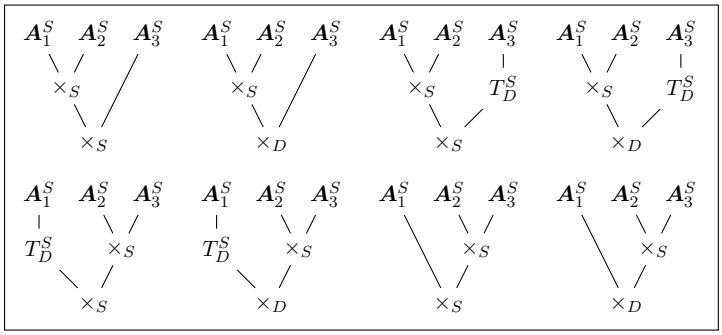


Abbildung 1.2: Acht der 128 möglichen Ausführungspläne für die Multiplikation von drei (potentiell dünnbesiedelten) Matrizen. Das Symbol $\times$ bezeichnet den Multiplikationsoperator, und T Speichertyp-Transformationen von Matrizen. Der obere/untere Index gibt die Eingangs-/Ausgangs-Matrixdarstellung des entsprechenden Operators an: S/D steht für eine *dünnbesetzte/vollbesetzte* Struktur.

Die acht unterschiedlichen Multiplikations-Kernel, die aus den verschiedenen Kombinationen von $Typ(\boldsymbol{A}) \times Typ(\boldsymbol{B}) \rightarrow Typ(\boldsymbol{C})$ entstehen, sind Grundlage für unseren Ausdrucks-Optimierer und die adaptive Matrixmultiplikation. Ein einfacher Ansatz zur Parallelisierung der Kernel in einem Shared-Memory-System ist die horizontale Partitionierung der Matrix $\boldsymbol{A}$. Teile des Materials in diesem Zusammenhang wurden zusammen mit Wolfgang Lehner und Frank Köhler entwickelt und auf der SSDBM'14-Konferenz präsentiert (Kernert et al., 2014).

2.3 Optimieren von Ausdrücken der linearen Algebra

Ausdrücke der linearen Algebra sind üblicherweise aus mehreren Operationen und Operanden zusammengesetzt. Ähnlich wie bei relationalen Anfragesprachen kann man basierend auf dem Anfrage-Ausdruck einen Ausführungsplan erstellen, dessen Effizienz maßgeblich von der Reihenfolge der Operationen sowie von der Auswahl der Algorithmen abhängt. Bei der linearen Algebra kommt dies insbesondere bei Matrixkettenmultiplikation zum Tragen, welche den Hauptteil der Berechnungszeit in einigen Anwendungen ausmacht (z.B. bei der randomisierten Singulärwertzerlegung).

Abbildung 1.2 zeigt einige verschiedene Pläne, um drei Matrizen miteinander zu multiplizieren. Der Suchraum wächst exponentiell mit der Anzahl der Matrizen, so dass ein Brute-Force-Ansatz für längere Ketten undurchführbar wird.

Für ausschließlich vollbesetzte Matrizen ist das Problem der Matrixkettenmultiplikation hinreichend bekannt und lösbar mit Dynamischer Programmierung (Godbole, 1973). Wir erweitern das Problem jedoch, indem ausdrücklich dünnbesetzte Matrizen zugelassen werden, welche in der Realität häufig vorkommen. Neu ist außerdem, dass Lapeg bei der Ausführung der Multiplikation Transformationen in andere Matrixdatenstrukturen zulässt, sofern es die Gesamtlaufzeit minimiert. Der vorgestellte Optimierer (SpMachO) zieht also nicht nur die Matrixdimensionen in Betracht, sondern auch die Matrix-Besetzungsdichte und die aktuelle Datenstruktur von Zwischenergebnissen. Für die Optimierung notwendig ist daher ein differenziertes Kostenmodell, welches die algorithmischen Eigenschaften der verschiedenen Multiplikations-Kernel abbildet. Basierend auf den Spei-

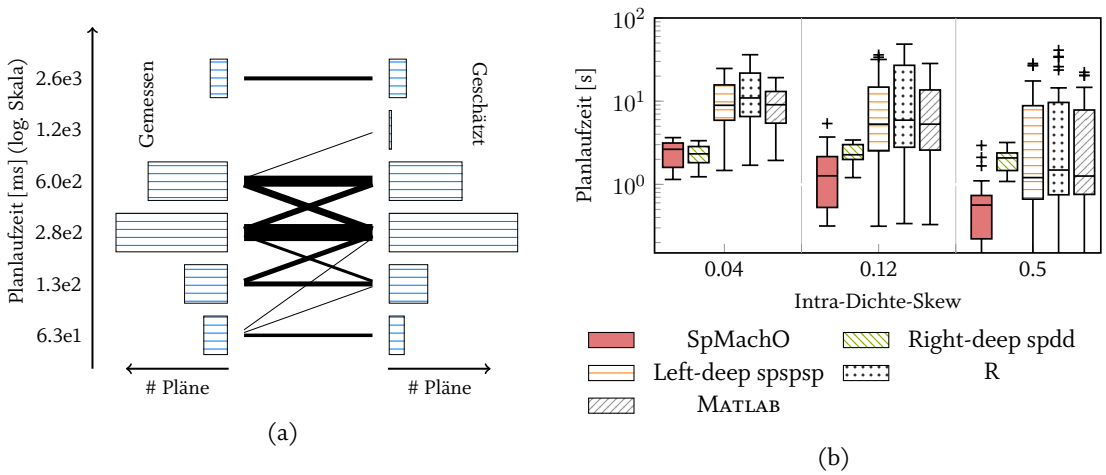


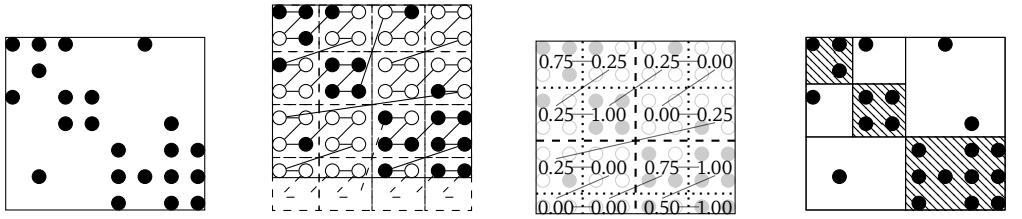
Abbildung 1.3: *a)* Übereinstimmung der geschätzten mit der tatsächlichen Planlaufzeit für alle 128 verschiedenen Pläne einer Kettenmultiplikation von 3 Matrizen. *b)* Laufzeitverteilungen für Matrixkettenmultiplikation. Mit dem *Intra-Density Skew* steigt die Ungleichheit der Nicht-Null-Matrixelemente der Matrizen.

cherzugriffen leiten wir die Kostenfunktionen für die acht vorkommenden Multiplikationsalgorithmen her und verifizieren diese mit einem Skalierungs-Experiment. In das Kostenmodell geht auch die Besetzungsdichte der Zwischenergebnisse ein, die allerdings im Voraus nicht exakt bestimmbar ist. Aus diesem Grund präsentieren wir ein neuartiges Schätzverfahren für die Matrixdichte dünnbesetzter Matrizen (SpProdest), welches auf einer probabilistischen Berechnung basiert und daher besonders effizient ist.

Abbildung 1.3a zeigt die Übereinstimmung der tatsächlichen Planlaufzeiten mit den von SpMachO geschätzten Planlaufzeiten für eine Kettenmultiplikation, wobei die Stärke der horizontalen Verbindungslinien zwischen den einzelnen Plangruppen den Grad der Übereinstimmung kennzeichnen. Es ist ersichtlich, dass die Laufzeiten eher überschätzt werden, und die geschätzten besten Pläne sich auch unter den tatsächlich besten Plänen befinden. Abbildung 1.3b zeigt einen systematischen Vergleich von SpMachO mit anderen Ansätzen, unter anderem R und Matlab. Das Experiment zeigt, dass SpMachO verzerrte Verteilungen der Matrixbesetzungsdichte (Skew) ausnutzen kann und robuster ist als die Alternativen. Teile des Optimierers und der Dichteschätzung wurden zusammen mit Wolfgang Lehner und Frank Köhler entworfen und auf der EDBT'15-Konferenz veröffentlicht (Kernert et al., 2015).

2.4 Adaptive Matrizen

Der Einsatz klassischer Strukturen für voll- und dünnbesetzte Matrizen, wie z.b. jene der Blas- oder anderen Bibliotheken (z.B. CuSparse, n.d.) birgt einige Nachteile: Zunächst ist es mühsam für den Endanwender, das geeignete Matrixformat für seine Daten zu finden, denn oft sind die Strukturen für bestimmte Matrix-Topologien (Diagonalmatrix, Bandmatrix, etc.) optimiert. Der gewichtigere Nachteil für Big-Data-Anwendungen ist aber, dass Algorithmen auf einfachen Matrixstrukturen wie



(a) Ausgangsmatrix $\boldsymbol{A}$ (b) Z-Kurven-Linearisierung (c) Dichtestruktur (d) Finale AT-Matrix

Abbildung 1.4: Schematische Illustration des Partitionierungsprozesses an einer 7×8 Beispielmatrix mit einer Blockgranularität von 2×2. Das diagonal gestreifte Muster kennzeichnet vollbesetzte Kacheln, die übrigen Kacheln sind dünnbesetzt.

Arrays oder CSR nicht beliebig skalieren (Patwary et al., 2015). Daher gehen skalierbare Lösungen (z.B. Ghoting et al., 2011; Buluç und Gilbert, 2010) üblicherweise dazu über, die Matrix in Blöcke oder *Kacheln* zu partitionieren. In diesem Zusammenhang stellen wir die im Rahmen der Arbeit entwickelte AT-Matrix-Struktur vor. Neu an der AT-Matrix-Darstellung ist, dass sie sich voll-adaptiv an die Verteilung der Nicht-Null-Elemente bzw. der topologischen Besetzungsdichte der Matrix anpasst: Die Kacheln können variable Größen annehmen und sind entweder vom Typ *dense* für überwiegend dichtbesetzte Matrixregionen, oder *sparse* (CSR) für dünnbesiedelte Matrixregionen. Abbildung 1.4 skizziert den Partitionierungsprozess der AT-Matrix. Die resultierende Heterogenität kann bei Matrixoperationen, wie zum Beispiel bei Matrixmultiplikationen, ausgenutzt werden. In diesem Zusammenhang präsentieren wir den dynamisch optimierenden Multiplikationsoperator ATmult. Letzterer wählt mit dem oben genannten Kostenmodell jeweils den effizientesten Multiplikations-Kernel für die einzelnen Kachel-Multiplikationen aus und transformiert gegebenenfalls einzelne Kacheln temporär in eine andere Struktur, sofern es sich positiv auf die Laufzeit auswirkt.

Für die Implementierung der ATmult-Operation auf einem Hauptspeichersystem mit Mehrkernprozessor haben wir Daten- und Task-Parallelität kombiniert (Abbildung 1.5). Um die Effekte von *Non-Uniform Memory Access* (NUMA) zu verringern, befinden sich immer jeweils zwei der drei beteiligten Kacheln einer Multiplikation in dem jeweiligen lokalen Speicher des verarbeitenden Prozessor-Sockels.

Die Auswertung von ATmult zeigt eine um bis zu Faktor zehn höhere Effizienz gegenüber alternativen Multiplikationsansätzen. Des Weiteren zeigt die Aufschlüsselung der Performance in Abbildung 1.6b, dass die einzelnen Komponenten der AT-Matrix-Struktur sich je nach Matrixinstanz unterschiedlich auf die Performance auswirken. Für sehr dünnbesiedelte Matrizen (R7) ist die Performance fast identisch mit der klassischen *sparse* Multiplikation (spspsp_gemm). Mittels Blocking (2), Dichteschätzung (3), Heterogenität (4), Adaptivität (5) und dynamischer Optimierung (6) lässt sich die Performance für die Matrixinstanzen (R2-R5) jedoch deutlich steigern.

Die AT-Matrix-Struktur und die ATmult-Multiplikation wurden in Teilen in Zusammenarbeit mit Wolfgang Lehner und Frank Köhler konzipiert und auf der ICDE'16-Konferenz präsentiert (Kernert et al., 2016).

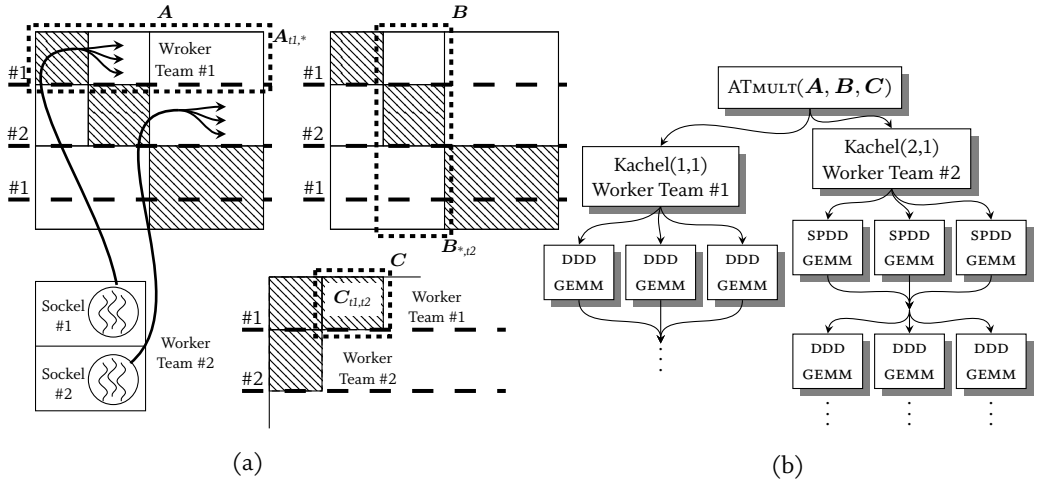


Abbildung 1.5: *a)* Verteilung der parallelen Ressourcen und NUMA-Partitionierung am Beispiel eines Zwei-Sockel-Systems. *b)* Gerichteter Task-Graph einer Kachel-Multiplikation ATMULT.

2.5 Matrix-Manipulation

Der letzte Schwerpunkt dieser Dissertation sind Strukturen und Methoden, um Matrizen effizient zu manipulieren. Insbesondere sollen damit Anwendungen, wie die zu Beginn vorgestellte Nuklearphysik-Analyse, ermöglicht werden, worin eine große Matrix iterativ manipuliert und gelesen wird: Im konkreten Fall werden Zeilen und Spalten der Matrix gelöscht und iterativ wieder hinzugefügt, wobei zwischen jedem Schritt mehrere Matrix-Vektor-Multiplikationen erfolgen (Roth, 2009). In diesem Zusammenhang definieren wir zunächst eine Manipulations-Programmierschnittstelle (engl. API), welche folgende Operationen enthält: *Wert setzen*, *Wert entfernen*, *ganze Zeilen/Spalten-Bereiche einfügen/löschen*. Sämtliche Operationen können sich auf eine beliebe rechteckige Referenzfläche in der Matrix beziehen (siehe Abbildung 1.6a). Die typischen Datenstrukturen für Matrizen in numerischen Bibliotheken (z.B. Saad, 1994) lassen sich nicht ohne Weiteres manipulieren. Insbesondere die sortierte CSR-Struktur erfordert eine Rekonstruktion, wenn Nicht-Null-Elemente der Matrix hinzugefügt werden. Dies ist jedoch hinderlich für die Performance von Ad-hoc-Analysen, bei welchen sich die Lesezugriffe mit Schreibzugriffen abwechseln.

Aus diesem Grund präsentieren wir eine update-fähige Datenstruktur für dünnbesetzte Matrizen, die konzeptuell aus einer Kombination von einer nativen CSR- und einer Tripel-Struktur besteht (siehe Abbildung 1.6b). Somit ist im Gegensatz zu verwandten Arbeiten (Macko et al., 2015) keine tiefgreifende Neugestaltung der CSR-Struktur notwendig ist. Insbesondere nutzt unsere Implementierung dabei die interne Spaltenstruktur des DBMS aus, welche bereits von sich aus eine Separation nach dem Prinzip der stapelweisen Aktualisierung aufweist (Sikka et al., 2012): dabei werden alle neu hinzugefügten Elemente zunächst in eine schreib-optimierte Deltakomponente hinzugefügt, welche in einer späteren Reorganisation in die lese-optimierte Hauptkomponente (Main) übergeführt wird. Durch die Verwendung der nativen Tabellen-Speicherstrukturen profitieren die Anwender von LAPEG außerdem von den inherenten transaktionalen Fähigkeiten des DBMS.

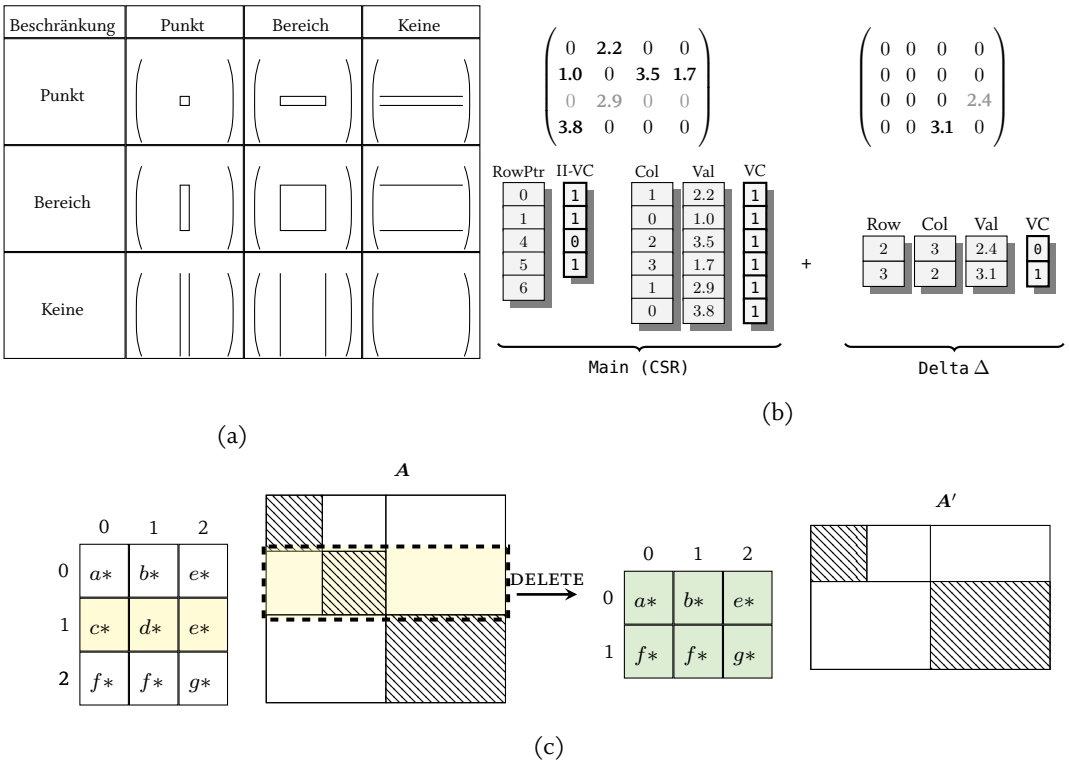


Abbildung 1.6: *a)* Zweidimensionale Matrix-Zugriffsmuster. *b)* Main-Delta Struktur für *sparse* Kacheln. *c)* Implementierung einer Block-Entfernungsoperation in der AT-Matrix.

Die veränderlichen Datenstrukturen lassen sich auf die AT-Matrix-Datenstruktur übertragen. Ein besonderer Vorteil ergibt sich durch die neuartige Abstraktionsschicht der adaptiven AT-Matrix. So werden z.B. beim Kachel-übergreifenden Löschen nur die Metadaten angepasst, und nur die partiell betroffenen Kacheln physisch manipuliert (siehe Abbildung 1.6c).

Abbildung 1.7b zeigt die Messung des Anfrage-Durchsatzes von abwechselnden Schreib- und Leseoperationen auf die Matrix R3. Hierbei bestehen die Leseoperation aus Matrix-Vektor-Multiplikationen und das Schreiben aus dem Einfügen von Nicht-Null-Elementen, wobei der Anteil der Schreiboperationen gegenüber den Leseoperationen variiert wird ($N_{\text{schreib}}/N_{\text{lese}}$). Die zwei Varianten der veränderlichen AT-Matrix (mit lokaler Delta- (*LD*) bzw. globale Delta-(*GD*) Struktur) sind den anderen Ansätzen deutlich überlegen, wobei bei häufigen Schreibanfragen AT-Matrix$^{\text{LD}}$ vorzuziehen ist. Teile der veränderlichen Matrixstruktur wurden zusammen mit Wolfgang Lehner und Frank Köhler entwickelt und auf der SSDBM'14-Konferenz präsentiert (Kernert et al., 2014).

3 SCHLUSSFOLGERUNGEN

In Zeiten der ständig wachsenden Datenmengen von verschiedensten Datenquellen in Wirtschaft und Wissenschaft gibt es eine wesentliche Nachfrage nach einem leistungsfähigen System, was so-

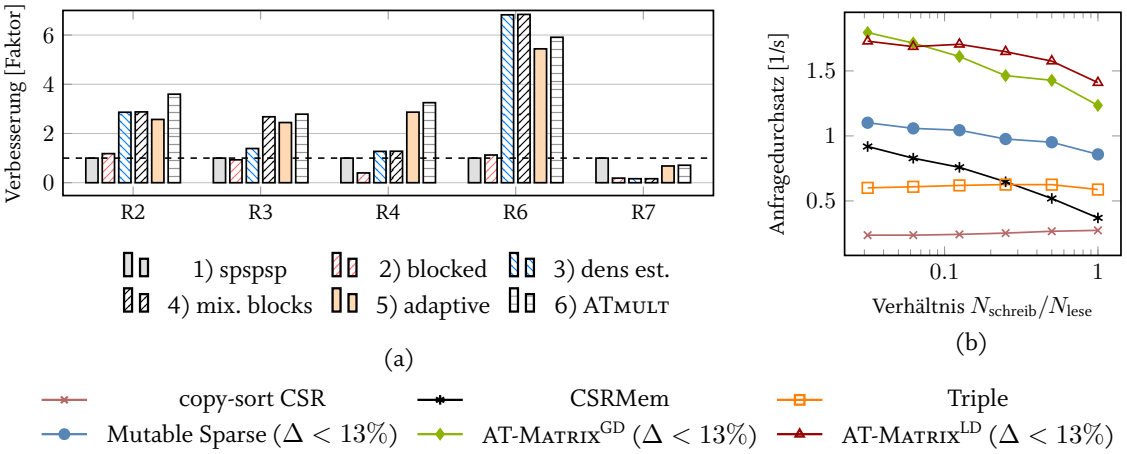


Abbildung 1.7: *a)* Komponentenweise Aufschlüsselung der ATMULT Performance. *b)* Anfragedurchsatz einer gemischten Lese- und Schreib-Arbeitslast auf Matrix R3 (TSOPF_RS_b2383) .

wohl ausgeprägte Datenmanagementfähigkeiten, als auch umfangreiche Analysewerkzeuge für Anwendungen auf großen Datenmengen bereitstellt. Da die komplexen Anwendungen der Datenanalyse meist auf linearer Algebra und Matrizen basieren, können sowohl herkömmliche Datenbankmanagementsysteme auf der einen Seite, als auch numerische Bibliotheken auf der anderen Seite diese Anforderungen nicht gleichzeitig zufriedenstellend erfüllen (Gray, 2007).

Diese Dissertation präsentiert eine Lineare-Algebra-Ausführungseinheit (LAPEG), eingebettet in eine spaltenorientierte Hauptspeicherdatenbank. Damit schließen wir die Lücke zwischen den Datenmanagementfähigkeiten eines DBMS und numerischen Hochleistungsalgorithmen. Diese Arbeit hat nicht nur gezeigt, dass effiziente Matrixdatenstrukturen nahtlos in eine spaltenorientierte Speicherverwaltung integriert werden können, sondern auch, dass einige Techniken der Datenbanktechnologie auf neuartige Weise bei matrix-basierten Analyseanwendungen angewendet werden können. Mittels der Schätzung der lokalen Matrixbesetzungsdichte von Zwischenergebnissen kann insbesondere die Laufzeit von Matrixmultiplikationen großer, dünnbesiedelter Matrizen deutlich reduziert werden. Dabei kann man ähnlich der Optimierung relationaler Join-Operationen die Heterogenität durch die Verwendung verschiedener Algorithmen ausnutzen. Unsere adaptive Partitionierung von Matrizen (AT-MATRIX) verbessert nicht nur die Performance bei Multiplikationen, sondern ermöglicht auch effiziente Manipulationen von großen Matrixdatensätzen, und komplettiert damit die wesentlichen Kernanforderungen an LAPEG.

Trotz der steigenden Nachfrage (Stonebraker et al., 2013) steckt die Entwicklung von Datenbanksystemen mit Unterstützung von Operationen der linearer Algebra und zur Matrixverarbeitung noch in den Kinderschuhen. Wir haben mit LAPEG hierzu einen Beitrag geleistet, doch sind für eine volle Funktionalität der linearen Algebra noch einige Erweiterungen notwendig, die über den Rahmen dieser Arbeit hinausgehen. Zum einen sollte man weitere Operationen berücksichtigen, zum Beispiel die Lösung von Gleichungssystemen, und untersuchen wie dabei unsere adaptive Struktur (AT-MATRIX) ausgenutzt werden kann. Zum anderen kann die Optimierung komplexer Ausdrücke der linearen Algebra um weitere Operationen erweitert werden. Bezüglich der physischen Struk-

tur von Matrizen sind darüber hinaus weitergehende Implementierungen für ultra-dünnbesetzte (*hypersparse*) Matrizen denkbar, oder inwiefern sich eine mögliche Vorverarbeitung positiv auf die adaptive Matrixdatenstruktur auswirkt, z.B. durch einen Algorithmus zur Graphpartitionierung (Karypis, 2011).

LITERATURVERZEICHNIS

Daniel J. Abadi, Mike Stonebraker, Edmond Lau, Amerson Lin, et al. C-Store: A Column-oriented DBMS. In *Proc. of 31st VLDB Conf.*, pages 553–564, 2005.

L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, et al. *ScaLAPACK Users' Guide*. SIAM, 1997.

Paul G. Brown. Overview of SciDB: Large Scale Array Storage, Processing and Analysis. In *Proc. of ACM SIGMOD*, pages 963–968. ACM, 2010.

Aydin Buluç und John R. Gilbert. Highly Parallel Sparse Matrix-Matrix Multiplication. *CoRR*, abs/1006.2183, 2010.

CuSparse. Nvidia CuSparse Library, n.d. http://docs.nvidia.com/cuda/cusparse [Accessed: 2016-04-05].

Jack Dongarra, Susan Blackford, Sven Hammarling, Iain Duff, und Jim Demmel. Basic Linear Algebra Subprograms Technical (BLAST) Forum Standard. *IJHPCA*, 16(1):1–111, 2001.

Jack Dongarra, Mark Gates, Azzam Haidar, Jakub Kurzak, Piotr Luszczek, Stanimire Tomov, und Ichitaro Yamazaki. Accelerating Numerical Dense Linear Algebra Calculations with GPUs. *Numerical Computations with GPUs*, pages 1–26, 2014.

Franz Färber, Jonathan Dees, Martin Weidner, Stefan Baeuerle, und Wolfgang Lehner. Towards a web-scale data management ecosystem demonstrated by SAP HANA. In *Proc. of 31st ICDE*, pages 1259–1267, April 2015.

H. Garcia-Molina und K. Salem. Main Memory Database Systems: An Overview. *IEEE Trans. Knowl. Data Eng.*, 4(6):509–516, December 1992.

Amol Ghoting, Rajasekar Krishnamurthy, Edwin Pednault, et al. SystemML: Declarative Machine Learning on MapReduce. In *Proc. of 27th ICDE*. IEEE, 2011.

Sadashiva S. Godbole. On Efficient Computation of Matrix Chain Products. *IEEE Trans. Comput.*, 22(9):864–866, September 1973.

Jim Gray. eScience – A Transformed Scientific Method. NRC-CSTB meeting, 2007.

Tony Hey, Stewart Tansley, und Kristin M. Tolle. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, 2009.

George Karypis. METIS and ParMETIS. In *Encyclopedia of Parallel Computing*, pages 1117–1124. 2011.

David Kernert, Frank Köhler, und Wolfgang Lehner. SLACID - Sparse Linear Algebra in a Column-oriented In-memory Database System. In *Proc. of 26th SSDBM*, pages 11:1–11:12. ACM, 2014.

David Kernert, Frank Köhler, und Wolfgang Lehner. SpMacho - Optimizing Sparse Linear Algebra Expressions with Probabilistic Density Estimation. In *Proc. of 18th EDBT*, pages 289–300, 2015.

David Kernert, Wolfgang Lehner, und Frank Köhler. Topology-Aware Optimization of Big Sparse Matrices and Matrix Multiplications on Main-Memory Systems. In *Proc. of 32nd ICDE*, 2016.

Moritz Kreutzer, Jonas Thies, Melven Röhrig-Zöllner, Andreas Pieper, et al. GHOST: Building Blocks for High Performance Sparse Linear Algebra on Heterogeneous Systems. *CoRR*, abs/1507.08101, 2015.

Peter Macko, Virendra J. Marathe, Daniel W. Margo, und Margo I. Seltzer. LLAMA: Efficient graph analytics using Large Multiversioned Arrays. In *Proc. of 31st ICDE*, pages 363–374, 2015.

Per-Gunnar Martinsson, Vladimir Rokhlin, und Mark Tygert. A randomized algorithm for the decomposition of matrices. *Applied and Computational Harmonic Analysis*, 30(1):47 – 68, 2011.

Tim McGuire, James Manyika, und Michael Chui. Why Big Data is the New Competitive Advantage. *Ivey Business Journal*, July 2012. http://iveybusinessjournal.com/publication/why-big-data-is-the-new-competitive-advantage/ [Accessed: 2016-04-05].

Md.Mostofa Ali Patwary, Nadathur Rajagopalan Satish, Narayanan Sundaram, et al. Parallel Efficient Sparse Matrix-Matrix Multiplication on Multicore Platforms. In *Proc. of 30th Int. Conf. ISC High Perf.*, volume 9137 of *LNCS*, pages 48–57. Springer, 2015.

Robert Roth. Importance Truncation for Large-Scale Configuration Interaction Approaches. *Phys.Rev.*, C79, 2009.

Yousef Saad. *SPARSKIT: A Basic Tool Kit for Sparse Matrix Computations*. University of Minnesota Department of Computer Science and Engineering, version 2 edition, June 1994.

Vishal Sikka, Franz Färber, Wolfgang Lehner, Sang Kyun Cha, Thomas Peh, und Christof Bornhövd. Efficient Transaction Processing in SAP HANA Database: The End of a Column Store Myth. In *Proc. of ACM SIGMOD*, pages 731–742. ACM, 2012.

Michael Stonebraker, Chuck Bear, Ugur Çetintemel, Mitch Cherniack, Tingjian Ge, et al. One Size Fits All? Part 2: Benchmarking Studies. In *Proc. of 3rd CIDR*, pages 173–184, 2007.

Michael Stonebraker, Sam Madden, und Pradeep Dubey. Intel „Big Data“ Science and Technology Center Vision and Execution Plan. *SIGMOD Rec.*, 42(1):44–49, May 2013.