

Hochschule für Technik, Wirtschaft und Kultur Leipzig

Fakultät Informatik, Mathematik und Naturwissenschaften

Masterstudiengang Informatik

Masterarbeit

zur Erlangung der akademischen Grades

Master of Science (M.Sc.)

Untersuchung quelloffener verteilter geografischer Informationssysteme zur Verarbeitung agrartechnischer Kennzahlen

Eingereicht von: Kurt Junghanns

Matrikelnummer: 59886

Leipzig 28. Mai 2015

Erstprüfer: Prof. Dr. rer. nat. Thomas Riechert

Zweitprüfer: M. Sc. Volkmar Herbst

Bibliografische Angaben

Kurt Junghanns: Untersuchung quelloffener verteilter geografischer Informationssysteme zur Verarbeitung agrartechnischer Kennzahlen, 79 Seiten, 30 Abbildungen, 20 Tabellen, Hochschule für Technik, Wirtschaft und Kultur, Fakultät Informatik, Mathematik und Naturwissenschaften

Die druckbare PDF Version dieser Arbeit sowie alle relevanten Datenquellen sind unter der Website <https://github.com/tboonx/masterthesis> verfügbar.

Masterarbeit, 2015

Satz: L^AT_EX

Abstrakt

In dieser Arbeit werden quelloffene verteilte geografische Informationssysteme für den Anwendungsfall des Einsatzes bei der Agri Con GmbH untersucht, bewertet und ausgewählt. Die Agri Con GmbH agiert als Berater, Verkäufer und Dienstleister für Agrarunternehmer mit dem Schwerpunkt der Modernisierung der Landwirtschaftlichen Vorgänge, um ein niedriges Verhältnis von Aufwand zu Ertrag zu erhalten. Dieses mittelständische Unternehmen zielt auf eine kostengünstige und effiziente Datenhaltung und -verarbeitung ab. Die Anforderungen der Agri Con GmbH zur Auswahl eines Systems werden in Form von Qualitätsmetriken messbar gemacht, wodurch mit einer Nutzwertanalyse eine Bewertung möglich ist. Diese Nutzwertanalyse wird auf die Frameworks GeoMesa, Postgres-XL und Rasdaman angewandt. An Hand dieser Bewertung erfolgt die Auswahl des Frameworks mit dem höchsten Nutzwert, hier Postgres-XL. Dieses wird anschließend genauer untersucht, wobei auf die Verwendung und den möglichen Einsatz bei der Agri Con GmbH eingegangen wird. Weiterhin werden Tests zur Feststellung von ausgewählter Funktionalität und Leistungsfähigkeit bezüglich Schwächen des aktuellen Systems durchgeführt. Um die Vergleichbarkeit zu gewährleisten, wurde das aktuelle System in eine virtuelle Maschine exportiert und Postgres-XL in eine identische virtuelle Maschine installiert. Die Arbeit endet mit einer Zusammenfassung, einer Empfehlung bzw. Wertung der Ergebnisse und einem Ausblick auf die zukünftige Handhabung der räumlichen Daten bei der Agri Con GmbH.

Im Ergebnis der Arbeit wird gezeigt, dass die untersuchte Funktionalität von Postgres-XL für den Anwendungsfall ausreichend ist, jedoch ein produktiver Einsatz nicht empfohlen werden kann. Gegen einen produktiven Einsatz spricht, dass Trigger in der aktuellen Version nicht unterstützt werden und die Erweiterung Plr nicht verteilt verwendet werden kann. Das Ergebnis hinsichtlich der Leistungsfähigkeit ist kein signifikanter Leistungszuwachs im Gegensatz zum aktuellen Stand. Signifikant meint 30% bis 100% Leistungszuwachs bei 200% Hardwareeinsatz. Ein weiteres Ergebnis ist, dass diese Arbeit als Handlungsempfehlung zur Analyse von Frameworks bei ähnlichen Anwendungsfällen verwendet werden kann.

Danksagung

Meinen Dank möchte ich Volkmar Herbst, den Leiter der Softwareabteilung der Agri Con GmbH, aussprechen, da er mich neben dieser Arbeit bereits im Praktikum und meiner Bachelorarbeit betreut hat. Weiterhin danke ich Herrn Professor Riechert für die fachliche Betreuung. Den größten Dank möchte ich aber meinen Eltern widmen, da sie mich nicht nur finanziell unterstützt haben sondern immer mit Rat und Tat zur Seite standen.

Inhaltsverzeichnis

Glossar	vi
Abkürzungsverzeichnis	ix
Abbildungsverzeichnis	xi
Tabellenverzeichnis	xiii
1 Einleitung	1
1.1 Zielsetzung	1
1.2 Einordnung der Arbeit	2
1.3 Aufbau der Arbeit	2
2 Grundlagen	4
2.1 Datenbankmanagementsysteme	4
2.2 Verarbeitung raumbezogener Daten	5
2.3 Alternativen zum relationalen Datenbankmodell	9
3 Methodisches Vorgehen	18
4 Ausgangsszenario	24
4.1 Anforderungen	24
4.2 Aktuelles System	31
4.3 Stand der Forschung	34
5 Systemauswahl	37
5.1 Definition der Nutzwertanalyse	37
5.2 Nutzwertanalyse	39
5.3 Zusammenfassung	44
6 Untersuchung von Postgres-XL	45
6.1 Verwendung	45

Inhaltsverzeichnis

6.2	Entwurf	48
7	Tests	51
7.1	Testumgebung	51
7.2	Funktionstests	53
7.3	Leistungstests	57
8	Fazit	74
8.1	Nutzwertanalyse	74
8.2	Zusammenfassung	76
8.3	Wertung	77
8.4	Ausblick	79
A	Anhang	I
A.1	GeoTools Funktionalitäten	I
A.2	Systembewertung	III
A.3	Postgres-XL Installationsskript	IX
A.4	Pl/R Installationsskript	X
A.5	PostGIS Installationsskript	XI
A.6	pgxcctl Konfigurationsskript	XII
A.7	Testdokument Funktionstests	XV
A.8	Mapfile Aggregation Punktdaten	XVI
	Literaturverzeichnis	XX

Glossar

Bonitur Die landwirtschaftliche Beurteilung des Ackers und der Pflanzen zum Zwecke der Planung des Einsatzes von Dünger, Pestiziden, Fungiziden und Herbiziden.

Cascading Das Java Framework Cascading steht unter der Apache License dient der Erstellung komplexer Datenverarbeitungsabläufe. Dafür wird MapReduce indirekt in vereinfachter Form zugänglich gemacht.

EPSG-Code Dies ist eine vier- bis fünfstellige Ziffer zur eindeutigen Identifikation von räumlichen Referenzsystemen. Sie werden von der EPSG herausgegeben und finden weltweit Anwendung.

FastCGI FastCGI ist ein Standard zur Auslieferung von dynamischen Inhalten im Internet in Zusammenarbeit mit einem Webserver. Dabei wurde dieser Standard auf Durchsatz optimiert, indem unter anderem der Interpreter nach Behandlung einer Anfrage im Arbeitsspeicher verbleibt und direkt nachfolgende Aufgaben übernimmt. Es wird als Weiterentwicklung des Common Gateway Interface (CGI) angesehen.

GeoServer Ist ein freier OGC konformer Mapserver der Open Source Geospatial Foundation, geschrieben in Java.

Manifold Internet Map Server Manifold ist eine proprietäre Produktpalette des gleichnamigen Unternehmens. Ein Produkt daraus ist der Internet Map Server, welcher Karten in Bildform aus räumlichen Daten erstellt und zur Verfügung stellt. Dies erfolgt über eine direkte Anwendung und eine Programmbibliothek.

Pig Als Apache Projekt dient Pig zur Abstraktion von Java MapReduce Jobs in der Sprache Pig Latin. Ziel ist eine Vereinfachung von MapReduce mit der gleichzeitigen Einbindung externer Funktionen.

Precision Farming Bedeutet eine individuelle Betrachtung und Bewirtschaftung einzelner Teile von Flurstücken, wodurch Unterschiede des Bodens und die variierende Ertragsfähigkeit innerhalb einer Nutzfläche berücksichtigt werden.

R Die plattformunabhängige Programmiersprache R steht unter der GNU General Public License und wird für statistisches Rechnen und dessen grafische Aufbereitung verwendet.

Scala Scala ist eine objektorientierte funktionale Programmiersprache mit einem statischen Typsystem und ist auf der JVM und LLVM lauffähig.

Spark Apache Spark steht unter der Apache License 2.0 und ist ein Framework zur Datenverarbeitung in Clustersystemen. Im Gegensatz zu Hadoop stellt es eine andere Ausführungsumgebung dar und arbeitet mit HDFS, Apache Cassandra, OpenStack Swift, Amazon S3 und Accumulo zusammen.

Storm Apache Storm ist ein Framework speziell für Stapelverarbeitung von Datenströmen durch verteilte Prozesse. Es steht unter der Apache License 2.0.

Thrift Thrift ist ein bis 2007 von Facebook entwickeltes Framework zur Erstellung skalierender Dienste in beliebigen Programmiersprachen. Es verbindet einen Softwarestapel mit einer generativen Engine.

UMN MapServer Mapserver des OGC unter MIT Lizenz, Erstentwicklung durch Universität von Minnesota, welcher als CGI Modul und für verschiedene Sprachen bereitsteht.

VMware ESXi ESXi ist ein Hypervisor Stufe 1 des Unternehmens VMware und wird als Betriebssystem installiert.

VMware vSphere vSphere ist eine Sammlung von Systemen und Werkzeugen des Unternehmens VMware zur umfassenden Virtualisierung.

Web Coverage Service Ein OGC konformer Dienst zum Abruf von multidimensionalen Daten mit Zeit- und Raumbezug. Diese sind über eine eigene Syntax mit ihren Metadaten abrufbar.

Web Processing Service Dieser OGC konformer Dienst ermöglicht die räumliche Analyse von Daten im geografischen Kontext. Dazu stellt der Dienst Clients Vorschriften und Modelle zur Verfügung.

ZooKeeper ZooKeeper ist ein Projekt der Apache Foundation, was es verteilten Prozessen ermöglicht miteinander zu kommunizieren. Es wird häufig mit Hadoop eingesetzt und zielt auf hohen Durchsatz, geringe Latenzen, Hochverfügbarkeit und effektiven Zugriff durch die Prozesse ab.

Abkürzungsverzeichnis

ACID Atomicity, Consistency, Isolation und Durability

AHP Analytic Hierarchy Process

BSON Binary JSON

DBMS Datenbankmanagementsystem

EPSG European Petroleum Survey Group Geodesy

GDAL Geospatial Data Abstraction Library

GIS Geoinformationssystem

GPL General Public License

HDFS Hadoop File System

IPMI Intelligent Platform Management Interface

JDBC Java Database Connectivity

JSON JavaScript Object Notation

JTS Java Topology Suite

LGPL Lesser General Public License

MPP Massively Parallel Processing

MTTF Mean Time to Failure

MVCC Multi Version Currency Control

Abkürzungsverzeichnis

ODBC Open Database Connectivity

ODBMS objektorientiertes Datenbankmanagementsystem

OGC Open Geospatial Consortium

SNMP Simple Network Management Protocol

TOPSIS Technique for Order Preference by Similarity to Ideal Solution

TPC Transaction Processing Performance Council

VM Virtuelle Maschine

Abbildungsverzeichnis

2.1	Übersicht der Ausführung von Googles MapReduce	13
2.2	Aufbau Postgres-XL	16
2.3	Aufbau Rasdaman	17
3.1	Nutzungsstatistik der Wikipedia Seite spatial database	19
4.1	Aktuelle Infrastruktur bei Agri Con	32
4.2	Aufbau Wunsch-Stand	33
4.3	Antwortzeiten von GeoMesa und PostGIS	35
4.4	TPC-H Benchmark von PostgreSQL und Postgres-XL	36
5.1	Übersicht relevanter GIS Frameworks	38
5.2	Zeitleiste der contributor von GeoMesa	40
5.3	Zeitleiste der commits von GeoMesa	40
5.4	Zeitleiste der contributor von Postgres-XL	42
5.5	Zeitleiste der commits von Postgres-XL	42
6.1	Aufwandsschätzung Umsetzung einer teilweisen Integration von Postgres-XL	49
6.2	Vorgehen zur Erstellung der Postgres-XL Testumgebung	50
7.1	Aufbau der Geräte des Testsystems	52
7.2	Aufbau der VMs des Testsystems	52
7.3	Kartenausschnitt mit 3 Schlägen	55
7.4	Kartenausschnitt mit 2 überlappenden Schlägen	56
7.5	Ergebnisse Verschneidungsfunktionen Intersection, Union und Sym- metric Difference	56
7.6	Ergebnisse Verschneidungsfunktionen Difference	57
7.7	Auslastung der Knoten 1 bis 6 im Lasttest der Aggregation	63
7.8	Auslastung des Knoten 7 im Lasttest der Aggregation	64
7.9	Auslastung der Knoten 1 bis 6 im Lasttest der Aggregation mit Karten- darstellung	65

Abbildungsverzeichnis

7.10	Auslastung des Knoten 7 im Lasttest der Aggregation mit Kartendarstellung	66
7.11	Auslastung der PostgreSQL <i>Virtuelle Maschine (VM)</i> im Lasttest der Aggregation	67
7.12	Auslastung der PostgreSQL <i>VM</i> im Lasttest der Aggregation mit Kartendarstellung	68
7.13	Auslastung der Knoten 1 bis 6 im Lasttest der Verarbeitung	70
7.14	Auslastung des Knoten 7 im Lasttest der Verarbeitung	71
7.15	Auslastung der PostgreSQL <i>VM</i> im Lasttest der Verarbeitung	72

Tabellenverzeichnis

2.1	Beispiele für räumliche Transformationen	7
4.1	Wertungstabelle Funktionsumfang	29
4.2	Wertungstabelle Dokumentation	29
5.1	Wertungsmaßstab der einzelnen Metriken	39
5.2	Nutzwertanalyse GeoMesa	40
5.3	Nutzwertanalyse Postgres-XL	41
5.4	Nutzwertanalyse Rasdaman	43
7.1	Anzahl der Einträge in n.sensorlogs nach fileid	62
7.2	Testergebnis Lasttest Aggregation	63
7.3	Testergebnis Lasttest der Aggregation mit Kartendarstellung	65
7.4	Testergebnis Lasttest der Aggregation mit PostgreSQL VM	66
7.5	Testergebnis Lasttest der Aggregation mit Kartendarstellung und PostgreSQL VM	67
7.6	Anzahl der Einträge in nutrients.samples nach fileid	68
7.7	Testergebnis Lasttest der Verarbeitung	69
7.8	Testergebnis Lasttest der Verarbeitung der PostgreSQL VM	71
7.9	Vergleich der Testergebnisse	72
7.10	Vergleich der CPU Auslastung	73
7.11	Vergleich der Testergebnisse der Verarbeitung	73
8.1	Neuer Wertungsmaßstab der einzelnen Metriken	74
8.2	Neue Nutzwertanalyse von Postgres-XL	75

1 Einleitung

Die Agri Con GmbH verwaltet als Akteur im Bereich *Precision Farming* täglich mehrere Millionen räumliche Datensätze. Diese werden von mit moderner Technik ausgestatteten Landwirtschaftsmaschinen und durch die Verarbeitung durch firmeninterne und firmenexterne Mitarbeiter sowie Systeme erzeugt. Weiterhin fallen dadurch indirekt Vektor- und Rasterdaten an, welche gespeichert und anschließend verarbeitet werden. Aus den Quelldaten werden beispielsweise Vektordaten der Verteilung der Grunddüngung erzeugt. Rasterdaten werden für die Stickstoffdüngung, auch „N-Düngung“ genannt, verwendet, was unter anderem die Biomasse, die Nährstoffaufnahme und die Nährstoffverteilung beinhaltet. Diese Menge an Daten ist essentiell für das Unternehmen und dessen Kunden, weshalb diese strukturiert gespeichert und kostengünstig verarbeitet werden müssen. Nicht nur Agri Con steht vor dieser Notwendigkeit, sondern der Großteil der Unternehmen, die sich mit komplexen Geodaten beschäftigen, wie Monsanto, Google, Facebook, BASF, etc.

1.1 Zielsetzung

PostgreSQL mit der Erweiterung PostGIS erfüllt nicht alle Anforderungen, wenn große Datenmengen zur Laufzeit aggregiert und verarbeitet werden müssen. Eine bereits durchgeführte vertikale Skalierung der Hardware verringerte die Laufzeit kritischer Operationen, jedoch muss das zu Grunde liegende System wachsenden Anforderungen standhalten. Es ist zu untersuchen, welche Vorteile andere Datenhaltungssysteme bieten bzw. welche alternativen Herangehensweisen wie NoSQL und die verteilte Datenhaltung geeignet sind, um die Anforderungen zu erfüllen. Dafür sind existierende *Geoinformationssystem (GIS)* zu untersuchen und deren Eignung für den in Kapitel 4 beschriebenen Anwendungsfall festzustellen. Die Schwerpunkte der Untersuchung sind die Möglichkeiten und die Leistungsfähigkeit der räumlichen Datenverarbeitung und nicht die Formen der Datendarstellung. Aus geeigneten Systemen wird eines ausgewählt. Dieses wird untersucht, für den Einsatz bei Agri Con GmbH validiert und hinsichtlich Funktionalität und

Leistung bewertet. Schlussendlich soll eine Entscheidungsgrundlage in Form einer Nutzwertanalyse anhand von bewerteten Qualitätsmerkmalen und durchgeführten Tests für die Eignung entsprechend des Anwendungsfalles gegeben werden.

Somit soll mit dieser Arbeit ein spezieller Vergleich von verschiedenen Systemen erfolgen und eine Wertung eines geeigneten Kandidaten gegeben werden. Dieses Forschungsziel ist von allgemeinem Interesse, da damit eine Erkenntnislücke befüllt wird, anerkannte Methoden verwendet werden und sich das Vorgehen auf ähnliche Ausgangsszenarios übertragen lässt.

1.2 Einordnung der Arbeit

Im Rahmen der Arbeit erfolgt eine Softwareauswahl mit Hilfe von Nutzwertanalysen. Weiterhin erfolgt eine ergänzende Bewertung eines Frameworks durch Funktions- und Leistungstests. Die untersuchten Frameworks entsprechen der Definition eines *GIS*, sie gehen somit über die Funktionalität und den Umfang über Frameworks zur reinen Datenhaltung und -verarbeitung hinaus. Zusätzlich handelt es sich um räumliche Daten und raumbezogene Datenverarbeitung. Die Bewertung von *GIS* erfolgt mit Bewertungsmaßstäben von Informationssystemen (vgl. Kapitel 4.1.1), mit Berücksichtigung des Spezialfalles des räumlichen Bezuges der Daten. Die Einschränkung auf verteilt arbeitende Frameworks führt allein genommen bereits zur Betrachtung von Alternativen des relationalen Modells, da bekannte Vertreter eine verteilte Arbeitsweise unterstützen. Die Methodik lässt sich auf ähnliche Ausgangsszenarios übertragen. Sie lässt sich somit als Handlungsempfehlung zur Auswahl eines *GIS* Frameworks verwenden.

1.3 Aufbau der Arbeit

Zum Verständnis der Arbeit werden in Kapitel 2 theoretische Grundlagen und Fachbegriffe zu Datenbanken, räumlicher Datenverarbeitung und NoSQL dargestellt. Daran schließt sich in Kapitel 3 die Darstellung und Begründung des methodischen Vorgehens an, was Definitionen der Begriffe Nutzwertanalyse und Funktions- sowie Leistungstest beinhaltet. Anschließend definiert Kapitel 4 das Ausgangsszenario, für welches die Frameworks analysiert und getestet werden sollen, sowie die dazugehörigen Anforderungen in Form von Softwarequalität. Darauf folgend bewertet eine Nutzwertanalyse ausgewählte Frameworks nach den Anforderungen des Anwendungsfalles (vgl. Kapitel 5). Das darauf folgende Kapitel 6 stellt das ausgewählte Framework unter den Punkten Installation, Schnitt-

1 Einleitung

stellen sowie Verarbeitung dar und präsentiert die Möglichkeiten zum Einsatz bei der Agri Con GmbH. Entsprechend den Anforderungen werden Funktions- und Leistungstest im Kapitel 7 durchgeführt und ausgewertet. Die Arbeit endet mit einer Zusammenfassung, einer Empfehlung bzw. Wertung der Ergebnisse und einem Ausblick auf die zukünftige Handhabung der räumlichen Daten bei der Agri Con GmbH in Kapitel 8.

2 Grundlagen

Dieses Kapitel stellt die für die weiteren Ausführungen notwendigen Begriffe und Systeme vor. Entsprechend dem Inhalt dieser Arbeit werden generelle Grundlagen zu Informationssystemen und zur geografische Datenverarbeitung dargelegt. Weiterhin werden verwendete Methoden und Systeme dargestellt.

Der zentrale Begriff des *GIS* bedingt eine Auseinandersetzung mit Datenbanken und Informationssystemen. Weiterhin ist die Mächtigkeit der zu untersuchenden Systeme abzugrenzen. Dies erfolgt mit der Definition von Frameworks und von *GIS*. Ein Framework ist eine Softwareumgebung zur wiederverwendbaren Herstellung einer Struktur oder Anwendung. Entweder werden Anwendung mit Frameworks vervollständigt oder aus daraus erstellt. In dieser Arbeit dienen Frameworks, oder auch Ordnungsrahmen genannt, zur Lösung spezieller Aufgaben und sind somit domänenspezifische Frameworks. Das heißt, dass notwendige Funktionen und Strukturen zur Lösung von speziellen Aufgaben bereits vorhanden sind, die konkreten Lösungen müssen jedoch mit Hilfe des Frameworks erstellt werden. Die am Ende dieses Kapitels vorgestellten Frameworks zählen zu Alternativen des relationalen Datenbankmodell, weshalb dieser und dazugehörige Begriffe definiert werden.

2.1 Datenbankmanagementsysteme

Grundlegende Kenntnisse zu Datenbankmanagementsystemen und deren Mechanismen sind Voraussetzung für das Verständnis von Informationssystemen. Dazu zählt Sharding. Bei Sharding wird eine Relation in disjunkte Partitionen aufgeteilt, die auf verschiedenen Platten gespeichert werden. Vorteile dieser Methode sind Anfrageoptimierung durch Auslastung der Partitionen, Vereinfachung der Administration der Partitionen und parallele Verarbeitung (vgl. [Kud07, S.296]). Dies setzt einen auf dieser Weise angepassten Query-Planer des *Datenbankmanagementsystem (DBMS)* voraus. Nach Kudraß wird ebenfalls nach drei Arten unterschieden. Konkret sind das Bereichspartitionierung, Round-Robin-Partitionierung und Hash-Partitionierung.

Eine Erweiterung des aus *Atomicity, Consistency, Isolation und Durability (ACID)* bekannten Transaktionsmanagement ist *Multi Version Currency Control (MVCC)*, welches im Framework Postgres-XL zum Einsatz kommt. In grundlegenden relationalen Systemen werden Transaktionen verzögert oder sogar gesperrt, um Konsistenz und Isolation zu gewährleisten. *MVCC* erhöht die Effizienz des blockierenden Verhaltens. Dabei werden von jedem Objekt mehrere Versionen verwaltet. Neue Versionen entstehen durch Änderungen einer anderen. Eine Transaktion verwendet die zu Transaktionsbeginn aktuelle Version. Dadurch werden die allgemeinen Sperrverfahren (siehe [Kud07, S.266 ff.]) verbessert, indem lesende Transaktionen sich nicht gegenseitig blockieren und schreibende- gegen lesende Transaktionen nicht mehr synchronisiert werden müssen. (vgl. [Kud07, S.270])

Da in dieser Arbeit verteilt arbeitende Frameworks untersucht werden, ist die Definition dieses Begriffes notwendig. Nach [Kud07, S.394] wird bei einem Mehrrechner-Datenbanksystem (MDBS) die Datenbankverwaltungsfunktionen auf mehreren Prozessoren bzw. Rechnern ausgeführt. Kudraß ergänzt dies durch folgende Unterscheidungen: Ein *DBMS* befindet sich auf eng gekoppelter Multiprozessor-Umgebung, was als *shared everything* bezeichnet wird. Erfolgt die Verarbeitung durch mehrere Rechner mit jeweils einem *DBMS*, wobei der Externspeicher unter den beteiligten Rechnern partitioniert ist, wird es *shared nothing* genannt. Bei *shared disk* handelt es sich um mehrere lokal angeordnete, lose oder nah gekoppelte Rechner mit je einem *DBMS* und einer gemeinsamen Speicherzuordnung. Lokal verteilte Systeme werden als parallele Datenbanksysteme bezeichnet. Ein Spezialfall stellen verteilte Datenbanksysteme dar. [Kud07, S.398] beschreibt Verteilte Datenbanksysteme (VDBS) als geografisch verteilte *Shared-Nothing* Systeme mit homogenen lokalen *DBMS*, die gemeinsam ein globales konzeptionelles DB-Schema unterstützen. Dagegen sind förderierte Datenbanksysteme (FDBS) ebenfalls geografisch verteilte *Shated nothing* systeme, wobei die beteiligten lokalen *DBMS* eine höhere Autonomie aufweisen, d.h. dass jeweils eine eigene lokale Datenbank mit lokalem DB-schema vorliegt.

2.2 Verarbeitung raumbezogener Daten

Diese spezielle Form der Datenverarbeitung berücksichtigt geografische und topologische Eigenschaften. Diese sind nach der Art und deren Bezug zueinander zu unterscheiden. Der in diesem Zusammenhang in der englischen Literatur verwendete Begriff „*spatial data*“ wird als „räumliche Daten“ ins deutsche übersetzt, weshalb das Attribut räumlich nachfolgend raumbezogen ersetzt.

2.2.1 Räumliche Bezugssysteme

Entsprechend [Kud07, S.506] erlauben räumliche Bezugssysteme die Interpretation der gespeicherten Koordinaten als Beschreibung von Lage- und Ausdehnungsinformationen in einem Datenraum. Ein räumliches Bezugssystem besteht aus einem Koordinatensystem, einem Geltungsbereich und Angaben, die es erlauben, Daten aus unterschiedlichen Koordinatensystemen auf ein globales System abzubilden. Lange ergänzt die allgemeine Definition von Kudraß indem er eine Unterscheidung der Koordinatensysteme nach kartesisch, homogen, Kugeltransformation und Ellipsoidentransformation durchführt, wobei den kartesischen einer hoher Stellenwert zugeordnet wird (vgl. [Lan13, S.141 ff.]).

Allen Bezugssystemen wird zur Identifikation ein weltweit eindeutiger Code zugeordnet. Dieser ist ein von der *European Petroleum Survey Group Geodesy (EPSG)* vergebener so genannter *EPSG-Code*. Das auf einem Ellipsoiden basierende Bezugssystem World Geodetic System von 1984 (WGS84) mit dem EPSG Code EPSG:4326¹ wird von der Agri Con GmbH verwendet.

2.2.2 Räumliche Objekte

Lange definiert räumliche Objekte bzw. Geoobjekte als Elemente die zusätzlich zu ihrer Sachinformationen geometrische und topologische Eigenschaften besitzen und zeitlichen Veränderungen unterliegen können. Dabei sind Geometrie, Topologie, Thematik und Dynamik kennzeichnend. Ein Geoobjekt enthält als Geometrie eine oder mehrere zwei- oder dreidimensionale Koordinaten, was die Lage, den Umfang und die Ausdehnung beschreibt. Zur Topologie zählt de Lange Umgebungen, Nachbarschaften, Teilmengen und Überlagerungen. Weiterhin werden Geoobjekte mit Sachinformationen gespeichert und je nach Anwendungsfall versioniert. (vgl. [Lan13, S.133])

Einfache Objekte

Ein Punkt besteht aus einer zwei- oder dreidimensionalen Koordinate und beliebigen Sach-, Topologie- und Dynamikinformationen. Mehrere Punkte bilden Linien. Bildet eine Linie eine geschlossene Fläche, handelt es sich um ein Polygon. Außerdem können Gruppen von Linien und Polygonen gebildet werden und Multilines und Multipolygone bilden. Multipolygone werden oft verwendet um Löcher in Polygonen abzubilden oder komplexe Polygone zu vereinfachen.

¹EPSG:4326: <http://spatialreference.org/ref/epsg/wgs-84/>

Vektorenmodell

Es besteht die Möglichkeit eine Menge von Punkten als Vektoren aufzufassen und daraus topologische Objekte entstehen zu lassen. Um damit geografisch zu modellieren, ist eine Diskretisierung d. h. eine Zuordnung der Vektoren untereinander notwendig.

Rastermodell

Ein Raster löst einen rechteckigen Bereich mit in einem Koordinatensystem gleichmäßig angeordneten quadratischen Bildelementen bzw. Pixeln fester Größe auf. Geodaten werden ergo mit einer indizierten Matrix abgebildet. Ein geografischer Punkt wird näherungsweise durch ein einzelnes Pixel dargestellt. Linienzüge werden durch entsprechende Anordnungen zusammenhängender Pixel angenähert erfasst. Diese können dann z.B. durch Folgen von Indexpaaren (Zeile, Spalte) der zugehörigen Pixel beschrieben werden. Eine Fläche ist ebenfalls durch zusammenhängende Pixel darstellbar. Somit sind keine weiteren Zusatzinformationen zur Modellierung von Flächen wie im Vektormodell notwendig. (vgl. [Lan13, S.136]) Ein dreidimensionales Raster heißt Voxel.

2.2.3 Räumliche Operationen

Räumliche Operationen, welche in den zu betrachtenden Frameworks eingesetzt werden, sind Transformationen. Dabei werden die Koordinatenreferenzsysteme, die Objekte und die topologischen Informationen geändert. Graeme führt Beispiele solcher Transformationen tabellarisch auf, siehe Tabelle 2.1. Darin werden ausgewählte Transformationen

Von/Nach	Punkte	Linien	Flächen
Punkte	Interpolation	Contouring	Thiessen Polygone
Linien	Linienüberschneidung	Linienglättung	Pufferzonen
Flächen	„sample at points“	Mediale Achse	Wiederholungsproben- nahme auf Flächen

Tabelle 2.1: Beispiele für räumliche Transformationen nach [BC94, S.139]

zwischen Objekten aufgeführt. Es ist zu erwähnen, dass Transformationen nicht nur mit den aufgezeigten Objekten verwendet werden. So ist die Interpolation auch für Punkte zu Linien möglich. Für die folgenden Betrachtungen sind jedoch nur die Transformationen Interpolation und Contouring von Interesse. In allen Transformationen werden Mittel der

Statistik verwendet, da neben geografischen und topologischen Informationen auch Sachinformationen in die Berechnungen einbezogen werden, welche in die Transformation verändert übernommen werden. Diese Verknüpfung von Statistik und räumlichen Operationen heißt Geostatistik.

In den weiteren Ausführungen wird die Transformationsmethode kriging als Spezialfall von Contouring angewandt. Der Entwickler war Kriging, welcher diese Transformation zur Vorhersage von Goldvorkommen in Süd Afrika verwendete (vgl. [Swa95, S.317]). Entsprechend Akin und Simes nach [Ros01, S.76] ist Kriging ein Schätzvorgang, der eine bestimmte Anzahl von Proben in einem Wichtungsprozess so einbezieht, dass die Schätzvarianz zu einem Minimum wird. Kriging wird bei Agri Con eingesetzt, um aus Punktdaten mit Sachinformationen durch Wichtung bei hoher Erwartungstreue der Schätzwerte Vektordaten mit Sachinformationen zu erzeugen.

2.2.4 Geografisches Informationssystem

Ein Informationssystem ist eine Softwareumgebung zur umfassenden Verwendung von Daten. Es stellt Möglichkeiten der Erfassung, Speicherung und Verarbeitung zur Verfügung. Außerdem können Daten analysiert, übertragen, angezeigt und gepflegt werden. Alle Daten und Ergebnisse sind Gegenstand der Verwendung von Informationssystemen. (vgl. [Kud07, S.32]) Wird ein Informationssystem im geografischen Kontext benötigt, wird ein dafür explizit programmiertes benötigt. Die geometrischen und topologischen Informationen der Geodaten müssen sich im Informationssystem widerspiegeln. So hat das System die topologischen Zusammenhänge zu berücksichtigen und die Daten bevorzugt optisch darzustellen.

Lange definiert *GIS* ähnlich:

Im Mittelpunkt der Geoinformatik stehen mit den Geoinformationssystemen raumbezogene Informationssysteme, die im Gegensatz zu den übrigen Informationssystemen Geoobjekte der realen Welt modellieren und diese in ein digitales Informationssystem abbilden [...]. Die Gegenstände eines Geoinformationssystems besitzen wie auch bei allen anderen Informationssystemen eine Thematik (und Dynamik). Das Besondere bei Geoinformationssystemen ist, dass Geoobjekte darüber hinaus Geometrie und Topologie als implizite und untrennbare Bestandteile aufweisen! Die Verarbeitung derartiger raumbezogener Informationen erfordert spezielle Werkzeuge bzw. Funktionen, die von den übrigen Informationssystemen nicht bereitgestellt werden [...]. [Lan13, S.337]

Die Klärung dieses Begriffes ist wichtig, da die zu untersuchenden Frameworks *GIS* sind und anhand der Definition eines *GIS* untersucht und bewertet werden.

2.2.5 Java Bibliothek GeoTools

GeoTools² ist eine in Java geschriebene Open Source Bibliothek, welche Standardkonforme Operationen zur Verarbeitung von geografischen Daten bereitstellt. Diese ist Teil der Open Source Geospatial Foundation (OSGeo)³. Sie wird von Java nahen Frameworks im Kontext von *GIS* verwendet, sodass die Bewertung dieser Frameworks die Kenntnis über GeoTools voraus setzt. Die Implementation erfolgte nach Anforderungen des *Open Geospatial Consortium (OGC)*, worauf beispielsweise Geometrien des *Java Topology Suite (JTS)* unterstützt werden und die OGC Filter Encoding Spezifikation von Attributen und räumlichen Filtern verwendet wird (vgl. [Geo15]). Eine detaillierte Auflistung der Funktionalitäten ist im Anhang A.1 zu finden.

2.2.6 PostgreSQL mit Erweiterung PostGIS

PostGIS⁴ ist eine geografische Erweiterung der Objekt-relationalen Datenbank PostgreSQL⁵. PostgreSQL wird dabei um geografische Datentypen, geografische Indizes und Funktionen erweitert. Konkret wird der Simple Feature Access Standard verwendet und um den Datentyp raster und weitere Funktionen zur Datenverarbeitung erweitert. (siehe [OSG15b]) Somit kann mit SQL direkt mit geografischen Daten gearbeitet werden. PostGIS ist ein Projekt der Refractions Research⁶ und steht unter der *General Public License (GPL)*v2.

2.3 Alternativen zum relationalen Datenbankmodell

Alternativen des relationalen Datenbankmodells werden unter dem Stichwort NoSQL zusammengefasst und dieses soll hier als Synonym dienen. NoSQL ist ein Begriff, dessen Kontext die Abkehr von klassischen relationalen Systemen fordert oder zumindest ein Umdenken bestehender Strukturen, Vorgehen und Grundsätze anstrebt. Dies wird durch

²GeoTools: <http://docs.geotools.org/>

³Open Source Geospatial Foundation: <http://www.osgeo.org/>

⁴PostGIS: <http://postgis.org/>

⁵PostgreSQL: <http://www.postgresql.org/>

⁶Refractions Research: <http://www.refractions.net/>

andere Abfragesprachen, nicht relationale Datenbankmodelle oder Neudefinitionen von Begriffen wie der Konsistenz zum Ausdruck gebracht. Zu den Gründen des Umdenkens zählen die im Web 2.0 anfallenden unstrukturierten Daten, welche kostenarm persistiert und Zugänglich gemacht werden müssen. Dafür wurden herkömmliche relationale Systeme nicht konzipiert. Die Menge an Daten und die geografische Verteilung dieser erzwingt die Einführung neuer Methoden und Prinzipien. Der Ursprung wird in der Literatur verschieden hergeleitet, jedoch wird immer zu den ersten Vertretern der NoSQL Bewegung Systeme mit einer anderen Abfragesprache und einfache Key Value Datenbanken gezählt. Auf einer Messe zu aktuellen Trends im Datenbankbereich wurde der Begriff NoSQL zuerst öffentlich für Lösungen dieser Bewegung verwendet (vgl. [Eva09]) und ist seitdem ein Sammelbegriff für eine hohe Anzahl an Systemen. Einen Überblick der bestehenden Systeme stellt Edlich auf einer eigenen Homepage bereit.⁷

In Bezug auf NoSQL kann *GIS* wie in 2.2.4 definiert werden, jedoch muss das zugrunde liegende System nicht relational sein. Im Rahmen dieser Arbeit ist mit *GIS* ein System oder die Teilsysteme zur räumlichen Datenhaltung, Datenverarbeitung und Bereitstellung gemeint, unabhängig des Konsistenzbegriffes und der Abfragesprache.

2.3.1 Kategorisierung

Edlich unterscheidet NoSQL Datenbanken nach den vier unten aufgeführten Kategorien [Edl11]. Jedoch kann eine eindeutige Zuteilung nicht für jedes System erfolgen, da Prinzipien verschiedener Kategorien auf eines zutreffen können.

Key Value Datenbank

Key Value Datenbanken speichern Daten in Tupeln aus Schlüssel und Wert. Der Key ist eine Zeichenkette oder ein Hashwert und der Datentyp von Value ist beliebig im Rahmen der Datentypen der Datenbank. Datenzugriff erfolgt über Key. Es existiert keine einheitliche Abfragesprache. Erste Datenbanken die zu NoSQL zugeordnet werden sind Key Value Datenbanken. Konkret DBM und BerkleyDB. Aktuelle Vertreter sind Amazon Dynamo, Riak, Voldemort und Redis. Diese Datenbanken eignen sich für heterogene Daten, horizontale Skalierung und Schemafreiheit, da diese einfach strukturierten Daten sich in keiner Relation zueinander befinden.

⁷Übersicht über NoSQL DBMS: <http://nosql-database.org/>

Dokumentenbasierende Datenbank

Hierbei werden strukturierte Daten, hier Dokumente, unter einem Hash abgelegt und sind über diesen abrufbar. Diese Dokumente sind im großteil der dokumentenbasierten Datenbanken versioniert. Häufige Formate sind *JavaScript Object Notation (JSON)*, *Binary JSON (BSON)* und YAML. Ziel ist hier schemafreie Daten zu speichern und den Zugriff zu skalieren. Dabei können zumeist keine Joins verwendbar. Bekannte Vertreter sind MongoDB, CouchDB und Terrastore.

Spaltenorientierte Datenbank

Im Gegensatz zu zeilenorientierten Datenbanken legen spaltenorientierte Datenbanken ihre Werte, hier Attribute einer Tabelle, spaltenweise ab. Dies eignet sich für OLAP und Data Warehouse, da Spalteneinfügungen kostengünstig und Garbage Collection effektiv ist. Dagegen besteht ein hoher Aufwand beim Lesen und Schreiben von zusammengehörigen Spaltendaten.

Googles Big Table erweitert diesen Ansatz und beschreibt es in dessen Paper wie folgt:

A Bigtable is a sparse, distributed, persistent multi-dimensional sorted map. The map is indexed by a row key, column key, and a timestamp; each value in the map is an uninterpreted array of bytes. [FC06, S.1]

Die mehrdimensionalen Tabellen oder Maps sind vom Format:

$n * [Domain/Keyspace] \times [item/Column\ Family] \times [Key\ x] * n * [key + Value]$ Googles Ansatz wurde OpenSource in HBase und Cassandra umgesetzt. Die konkrete Implementierung von Google wurde jedoch nicht veröffentlicht. HBase verwendet folgendes Format: Pro Table Zugriff auf Zeilen per Rowkey, diese enthalten Column Familys oder Spalten welche wiederum eine Map namens Column Qualifier mit Tupeln aus der Version als Schlüssel und ein Byte-Array als Wert besitzen (vgl. [Hä13, S.13]).

Graphenbasierte Datenbank

Der bekannteste Vertreter der graphenbasierten Datenbanken ist Neo4J. Alle Daten und deren Beziehungen werden in Form von Graphen persistiert. Ein Graph besteht dabei aus Knoten und gerichteten Kanten. Knoten sind dabei strukturierte Objekte und Kanten Beziehungen zwischen den Objekten. Diese strukturierten Objekte sind Key Value Tupel.

Kanten können typisiert sein. Somit lassen sich direkt Beziehungen zwischen Daten definieren, was sich für semantic web, social network, Bioinformatik und Internetrouting eignet.

Diese Datenbanken sind nur optional mit einem Schema versehen und besitzen keine einheitliche Abfragesprache. Auch sind im allgemeinen keine Joins vorgesehen.

2.3.2 Hadoop

Hadoop⁸ ist ein unter der Apache Lizenz 2.0 stehendes Java-Framework zur Datenhaltung und Verarbeitung von großen Datenmengen auf einem Verbund von mehreren Computern. Es basiert auf MapReduce und dem Dateisystem HDFS.

Das *Hadoop File System (HDFS)* ist ein verteiltes Dateisystem, welches keine besonderen Anforderungen an die Hardware stellt und für die Verwendung von mehreren hundert bis mehreren tausend Computern ausgelegt ist. Die in einem verteilten System teilnehmenden Computer heißen Knoten. Es besitzt eine hohe Fehlertoleranz und ist für den Einsatz auf kostengünstiger Hardware ausgelegt. Hoher Datendurchsatz und die Verwendung großer Dateien⁹ sind wesentliche Merkmale. (vgl. [Bor07, S.3]) Die Datei-Blöcke werden redundant auf die Knoten verteilt und sind mit Hilfe des Name-Node abrufbar (vgl. [Hä13, S.7]).

Die verteilte Verarbeitung übernimmt MapReduce. Entsprechend dem Namen entspringt der Name MapReduce aus der funktionalen Programmierung, in welcher die Funktionen „map“ und „reduce“ zum Einsatz kommen. So werden hier die Daten mit einer map-Funktion modifiziert gesammelt und mit reduce-Funktion aggregiert. Ein Master weist die Daten und Funktionen den Slaves zu, in diesem Zusammenhang werden die Slaves Worker genannt. Die Slaves führen die Funktionen mit den ihnen zugewiesenen Daten aus und speichern ihre Ergebnisse auf deren Festplatte ab. MapReduce wurde von Google definiert. In Abbildung 2.1 ist der beschriebene Ablauf dargestellt. Auch hier werden keine besonderen Anforderungen an die Hardware gestellt (vgl. [JD04, S.3]). Hadoop besitzt eine Master-Slave Architektur, wobei der Name-Node¹⁰ ankommende Anfragen bearbeitet und die Slave-Knoten organisiert. Hadoop ist per API verwendbar und bietet sich somit zur Stapelverarbeitung an. Es wird meist nur als Grundgerüst verwendet und

⁸Apache Hadoop: <https://hadoop.apache.org/>

⁹Eine Datei kann mehrere Gigabyte bis mehrere Terrabyte groß sein und wird in Blöcke gleicher Länge aufgeteilt.

¹⁰Mit Name-Node ist der Master-Knoten gemeint, welcher in der Literatur auch das Synonym Jobtracker besitzt.

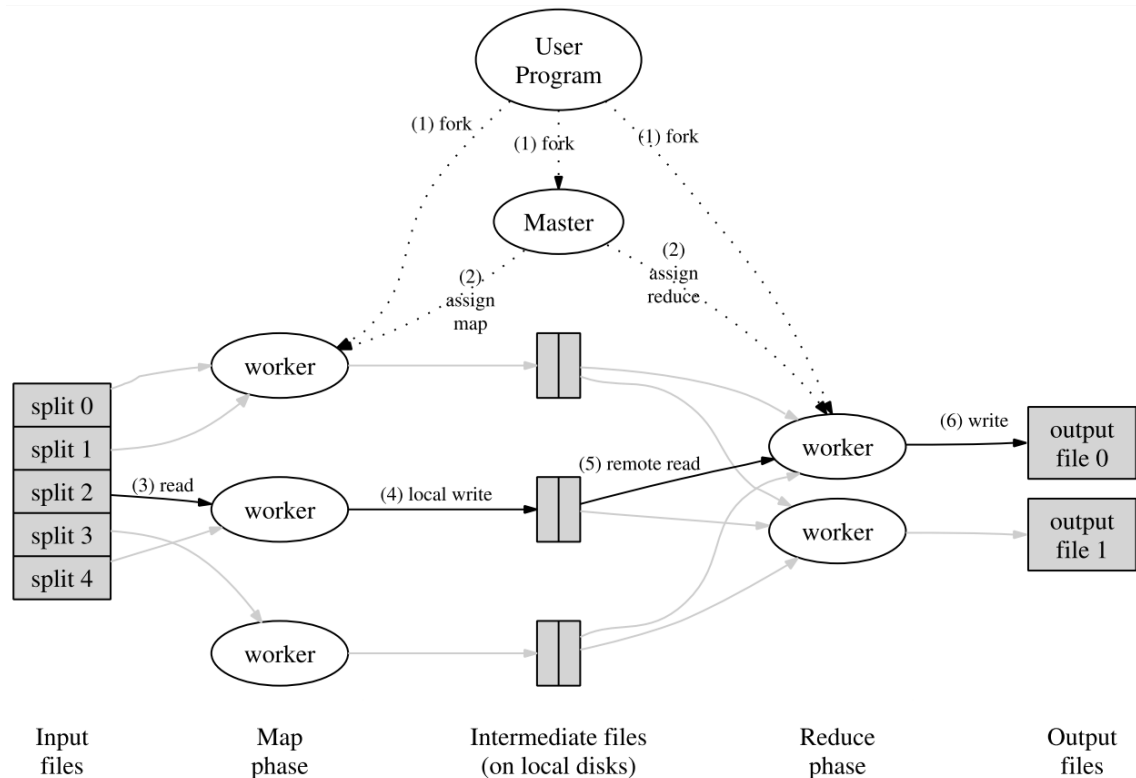


Abbildung 2.1: Übersicht der Ausführung von Googles MapReduce, Quelle: [JD04] S. 3

mit Datenbanken wie HBase, MongoDB oder PostgreSQL sowie mit Frameworks für die Nutzung wie Hive, *Pig*, *Spark* oder Scalding erweitert.

Gegenüber der Möglichkeit auf unterschiedlicher Hardware direkt und gleichzeitig mit Terrabyte großen Daten zu arbeiten, steht die Kritik das MapReduce eine hohe IO auf Festplatten der einzelnen Systeme erzeugt, da alle Zwischenergebnisse auf den Festplatten abgelegt und anschließend gelesen werden.

2.3.3 Accumulo

Hierbei handelt es sich seit 2008 um ein Apache Projekt¹¹ und eine Java Open-Source Implementation des BigTable Ansatzes von Google. Es verwendet Hadoop, *ZooKeeper* und *Thrift*. Der BigTable Ansatz wird um Iteratoren, Zellenbezeichnungen, Constraints, physische Verteilungsmöglichkeiten einer dokumentbasierten Datenbank und die Unterstützung der gleichzeitigen Verwendung mehrerer HDFS namenodes erweitert. Weitere Funktionen sind folgende:

- Verwendung mehrerer Master

¹¹ Apache Accumulo: <https://accumulo.apache.org/>

- Verwendung einer eigenen Zeitsynchronisation
- Eingebaute temporäre Datenhaltung im Arbeitsspeicher
- Bereitstellung von Testimplementierungen per API

Es existieren weiterhin verschiedene Erweiterungen zum Datenmanagement und Änderung des Ordnungsrahmens zur Verfügung. (vgl. [Sof14a])

2.3.4 GIS GeoMesa

GeoMesa¹² ist eine unter Apache License Version 2.0 stehende geografische Datenbank der Firma LocationTech¹³ mit den Möglichkeiten der verteilten Verarbeitung und Versionierung von geografischen Daten. Dieses Framework ist in *Scala* geschrieben. Es erweitert Accumulo, unterstützt die GeoTools API und bietet ein Plugin für den Mapserver *GeoServer* an. Die Daten werden nach Geohash verwaltet. (vgl. [Fox14])

GeoMesa wird in Verbindung mit stream processing¹⁴ und batch processing¹⁵ verwendet. Zur räumlichen Datenverarbeitung werden Scala Bibliotheken wie *JTS* und *GeoTools* eingesetzt. Vorrangig werden Vektordaten von GeoMesa verarbeitet, durch eine optionale Erweiterung sind auch Rasterdaten verwendbar. Datenimport wird ingest genannt, erfolgt über die Kommandozeile und unterstützt die Datenformate CSV, TSV und SHP. CSV, TSV, Shapefile, GeoJSON, and GML können dagegen über den selben Weg exportiert werden. Weiterhin erfolgt der Datenexport und -import über Scala. (vgl. [Loc14a]) GeoMesa ist konzipiert, um initial große Datenmengen per Ingest zu laden und diese anschließend mit Frameworks zur verteilten Datenverarbeitung wie Spark und dafür vorgesehenen Bibliotheken zu verarbeiten.

2.3.5 Postgres-XL

Postgres-XL¹⁶ ist ein frei verfügbares Clustersystem für PostgreSQL unter der Mozilla Public License des Unternehmens TransLattice¹⁷. XL steht dabei für eXtensible Lattice, erweiterbarer Verbund. Damit soll es ermöglicht werden, mit PostgreSQL verteilt Schreiboperationen zu skalieren sowie parallele Datenverarbeitung auf mehreren physischen

¹²GeoMesa: <http://www.geomesa.org/>

¹³LocationTech: <https://www.locationtech.org/>

¹⁴Bspw. Spark oder Storm

¹⁵Bspw. Pig oder Cascading

¹⁶Postgres-XL: <http://www.postgres-xl.org/overview/>

¹⁷TransLattice: <http://www.translattice.com/>

und virtuellen Systemen gleichzeitig zu betreiben. Es handelt sich um ein shared nothing Mehrrechner-Datenbanksystem. Dafür wird zur verteilten Datenhaltung *ACID* mit *MVCC* und zur parallelen Verarbeitung ein *Massively Parallel Processing (MPP)* Mechanismus eingesetzt. Die Postgres-XL Umgebung nutzt mehrere PostgreSQL Instanzen und bietet Schnittstellen für alle Instanzen an. (siehe [Tra15b])

Abbildung 2.2¹⁸ verbildlicht den Aufbau. Danach wird als erstes ein Load-Balancer angesprochen und es existieren mehrere GTM Instanzen. (siehe [Tra15b]) Dies wird in der Dokumentation nicht belegt. Die Elemente sind nach [Tra15b] wie folgt beschrieben:

Global Transaction Manager Dient als Verwaltungselement der Transaktionen und realisiert *MVCC* über das System. Laut Dokumentation existiert genau ein GTM pro Cluster, um *MVCC* mit einem globalen Kontext realisieren zu können. Der GTM wird von jeder Coordinator Instanz angesprochen. Es wird jedoch zusätzlich pro physischem Knoten des Clusters ein GTM-Proxy empfohlen, welcher die Anfragen an den GTM bündelt.

Coordinator Jeder Coordinator dient als Eintrittspunkt in den Cluster, ruft von der GTM Instanz pro SQL Statement eindeutige Transaktionsnummern und globale Informationen ab sowie formuliert Subqueries für die relevanten DataNodes entsprechend seines Query-Planers.

Data Node Diese Elemente sind PostgreSQL Instanzen, welche die konkreten Daten vorhalten und das Transaktionsmanagement an den GTM abgegeben haben. Die Datenbanken und Tabellen der DataNodes werden entweder per partition verteilt oder repliziert. Anfragen können von verschiedenen Coordinators gleichzeitig in unterschiedlichen Sitzungen erfolgen. Auf Grund der Kapselung besitzt jeder Data Node seinen eigenen Kontext zur Transaktion.

Dabei besitzt jeder Coordinator und jeder DataNode einen ConnectionPool, welcher die Verbindungen verwaltet. Veränderungen des Datenbankschemas werden auf alle Coordinators und DataNodes propagiert. Es besteht jedoch die Möglichkeit einzelne DataNodes aus Relationen des Schemas zu entfernen. Damit handelt es sich bei Postgres-XL um ein förderiertes Datenbanksystem. Es wird analog einer PostgreSQL Installation angesprochen. Zur Erhöhung der Ausfallsicherheit, besteht die Möglichkeit, für jedes Element im Cluster eine dazugehörige inaktive Instanz zu erzeugen, welche bei Ausfall der eigentlichen Instanz für diese einspringt. Jedes Element im Cluster ist einzeln zu konfigurieren, was zu $n * 3 + 1$ verschiedenen Konfigurationen bei n DataNodes führt. Zur Vereinfachung

¹⁸Link zur Abbildung: http://www.postgres-xl.org/wp-content/uploads/2014/04/xl_cluster_architecture1.jpg

der Erstellung und Verwaltung eines Clusters sind neben den von PostgreSQL gelieferten Kommandozeilentools weitere durch Postgres-XL gegeben. Zur Erstellung und Verwaltung eines Clusters kann pgxc-ctl eingesetzt werden. Damit kann mit einer Konfigurationsdatei ein Cluster definiert sowie erstellt werden oder das Cluster mit einem Befehl um Knoten ergänzt werden. Ebenso sind Erweiterungen wie PostGIS, DBLink oder PL/R

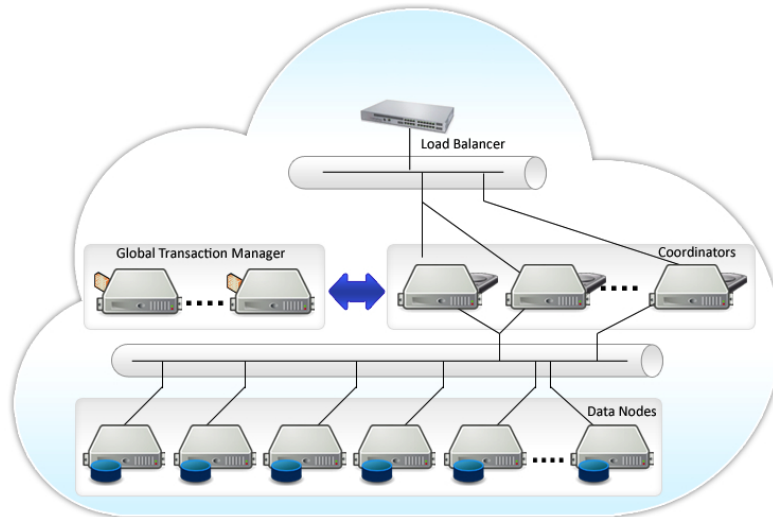


Abbildung 2.2: Aufbau Postgres-XL, nach [Tra15a]

installierbar. Es ist zu erwähnen, dass die Version 9.2.34 keine Trigger unterstützt¹⁹ und der Transition Typ internal nicht verwendet werden kann.

2.3.6 Array DBMS Rasdaman

Rasdaman²⁰ ist ein Array-Datenbanksystem speziell zum speichern und verarbeiten von Rasterdaten der rasdaman GmbH. Es erweitert eine relationale Datenbank und wird mit multi-dimensionalität der Daten, einer eigenen SQL ähnlichen Abfragesprache, Parallelisierung und Skalierbarkeit in beliebigen Maßstab sowie OGC konformen Diensten beworben. Es ist als Client bzw. API unter der *Lesser General Public License (LGPL)* 3 und als Server unter der *GPL* 3 für Linux, MacOS und Solaris verfügbar. Als OGC konforme Dienste werden WMS 1.3, WCS 2.0, WCS-T 1.4, WCPS 1.0 und WPS 1.0 bereitgestellt. Die API kann in Java, C++ und über die eigene Abfragesprache rasql verwendet werden. (vgl. [OSG14]) Der beschriebene Aufbau ist unter Abbildung 2.3²¹ dargestellt.

¹⁹Hinweis über fehlende Trigger in Postgres-XL: <http://files.postgres-xl.org/documentation/intro-what-is.html> Ergänzung 4

²⁰Rasdaman: [Ras15c]

²¹Link zur Abbildung: <http://www.rasdaman.org/raw-attachment/wiki/Technology/wcps-stack.png>

2 Grundlagen

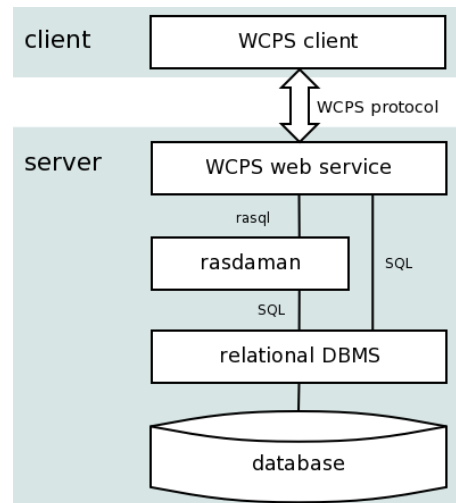


Abbildung 2.3: Aufbau Rasdaman

Es besteht die Möglichkeit, Rasdaman zu einer bestehenden PostgreSQL zu installieren und direkten Datenaustausch zwischen den beiden Systemen zu ermöglichen. Weiterhin kann Rasdaman in Verbindung mit GDAL verwendet werden. Momentan existiert eine Community und eine Enterprise Variante. Dabei verfügt die Enterprise Variante über mehr Features wie beispielsweise Datenkomprimierung, Serververwaltung per Webbrowser, Laufzeitoptimierungen und verschiedene Datenbankschnittstellen. Von der verwendeten Datenbank wird BLOB als Datenbankinterner Datentyp verwendet. (vgl. [Ras14d])

3 Methodisches Vorgehen

Dieses Kapitel führt die angewandte Methodik dieser Arbeit an. Um die umfangreiche Fragestellung methodisch angehen zu können, wird sie in Teilfragen zerlegt. Das Thema "Untersuchung quelloffener verteilter geografischer Informationssysteme zur Verarbeitung agrartechnischer Kennzahlen" besteht aus vier Unteraufgaben:

Untersuchung bestehender Frameworks anhand von Qualitätsmerkmalen

Die erste Hälfte dieser Arbeit ist eine Softwareauswahl im Bereich der Datenverarbeitung. Es handelt sich dabei nicht um Anwendersoftware, wodurch keine empirischen Studien der Benutzung der Frameworks durch Anwender und keine Kopplung mit der Unternehmensstruktur notwendig ist. In diesem speziellen technischen Kontext sind wissenschaftliche Vorgehensweisen zur Softwareauswahl notwendig. Darin soll mit Mitteln des Softwarequalitätsmanagement und allgemein Prinzipien der Softwareentwicklung eine nachvollziehbare, auch auf ähnliche Projekte übertragbare und wissenschaftlich korrekte Vorgehensweise erreicht werden. Dafür geeignet ist eine Nutzwertanalyse.

Aus gegebenen Anforderungen¹ sind Qualitätsmerkmale zu erstellen. Davon ist die Mindestmenge zu definieren, welche zur Eignung eines Frameworks notwendig ist bzw. die notwendige Qualität. Darauf aufbauend sind Qualitätsmetriken zu erstellen, welche die einzelnen Qualitäten messbar machen. Eine Menge von scheinbar geeigneten Frameworks ist anhand der definierten Metriken zu untersuchen. Dabei sind jedoch nur die für den Anwendungsfall relevanten Qualitäten zu untersuchen. Aus bekannten GIS Frameworks werden vielversprechende ausgewählt und somit die Menge der zu untersuchenden Frameworks erstellt. Diese Vorauswahl erfolgt anhand der Tabelle „Table of free systems especially for spatial data processing“ in [Wik15b]. Nicht einheitlich messbare Kriterien zur Softwareauswahl wie Marktposition, Support und Community fließen in diese Bewertung nicht ein, werden jedoch im Sinne der Agri Con ergänzend festgehalten.

¹siehe 4.1

3 Methodisches Vorgehen

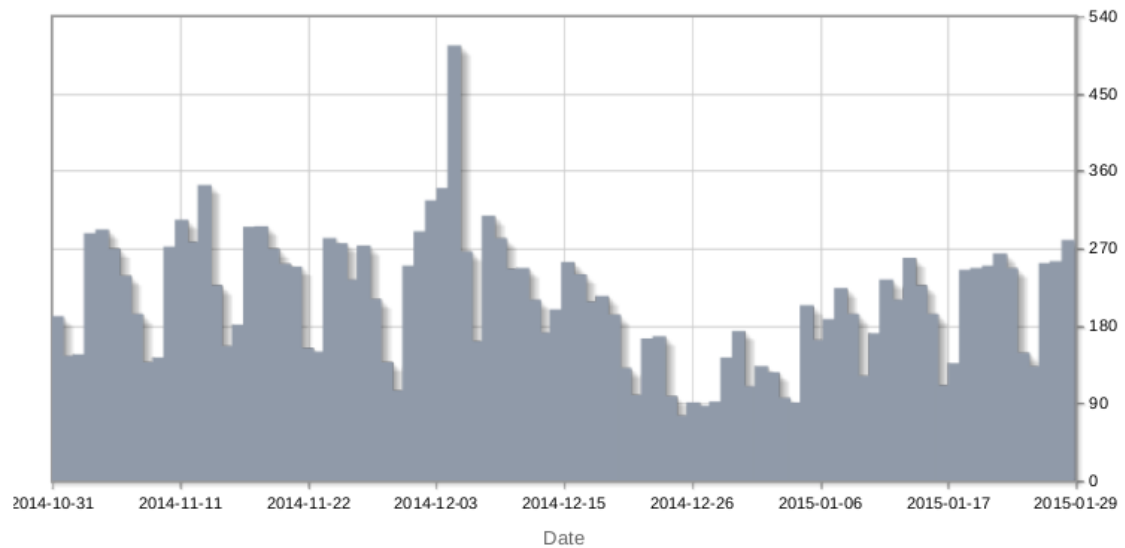


Abbildung 3.1: Nutzungsstatistik von 90 Tagen der Wikipedia Seite „Spatial database“ nach [Wik15a]

Wikipedia dient zur Vorauswahl der Frameworks als Quelle, da diese Website von Arbeitnehmern genutzt wird und somit diese Community die Tabelle aktuell und ausreichend vollständig hält. Der Unternehmensbezug ist in der Aufrufstatistik begründet, welche in Abbildung 3.1 dargestellt ist. So wurde die Website am Mittwoch dem 21.1.2015 250 mal, dagegen am Samstag dem 17.1.2015 111 mal aufgerufen. Allgemein ist an Tagen an Wochenenden etwa die Hälfte der Aufrufe pro Tag gegenüber Montag bis Freitag zu beobachten. Weiterhin war zu den Weihnachtsfeiertagen eine wesentlich geringere Anzahl an Aufrufen zu verzeichnen: 1650 Aufrufe zwischen dem 22.12.2014 und 4.1.2015 im Gegensatz zu 3005 Aufrufe zwischen dem 8.12.2014 und 21.12.2014. Diese Aufrufstatistik belegt das die Nutzung der Website etwa zur Hälfte durch Arbeitnehmer erfolgt und somit für Unternehmen interessant ist.

Auswahl eines Frameworks

Aus der untersuchten Menge ist eines anhand der gemessenen Nutzwertes auszuwählen. Der Ist-Stand unter 4.2 der Agri Con ist über Jahre hinweg durch wachsende Anforderungen im technischen und organisatorischen Bereich entstanden. Aus diesem Grund ist die Auswahl eines Frameworks für den gesuchten Anwendungsfall aufwendig. Die Wahrscheinlichkeit der Eignung mehrerer Frameworks scheint daher nach der Einschätzung des Autors gering. Diese induktive Hypothese gilt es in dieser Teilaufgabe zu belegen. Eine detaillierte Bewertung der Frameworks anhand aller Qualitäten würde danach dazu führen, dass kein Framework geeignet erscheint. Deshalb ist die Nutzwertanalyse zunächst anhand deren Spezifikationen durchzuführen.

3 Methodisches Vorgehen

Für derartige Entscheidungen kann stattdessen das Vorgehen *Analytic Hierarchy Process (AHP)* verwendet werden. Dabei werden verschiedene Alternativen für eine Anforderung entsprechend Kriterien hierarchisch bewertet. Schlussendlich wird der Wert einer Alternative über Matrizenberechnungen der Kriteriumswerte unter Berücksichtigung der Wertigkeit der einzelnen Kriterien und Unterkriterien erstellt (siehe [Gau]). Dieser Entscheidungsvorgang ist für Szenarien geeignet, in welchen eine Vielzahl von etwa gleich geeigneten Alternativen vorliegt und die Bewertung eines Kriteriums Teilbewertungen anderer Alternativen berücksichtigen muss, um die beste Alternative herausarbeiten zu können. Dies ist im Szenario dieser Arbeit nicht gegeben:

- Die Menge an für den Vergleich verwendbaren Kriterien ist gering, da nur die Spezifikation als Quelle dienen soll.
- Die Kriterien sind zwar unterschiedlich zu werden, aber nicht direkt hierarchisch abbildbar.
- Defizite der Frameworks sind im Anschluss an die Auswahl zu implementieren, wobei der Aufwand abhängig von Kriterium und Framework ist. Eine Berücksichtigung dessen ist in *AHP* nicht gegeben.
- Die Menge an Alternativen ist gering.
- Die Alternativen unterscheiden sich in Hauptelementen wesentlich voneinander, wodurch die Bewertung eines Kriteriums bereits das Framework ausscheiden lassen kann.

Das Vorgehen *Technique for Order Preference by Similarity to Ideal Solution (TOPSIS)* eignet sich auf Grund der Ähnlichkeit zur Nutzwertanalyse ebenfalls. Die Ähnlichkeit besteht darin, dass beide Vorgehen Alternativen entsprechend ihrer Erfüllung oder dem Abstand zur schlechtesten und besten Möglichkeit bewerten. Der Autor entschied sich gegen *TOPSIS*, da die Nutzwertanalyse übersichtlicher und nachvollziehbarer erscheint sowie die Darstellung der abstrakten Entfernung zur bestmöglichen theoretischen Alternative keinen Erkenntnisgewinn darstellt. Dieser Abstand bildet nicht den Aufwand zum Erreichen des bestmöglichen Zustandes ab. Höchstwahrscheinlich müssen Funktionen manuell hinzugefügt werden. So kann der Abstand von Alternative A geringer sein als jener einer Alternative B, jedoch ist der Aufwand zur Umsetzung von A doppelt so hoch wie für B. Dieses Fehlen der Integration des Aufwandes macht den Abstand und somit *TOPSIS* uninteressant für diese Untersuchung.

Entwurf eines Prototypen

Das ausgewählte Framework ist detailliert zu untersuchen. Aus dieser Untersuchung soll ein Entwurf zum Einsatz bei der Agri Con entstehen. Dabei ist besonders dessen Architektur, die Konfiguration und fehlende Funktionalitäten zu beleuchten, welche mit nachträglicher Implementierung in das Framework einzubinden sind. Auf Grund der wesentlich unterschiedlichen Frameworks, kann vor der Auswahl keine konkrete Architekturempfehlung verwendet werden. Dieser Entwurf ist nach den Anforderungen und den Eigenheiten des Frameworks zu erstellen. Eine wesentliche Grundlage sind dabei Referenzimplementierungen und Guidelines des entsprechenden Rahmenwerkes.

Prototypische Implementierung

Der Entwurf wird schlussendlich umgesetzt und anhand der Funktions- und Leistungstests bewertet. Auch diese Bewertung erfolgt im Rahmen einer Nutzwertanalyse, jedoch Anhand des bestehenden Entwurfs und der gemessenen Daten aus den Tests. Ziel ist dabei die Eignung des Prototypen hinsichtlich des geforderten Einsatzzweckes darzustellen. Dafür werden die definierten Qualitäten herangezogen. Außerdem soll eine Einschätzung aus Entwicklersicht gegeben werden, welche Preis, Marktposition, Support und Community einschätzt. Diese Ergänzung ist notwendig, um weiterhin den Aufwand zur Einbindung des Frameworks in die Unternehmensstruktur und die Zukunftsfähigkeit einschätzen zu können.

Für die grobe Nutzwertanalyse zur Auswahl eines Frameworks ist im nachfolgenden Kapitel die Darstellung des Ist-Standes sowie die Anforderungen an das Framework zu finden. Die Definition der Testfälle zur Datenerhebung für die detaillierte Nutzwertanalyse des Prototypen schließt sich daran an.

Es folgt die Definition der Tests abhängig des Anwendungsfalles. Die Ein- und Ausgaben sind in Kapitel 4.1 dargestellt. Die Tests müssen einerseits vergleichend sein, ergo bei unterschiedlichen Systemen die gleichen Merkmale testen und wiederholbar sein, andererseits die relevanten Merkmale testen.

Es handelt es sich somit um systematische Tests:

*Ein systematischer Test ist ein Test, bei dem
die Randbedingungen definiert oder präzise erfasst sind,
die Eingaben systematisch ausgewählt wurden,
die Ergebnisse dokumentiert und nach Kriterien beurteilt werden, die vor dem Test
festgelegt wurden.*[Lud07, S.446]

Um die Systeme auf Softwarequalität, beschrieben in Kapitel 4.1.1, zu testen, sind Funktionstests notwendig. Im folgenden bezieht sich „Systeme“ auf die untersuchten Frameworks. Ludwig verweist auf Funktionstests wie folgt:

Werden die Testfälle auf Basis der in der Spezifikation geforderten Eigenschaften des Prüflings ausgewählt (z.B. Funktionalität, Antwortzeit), dann spricht man von einem Block-Box-Test oder auch von einem Funktionstest (19.5). [Lud07, S.455]

Dazu wird auf Seite 471 der Umfang des Funktionstestes wie folgt umrissen:

*Ein umfassender Black-Box-Test sollte
alle Funktionen des Programms aktivieren (Funktionsüberdeckung),
alle möglichen Eingaben bearbeiten (Eingabeüberdeckung),
alle möglichen Ausgabeformen erzeugen (Ausgabeüberdeckung),
die Leistungsgrenzen ausloten,
die spezifizierten Mengengrenzen ausschöpfen,
alle definierten Fehlersituationen herbeiführen.*

Im Rahmen dieser Arbeit soll dieser Umfang wie folgt begrenzt werden:

Funktionsüberdeckung Zur Einschätzung der Eignung eines Systems für den Anwendungsfall sind ausgewählte Funktionen zu testen, Im allgemeinen besitzen die ausgewählten Systeme Funktionen die in diesem Rahmen nicht genutzt werden sollen, jedoch auch Funktionen die von Hand ergänzt werden müssen. Die notwendige Menge an Funktionen soll einzig getestet und somit abgedeckt werden.

Eingabeüberdeckung Auch hierbei stehen verschiedene von den Systemen bereitgestellte Schnittstellen und Austauschformate zur Verfügung, aber es sollen nur die relevanten untersucht werden.

Ausgabeüberdeckung siehe Eingabeüberdeckung

Leistungsgrenzen Dafür werden eigene Tests definiert und verwendet.²

Mengengrenzen Die zu untersuchenden Systeme eignen sich zum Speichern großer Datenmengen die mehrere Terrabyte bis Petabyte umfassen. Da die vorhandenen Datenmengen diese Größe unterschreitet, müssen die Mengengrenzen nicht wesentlich getestet werden.

Fehlersituationen Es besteht nicht das Ziel die Fehleranfälligkeit als einzelnes Merkmal zu untersuchen, weshalb dafür keine Testfälle erstellt oder durchgeführt werden. Einzig die Korrektheit der Berechnungen wird überprüft, indem die Ergebnisse des aktuellen ist-Standes zum Vergleich herangezogen werden.

²siehe 3

3 Methodisches Vorgehen

Ein Kriterium der Untersuchung in dieser Arbeit ist die Leistung. Nach Kesselman in [Han95, S.20] ist Leistung die gewichtete Summation von Leistungsindizes, wobei ein Leistungsindex die Quantifizierung einer Eigenschaft eines Systems ist. Wesentliche Leistungsindizes sind Laufzeiten von einfachen oder komplexen Operationen. Es wird hier nur die spezielle Leistung gemessen, da ausgewählte Eigenschaften betrachtet werden und somit nicht die Leistung für einen allgemeinen Anwendungsfall. Es existieren vordefinierte Leistungstests, dabei sind jene des *Transaction Processing Performance Council (TPC)* sowie die Benchmarks 001, 007, HyperModel und Justitia zu nennen. Diese sind jedoch nicht für den zu untersuchenden Anwendungsfall geeignet, da sie die allgemeine Leistungsfähigkeit und Effektivität messen und somit nicht die gesuchten Werte der GIS ermitteln.

Der in diesem Zusammenhang in der Literatur verwendete Begriff Benchmark ist hier jedoch ungeeignet, da die Software und nicht die Hardware untersucht werden soll:

Benchmarking ist eine Methode der Analyse des Leistungsverhaltens von Rechensystemen anhand von Referenzprogrammen oder Sätzen von Referenzprogrammen (Benchmarks). Dabei wird das Leistungsverhalten verschiedener Rechensysteme in Relation gesetzt, um so Vergleichskriterien für Rechensysteme zu erhalten. [Han95, S.24]

Eine hier betrachtete Leistungs- und Laufzeitmessung ist dabei wie folgt definiert:

Unter Leistungsmessung versteht man die Beobachtung des Ablaufverhaltens eines Programms bei der Ausführung auf einem realen System. Das Ziel das dabei verfolgt wird, ist die Gewinnung von Erkenntnissen, die zur Optimierung eines Programms genutzt werden können. [Han95, S.28]

Die aus der Leistungsmessung gewonnenen Erkenntnisse dienen hierbei als Qualitätsmerkmale und werden nach definierten Metriken gewichtet.

Gleichzeitig zur Durchführung der Leistungstests ist die Auslastung des Systems zu überwachen und zu protokollieren. Im Allgemeinen wird die Auslastung der CPU, der Anteil des reservierten Arbeitsspeichers und die Anzahl an Operationen auf der Festplatte überwacht. Die Durchführung unterscheidet sich je nach Betriebssystem und Art des logischen Systems. Jedoch ist darauf zu achten die eigentlichen Messungen nicht nennenswert zu beeinflussen. Ziel ist ein Kompromiss aus hochfrequenter Auflösung der Daten der Systemauslastung und dem dafür notwendigen Aufwand in Form von Rechenzeit. Außerdem ist die Auslastung des Systems vor dem Start der Leistungstests festzuhalten, um den Grundwert zu erhalten.

4 Ausgangsszenario

Nach Darlegung der Methodik im voran gegangenen Kapitel, wird in diesem präsentiert worauf sie angewendet wird. So werden die Anforderungen definiert und messbar gemacht wie auch Funktions- und Leistungstests skizziert. Zur Einordnung der Gewichtung der zu untersuchenden Qualitäten und Heranführung an die Absichten der Agri Con, wird der Ist Stand und die gewünschte Integration eines neuen Frameworks ausgeführt.

4.1 Anforderungen

Aktuelle Möglichkeiten der Datenerfassung über Sensoren und moderne Probenahmegeräte führen zu mehr und mehr Datensätzen, die für einen Landwirtschaftsbetrieb ausgewertet werden müssen. Darüber hinaus besteht die Notwendigkeit, Daten Jahresübergreifend und betriebsübergreifend auszuwerten, um pflanzenbauliche Zusammenhänge über statistische Methoden untersuchen zu können. In den letzten 3 Jahren wurde beispielsweise nur zum Thema N-Versorgung¹ für einen Betrieb etwa 800 Datensätze mit 1,9 Mio Einträgen erfasst. Alle diese Daten haben einen räumlichen Bezug, sie müssen weiterverarbeitet, kartographisch aufbereitet und dargestellt werden.

Daraus ergeben sich verschiedenen Anforderungen an die Technologie, die für die Verarbeitung, Analyse und Darstellung verwendet wird:

- PostgreSQL mit PostGIS zum Datenimport und -export nutzbar
- Gruppieren und Filtern mit geringer Laufzeit
- parallele Berechnung² über große Datenmengen mit geringer Laufzeit
- Räumliche Berechnungen wie Verschneiden und Overlays
- nutzbare Schnittstelle zur Darstellung mit dem *UMN MapServer*

¹Stickstoffdüngung und -aufnahme

²hier Geostatistik

Konkret handelt es sich bei den Eingangsdaten um folgende:

- Pflanzenbauliche Daten³: Punktdaten
- Basisdaten wie Feldgrenzen sowie Interpolationen⁴: Vektordaten
- Externe Satelliteninformationen und Multispektralanalysen: Rasterdaten

4.1.1 Softwarequalität

Qualitätsmerkmale sind nach DIN 9126⁵ in [Bal98, S.258 f.] Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit und Übertragbarkeit. Diese Merkmale werden durch Qualitätskriterien für jeden Anwendungsfall konkretisiert. Nachfolgend werden die Qualitätsmerkmale für diesen Anwendungsfall dargestellt und darauf die zu untersuchenden aufgelistet. Da die zu analysierenden Systeme eine Datenbank beinhalten, welche mit räumlichen Datentypen arbeitet, wurde die im Anhang C von [Hoh96] enthaltene Checkliste zur Auswahl eines *Objektorientiertes Datenbankmanagementsystem (ODBMS)* berücksichtigt.

Funktionalität

Das System stellt alle geforderten Funktionen mit den definierten Eigenschaften zur Verfügung.

- **Richtigkeit:** Ergebnisse sind korrekt oder ausreichend genau. Die originalen räumlichen Daten werden von Sensoren erfasst, in wessen Toleranzbereich die Ergebnisse der Verarbeitung durch das Framework liegen müssen.
- **Interoperabilität:** Es sind Schnittstellen zur Ein- und Ausgabe vorhanden. Dabei soll es sich um PostgreSQL Import sowie PostgreSQL und *UMN MapServer* Export handeln.
- **Funktionsumfang:** Mindestens die benannte und essentielle Menge an Funktionalitäten wird bereitgestellt. Dazu zählt: parallele Verarbeitung, Gruppierungs-, Filter-, Verschneidungs- sowie Overlayfunktionen, Geostatistik und Umrechnung zwischen Koordinatensystemen und -formaten. Außerdem sind vorhandene Datentypen und Schemaversionierung von Interesse.

³Sensoren, Bodenuntersuchung, *Bonitur*, Logger

⁴Interpolationen als Ergebnis von Nährstoffverteilung und Bodenscanner

⁵DIN 9126 wurde durch ISO/IEC 25000 ersetzt, jedoch sind beide nur proprietär verfügbar

- **Ordnungsmäßigkeit:** Die Implementation des Systems und dessen Funktionen erfüllt Normen, Vereinbarungen, gesetzliche Bestimmungen und andere Vorschriften. Hierzu ist zu nennen, dass besonders Berechnungsfunktionen nach mathematischen Gesetzen implementiert sein müssen. Konkret sind Berechnungen der räumlichen Verarbeitung nach anerkannten definierten Algorithmen durchzuführen. Weiterhin sind Definitionen der Koordinatenreferenzsysteme⁶, die mathematisch korrekte deterministische sowie stochastische Interpolation und Interpolation nach Krige einzuhalten. Auch Allgemeine Anforderungen an *DBMS* wie Integritätssicherung, Datensicherheit, Mehrbenutzerbetrieb, Datenunabhängigkeit und Zugangssicherung müssen erfüllt werden.

Zuverlässigkeit

Nach [Bal98, S.259] wird Zuverlässigkeit als die Fähigkeit einer Software ihr Leistungsniveau unter festgelegten Bedingungen über einen festgelegten Zeitraum bewahren definiert.

- **Fehlertoleranz:** Das System sollte auftretende Fehler des Tagesgeschäftes abfangen und weiterarbeiten. Besonders Fehler in den Quelldaten können zu Fehlern während der Ausführung von Berechnungen führen, was per sé abgefangen werden muss.
- **Wiederherstellbarkeit:** Auch die Möglichkeit bei einem schwerwiegendem den Zustand der abgebrochenen Operationen wiederherzustellen ist ein zu betrachtendes Qualitätskriterium.
- **Mean Time to Failure (MTTF):** Diese statische Kenngröße der erfahrungsgemäßen mittleren Lebensdauer ist für kritische Systeme relevant.

Benutzbarkeit

Qualität des Zugangs für Benutzer sowie Eignung für eine oder mehrere Benutzergruppen.

- **Verständlichkeit**
- **Bedienbarkeit**
- **Dokumentation:** Eine ausführliche, aktuelle und korrekte Dokumentation ist Voraussetzung zur produktiven Verwendung.

⁶siehe *EPSG-Codes*

- **Eignung:** Die angestrebte Benutzergruppe muss mit der aktuellen Benutzergruppe übereinstimmen. Die aktuelle Benutzergruppe ist Programmierer bzw. Administrator.

Effizienz

Effizienz ist das Verhältnis zwischen Auslastung der Hardware und erfolgreich bearbeiteten Aufgaben. Nach [Han95, S.21] ist Leistung paralleler Programme das Verhältnis des Speedups zur Anzahl der verwendeten Prozessoren. Wobei Speedup als Verhältnis der Ausführungszeiten zwischen der auf N Prozessoren ausgeführten parallelen Version eines Programms und der sequentiellen Version des Programmes definiert ist. Diese Definitionen treffen für die zu untersuchenden Systeme zu, da es sich um parallelisierende GIS handelt.

- **Zeitverhalten:** Oder auch Laufzeitverhalten genannt, dient allgemein zur Darstellung des Durchsatzes. Die Skalierung des Systems zählt hier dazu. Dies wird speziell durch zusätzliche Leistungstests beurteilt.
- **Verbrauchsverhalten:** Das Verhältnis aus erbrachter Leistung und dem dafür notwendig gewesenem Aufwand in Form von Hardwarenutzung.
- **Skalierbarkeit:** Anzahl der zu verwendenden Computer um nach dem Speedup eine Effizienzsteigerung im Gegensatz zum Einsatz bei einem Computer zu erreichen.

Änderbarkeit

Aufwand zur Verbesserung oder Anpassung der Umgebung und der Spezifikationen, auch Wartungsaufwand genannt.

- **Analysierbarkeit:** „Aufwand, um Mängel oder Ursachen von Versagen zu diagnostizieren oder um änderungsbedürftige Teile zu bestimmen.“ [Bal98, S. 260]
- **Modifizierbarkeit:** Notwendiger Aufwand für Änderungen zum Ziele der Verbesserung und Fehlerbehebung.
- **Stabilität:** Wahrscheinlichkeit von ungewollten Auswirkungen durch Änderungen.
- **Prüfbarkeit:** Oder Testbarkeit als Merkmal, welches die Möglichkeiten und den Aufwand zum testen der originalen und geänderten Systeme darstellt.

Übertragbarkeit

Die Fähigkeit das System auf andere Hard- und Software sowie andere Vorgehensweisen zu migrieren.

- **Anpassbarkeit:** Möglichkeiten des unveränderten Systems Änderungen vorzunehmen.
- **Installierbarkeit:** Systemvoraussetzung und Aufwand zur Installation des Systems.

Nichttechnische Kriterien

Erweiterte Qualitätskriterien, welche nicht nach der DIN 9126 zugeordnet werden können.

- **Herstellerfirma und Produkt:** Dazu zählt die Marktposition, der Preis, die Produktplanung und der Service.

Die zu untersuchenden Qualitätskriterien für die Softwareauswahl sind Funktionsumfang, Fehlertoleranz, Dokumentation, Zeitverhalten, Analysier- und Modifizierbarkeit.

4.1.2 Qualitätsmetriken

Zu den wichtigen Qualitätsmerkmalen aus Kapitel 4.1.1 sind folgend Kriterien definiert, sowie je eine Bewertungsvorschrift und die geforderte Mindestbewertung für den Anwendungsfall angegeben.

Richtigkeit:

Berechnungen sind zu 100% korrekt. Ausnahme ist dabei die Berechnung von Koordinaten. Dabei haben die Ergebnisse bis acht Stellen nach dem Komma korrekt zu sein. Die statische Abbildung ist dabei [*korrekt, nicht korrekt*] nach [1,0] und die geforderte Mindestbewertung 1.

Interoperabilität:

Ist der Import und Export von räumlichen Daten aus PostgreSQL sowie eine Anbindungsmöglichkeit an den *UMN MapServer*. Statische Abbildung:

[*Datenschnittstelle und UMN Schnittstelle vorhanden, Datenschnittstelle vorhanden, UMN Schnittstelle vorhanden, keine Schnittstelle vorhanden*] nach [12,7,5,0]

Der Bereich bis 12 soll die Wichtigkeit des Vorhandenseins der Schnittstellen verdeutlichen. Die geforderte Mindestbewertung ist 7.

Funktionsumfang:

Tabelle 4.1 gibt die Wertung bei Existenz der einzelnen Funktionen wieder. Existiert die Funktion nicht, ist die Wertung Null. Ein Zwischenwert bei eingeschränkter Funktionalität ist möglich. Maximale Wertung: 61 Es müssen mindestens geografische Datentypen, Filterfunktionen und parallele Verarbeitung vorhanden sein.

Funktion	Wertung
parallele Verarbeitung	2
geografische Datentypen	14
Umrechnung zwischen Koordinatensystemen	10
Gruppierungsfunktionen	10
Verschneidungsfunktionen	4
Overlayfunktionen	4
Geostatistik	6
Filterfunktionen	10
Schemaversionierung	1

Tabelle 4.1: Wertungstabelle Funktionsumfang

Fehlertoleranz:

Es gilt zu messen, ob Fehler bei einer Berechnung andere verschränkt gleichzeitig laufende Berechnungen beeinträchtigen. Aus diesem Grund wird Unabhängigkeit auf eins und Abhängigkeit auf null abgebildet. Die geforderte Mindestbewertung ist 1.

Dokumentation:

Vorhandene Dokumentation ist nach einzelnen Themen zu bewerten. Dabei kann ein maximaler Wert von 13 erreicht werden. Die geforderte Mindestbewertung ist 6.

Dokumentation zu	Wertung je Eintrag
Installation, Zeitverhalten	1
Funktionsumfang	2
Interoperabilität, Best practise, Anpassbarkeit	3

Tabelle 4.2: Wertungstabelle Dokumentation

Zeitverhalten:

Konkret wird eine Beschleunigung aufwendiger Vorgänge angestrebt. Die statische Abbildung auf Bezug auf die Laufzeiten des Ist-Standes:

[Zeit wird berschritten, Zeit ist gleich, Zeit wird unterschritten] nach [0,1,3]

Modifizierbarkeit:

Mögliche Anpassungen des Frameworks hinsichtlich der folgenden Punkte erhöhen den

Wert um eins:

Verwendung eigener Datentypen, Erstellung eigener Schnittstellen, Erstellung eigener Funktionen, Verwendung der Programmiersprachen Scala oder R, Anlegen eigener Berechnungsvorgängen zur späteren Abarbeitung

Die geforderte Mindestbewertung ist abhängig vom Funktionsumfang und der Interoperabilität. Jedoch sollten fehlende Funktionen und Schnittstellen erstellt werden können.

4.1.3 Testfälle

Zur definierten und wiederholbaren Erfassung der Erfüllung von speziellen Kriterien wie Funktionalität und Zeitverhalten, folgt die Definition der Funktions- und Lasttests.

Funktionstests

Nach der Definition in Kapitel 3 werden hiermit spezielle Funktionalitäten auf Vorhandensein und die Menge der Schnittstellen sowie der Austauschformate getestet. Diese Funktionstest sind nicht automatisiert für unterschiedliche Frameworks durchführbar. Deshalb werden sie manuell anhand der Spezifikation und des Quellcodes durchgeführt und die Ergebnisse tabellarisch festgehalten.

Folgende Schnittstellen sind zu testen:

- PostgreSQL
- PostGIS
- *UMN MapServer*

Als Austauschformat ist mindestens eines der folgenden Datentypen zum Datenaustausch notwendig:

- Simple Feature Access
- Objekte der JTS
- PostGIS geometry

Speziell in GIS gilt es die Koordinatenreferenzsysteme zu analysieren. Die EPSG Codes 3578 und 4326 werden im Anwendungsfall verwendet.

Für folgende Aufgaben sind die Funktionen zu analysieren:

- Umwandlung zwischen Koordinatensystemen

- Verschneidung von räumlichen Daten
- Interpolation
- Kriging
- topologische Filterung
- räumliche Filterung

Dabei ist stets zu berücksichtigen mit welchen Datentypen die Funktionen verwendet werden können.

Lasttests

Die Basis für die Erstellung von Lasttests ist Kapitel 4.2. Eine exakte Definition der Lasttests ist in diesem Kapitel nicht möglich, da das zu verwendende Framework noch nicht ausgewählt wurde und somit die darin zu verwendenden Werkzeuge und Daten nicht bekannt sind. Es werden jedoch die zu testenden Szenarien benannt.

Ein zu testendes Szenario ist die Aggregation von Daten. Dabei ist eine Menge von mehreren hundert Megabyte an Punktdaten mit den gegebenen Werkzeugen abzurufen und die Laufzeit zu messen. Die Werkzeuge sollten dabei eine Abfragesprache wie SQL und bei Vorhandensein der Schnittstelle der *UMN MapServer* sein. Über eine Abfragesprache sind neben der Laufzeit weitere Informationen abrufbar. Mit SQL kann das Vorgehen des Query Planers ausgegeben werden, was für die Auswertung herangezogen werden kann. Die Verwendung des *UMN MapServer* zielt auf eine anwendungsnahe Darstellung der Leistungsfähigkeit ab. Die hohe Leistungsfähigkeit des *UMN MapServer* für ein solches Szenario wurde bereits in [Jun12] dargestellt, sodass von einer hohen Auslastung des *DBMS* ausgegangen werden kann.

Analog zu diesem ersten Szenario ist ein weiterer zur Interpolation bzw. Kriging durchzuführen. Darin sollen Punktdaten zu Vektordaten mit Interpolation im Rahmen eines Krigings verarbeitet werden. Damit soll die Verarbeitungsleistung bewertet werden.

Da es sich bei den zu untersuchenden Frameworks um verteilte Systeme handelt, ist in allen Tests die Verteilung der Daten und der Verarbeitungsschritte zu berücksichtigen.

4.2 Aktuelles System

Die Durchführung einer Softwareauswahl zum teilweisen Ersatz eines bestehenden Systems setzt die Analyse des Systems voraus. In diesem Unterkapitel wird das aktuelle

System sowie der der angestrebte Zustand nach Einbau des Prototypen dargestellt. Firmeninterne Schnittstellen mit dem ist-Stand werden nicht konkretisiert, da sie den Anwendungsfall nicht schneiden.

In Abbildung 4.1 ist die Übersicht des Aufbaus ersichtlich. Das Herzstück bildet die objektrelationale Datenbank PostgreSQL mit der geografischen Erweiterung PostGIS. Diese dient zur Datenhaltung und wesentlich auch zur Datenverarbeitung. In den Programmiersprachen Delphi und *R* werden zusätzlich automatische Verarbeitungsvorgänge durchgeführt. Daten werden von extern und intern eingespeist. Dabei handelt es sich um Punkte, Vektoren und Raster mit dazugehörigen Metadaten. Im Rahmen der Datenverarbeitung werden Punktdaten interpoliert und mit Hilfe von Geostatistik Auswertungen aus originalen und verarbeiteten Daten erstellt. Die Darstellung der Daten erfolgt über *UMN Map-Server* und *Manifold Internet Map Server*. Als zentrales Element enthält die Datenbank neben den agrartechnischen Kennzahlen alle weiteren Daten des Unternehmens. In dieser Betrachtung werden einzig die agrartechnischen Kennzahlen berücksichtigt.

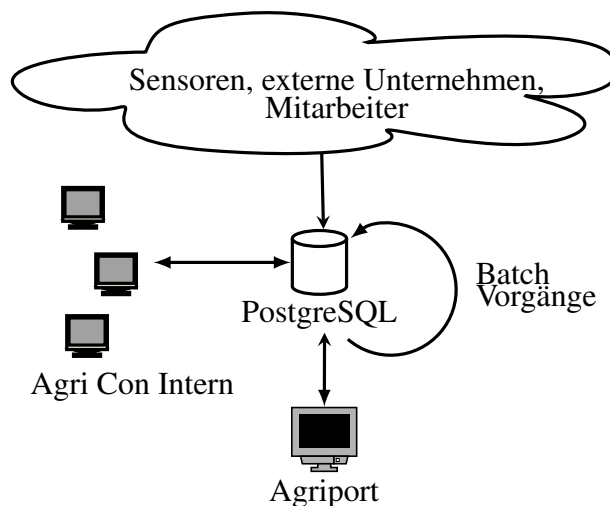


Abbildung 4.1: Aktuelle Infrastruktur bei Agri Con

Es existiert eine Vielzahl von Vorgängen, welche zur Erhöhung der Useability in Hinblick auf ihre Laufzeit verbessert werden können. Für die Bewertung der Frameworks und schließlich zum Entwurf des Prototypen, ist es notwendig eine Auswahl der zu untersuchenden Vorgängen zu treffen. Die Auswahl erfolgt durch Mitarbeiter der Agri Con und hat Vorgänge mit der längsten Laufzeit als Ergebnis. Dazu zählt das Laden von Daten zum Zwecke der Verarbeitung und Anzeige. Die größten Datenmengen sind bei Punkten aus dem N-Sensor Bereich zu finden. Die Anzeige der Punktdaten für einen Betrieb kann bis zu neun Sekunden dauern. Punktdaten aus dem Docu Bereich werden ebenfalls nach mehreren Sekunden ausgegeben. Die Punktdaten aus dem N-Sensor Bereich sind jedoch

repräsentativ und werden deshalb betrachtet. Weiterhin zählt Interpolation zu diesen Vorgängen. Punkte und Fahrspuren werden mit einem angepassten Kriging interpoliert und als Vektoren oder Raster abgespeichert. Dies wird mit einer R Bibliothek realisiert und bei großen Datenmengen Nachts angestoßen. Das spezielle Kriging im jeweiligen Framework zu implementieren ist aufwendig, weshalb ein unveränderter Kriging Algorithmus für den Vergleich verwendet werden muss, sofern R nicht verwendet werden kann.⁷ Diese zwei charakteristischen Vorgänge sind durch ein Framework zu realisieren.

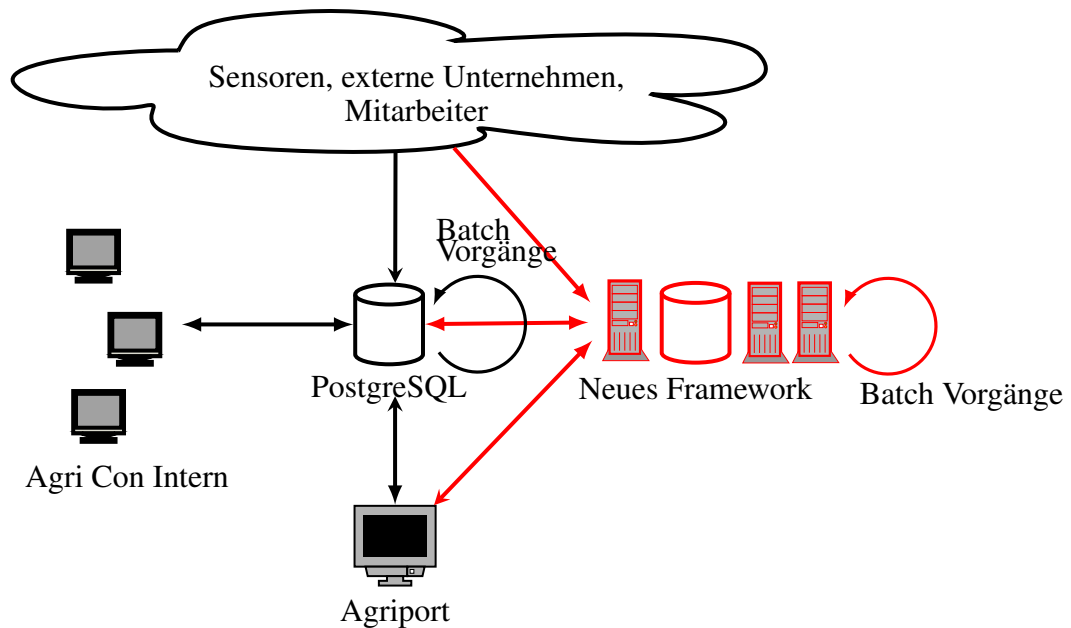


Abbildung 4.2: Übersicht des Aufbaus des Wunsch-Standes bei Agri Con

Das Ziel ist es, ein Framework zusätzlich in den Ist-Stand zu integrieren. Es soll dabei die aufwendigen Vorgänge durchführen, als Permanentspeicher für historische Daten dienen und Daten zur späteren Anzeige aufbereiten und bereitstellen. Die aufwendigen Vorgänge werden in den Lasttests untersucht. In welcher konkreten Form das Framework als Speicher für historische und aufbereitete Daten dient, ist abhängig vom Framework. Auf Grund dessen ist dies nach Auswahl des Frameworks im Entwurf des Prototypen zu konkretisieren.

Für das Verständnis des Kapitels 6 wird nachfolgend das Datenbankschema knapp beschrieben. Es handelt sich um ein umfangreiches Schema, weshalb es aus mehreren Schemata besteht: adwork, apps, bu, bucardo, catalogs, checks, common, demo, docu, farm, files, import, information_schema, ips, krig, log, n, nutrients, ogeo, pg_catalog, pp, public, rax, statistics, tasks, temp, topology, umn, utils und yield. catalogs enthält statische Einträge analog eines globalen Kataloges. common enthält Benutzerspezifische, farm dagegen

⁷Dieser Algorithmus wurde im Auftrag durch dritte erstellt.

Betriebsspezifische Daten. Das Schema files beinhaltet Informationen zu Dateien, welche von Kunden auf den Server hochgeladen wurden sowie Metainformationen zu Dateiarten. In den Schemata n und nutrients sind Daten zu N-Sensor zu finden. Hilfsfunktionen und -tabellen des *UMN MapServer* sind im Schema umn zu finden. Alle Schemata beinhalten Funktionen, Trigger, Tabellen, Typen, Indizes und Constraints.

4.3 Stand der Forschung

Die jährlich stattfindende Internationale Messe FOSSGIS⁸ zeigte das PostgreSQL mit PostGIS in der Regel als freies *DBMS* für *GIS* eingesetzt wird. Zwar existiert eine Vielzahl von Elementen zur Darstellung und Kartenverwaltung, aber es scheint keine geeignete Alternative zur Datenhaltung und Verarbeitung von komplexen geografischen Daten zu geben.

Diese unternehmerische Sicht wird durch wissenschaftliche Arbeiten bestätigt.

So vergleicht Ahlers in seiner Bachelorarbeit⁹ PostGIS mit Oracle Spatial, da auch für ihn PostGIS der größte freie Vertreter eines GIS ist.

Thurm untersucht in [Thu12] gängige NoSQL *DBMS* für Eignung mit geografischen Daten im Vergleich mit PostGIS und Oracle Spatial. Darin kommt der Autor zu dem Schluss, dass es für einfache Datenobjekte in begrenzten Mengen nützlich ist die NoSQL Alternativen in Betracht zu ziehen. Beispielsweise eignet sich die graphenbasierte Datenbank Neo4J für Operationen mit Strecken. Für weniger spezielle und mehr komplexe Anforderungen sind einzig PostGIS und Oracle Spatial zu empfehlen. Die NoSQL Systeme im Bereich der Datenverarbeitung von räumlichen Daten sind in den letzten Jahren entstanden und somit nicht ausgereift und umfangreich, so Thurm auf S.51.

[Baa12] vergleicht dagegen Neo4J mit PostGIS und kommt auch zu dem Schluss, dass graphenbasierte Datenhaltung zwar Vorteile besitzt, aber die Eignung individuell validiert werden muss.

Wissenschaftliche Dokumente in Form von Paper sind zu diesem Thema vorhanden. Diese beschreiben aber nur knapp Entwürfe eines Systems für einen speziellen Anwendungsfall, weshalb sie in diesem Zusammenhang nicht gebraucht werden können. [YZ12] beschreibt einen Entwurf zur Speicherung und Verarbeitung räumlichen Daten mit Hadoop, VegaGiStore genannt. [YZ11] erläutert dagegen Methoden zur Verteilung von Daten in einem verteilten System auf Grund der räumlichen Informationen.

⁸Industriemesse für freie und Open Source Software für Geoinformationssysteme <https://www.fossgis.de/>

⁹[Ah13]

4 Ausgangsszenario

Es existieren Arbeiten zu Randthemen und -betrachtungen sowie Teilproblemen, aber nicht zu den Fragen *Gibt es eine freie Alternative zu PostGIS?* und *Welche verteilten GIS eignen sich für einen komplexen Anwendungsfall?*.

Neben wissenschaftlichen Dokumenten finden sich Werbetexte zu scheinbar geeigneten Systemen. Das Fehlen von verlässlichen konkreten Informationen erschwert die Literaturrecherche. Speziell Leistungsvergleiche sind in diesem Zusammenhang interessant, sind aber auf Grund fehlender Informationen nicht reproduzierbar. Nachfolgend zwei Beispiele einer solchen Werbung:

GeoMesa: Zehn bis 60 fache Antwortgeschwindigkeit bei GeoMesa gegenüber PostGIS, entsprechend der Abbildung 4.3.

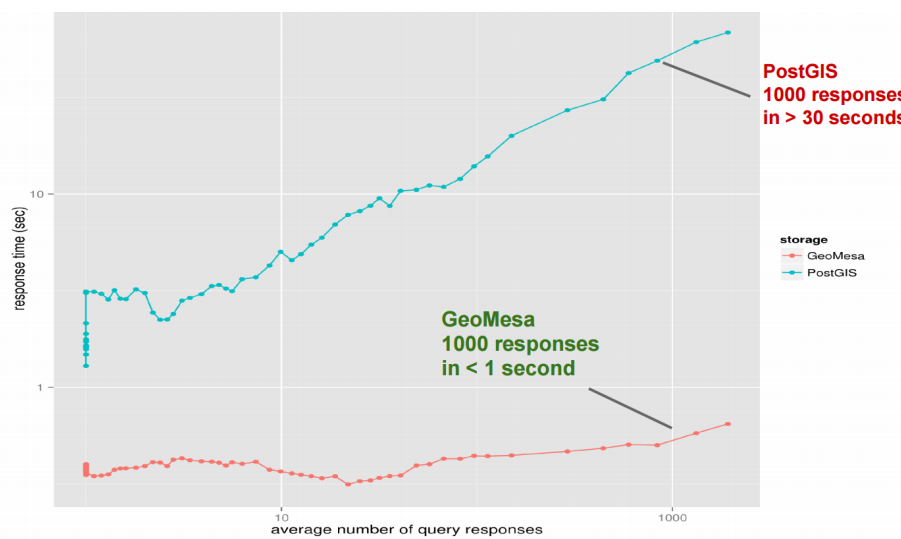


Abbildung 4.3: Antwortzeiten von GeoMesa und PostGIS, Quelle [CCR13, S.24]

Postgres-XL: Bis zu sechs fache Antwortdauer von PostgreSQL gegenüber Postgres-XL in einem standardisierten TPC-H¹⁰ Benchmark bei Verwendung von vier DataNodes. Abbildung 4.4 zeigt das Diagramm mit dem genannten Wert. Die x-Achse ist nicht beschriftet, weswegen die Zuordnung der 15 Ergebnisse zu den 22 vorgegebenen Abfragen des TPC-H Benchmark nicht möglich ist. Somit kann keine Bewertung der Laufzeiten erfolgen.

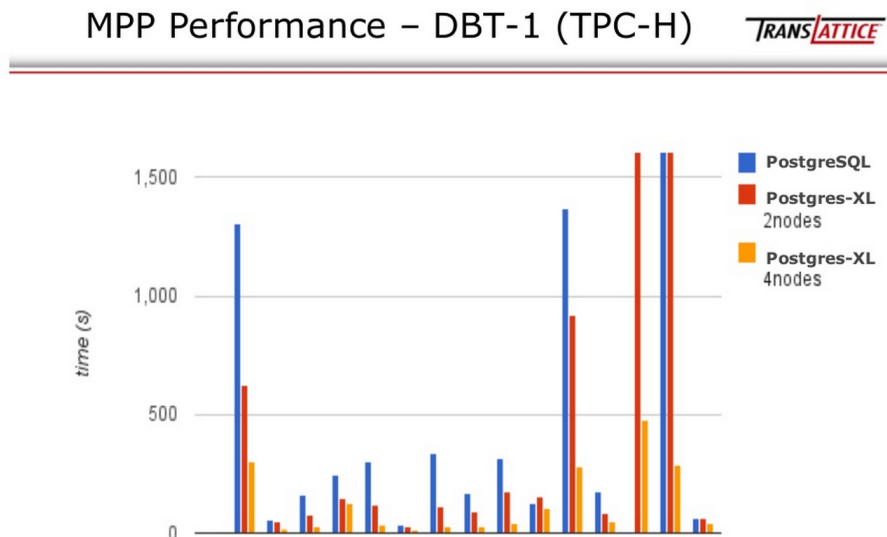


Abbildung 4.4: TPC-H Benchmark von PostgreSQL und Postgres-XL, Quelle [mas13, S.12]

¹⁰<http://www.tpc.org/tpch/>

5 Systemauswahl

Mit den unter 4.1.2 erstellten Metriken sind die Frameworks GeoMesa, Postgres-XL und Rasdaman zu vergleichen. Der Vergleich findet im Rahmen einer Nutzwertanalyse statt. Hierbei werden keine Daten von durchgeführten Tests herangezogen, sondern es wird anhand der Spezifikation der einzelnen Frameworks untersucht ergo eine Inspektion als Prüfmethode verwendet.

5.1 Definition der Nutzwertanalyse

Die drei Frameworks wurden aus der Tabelle der Abbildung 5.1 ausgewählt. Darin sind GIS zur räumlichen Datenverarbeitung mit wesentlichen Eigenschaften wie PostgreSQL Schnittstelle und räumliche Datentypen aufgelistet. Entsprechend den Anforderungen wurden daraus drei Frameworks für die Nutzwertanalyse ausgewählt. Anforderung war dabei, dass Schnittstellen zu PostgreSQL und *UMN MapServer* gegeben sind und es sich um ein Open-Source Frameworks handelt.

Abbildung 5.1 stammt von der Wikipedia Seite https://en.wikipedia.org/wiki/Spatial_database und ist für Unternehmen relevant, wie unter Kapitel 3 beschrieben. Der Autor erschuf die abgebildete Tabelle durch Recherche und stellte sie am 1.2.2015 in den Artikel. In der Annahme, dass unternehmensbezogene Besucher der Seite fehlendes ergänzen oder falsches korrigieren würden, dient diese zur Auswahl geeigneter Frameworks.

Tabelle 5.1 zeigt die für die Nutzwertanalyse notwendige Wertung der einzelnen Metriken. Die Metriken Richtigkeit, Fehlertoleranz und Zeitverhalten werden nicht in die Analyse aufgenommen, da sie über die Spezifikation nicht belegbar sind.

Für jedes Framework wird eine Nutzwertanalyse durchgeführt und die dazugehörige Tabellen dazu präsentiert. Zu jeder Metrik wird der erreichte Wert, die ungewichtete Erfüllung, die gewichtete Erfüllung und ein Kommentar angegeben. Die ungewichtete Erfüllung bezieht sich auf den maximal zu erreichenden Wert der Metrik, die gewichtete

Table of free systems especially for spatial data processing

DBS	License	Distributed	Spatial objects	Spatial functions	PostgreSQL interface	UMN MapServer interface	Documentation	Modifiable	HDFS
AsterixDB	Apache License 2.0	yes	yes (custom)	center, radius, distance, area, intersect and cell	no	no	good in Google Code	own datatypes, functions and indexes	possible
ESRI GIS Tools for Hadoop	Apache License 2.0	yes	yes (own specific API)	yes (union, difference, intersect, clip, cut, buffer, equals, within, contains, crosses, and touches)	no	no	just briefly	forking	yes
GeoMesa (http://www.geomesa.org/)	Apache License 2.0	yes	yes (Simple Features)	yes (JTS)	no (manufacturable with GeoTools)	no	parts of the functions, a few examples	with Simple Feature Access in Java Virtual Machine and Apache Spark are all kinds of tasks solvable	yes
H2GIS (http://www.h2gis.org/)	GPL 3	no	yes (custom, no raster)	Simple Feature Access and custom functions for H2Network	yes	no	yes (homepage)	SQL	no
Ingres	GPL or proprietary	yes (if extension is installed)	yes (custom, no raster)	Geometry Engine, Open Source (http://trac.osgeo.org/geos/)	no	with MapScript	just briefly	with C and OME	no
Neo4J-spatial (https://github.com/neo4j-contrib/spatial)	GNU Affero general public license	no	yes (Simple Features)	yes (contain, cover, covered by, cross, disjoint, intersect, intersect window, overlap, touch, within and within distance)	no	no	just briefly	fork or JTS	no
Postgres-XL (http://www.postgres-xl.org/) with PostGIS	Mozilla public license and GNU general public license	yes	yes (Simple Features and raster)	yes (Simple Feature Access and raster functions)	yes	yes	PostGIS: yes, Postgres-XL: briefly	SQL, in connection with R or Tcl or Python	no
PostgreSQL with PostGIS	GNU General Public License	no	yes (Simple Features and raster)	yes (Simple Feature Access and raster functions)	yes	yes	detailed	SQL, in connection with R	no
Rasdaman	server GPL, client LGPL, enterprise proprietary	yes	just raster	raster manipulation with rasql	yes	with Web Coverage Service or Web Processing Service	detailed wiki	own defined function in enterprise edition	no

Abbildung 5.1: Übersicht relevanter GIS Frameworks nach [Wik15b] vom 13.2.2015

Metrik	Gewichtung in %
Interoperabilität	30
Funktionsumfang	20
Dokumentation	35
Modifizierbarkeit	15

Tabelle 5.1: Wertungsmaßstab der einzelnen Metriken

Erfüllung dagegen auf die Erfüllung der Metrik in Bezug auf Tabelle 5.1. Die Kommentarspalte dient der Darstellung des Erreichens der Mindestanforderungen. Der schlussendliche Nutzwert ergibt sich nach Zangemeister in [Ben13] aus der Summe der Produkte des Teilnutzens des jeweiligen Kriteriums mit der Gewichtung des Kriteriums. Der Teilnutzen ist hier der Prozentuale Anteil der erreichten Punktzahl an der maximalen Punktzahl des Kriteriums. Diese Prozentangabe wird als Wert mit der Gewichtung des Kriteriums multipliziert, woraus sich der Nutzwert für das Kriterium ergibt. Die Summe aller dieser Teilnutzwerte ergibt den Nutzwert des Frameworks für den Anwendungsfall.

Die granulare Bewertung der einzelnen Kriterien der drei Systeme ist im Anhang A.2 zu finden.

5.2 Nutzwertanalyse

5.2.1 GeoMesa

Der Nutzwert von GeoMesa ist nach Tabelle 5.2 56. Die detaillierte Analyse und Bewertung ist im Anhang A.2.1 zu finden. Darin waren die wichtigsten Quellen die offiziellen Webseiten von GeoMesa [Loc14b] und [Loc14c] sowie der Artikel [Fox14].

Neben dem messbaren Nutzwert sind die nichttechnischen Kriterien zu nennen, welche auf die Auswahl eines Frameworks Einfluss haben. Dazu zählt die Herstellerfirma mit Marktposition, Produktplanung und Service sowie das Produkt in Hinsicht auf Preis, Lebendigkeit in Form von Entwickleraktivität und Größe der Benutzer.

<https://github.com/locationtech/geomesa> zählt am 17.2.2015 271 commits, 20 contributors und 186 branches. Eine solche hohe Anzahl an branches spricht normalerweise für eine hohe Nutzung und Lebendigkeit des Projektes. Jedoch wurde die Mehrzahl der branches nicht in den master Zweig übernommen. Das GeoMesa Projekt auf GitHub hat nach Abbildung 5.2 eins bis vier Stammprogrammierer und ist im zweiten und dritten Quartal gegenüber mit der doppelten Anzahl an contributors gegenüber den anderen Quartalen fragmentiert. Die drei contributors mit dem größten Anteil an Änderungen

5 Systemauswahl

Metrik	erreichter Wert	Erfüllung in %	Kommentar	gewichteter Teilnutzen
Interoperabilität	7	58	Implementationen für beide Schnittstellen notwendig.	17
Funktionsumfang	48	79	Die meisten Funktionen sind nur mit Scala verfügbar, Mindestabdeckung jedoch gegeben.	16
Dokumentation	4	31	Mindestabdeckung nicht erfüllt.	11
Modifizierbarkeit	4	80	Mit Simple Features und Spark umfangreiche Problemlösungen erstellbar. Fehlende Funktionen können nachgerüstet werden.	12

Tabelle 5.2: Nutzwertanalyse GeoMesa

Feb 16, 2014 – Feb 17, 2015

Contributions to accumulo1.5.x/1.x, excluding merge commits

Contributions: Commits ▾

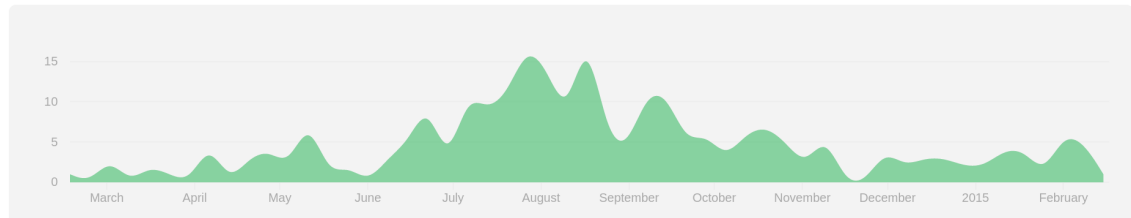


Abbildung 5.2: Zeitleiste der contributor von GeoMesa vom 17.2.2015 nach <https://github.com/locationtech/geomesa/graphs/contributors>

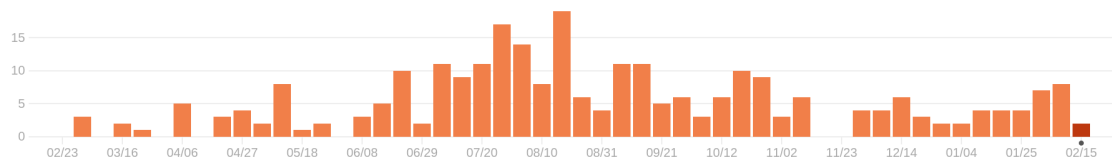


Abbildung 5.3: Zeitleiste der commits von GeoMesa vom 17.2.2015 nach <https://github.com/locationtech/geomesa/graphs/commit-activity>

sind vorwiegend in Projekten von LocationTech aktiv was darauf schließen lässt, dass sie für das Unternehmen arbeiten. Daraus folgt das zum wesentlichen Teil das Unternehmen LocationTech das Projekt wartet. Die Anzahl der commits geht mit dem Verlauf der ak-

tiven Programmierer einher. Abbildung 5.3 zeigt die selbe Quartalsweise Verteilung wie Abbildung 5.2. Dabei ist der Unterschied zwei zu neun commits pro Woche.

LocationTech ist eine Arbeitsgruppe der non-for-profit Stiftung Eclipse. In diesem Rahmen erhält diese Arbeitsgruppe 20 Mitglieder für Projektplanung und Projektumsetzung. Weiterhin findet die Finanzierung im Rahmen von Mitgliedschaft an der Arbeitsgruppe statt. Darin können Mitglieder je nach Beitrag Teile der Entscheidungsorgane der Arbeitsgruppe werden und Zugang zu Ergebnissen dieser erhalten. [Loc] LocationTech ist mit GeoMesa mitten in der Entwicklung und hat keine permanenten aktive Unterstützer. Dieser Stand spricht gegen eine Auswahl von GeoMesa zum produktiven Einsatz.

5.2.2 Postgres-XL

Metrik	erreichter Wert	Erfüllung in %	Kommentar	gewichteter Teilnutzen
Interoperabilität	12	100	Analog des Ist-Standes.	30
Funktionsumfang	53	87	Mindestabdeckung erfüllt, jedoch sind Geostatistik und Versionierung nicht vorhanden.	17
Dokumentation	9	69	Dokumentation zu PostGIS ist sehr gut, zu Postgres-XL grob. Mindestabdeckung ist erfüllt.	24
Modifizierbarkeit	5	100	Vollständige Abdeckung vorhanden. Möglichkeiten sind in SQL gegeben.	15

Tabelle 5.3: Nutzwertanalyse Postgres-XL

Aus Tabelle 5.3 ergibt sich ein Nutzwert von 86. Das Ergebnis bezieht sich auf Anhang A.2.2. Es wurde vorrangig die Postgres-XL Dokumentation [Trac] und jene von PostGIS [OSGb] verwendet.

Dazu sind ebenso nichttechnische Faktoren zu berücksichtigen.

<https://github.com/snaga/postgres-xl> zählt am 17.2.2015 35.266 commits, 23 contributors und drei branches. Abbildung 5.4 zeigt einerseits, dass dieses Projekt seit 1998 besteht, andererseits dass die Zahl der aktiven contributors im Gegensatz der Jahre 1998 bis 2012 zu 2014/2015 in etwa ein viertel beträgt. Diese deutliche abrupte Abnahme

Jul 7, 1996 – Feb 17, 2015

Contributions to master, excluding merge commits

Contributions: Commits ▾

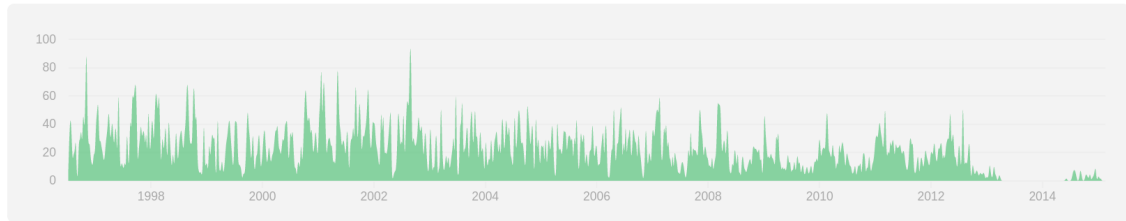


Abbildung 5.4: Zeitleiste der contributor von Postgres-XL vom 17.2.2015 nach <https://github.com/snaga/postgres-xl/graphs/contributors>

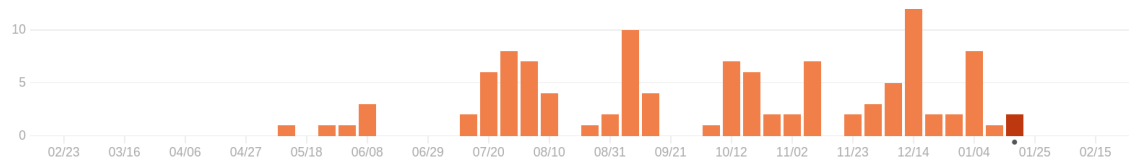


Abbildung 5.5: Zeitleiste der commits von Postgres-XL vom 17.2.2015 nach <https://github.com/snaga/postgres-xl/graphs/commit-activity>

der aktiven Programmierer deutet eine Veränderung im Projekt oder den Projektverantwortlichen an. Die commits des vergangenen Jahres sind in Abbildung 5.5 dargestellt. Danach wurden im ersten Halbjahr 2014 nur insgesamt 6 commits und im zweiten Halbjahr 2014 etwa täglich ein commit durchgeführt.

Das Unternehmen TransLattice¹ übernahm im Mai 2014 das Unternehmen StormDB. Die Übernahme schloss das Projekt Postgres bzw. Postgres-XC ein. (siehe [Tra13]) Dieses wurde darauf in Postgres-XL umbenannt und erweitert. Diese Änderung rief die Verringerung der contributors seit Anfang 2014 hervor. TransLattice verwaltet seitdem das Projekt und stellt technischen sowie theoretischen Support. Postgres-XL ist durch die langjährige Entwicklung empfehlenswert für den produktiven Einsatz. Jedoch ist die Aktivität der TransLattice Entwickler zu beobachten, da die Gefahr besteht das dieses Projekt vom Unternehmen nicht mehr gefördert wird und somit Fehler und Verbesserungen nicht eingepflegt werden und neue PostgreSQL Versionen nicht unterstützt werden.

5.2.3 Rasdaman

Aus Tabelle 5.4 ergibt sich ein Nutzwert von 51. Die granulare Bewertung befindet sich im Anhang A.2.3. Als Quelle diente dabei [Ras14a] mit verlinkten Websites des gleichem Domain Namen.

¹<http://www.translattice.com/>

5 Systemauswahl

Metrik	erreichter Wert	Erfüllung in %	Kommentar	gewichteter Teilnutzen
Interoperabilität	7	58	<i>UMN MapServer</i> Schnittstelle ist nicht vorhanden.	17
Funktionsumfang	10	16	Umfangreiche Rasterverarbeitung möglich. Kostenlose Version enthält keine Optimierungen. Abbildung von Simple Features auf Arrays mit Aufwand verbunden und nur bedingt sinnvoll. Mindestabdeckung wird nicht erfüllt.	3
Dokumentation	8	62	Mindestabdeckung erfüllt.	22
Modifizierbarkeit	3	60	Einfache Java und C++ API bietet zwar Erweiterungsmöglichkeiten, aber die Mindestabdeckung ist durch fehlende Datentypen nicht gegeben.	9

Tabelle 5.4: Nutzwertanalyse Rasdaman

Die Statistiken der Entwicklung des Repository müssen händisch gewonnen werden, da es auf einem Trac Verwaltungssystem mit Git basiert. Das Repository ist unter kahlua.eecs.jacobs-university.de/rasdaman.git verfügbar. Es kann mit der Konsolenanwendung Git heruntergeladen und ausgewertet werden. So erhält man mit `git log --pretty=format:“%h - %an, %ad : %s“ | tail -1` das der erste commit 2009 erstellt wurde:

0f1055b - Constantin Jucovschi, Tue Mar 31 06:18:54 2009 -0400 : Initial commit

Somit sind auch die commits des vergangenen Jahres ermittelbar. So wurden vom 19.2.2014 bis zum 19.2.2015 1949 commits durchgeführt. `git shortlog -sne` liefert dagegen alle Autoren der vorhandenen commits. Die Autoren der meisten commits sind Dimitar Misev mit 456, Piero Campalani mit 301 und Andrei Aiordachioaie mit 74 commits, Stand 19.2.2015 14:00 Uhr. Herr Misev ist Director of Product Development der

rasdaman GmbH. Herr Campalani und Herr Aiordachioaie haben wie Herr Misev an der Jacobs Universität in Bremen studiert, arbeiten aber nicht bei der rasdaman GmbH.

Rasdaman ist laut der Meldung „Führender Rasterserver kostenfrei zum Download“ in [Ras12] seit September 2008 in einer freien Version verfügbar. Außerdem ist es aus Forschungsarbeiten der TU Darmstadt, der TU München und der Jacobs University Bremen entstanden. Förderer war dabei das Community Research and Development Information Service der EU. [Com98]

5.3 Zusammenfassung

Auf Grund der Ähnlichkeit von Postres-XL zum Ist-Stand erfüllt es nicht nur alle Mindestanforderungen, sondern erzielt auch den höchsten Nutzwert der untersuchten Frameworks. Der Nutzwert von 86 ist auf 86% Erfüllung der untersuchten Qualitätsmetriken abbildbar. Da die Anforderungen neben vorhandenen Qualitäten des Ist-Standes fehlende dessen enthalten, betont diese hohe Erfüllung die Eignung für zukünftige Anforderungen an Zeitverhalten und Modifizierbarkeit. Weiterhin spricht die Existenz seit 1996 für Postgres-XL. Einzig die Übernahme der Firma TransLattice und der damit einhergegangene Einbruch der Anzahl an commits und contributors ist negativ und für die Zukunft zu beobachten.

GeoMesa und Rasdaman sind für andere spezielle Anwendungsfälle geeignet. So ist Rasdaman in der kommerziellen Version bei verteilter Rasterdatenverarbeitung zu empfehlen. GeoMesa eignet sich auf Grund des BigTable Ansatzes für enorm große Datenmengen die im Peta Bereich liegen. So können diese Daten nicht nur gespeichert, sondern auch mit Scala und dazugehörigen Frameworks und Bibliotheken verteilt und parallel nach selbst erstellten Algorithmen und Vorgängen verarbeitet werden.

Die Systemauswahl hat nicht nur ein geeignetes Framework als Ergebnis, sondern auch ein Grundgerüst zur Beurteilung anderer Frameworks und Anwendungsfälle anhand von Nutzwertanalysen. In Folge ist Postgres-XL detailliert zu untersuchen und erneut zu bewerten.

6 Untersuchung von Postgres-XL

Nach der Auswahl von Postgres-XL im vorangegangenen Kapitel, wird es in diesem hinsichtlich der Verwendung erläutert und eine Empfehlung für den Einsatz bei Agri Con gegeben.

6.1 Verwendung

Im Abschnitt 2.3.5 der Grundlagen wurde Postgres-XL allgemein beschrieben und im Unterkapitel A.2.2 wurde es hinsichtlich des Anwendungsfalles mit einer Nutzwertanalyse bewertet. Darauf aufbauend folgt die Darstellung der konkreten praktischen Verwendung entsprechend den Abschnitten Installation, Schnittstelle und Verarbeitung. Für das Verständnis werden grundlegende Linux Kenntnisse und ein root Zugang voraus gesetzt.

6.1.1 Installation

Systemvoraussetzungen

Die Dokumentation¹ verwendet hierbei deckungsgleich die offizielle Dokumentation zu den Systemanforderungen von PostgreSQL². Dabei wird ein Linux Betriebssystem, 155MB freien Festplattenspeicher für die Übersetzung, Installation und Erstellung eines leeren Datenbankclusters sowie eine Menge von Paketen genannt. Diese ist: GNU make, gcc, tar und zlib als benötigte sowie libperl oder libpython als optionale Pakete. Zusätzlich werden für die Erzeugung der Dokumentation oder über Übersetzung des Quellcodes weitere Pakete benötigt. Diese Anforderungen setzen eine Standard Installation voraus. Neben Linux Derivaten wird auch FreeBSD und Max OS X unterstützt. Die Prozessorarchitektur von Intel wird unterstützt, andere sind laut Dokumentation ebenso verwendbar.

¹Dokumentation Postgres-XL:
[install-requirements.html](http://files.postgres-xl.org/documentation/install-requirements.html)

²Dokumentation PostgreSQL:
[install-requirements.html](http://www.postgresql.org/docs/9.2/static/install-requirements.html)

<http://files.postgres-xl.org/documentation/>

<http://www.postgresql.org/docs/9.2/static/>

Im Rahmen dieser Arbeit konnte Postgres-XL auch auf einem Raspberry Pi 1 Model B, basierend auf einem ARMv6 Prozessor und dem Linux Derivat Raspbian, übersetzt und verwendet werden.

Installation

Postgres-XL steht als RPM und direkt als Quellcode bereit. Davon verfügbare RPM Pakete sind jedoch von Mai 2014 und somit veraltet. Im Rahmen dieser Arbeit wurde der aktuelle Quellcode von github verwendet. Die Übersetzung des Quellcodes erfolgt mit einer im Linux Umfeld oft verwendeten `configure`, `make` und `make install` Routine. Der aktuelle Quellcode wird mit dem Kommandozeilen- und Versionsverwaltungstool `git` auf den Computer geladen und die darin enthaltene Datei `configure` mit zusätzlichen Parametern zum Zwecke der Einrichtung der anschließenden Installation ausgeführt. Die Ausführung von `make` übersetzt den Quellcode und der Parameter `install` kopiert die Übersetzungen in die mit `configure` festgelegten Ordner. Im Anhang A.3 ist das Skript zur Installation von Postgres-XL auf dem Testsystem zu sehen. Das Installationsskript muss für andere Systemumgebungen angepasst werden, da Pfade und notwendige Pakete unterschiedlich sein können. Weiterhin ist zu erwähnen, dass Änderungen an der Kernel-Konfiguration vorzunehmen sind, jedoch ebenso abhängig von der Systemumgebung. Im Testsystem musste der Wert des für jede Anwendung nutzbaren geteilten Speichers erhöht werden, um Postgres-XL starten zu können.

Um das Kommandozeilentool `pgxc_ctl` nutzen zu können, muss im Quellcode Ordner in `./contrib/pgxc_ctl` gewechselt und dort `make` sowie `make install` ausgeführt werden. Damit wird das Tool übersetzt und in den in der vorangegangenen Installation festgelegten Ordner kopiert.

Einrichtung

Postgres-XL ist für den Einsatz als Cluster konzipiert. So muss die Installation für jeden Knoten vorgenommen werden. Die Einrichtung der einzelnen Knoten variiert je nach Art des Knotens, wobei ein Knoten entweder eine GTM Instanz oder mehrere Coordinator sowie DataNodes Instanzen mit einer GTM-Proxy Instanz enthält. Jede Instanz kann automatisiert mit `pgxc_ctl` oder manuell erstellt und konfiguriert werden. Das Testsystem wurde mit `pgxc_ctl` eingerichtet. Voraussetzung der Nutzung dieses Kommandozeilentools ist der Zugang zu allen Knoten per SSH ohne Passwortabfrage für

den selben Benutzer und die Vergabe eindeutiger Hostnames an die Knoten. Ist dies gegeben, kann `pgxc_ctl` in der Kommandozeile gestartet werden. Mit dem Kommando `prepare config` erzeugt `pgxc_ctl` eine Konfigurationsdatei unter dem in der Umgebungsvariable `PGXC_CTL_HOME` festgelegten Ordner. In dieser Datei werden alle Elemente des Clusters definiert. Dazu zählt: GTM, GTM-Proxys, Coordinators und DataNodes. Außerdem können zu allen vier Typen Slaves definiert werden, welche bei Ausfall des Elementes dessen Aufgaben übernehmen. So wird pro Element das Arbeitsverzeichnis, der Name, der Host, der Port, optionale Konfigurationsparameter, `pg_hba` Einträge, `pgPool` Port und der Ordner der Logdateien festgelegt. Eine detaillierte Beschreibung befindet sich in der Dokumentation.³ Anhang A.6 enthält die Konfigurationsdatei des Testsystems mit zwei ergänzenden Dateien.

6.1.2 Schnittstelle

Der Zugriff auf Daten eines Postgres-XL Clusters erfolgt über die Coordinators. Dazu sind Programme und Tools aus dem PostgreSQL Umfeld zu verwenden. Dazu zählen *Java Database Connectivity (JDBC)*, das Kommandozeilentool `psql` und das grafische Programm zur Datenbankverwaltung `pgAdminIII`. So wird im Testsystem der erste Coordinator mit folgendem Aufruf angesprochen:

LISTING 6.1: Verbindung zu erstem Coordinator herstellen

```
psql -h node1 -p 20004 agrodb
```

Ebenso sind die SQL Befehle bis auf ein paar Ausnahmen deckungsgleich. Diese Ausnahmen beziehen sich auf die Verteilung der Daten und Verwaltung des Clusters. Beispielsweise das SQL Statement in Listing 6.2 weicht durch die Ergänzung `Distribute by` vom PostgreSQL SQL Syntax ab.

LISTING 6.2: SQL Query zum erzeugen einer Tabelle in Postgres-XL

```
Create Table tblname (serial id, text data) Distribute by Hash(id)
;
```

Jede Tabelle wird in jeder Coordinator und DataNode Instanz erzeugt. Dabei werden die Daten entweder nach einem Attribut verteilt oder zwischen den Datenbankinstanzen gespiegelt. Das Schlüsselwort `Distributed` veranlasst eine Verteilung der Daten,

³<http://files.postgres-xl.org/documentation/pgxc-ctl.html>

Replicate dagegen eine Replikation der Tabelle über alle Nodes. Die unterstützten Datentypen sind der Dokumentation zu entnehmen. Weiterhin zu erwähnen ist der Befehl des Listings 6.3, welcher direkt als SQL Statement verwendet werden kann und dem Cluster einen Knoten hinzufügt.

LISTING 6.3: SQL Query zum hinzufügen eines aktiven Knoten in einen Postgres-XL Cluster

```
Create Node nodename With (TYPE=, HOST=, PORT=)
```

Analog dazu existiert der Befehl `Drop Node nodename` zum entfernen eines Knotens. Wurde das Cluster verändert, ist dies mit Listing 6.4 für alle Knoten zu propagieren.

LISTING 6.4: SQL Query zur Aktualisierung der Cluster Zusammensetzung

```
Select * From pgxc_pool_reload();
```

Weitere Abweichungen sind der Dokumentation zu entnehmen.⁴

Das Datenbankschema wurde mit `pg_dump` in eine Textdatei geladen und als SQL Befehle in Postgres-XL übernommen. Die Übernahme der Daten erfolgte mit `db_link`, welches auch zwischen unterschiedlichen PostgreSQL Versionen funktioniert. Die Daten können auch analog des Schema übertragen werden, dies dauert jedoch auf Grund der Umwandlung von zu zu Textformaten länger.

6.1.3 Verarbeitung

Die Datenverarbeitung erfolgt analog des Ist-Standes bei Agri Con. Mit der Installation von PostGIS als Erweiterung wie in Skript A.5 dargestellt, können die vorhandenen SQL Funktionen übernommen werden, ebenso die R Bibliotheken des speziellen Krigings. Dafür ist neben R als Programmiersprache mit notwendigen Paketen auf den Systemen, PI/R als Erweiterung in Postgres-XL, zu installieren. Die Installation von PI/R mit dem Quellcode als Grundlage ist in Anhang A.4 zu finden.

6.2 Entwurf

Dieses Unterkapitel enthält die Definition der Postgres-XL Konfiguration für den Einsatz bei Agri Con. Als erstes werden die Gründe für diese Art der Konfiguration dargelegt.

⁴siehe <http://files.postgres-xl.org/documentation/sql-commands.html>

Die Erfüllung der Anforderungen liegt entsprechen der Nutzwertanalyse von Postgres-XL bei 86%. Aus diesem Grund besteht die geeignete Möglichkeit einen Entwurf zu erzeugen, welcher wesentliche Aufgaben des Ist-Standes übernimmt oder den Ist-Stand ersetzt.

Gegen eine Ersetzung spricht das Trigger in der aktuellen Postgres-XL Version nicht verwendet werden können. Zwar können Trigger in Funktionen ausgelagert und bei Verwendung der Tabellen aufgerufen werden, aber dies würde einen erheblichen Implementationsaufwand bedeuten und nicht zu 100% die bestehende Funktionalitäten abbilden. Jedoch ist eine grundlegende Integration in den Ist-Stand notwendig. Dies ist in der Datenabhängigkeit der Bestandsdaten und der Eignung von Postgres-XL begründet. Das Datenbankschema befindet sich in der ersten und zweiten Normalform, wodurch eine Abhängigkeit über Fremdschlüssel zu häufig verwendeten Tabellen wie farm.farms oder farm.fields besteht, was eine Auslagerung von Tabellen und Funktionen nicht trivial macht. Eine teilweise Schemaintegration mit anschließender Normalisierung der zwei Datenbankschemata würde Änderung aller Programme nach sich ziehen und in diesem Rahmen zu aufwendig ausfallen. Abbildung 6.1 skizziert den notwendigen Zeitraum zur Umsetzung dessen. Deshalb ist die Erstellung eines umfassenden Entwurfs hinsichtlich

	Q3 2015	Q4 2015	Q1 2016	Q2 2016
Schemaintegration				
Normalisierung				
Anpassung der Programme				

Abbildung 6.1: Aufwandsschätzung Umsetzung einer teilweisen Integration von Postgres-XL in den Ist-Stand

des speziellen Szenarios bei Agri Con im Rahmen dieser Arbeit nicht möglich. Ebenso ist der Aufwand zur Implementierung der Trigger Funktionalität in Postgres-XL zu hoch.

Der Entwurf wird daher als Klon des Ist-Standes, jedoch ohne Trigger, erstellt. Entsprechend den Leistungstests soll der Entwurf die Produktivdaten liefern und anwendungsnahe Berechnungen ausführen. Somit sind Aussagen zur Leistungsfähigkeit gegenüber dem Ist-Stand treffbar. Die Speicherung von historischen Daten wird aus Zeitgründen nicht betrachtet. Eine Übernahme aller vorhandenen Daten wurde mit Abschnitt 6.1.2 erläutert, sodass die Möglichkeit der Speicherung von historischen Daten damit gezeigt ist.

Der Ist-Stand wurde unter Kapitel 4.2 dargelegt, somit sind die Unterschiede des für die Testumgebung umgesetzten Entwurfes zu erläutern. Das Datenbankschema unterscheidet sich von dem des Ist-Standes durch das Fehlen von Triggern und einigen Fremdschlüssel Constraints. Diese Fremdschlüsselbeziehungen müssen entfernt werden, da Primär-

und Fremdschlüssel immer in `distribute by` enthalten sein müssen, was entweder auf Grund der Stufe der Datennormalität nicht umsetzbar ist oder diese Beziehungen in den Tests nicht verwendet werden. Diese Änderungen sind auch im Vergleichssystem mit PostgreSQL 9.3 enthalten. Weiterhin wurden die zu testenden Funktionen an den Funktionsumfang von Postgres-XL angepasst. Dabei wurden interne Transaktionen und inserts aus den verwendeten Funktionen entfernt. Das Vorgehen zur Erstellung dieses Aufbaus war zusammengefasst folgender: Postgres-XL auf sieben VMs installiert, eine Konfigurationsdatei für `pgxc_ctl` erstellt, den Cluster mit `pgxc_ctl` erzeugt, eine Datenbank mit dem Namen `agrodb` erzeugt, darin die Erweiterungen `plr`, `dblink` und `PostGIS` installiert, das Datenbankschema mit Funktionen angepasst, das Datenbankschema und Hilfsfunktionen eingespielt und die Daten mit `dblink` für ausgewählte Betriebe übernommen. Die Schritte sind mit entsprechender Zeitschätzung in Abbildung 6.2 zu finden.

	1h	2h	3h	4h	5h	6h	7h	8h	9h
Installation Postgres-XL	■								
Erstellung Konfigurationsdatei		■							
Erzeugung Datenbank			■						
Installation Erweiterungen			■						
Anpassung Datenbankschema				■	■	■			
Übernahme Datenbankschema							■		
Übernahme Daten							■	■	■

Abbildung 6.2: Vorgehen zur Erstellung der Postgres-XL Testumgebung mit Zeitschätzung

Es handelt sich um eine Schätzung des Zeitaufwandes, da der Aufwand einzelner Schritte vom Szenario abhängt, ergo nicht allgemein bestimmbar ist. Außerdem kann es dazu kommen, dass bei einer Fehlkonfiguration von Postgres-XL der Cluster neu erstellt werden muss, um die vollständige Umsetzung zu realisieren.⁵

⁵Während der Erstellung durch den Autor waren falsche Einstellungskonstellationen aus Knoten und Datenbankschema irreversibel, sodass nur eine Neuerstellung die angestrebte Konfiguration ermöglichen konnte.

7 Tests

Postgres-XL wird hinsichtlich Funktionalität und Leistungsfähigkeit bezüglich der Anforderungen getestet. Die systematischen Tests werden in diesem Kapitel vorgestellt. Die Kriterien wurden in Kapitel 4.1.2 definiert, somit werden hier die Randbedingungen, Eingaben, Durchführung und Ergebnisse dargelegt.

7.1 Testumgebung

Die wesentliche Randbedingung Testumgebung wurde nach der Softwareauswahl und vor der Durchführung der Tests erstellt. Die Testumgebung ist Hardware aus dem Jahr 2007 und soll dazu dienen, relative Aussagen über die Leistung von Postgres-XL zu treffen. Dafür steht ein IBM Rack Server x3850 M2¹ mit folgender Ausstattung zur Verfügung: vier Xeon E7330 Quad-Core mit 2,4 GHz, 64GB DDR2 RAM, vier 500GB 2,5 Zoll SATA Festplatten von Western Digital mit 7.200 U/min und einem MR10k Raid-Controller. Der Postgres-XL Cluster wird mit Virtualisierung der einzelnen Knoten erstellt. Als Virtualisierungssoftware kommt *VMware ESXi* in der kostenlosen Version 6.0 mit *VMware vSphere* 6.0² als Testversion zum Einsatz. Mit dieser Virtualisierungslösung ist es möglich, Ressourcen explizit und ausschließlich einer *VM* zuzuordnen. So werden die Prozessorkerne in Paaren und der Arbeitsspeicher direkt und mit exklusiver Verwendung den einzelnen *VMs* zugeordnet.

Ziel dieser unter Abbildung 7.1 skizzierten Testumgebung ist es einen homogenen Cluster zu erzeugen und Postgres-XL mit der PostgreSQL Konfiguration des Ist Standes zu vergleichen.

Die Ausstattung des Computers zur Verwaltung der Virtualisierung wird nicht definiert, da dies keinen Einfluss auf Messungen hat. Die *VMs* sind in Abbildung 7.2 dargestellt. Typ I enthält eine GTM Instanz und Typ II eine GTM-Proxy, eine Coordinator und zwei DataNode Instanzen. Dabei stehen sechs Typ II VMs zur Verfügung. Jede *VM* besitzt

¹<http://www.redbooks.ibm.com/redpapers/pdfs/redp4362.pdf>

²<https://www.vmware.com/de/products/vsphere>

als Betriebssystem Ubuntu 14.04 LTS und alle für die Installation und Ausführung des Prototypen notwendigen Pakete. Weiterhin erhalten die Typ II *VMs* folgende Hardware Zuordnung: zwei Prozessorkerne, sieben GB RAM und 100 GB Festplattenspeicherplatz. Typ I erhält dagegen zwei Prozessorkerne, sieben GB RAM und 20 GB Festplattenspeicherplatz.

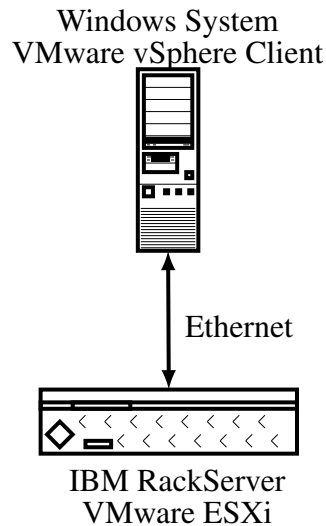


Abbildung 7.1: Aufbau der Geräte des Testsystems

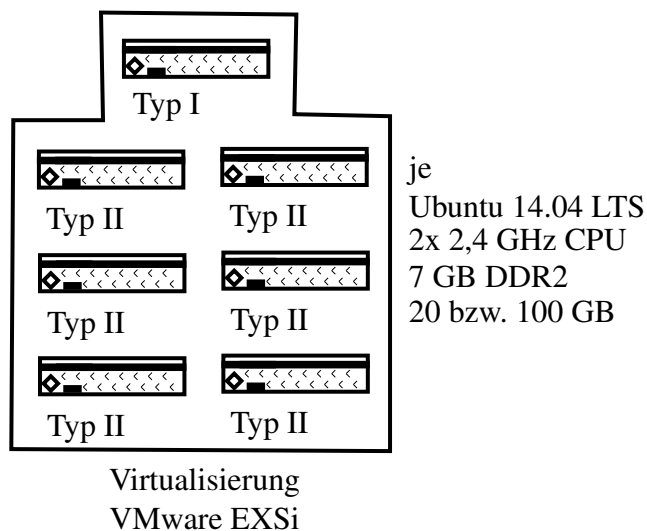


Abbildung 7.2: Aufbau der VMs des Testsystems

Als Referenzsystem wird ein Ubuntu System mit PostgreSQL 9.3.5, PostGIS 2.1.5 und R sowie der Hardwareausstattung einer Typ II *VM* verwendet. Mit der Übereinstimmung der Hardwareausstattung der *VMs* ist eine Vergleichbarkeit von PostgreSQL und Postgres-XL gegeben.

Die physischen Festspeicher des Testservers stehen den *VMs* nicht exklusiv zur Verfügung. Die Anzahl an Operationen pro Zeiteinheit auf diesem Medium soll für alle *VMs* annähernd gleich sein. Als Kompromiss aus Verteilung der Last auf alle Festspeicher, Ausfallsicherheit und Kosten, werden die Festspeicher in einem RAID 5 Verbund zusammengefasst. So stehen 1,4 TB Festspeicher zur Verfügung. Die *VMs* sind mit virtuellen Netzwerkkarten von 10Gbit/s verbunden, wodurch die Kosten der Datenübertragung im Netzwerk vernachlässigt werden können. Um die Vergleichbarkeit der zeitlich abhängigen Messwerte zu gewährleisten, wird ESXi als Zeitgeber für alle *VMs* eingerichtet.

Zur Überwachung der Auslastung der einzelnen *VMs* bietet VMware mit vSphere einfache Graphen für die zeitabhängige Auslastung der einzelnen Kerne, des Arbeitsspeichers und weiterer Komponenten. Diese Graphen besitzen als zeitliche Achse einen beliebigen Bereich bis zum momentanen Zeitpunkt. Damit sind historische Daten für einzelne *VMs* und ausgewählte Zeiträume nicht darstellbar. Deshalb wird zur Überwachung Zabbix eingesetzt.

Zabbix ist ein freies Framework zur Überwachung von Computern und Netzwerken. Auf einem Zabbix Server werden Daten zentral von beliebigen Zabbix Clients empfangen. Dabei sind die Art der zu überwachenden Daten beliebig, standardmäßig wird die Systemauslastung und allgemeine Daten zum jeweiligen System an den Server gesendet. Dazu können eigene Skripts installiert werden und Protokolle wie *Simple Network Management Protocol (SNMP)* oder *Intelligent Platform Management Interface (IPMI)* zur Datenerhebung und Verwaltung genutzt werden. Ein Zabbix Server wird zur Analyse und Verwaltung der Zabbix Clients mit einem PHP Web-Frontend versehen. Über dieses Frontend können Clients konfiguriert, deren Status eingesehen, Diagramme zu erhobenen Daten definiert und angezeigt werden. Weiterhin ist Zabbix ein Mehrbenutzer System und reagiert mit Funktionsaufrufen oder E-Mails auf definierte Ereignisse wie den Ausfall einzelner Clients.

7.2 Funktionstests

Im Abschnitt Prototypische Implementierung auf Seite 22 wurden die Funktionstests bereits als Black-Box Tests definiert und der Umfang im Kapitel 4.1.3 genauer festgesetzt. In diesem Abschnitt werden die Funktionstests für die Durchführung spezifiziert und deren Ergebnis festgehalten. Es wird eine vollständige Erfüllung der Funktionstests erwartet, da PostgreSQL 9.2 und PostGIS 2.1.5 an sich die gesetzten Anforderungen erfüllen. Das

Fehlen von Triggerfunktionen in Postgres-XL wird durch die Funktionstests nicht aufgedeckt. Jedoch ist dies als wesentliche Einschränkung festzuhalten und für die Auswertung heran zu ziehen.

Alle Funktionstests werden im Anhang A.7 in Form eines Testdokumentes³ zusätzlich dargelegt. Darin wird der Testfall beschrieben, die Testdaten konkretisiert, das Sollergebnis vordefiniert, das schlussendliche Ergebnis dargestellt und die Akzeptanz des Tests erläutert. Entsprechend der Reihenfolge der Testfälle im Anhang werden nachfolgend die Tests mit deren Durchführung und deren Ergebnissen präsentiert. Die Testfälle werden durchnummeriert und mit einem vorangestellten FT für Funktionstest bezeichnet. Dabei werden die Testfälle mit absteigender Priorität aufgelistet. Die Priorität hängt vom geschätzten Implementationsaufwand zur Behebung des Fehlschlages des Testfalles ab.

FT01:

Die Schnittstellen sind nicht explizit zu testen, da Postgres-XL und PostgreSQL wie in Abschnitt 6.1.2 demonstriert direkt Daten austauschen können.

FT02:

Für diesen Funktionstest wird der *UMN MapServer* als *FastCGI* Modul und einem geeigneten Mapfile verwendet. Es wird ein Kartenausschnitt vom *UMN MapServer* angefordert, welcher drei Schläge enthält. Arbeitet Postgres-XL mit dem *UMN MapServer* korrekt zusammen, liegt ein Bild mit den Schlägen in Form von grauen Polygonen als Ergebnis vor. Diese Schnittstelle ist nutzbar. Abbildung 7.3 ist die Rückgabe des *UMN MapServer*.

FT03:

Da beide Systeme PostGIS 2.1.5 verwenden, sind die Austauschformate für geographische Daten identisch. Der Beweis stellt die erfolgte Übernahme der Daten des Ist-Standes dar.

FT04:

Mit PostGIS kann mit der Funktion `st_transform` direkt zwischen den Koordinatenreferenzsystemen umgerechnet werden:

LISTING 7.1: SQL Query zur Umrechnung zwischen Koordinatenreferenzsystemen

```
select id, st_astext(st_transform(geom, 3857)) as transformed,
       st_srid(st_transform(geom, 3857)) as newsrid, st_astext(
       st_transform(st_transform(geom, 3857), 4326)) as original,
```

³Ein Testdokument mit Testfällen in tabellarischer Form, wie es in der Softwareentwicklung zum Einsatz kommt



Abbildung 7.3: Kartenausschnitt mit 3 Schlägen

```
st_srid(st_transform(st_transform(geom, 3857), 4326)) as srid
from nutrients.samples limit 10;
```

So werden die Punkte der Tabelle `samples` im Schema `nutrients` mit dem EPSG Code 4326 zum EPSG Code 3857 sowie wieder zurück umgerechnet. Die Funktionsdeklaration lautet: `geometry ST_Transform(geometry g1, integer srid);` `srid` ist dabei ein EPSG Code aus der Tabelle `SPATIAL_REF_SYS`, welche 3911 Einträge im Testsystem enthält. `geometry` steht dagegen für einen beliebigen räumlichen Datentyp.

FT05:

Auch in diesem Funktionstests findet die Validierung anhand von Bildern statt. Es werden sich überlappende Polygone aus `farm.fields` heraus gesucht und mit Hilfe von PostGIS Funktionen verschnitten. Dafür werden folgende Funktionen verwendet:

LISTING 7.2: PostGIS Funktionen der Verschneidung

```
geometry ST_Intersection( geometry geomA , geometry geomB );
geometry ST_Union(geometry g1, geometry g2);
geometry ST_Difference(geometry geomA, geometry geomB);
geometry ST_SymDifference(geometry geomA, geometry geomB);
```

Das Ergebnis jedes Funktionsaufrufes wird gespeichert und zu einer Karte gerendert. Zur Demonstration wurden zwei Schläge, in Abbildung 7.4 dargestellt, verwendet. Der Rechteckige Schlag erhält die Nummer 1 und der darüber liegende die Nummer 2. Das Ergebnis

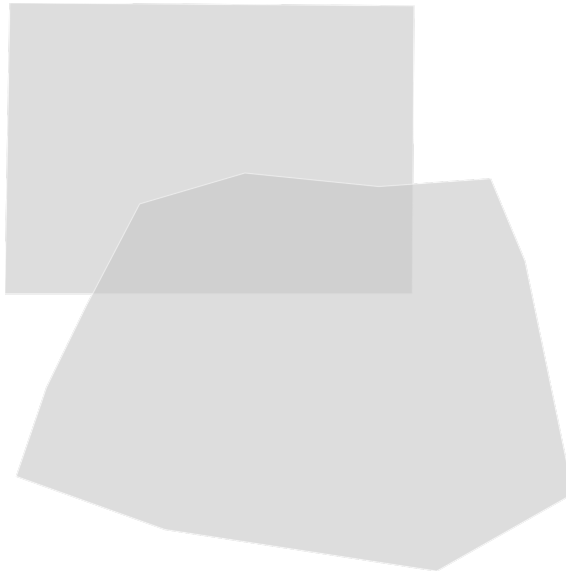


Abbildung 7.4: Kartenausschnitt mit 2 überlappenden Schlägen

der Funktionen ist in den Abbildungen 7.5 und 7.6 abgebildet. Bei Difference ist das Ergebnis abhängig der Reihenfolge der Parameter. Die Geometrien der Schläge 1 und 2 wurden getauscht und beide Ergebnisse mit dargestellt. In allen Fällen sind die Ergebnis-

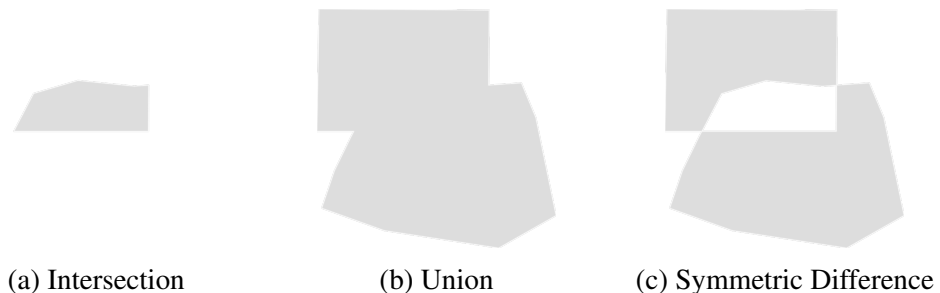


Abbildung 7.5: Ergebnisse Verschneidungsfunktionen Intersection, Union und Symmetric Difference

se korrekt und Die Funktionalität der Verschneidung bewiesen

FT06:

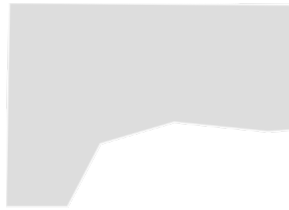
Dies wird im zweiten Szenario der Lasttests im nachfolgenden Abschnitt durchgeführt. Da das Ergebnis nicht überprüft werden kann, gilt der Test als nicht bestanden.

FT07:

Userlayer einer Halbjahresplanung besitzen begrenzende Rechtecke, welche im WKT Format wie folgt beschrieben sind:

LISTING 7.3: Begrenzendes Rechteck im WKT Format

```
POLYGON ((13.2547868501418 50.7878870616778 ,
```



(a) Difference mit Schlag 1 zu Schlag 2



(b) Difference mit Schlag 2 zu Schlag 1

Abbildung 7.6: Ergebnisse Verschneidungsfunktionen Difference

```
13.2833057640198 50.7878870616778 , 13.2833057640198
50.7935526493687 ,
13.2547868501418 50.7935526493687 , 13.2547868501418
50.7878870616778))
```

Mit diesem können alle Schläge abhängig des Userlayers räumlich gefiltert werden. So sind im Testsystem 13803 Schläge vorhanden. Mit der Funktion *boolean ST_Intersects(geometry geomA , geometry geomB);* werden mit dem begrenzenden Rechteck überlappende Schläge gefiltert. Dies ergibt mit dem oben genannten Rechteck 382 Schläge. Mit *boolean ST_Disjoint(geometry A , geometry B);* erhält man dagegen räumlich disjunkte Schläge, 13421 in diesem Fall. Dies soll als Beweis der Verwendbarkeit und Korrektheit der PostGIS Funktionen dienen. Da die PostGIS Version des Ist-Standes mit der des Testsystems übereinstimmt, kann von voller Funktionalität ausgegangen werden. Davon ausgenommen sind Teile der Funktionsmenge zur Verarbeitung von Rasterdaten. Da Postgres-XL internal als Transition Typ nicht unterstützt, ist die Funktionen zur Bildung der Vereinigung mit einem Raster nicht verfügbar⁴.

7.3 Leistungstests

Die hier verwendeten Leistungstests wurden in Kapitel 3 als spezielle Tests zur Messung der Leistungsfähigkeit und Effizienz definiert. Kapitel 4.1.3 beschreibt unabhängig des Testsystems die Arten der Leistungstests. Nachfolgend wird auf den Aufbau der Tests, deren Durchführung und deren Ergebnis eingegangen. Zusammenfassend ist das Ziel dieser Leistungstests, die spezielle Leistung gegenüber des Ist-Standes zu ermitteln. Es wird

⁴Installation von PostGIS wirft Fehler und legt die entsprechenden Funktionen nicht an.

von einer höheren Leistung, abhängig der Anzahl der verwendeten Knoten, bei der Aggregation und der Verarbeitung im Gegensatz zum Ist-Stand ausgegangen. Zwar ist bei Postgres-XL die Verarbeitung lokaler Anfragen mit mehr administrativen Aufwand verbunden, wird somit langsamer gegenüber einer PostgreSQL Instanz bearbeitet, jedoch stehen mehrere Zugriffspunkte auf exklusiven Hardwaresystemen zur Verfügung.

Analog des Testaufbaus der Bachelorarbeit des Autors⁵ wird zur Generierung der Last das freie Java Programm JMeter verwendet, welches mehrere vorbereitete Anfragen parallel und verschränkt erzeugt sowie eine Auswertung der Laufzeiten zur Verfügung stellt. Ein Testlauf wird dabei per Kommandozeile gestartet und die Ergebnisse in einer .csv Datei festgehalten. Diese Datei wird anschließend mit der grafischen Oberfläche von JMeter geöffnet und die Daten mit Graphen und Tabellen ausgewertet. JMeter wird von einem externen Computer ausgeführt. Dieser ist mit 1Gbit/s an das Testsystem angebunden und besitzt die folgende Ausstattung: 16GB DDR3 RAM, 256GB mSata3 SSD und eine Intel i5-3320M Dual-Core CPU mit 2,6GHz und Hyperthreading.

Die PostgreSQL Instanz, welche auf einer dedizierten *VM* als Vergleichssystem zum Postgres-XL Cluster dient, besitzt die gleiche Konfiguration wie die Coordinator Instanzen. Einzig `max_connections` und `max_prepared_transactions` wurden auf 160 erhöht, um der erhöhten Anzahl an Anfragen pro Instanz gerecht zu werden.

Um ein statistisches Mittel zu erreichen und Nebeneffekte des Caching der Datenbank und des Betriebssystems zu reduzieren, wird ein Test elf mal ausgeführt. Es werden die zehn letzten Tests gemittelt. Außerdem wird Postgres-XL vor jeder Testreihe neu gestartet. Dieses Vorgehen orientiert sich an der Masterarbeit von Baas Kapitel 6.1 Objective measurements⁶. Die Graphen der Auslastung der Knoten liegen als Bilder vor. Zur Einschätzung der Auslastung wird der Durchlauf einer gesamten Testreihe mit einem Graph pro Knoten und den dazugehörigen Laufzeiten festgehalten. Ein Graph enthält die Auslastung der CPU und die Menge des verwendeten Arbeitsspeichers in MB. Der Wert der CPU Auslastung geht von null bis unbegrenzt, was die Auslastung auf Linux Systemen wiedergibt. Da es sich pro Knoten um zwei Prozessorkerne handelt, ist der Wert durch 2 geteilt, um mit einer Linie die Durchschnittsauslastung anzeigen zu können. Der Wert eins bedeutet eine Auslastung aller Kerne von 100%. Werte über eins zeigen an, dass das Vielfache der momentan verrichteten Arbeit noch für den Prozessor anliegt.

Es besteht die Möglichkeit die theoretische Leistung anhand des Kostenmaßes zu bestimmen. Nach [Kud07, S.300 f.] setzen sich die Kosten für die Verarbeitung einer Anfrage aus folgendem zusammen:

⁵[Jun12]

⁶[Baa12, S.51]

Rechenzeit Dazu zählt die Zeit des Syntaxprüfers, des Ausführungsplaners, der Kombination und der Sortierung.

Ein- und Ausgabe Besteht aus der Anzahl der Aufrufe an das Speichersystem und der Protokollierung mit internem Speichermanagement.

Datenübertragung Meint die Übertragung von Befehlen und Daten zwischen Komponenten.

Weiterhin sind Einflussfaktoren das logische Datenbank Design, die Anwendung, die Abfrage, der Ausführungsplan, die Datenmenge der Abfrage, das physisches Datenbank Design, die Größe der Datenbank, das *DBMS*, die Systemlast, das Netzwerk, die Parallelität und das Betriebssystem. Eine Berechnung des einheitenlosen Kostenmaßes fällt auf Grund der Komplexität des Anwendungsfalles und der Menge der Einflussfaktoren aufwendig aus. Deshalb wird die Verarbeitungsleistung empirisch mit Tests festgestellt.

Postgres-XL arbeitet mit verteilten Daten. Diese werden entweder auf jeden Knoten repliziert oder zwischen den Knoten aufgeteilt. Abhängig von der Verteilung der Daten und deren Aufrufen ergeben sich unterschiedliche Ergebnisse der Leistungsmessung. Entscheidend für die Laufzeit einer Anfrage an die Datenbank ist die Aufteilung dieser durch den Query Planer. Die erste Query Planer Instanz ist jene des angesprochenen Coordinators. Diese erstellt Unterabfragen entsprechend der Verteilung der Daten auf den DataNodes. Diese Unterabfragen werden lokal vom entsprechenden DataNode verarbeitet und das Ergebnis an den Coordinator zurück gesendet. Diese Zwischenergebnisse werden schlussendlich vom Coordinator zusammengefasst. Somit sind die Daten entsprechend ihrer Verknüpfung hinsichtlich des Datenbankschemas und des Anwendungsfalles für ein geeignetes Zerlegen von Aufgaben durch den Query Planer zu verteilen. Geeignet meint hier das die Unterabfragen bereits wesentlich die Daten filtern, damit der Coordinator einzig die Untermengen zu einer Menge zusammenfassen und nicht mehr als notwendig Datenmengen aggregiert und diese selbst verknüpfen sowie filtern muss. Auf das Schema angewendet bedeutet dies: Die Tabellen *fields* und *farms* des Schemas *farm* werden per Replikation auf alle Knoten verteilt, da diese häufig zur numerischen und räumlichen Filterung verwendet werden und somit jedem DataNode zur Vorfilterung zur Verfügung stehen. Die Tabelle *nsensorlogs* im Schema *n* und die Tabelle *samples* im Schema *nutrients* werden anhand des Attributes *fileid* mit modulo auf die DataNodes verteilt, da dieses Attribut als primäres Filterargument verwendet wird. Das Attribut *fileid* ist für eine modulo Verteilung geeignet, da Gruppen von *files* mit aufsteigender *id* zu einem Betrieb gehören und bei modulo Verteilung die Wahrscheinlichkeit erhöht wird, dass ein DataNode die *files* eines Betriebes vorhält.⁷

⁷Bezogen auf eine betriebsübergreifende Verarbeitung.

Diese Verteilung sollte sich positiv auf die Ergebnisse der Leistungstests auswirken. Es wurden einzig Tabellen berücksichtigt, welche in den Tests verwendet werden. Für den produktiven Einsatz bei der Agri Con entsprechend des Anwendungsfalles sind weitere Änderung in der Verteilung der Daten durchzuführen.

Entscheidend für die Leistungsfähigkeit des *DBMS* ist ebenfalls die Konfiguration des Query Planers. Dazu zählt die Vergabe von Kostenwerten an Funktionen und das Setzen von Konfigurationen wie den Kostenfaktoren und der Algorithmenutzung in den postgresql.conf Dateien. Diese Einstellungen werden im Rahmen dieser Arbeit auf Standardwerten belassen. Da das Vergleichssystem mit PostgreSQL die selbe Konfiguration besitzt ist eine relative Vergleichbarkeit gewährleistet. Für einen produktiven Einsatz ist die Berücksichtigung dessen zu empfehlen. So besteht auch die Möglichkeit Coordinator mit unterschiedlichen Konfigurationen auszustatten, um mehrere Aspekte des Anwendungsfalles abzudecken.

Postgres-XL bietet mit mehreren Coordinator Instanzen mehrere Verbindungspunkte zum Datenbanksystem. Bei gleicher Konfiguration aller Instanzen ist eine gleichmäßige Verteilung der Anfragen an alle Instanzen für die Antwortzeit gewinnbringend. In den nachfolgenden Tests werden alle Instanzen durch JMeter gleich belastet. Für den produktiven Einsatz bietet sich ein Lastverteilungsprogramm wie pgpool an, welches als Verbindungspunkt zwischen Programmen und Postgres-XL dient und die Anfragen an alle Coordinator weiterreicht.

Zur Bewertung eines verteilt arbeitenden Algorithmus ist die Skalierbarkeit bzw. die Effizienz zu ermitteln. Skalierbarkeit ist die Differenz der Laufzeit T bei Verwendung eines Knotens und der Laufzeit bei Verwendung von p Knoten: $T(1)/T(p)$. Dies sagt aus, um welches Maß sich die Laufzeit ändert bei Verminderung oder Erhöhung der Anzahl der Knoten. Ein idealer Wert ist bei n Knoten n . Auf Grund von sequenziellen Anteilen in Subalgorithmen und Mehrkosten für Kommunikation zwischen den Knoten liegt der Wert in realen Szenarios unter n . Die Teilung des Wertes der Skalierung durch die Anzahl der Knoten ergibt die Effizienz. Dieser Wert beurteilt die Skalierung. Eine Effizienz von eins ist bei verteilt arbeitenden Systemen ein Zielwert. Da die Skalierung wie benannt begrenzt ist, befindet sich die Effizienz in realen Szenarios unter eins.

Diese Bewertung der Skalierung wird auch auf verteilt arbeitende Systeme angewandt. Für Postgres-XL ist die Anzahl an Knoten, welche Coordinator und DataNode Instanzen sowie eine GTM Proxy Instanz besitzen, zu ändern und die Differenz der Laufzeiten des unveränderten und geänderten Clusters zu messen. Dabei sind Techniken zur Lastverteilung oder auf gleiche Belastung aller Elemente des Clusters gerichtete Funktionen

zu verwenden. Die Tests mit dem Testsystem zeigten Fehleranfälligkeiten der Verbindungen zwischen den Elementen des Clusters, weshalb auf empirische Ermittlung der Skalierbarkeit von Postgres-XL im Cluster abgesehen wird. Jedoch wird die theoretische Vorgehensweise für das Entfernen von Instanzen gegeben. Das Hinzufügen von Instanzen erfolgt analog mit kleinen Änderungen. Die Vorgehensweise stammt von der Postgres-XL Dokumentation⁸.

Coordinator Eine Verbindung mit einem anderen Coordinator herstellen und darin `DROP NODE coordinatorname;` und `SELECT pgxc_pool_reload();` ausführen.

DataNode Mit beliebigen Coordinator verbinden und für jede Relation, welche auf die zu löschende DataNode verteilt ist, `ALTER TABLE tablename DELETE NODE (datanodename);` ausführen. Betroffene Relationen können mit Listings ?? ermittelt werden. Das Tool `pgxc_ctl` starten und darin `stop datanode datanodename` ausführen. Auf jedem Coordinator `DROP NODE datanodename;` und `SELECT pgxc_pool_reload();` ausführen.

LISTING 7.4: SQL Query zur Ermittlung betroffener DataNodes ausgewählter Relationen

```
\SELECT c.pcrelid FROM pgxc_class c, pgxc_node n WHERE n.node_name
      = 'datanodename' AND n.oid = ANY (c.nodeoids);
```

7.3.1 Aggregation

Der erste Lasttest misst die Laufzeit der Aggregation von Punktdaten aus der Tabelle `nsensorlogs` des Schemas `n`. Die Abfrage und Messung erfolgt mit dem SQL Statement `Select * From n.nsensorlogs Where fileid=%;` sowie `Explain Verbose Select * From n.nsensorlogs Where fileid=%;`. Die erste Query liefert die Daten sowie Laufzeit⁹ und die zweite die Aufteilung des SQL Statements an die Knoten des Clusters. Der Aufruf der ersten Query erfolgt parallel an alle Coordinator und wird für den Lasttest verwendet. Diese zweite Query wird ein mal pro Coordinator abgerufen, um die Verteilung der Daten zu validieren. `%` steht dafür für einen Integer Wert und ist einer der Einträge in `files.soilsamplefiles.id`. Es werden die sechs Werte mit der größten Menge an Einträgen, aufgelistet in Tabelle 7.1, in `n.nsensorlogs` verwendet. Die `fileids` wurden gleichzeitig so gewählt, dass jeder Datensatz einer `fileid` auf einem anderen Knoten vorhanden ist. Somit werden alle Knoten gleichmäßig angesprochen. Der JMeter Lasttest

⁸<http://files.postgres-xl.org/documentation/index.html>

⁹ Sofern in PostgreSQL die Zeiterfassung mit `\timing` eingeschaltet wurde.

fileid	Anzahl Einträge	Betroffener Knoten	Betroffener DataNode
10591	37.926	3	5
9396	36.873	1	1
34791	33.913	6	12
9394	33.391	4	8
10595	28.342	5	9
44982	20.494	2	4

Tabelle 7.1: Anzahl der Einträge in n.sensorlogs nach fileid

ist wie folgt geplant: Pro Coordinator werden drei Threads erzeugt, welche je fünf Wiederholungen folgender Abfragen durchführen: Fünf der sechs fileids¹⁰ werden für die beschriebene Anfrage verwendet und die Ergebnisse in den Arbeitsspeicher geladen.

Vor Ausführung des Tests ist der Query Plan der Aggregation zu untersuchen.

Explain Verbose Select liefert für die fileid 44982 folgenden Query Plan:

LISTING 7.5: Explain Verbose Ergebnis eines Coordinator

```
Remote Subquery Scan on all (datanode4)
-> Index Scan using idx_fileid on n.sensorlogs
    Index Cond: (sensorlogs.fileid = 44982)}
```

Das Ergebnis ist auf allen Coordinatoren gleich.¹¹ Bei unterschiedlichen fileids ändert sich der datanode entsprechend der Tabelle 7.1. Damit kann von einer gleichen Auslastung aller Knoten ausgegangen werden.

Die einzelnen Ergebnisse des ersten Lasttest sind in Tabelle 7.2 enthalten. Wird der erste Durchgang ignoriert und die restlichen zehn Werte gemittelt, lautet das Ergebnis 3619ms durchschnittliche Antwortzeit pro Anfrage.

Die Auslastung der sieben Knoten ist in Abbildung 7.7 und 7.8 dargestellt. Die Auslastung der CPUs der Knoten eins bis sechs schwankt zwischen 0.05 und 0,55 innerhalb des zwanzig minütigen Testlaufes. Die Arbeitsspeicherauslastung dieser Knoten erhöht sich zu Beginn der Durchläufe um etwa 50MB und ändert sich danach nicht mehr messbar. Der siebte Knoten stellt als GTM eine besondere Instanz dar. Bei dieser sind bei der CPU einzig zwei Ausschläge zu verzeichnen, welche bis 0,04 bzw. 0,075 gehen. Die Ursache dieser Spitzen ist nicht feststellbar. Da sie auch vom Betriebssystem stammen können, kann von einer nicht messbaren Auslastung der CPU des GTM enthaltenden Knoten ausgegangen werden. Die Arbeitsspeichernutzung hat sich während der Durchläufe stetig um 180MB erhöht.

¹⁰Jeder Coordinator verwendet andere fünf fileids, sodass alle fünf mal verwendet werden.

¹¹Tabelle wurde mit *vacuum* und *analyze* vorbereitet.

Durchlauf	Durchschnittliche Antwortzeit in ms	Laufzeit des Tests in s
1	3771	101
2	3561	99
3	3472	97
4	3704	103
5	3640	100
6	3644	102
7	3507	97
8	3602	101
9	3813	104
10	3707	102
11	3541	98

Tabelle 7.2: Testergebnis Lasttest der Aggregation

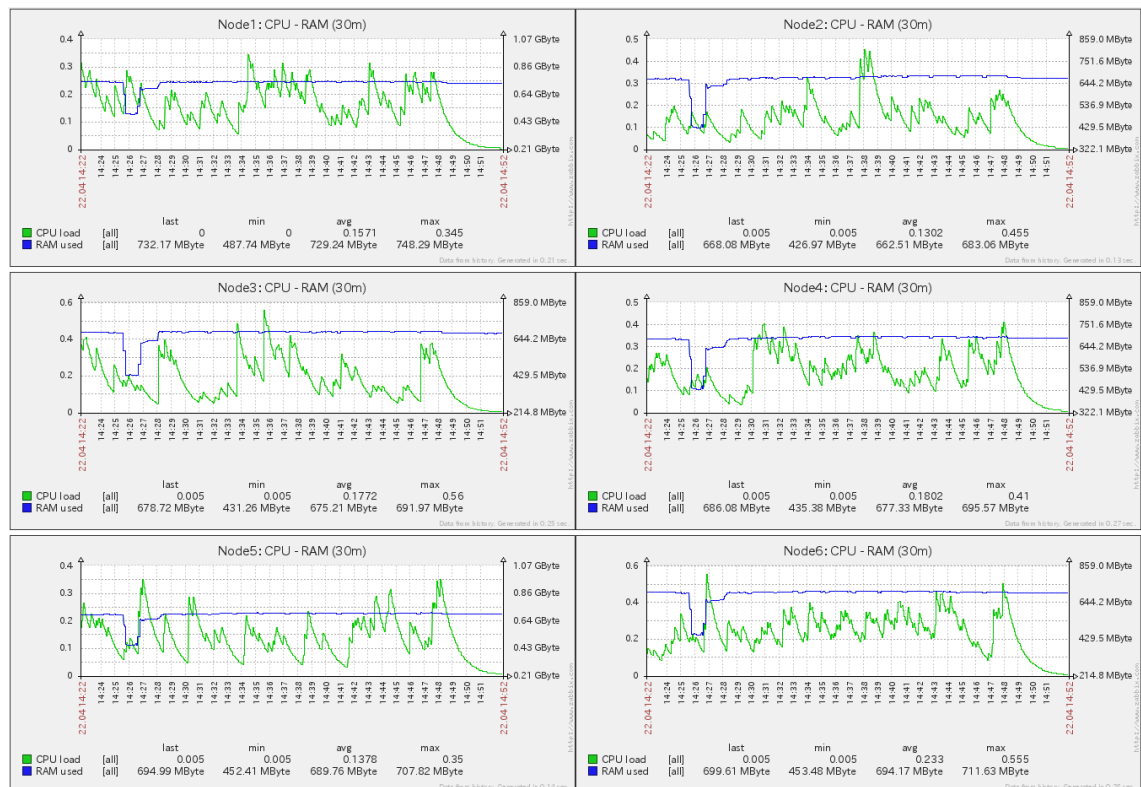


Abbildung 7.7: Auslastung der Knoten 1 bis 6 im Lasttest der Aggregation, 14:28-14:48 Uhr

Zusätzlich erfolgt die Aggregation in einem gesonderten Test entsprechend des Anwendungsfalles mit dem *UMN MapServer*, mit der Darstellung in Kartenform als Ergebnis. Dafür wird der *UMN MapServer* mit einem Mapfile als *FastCGI* Modul verwendet, siehe Anhang A.8. Das Mapfile stellt Kartenausschnitte der Punktdaten dar und die Karte wird farbig anhand der Metadaten erzeugt. Die Testdefinition baut auf jener aus [Jun12] bei Verwendung des *UMN MapServer* auf. Es werden 32 Clients simuliert, indem gleichzeitig 32 Threads in zwei Wiederholungen fünf ausgewählte Kartenbereiche in zufälli-

7 Tests

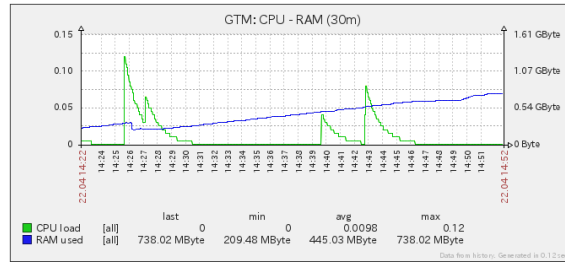


Abbildung 7.8: Auslastung des Knoten 7 im Lasttest der Aggregation, 14:28-14:48 Uhr

ger Reihenfolge per WMS abfragen. Diese Kartenausschnitte enthalten die Einträge aus `n.sensorlogs` entsprechend des angeforderten Kartenausschnittes. Darin wird jeder Wert als Punkt dargestellt und entsprechend des Wertes `applraten` eingefärbt. Diese Darstellung wurde aus AgriPort übernommen, färbt die Punkte weiß bis dunkelblau und stellt die Aufnahmemenge von Stickstoff der Pflanzen dar. Ein Kartenbereich besteht dabei aus fünf Ausschnitten und somit aus fünf WMS Anfragen. Bei der Erstellung der Kartenausschnitte wurden die aufgeführten `fileids` berücksichtigt, um eine gleichmäßige Verteilung auf die `DataNodes` zu erreichen. Die angeforderten Kartenausschnitte werden als Bilder in den Arbeitsspeicher geladen.

In Vorbereitung des Testes wurde eine sehr hohe Auslastung der CPU in der *VM* beobachtet, welche die *UMN MapServer* Instanz beinhaltet, im Gegensatz zu einer geringen Auslastung der restlichen *VMs*. Um einen höheren Durchsatz zu erreichen und das Postgres-XL Cluster mehr auszulasten, wurden der *UMN VM* zwei weitere CPU Kerne zugeordnet. Da bereits alle 16 Kerne vergeben sind, wurde die Prozessor Affinität der *GTM VM* so geändert, dass diese sich ihre zwei Prozessorkerne mit der *UMN VM* teilt. Es wurde dafür die *GTM VM* verwendet, da sie auch bei Benutzung des Postgres-XL Clusters eine geringe CPU Auslastung aufwies¹². Diese Ressourcenaufteilung beeinträchtigt die Leistungsfähigkeit des Clusters somit nicht.

Die Ergebnisse der Testdurchläufe sind in Tabelle 7.3 aufgeführt. Entsprechend der genannten Berechnungsvorschrift ergibt sich eine mittlere Antwortdauer von 2393ms.

Die Auslastung aller *VMs* ist in Abbildung 7.9 sowie 7.10 abgebildet. In den Knoten eins bis sechs ist zu Beginn eine Erhöhung des genutzten Arbeitsspeichers von etwa 100 MB zu verzeichnen. Nach zwei Minuten ändert sich die Auslastung nicht mehr. Die CPU Auslastung schwankt stark, enthält regelmäßige Leistungsspitzen und ist nicht deckungsgleich zwischen den Knoten. Dabei befindet sich die Auslastung je nach Knoten zwischen 0.01 und 0.31. Überblicksartig kann von einer CPU Auslastung 0,1 gesprochen werden. Bezüglich der *GTM VM* steigt auch in diesem Test die Arbeitsspeicher Auslastung stetig

¹²siehe Abbildung 7.8

7 Tests

Durchlauf	Durchschnittliche Antwortzeit in ms	Laufzeit des Tests in s
1	2418	155
2	2389	153
3	2414	152
4	2415	154
5	2402	152
6	2370	153
7	2391	152
8	2404	154
9	2380	153
10	2394	152
11	2375	152

Tabelle 7.3: Testergebnis Lasttest der Aggregation mit Kartendarstellung unter Nutzung des *UMN MapServer*

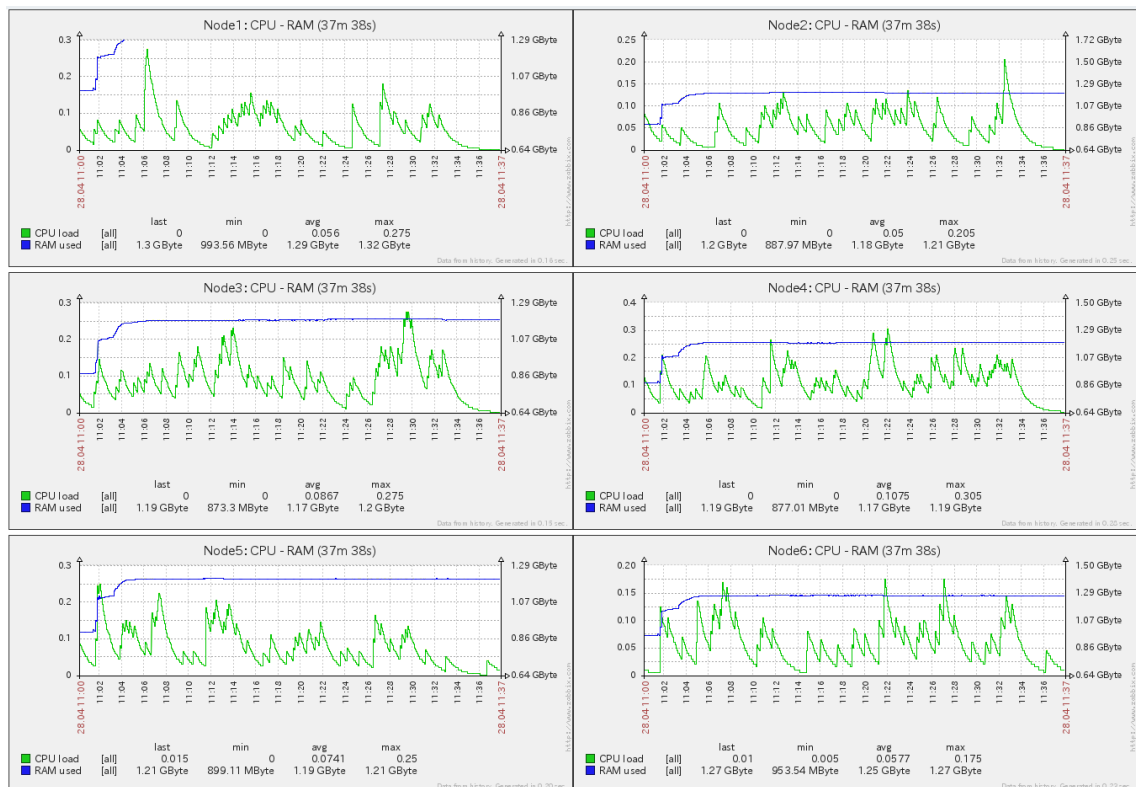


Abbildung 7.9: Auslastung der Knoten 1 bis 6 im Lasttest der Aggregation mit Kartendarstellung, 11:03-11:33 Uhr

und die CPU Auslastung ist sehr gering. Der Arbeitsspeicher wird mit etwa 1GB belastet und die CPU wird einzig von 11:15 bis 11:21 Uhr bis zu einem Maximalwert von 0,06 genutzt. Trotz der Nutzung von vier Prozessoren ist die Auslastung der CPU durch die *UMN MapServer* Instanz sehr hoch. Diese bewegt sich während der Tests zwischen 4.5 und 6.5.

7 Tests

Zu Beginn wird rund 260MB Arbeitsspeicher der *UMN MapServer* Instanz zugewiesen, ist zwischen den Tests 40MB niedriger und ansonsten gleich.

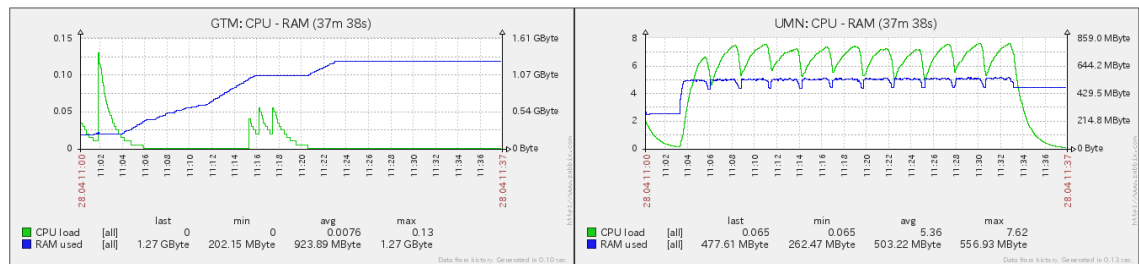


Abbildung 7.10: Auslastung des Knoten 7 im Lasttest der Aggregation mit Kartendarstellung, 11:03-11:33 Uhr

Für die Lasttests mit der PostgreSQL *VM* wurden die Datenbankverbindungen zu den Coordinators auf die IP und den Port der PostgreSQL *VM* geändert. PostgreSQL und Postgres-XL werden somit mit den gleichen Anfragen in der gleichen Menge belastet.

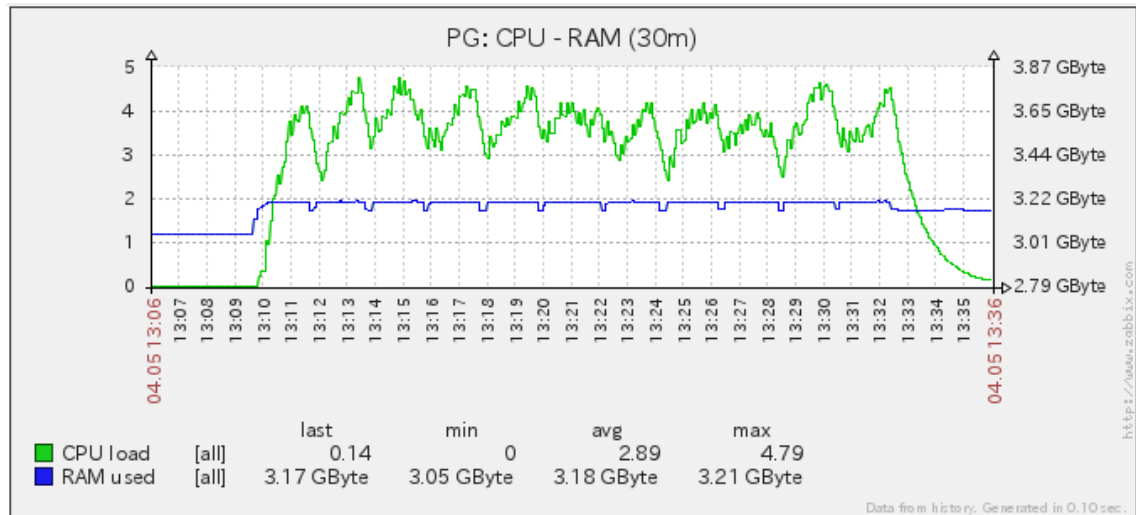
Die Aggregation mit SQL ergab nach Tabelle 7.4 für die PostgreSQL *VM* 4319ms durchschnittliche Laufzeit.

Durchlauf	Durchschnittliche Antwortzeit in ms	Laufzeit des Tests in s
1	4348	113
2	4214	110
3	4215	110
4	4317	112
5	4288	112
6	4631	122
7	4412	116
8	4369	114
9	4363	114
10	4154	109
11	4226	110

Tabelle 7.4: Testergebnis Lasttest der Aggregation mit PostgreSQL *VM*

Die Auslastung der *VM* ist in Abbildung 7.11 abgebildet. Darin ist zu sehen, dass die Auslastung des Arbeitsspeichers um etwa 180MB zu Beginn des Testdurchlaufes zunahm und nur zwischen den Testschritten temporär um 40MB einbrach. Weiterhin ist in dieser Abbildung eine bis zu 460 prozentige Auslastung der CPU zu verzeichnen. Die Auslastung der CPU schwankt zwischen 250% und 460%.

Die Aggregation mit dem *UMN MapServer* ergab für die PostgreSQL *VM* 2595ms durchschnittliche Laufzeit, entsprechend der Tabelle 7.5.

Abbildung 7.11: Auslastung der PostgreSQL *VM* im Lasttest der Aggregation, 13:09-13:32 Uhr

Durchlauf	Durchschnittliche Antwortzeit in ms	Laufzeit des Tests in s
1	2302	151
2	2370	150
3	2377	151
4	2379	150
5	2364	150
6	2388	150
7	2357	150
8	2352	150
9	2352	150
10	2348	150
11	2365	150

Tabelle 7.5: Testergebnis Lasttest der Aggregation mit Kartendarstellung und PostgreSQL *VM*

Die Auslastung der PostgreSQL und UMN *VM* besitzt Ähnlichkeit mit jener der Knoten in Abbildung 7.9 und 7.10. Die UMN *VM* wird bezüglich der CPU um ein Vielfaches der Kapazität belastet. Die Auslastung beträgt 165MB bis 270MB der Arbeitsspeichers und 450% bis 680% der CPU.

7.3.2 Verarbeitung

Der Lasttest zum messen der Verarbeitungsleistung ruft die SQL Funktion *nutrients.contouringcorrectedatop(integer)* auf. Diese führt den speziellen Kriging Algorithmus anhand der übergebenen fileid mit den Werten in *nutrients.samples* durch. Es werden fileids aus der Tabelle 7.6 verwendet. Neben der Laufzeit ist gesondert die Auslastung der einzelnen Knoten zu beobachten und zu bewerten. Zu Beginn dieser Untersuchung

7 Tests

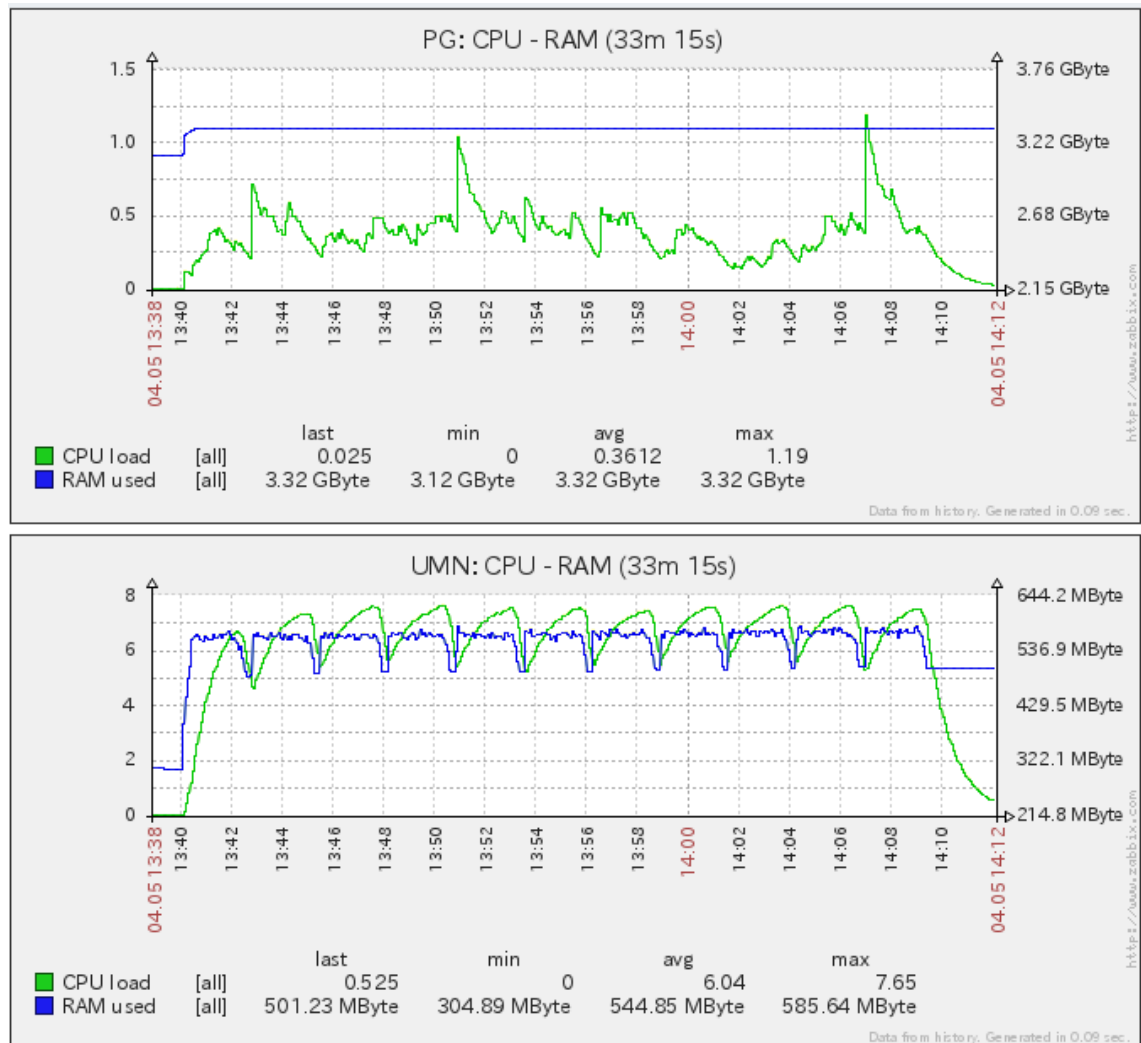


Abbildung 7.12: Auslastung der PostgreSQL *VM* im Lasttest der Aggregation mit Kartendarstellung, 13:40-14:09 Uhr

fileid	Anzahl Einträge	Betroffener Knoten	Betroffener DataNode
4004	494	3	6
3110	404	6	11
3114	369	2	4
4186	160	4	8
5796	143	1	1
5915	110	5	9

Tabelle 7.6: Anzahl der Einträge in nutrients.samples nach fileid

steht nicht fest, in wie weit die DataNodes mit in die Berechnungen der Coordinator mit einbezogen werden. Eine automatisierte Verteilung der Berechnungen ist für einen höheren Durchsatz wünschenswert.

Es zeigte sich, dass mit Postgres-XL zwar Daten per SQL *Insert into* gespeichert werden können, es jedoch bei einer Speicherung innerhalb einer SQL Funktion zu Fehlern kommt. In der Fehlermeldung wird auf unbekannte Parameter verwiesen, obwohl alle Daten in Parametern korrekt sind. Aus diesem Grund wurden Nebenwirkungen der Funktionen der Berechnungen entfernt, sodass zwar alle Berechnungen durchgeführt werden, die Ergebnisse aber nicht verfügbar sind. Weiterhin zeigte sich, dass Funktionen welche die Erweiterung plr für die Nutzung der Sprache R verwenden, zufällige Fehler werfen. Die Wahrscheinlichkeit des Auftretens des Fehlers konnte durch neu laden der verwendeten R Bibliotheken bei jedem Funktionsaufruf verringert werden. So wird `nutrients.contouringcorrectedatop(integer)` ohne Nebenwirkungen ausgeführt.

Die Testdefinition wurde aus dem Leistungstest der Aggregation übernommen. Geändert wurde dabei die SQL Anfrage und die Anzahl der Anfragen an Postgres-XL. Pro Coordinator wird gleichzeitig die Interpolation von sechs fileids aufgerufen. Diese Definition ist jedoch nicht verwendbar, da bei gleichzeitigen parallelen Aufruf Fehler geworfen werden. Dabei verlieren die Coordinator die Verbindung zu den DataNodes. Aus diesem Grund wird ein Coordinator verwendet und die Anfragen erfolgen sequenziell mit den fileids aus Tabelle 7.6 beginnend mit dem ersten Eintrag.

Dieser Lasttest hat eine durchschnittliche Laufzeit von 975s im Postgres-XL Cluster. Tabelle 7.7 enthält die einzelnen Laufzeiten und die maximale und minimale Antwortzeit der Anfragen. Die durchschnittliche Antwortzeit wird nicht aufgeführt, da 6 Anfragen gestellt werden, der Durchschnitt somit leicht berechenbar ist, und sich die Laufzeiten der Berechnungen bis zum 20 fachen unterscheiden. Die Laufzeit der Anfragen ist direkt proportional Abhängig zur Anzahl der betroffenen Einträge in `nutrients.samples`.

Durchlauf	Maximale Antwortzeit in ms	Minimale Antwortzeit in ms	Laufzeit des Tests in s
1	563130	24992	987
2	553914	24948	978
3	552416	25027	970
4	551919	25097	970
5	553892	24919	977
6	552905	25108	971
7	553884	24985	977
8	553931	24843	976
9	554989	24947	979
10	554304	24916	978
11	552536	24915	975

Tabelle 7.7: Testergebnis Lasttest der Verarbeitung

7 Tests

Abbildung 7.13 zeigt eine durchgehend starke Auslastung des ersten Knoten¹³ im Gegensatz zu einer geringen Auslastung der restlichen Knoten. Knoten eins, in Abbildung Node1 genannt, verwendet für die Testreihe 400MB und für einen Testdurchlauf etwa 100MB Arbeitsspeicher. Die Auslastung der CPU bewegt sich während der Testreihe zwischen 33% und 78%. Die restlichen Knoten reservierten während des Testlaufes stetig Arbeitsspeicher, bis zu 100MB pro Knoten. Bezüglich der CPU kann von einer nicht relevanten Auslastung ausgegangen werden. Zwar existieren bei jedem Knoten Leistungsspitzen bis zu 80%, jedoch dauert eine solche Auslastung maximal 5 Minuten und das Auftreten dieser Spitzenwerte ist während der Testreihe unterschiedlich sowie nicht deckungsgleich zwischen den Knoten. Der Knoten sieben zeigt erneut eine charakteristische



Abbildung 7.13: Auslastung der Knoten 1 bis 6 im Lasttest der Verarbeitung, 10:07-13:09 Uhr

geringe Auslastung der CPU sowie einer steigenden Auslastung des Arbeitsspeichers, wie in Abbildung 7.14 zu sehen. So wird etwa 1 GB verwendet, zusätzlich vom Grundwert von 610MB.

Das Testergebnis mit der PostgreSQL *VM* ist 969 Sekunden als durchschnittliche Laufzeit.

¹³Knoten eins wurde im Test angesprochen.

7 Tests

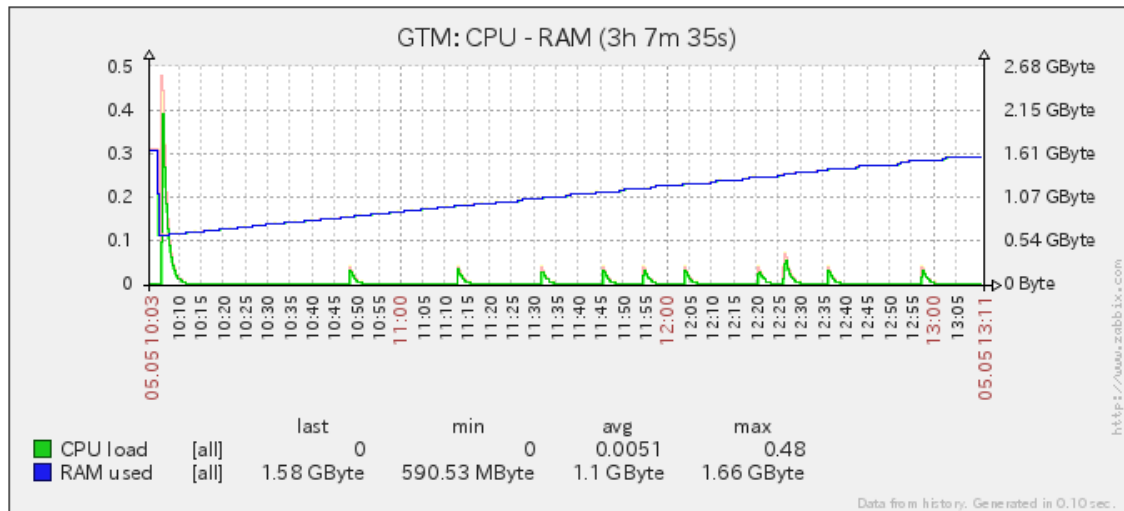
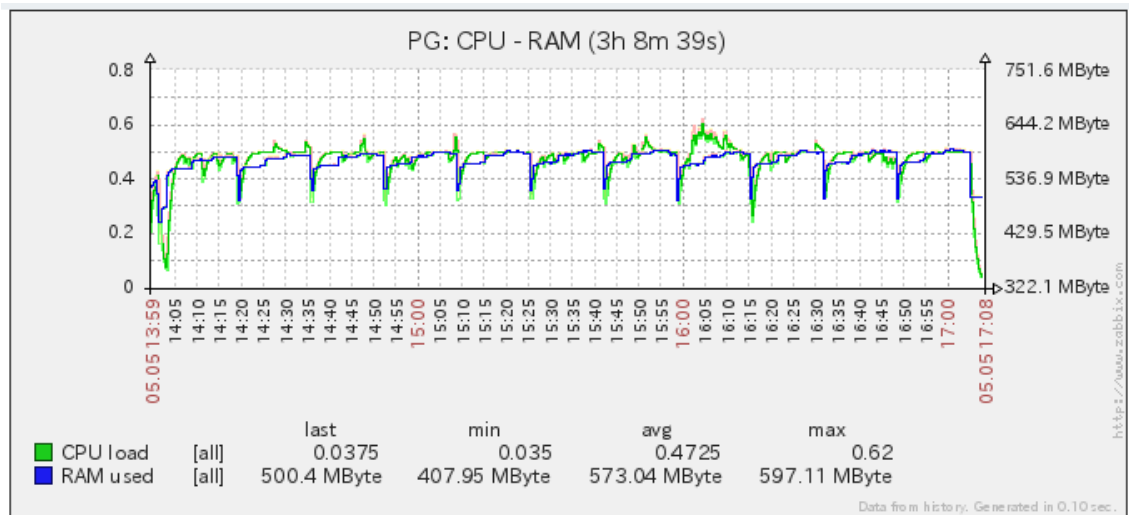


Abbildung 7.14: Auslastung des Knoten 7 im Lasttest der Verarbeitung, 10:07-13:09 Uhr

Durchlauf	Maximale Antwortzeit in ms	Minimale Antwortzeit in ms	Laufzeit des Tests in s
1	551859	23890	966
2	551287	23919	966
3	553798	23897	967
4	553048	23981	968
5	553198	23950	969
6	554308	23957	969
7	553516	23938	968
8	553376	23946	969
9	552566	23947	968
10	553648	24010	970
11	556751	24039	974

Tabelle 7.8: Testergebnis Lasttest der Verarbeitung der PostgreSQL VM

Die Auslastung der VM ist ähnlich, wie in Abbildung 7.15 dargestellt. Dabei liegt die Auslastung der CPU zwischen 30% und 62%. Pro Testdurchlauf wird etwa 150MB Arbeitsspeicher verwendet und bei Ende des Durchlaufes wieder frei gegeben.

Abbildung 7.15: Auslastung der PostgreSQL *VM* im Lasttest der Verarbeitung, 10:07-13:09 Uhr

7.3.3 Zusammenfassung

Nach der Definition der Leistungsmessung und der Testszenarien erfolgte die Durchführung dieser mit anschließender Wiedergabe der Messergebnisse. Als Testsystem kam ein Postgres-XL Cluster mit zwölf DataNodes, sechs Coordinator, sechs GTM Proxys und einem GTM sowie eine PostgreSQL Installation zum Einsatz. Diese zwei Systeme werden gegenüber gestellt.

Tabelle 7.9 stellt die durchschnittlichen Antwortzeiten der beiden Systeme für die Aggregation mit SQL und die Aggregation mit dem *UMN MapServer* gegenüber. Bezüglich der Aggregation mit SQL Abfragen besteht ein Unterschied von 0,7 Sekunden. Ebenso ist die Antwortzeit bei der Aggregation über den *UMN MapServer* mit Postgres-XL um 0,2 Sekunden kürzer. Postgres-XL besitzt somit bei sechsfacher Hardwareausstattung 16% bzw. 8% mehr Leistung bezüglich der Aggregation. Die Auslastung des Arbeitsspeichers

Durchschnittliche Antwortzeit	Aggregation	Aggregation mit Kartendarstellung
Postgres-XL	3,6s	2,4s
PostgreSQL	4,3s	2,6s

Tabelle 7.9: Vergleich der Testergebnisse

beträgt auf beiden System bei allen Testdurchläufen maximal 180MB pro *VM*, was 3% des verfügbaren Arbeitsspeichers darstellt. Diese geringe Auslastung ist positiv zu bewerten und wird nicht als Vergleichskriterium verwendet. Die Auslastung der CPU ist in Tabelle 7.10 aufgeführt. Entsprechend den abgebildeten Graphen wurden die Werte der

Mittlere CPU Auslastung	Aggregation	Aggregation mit Kartendarstellung	Verarbeitung
Postgres-XL	0,2	0,1	0,5
PostgreSQL	3,6	0,4	0,5

Tabelle 7.10: Vergleich der CPU Auslastung, GTM *VM* nicht berücksichtigt

CPU Auslastung gemittelt. Die Mittlung fand je Graph statt. Bei Postgres-XL wurden aus sechs gemittelten Werte ein Durchschnittswert erzeugt.

Die Verarbeitungsleistung konnte bei Postgres-XL nur mit einem Coordinator gemessen werden. Aus diesem Grund ist unterscheidet sich die Laufzeit eines Testlaufes nicht relevant zwischen den Systemen. 975 Sekunden zu 969 Sekunden beträgt 99,5% zu 100% Leistung, entsprechend Tabelle 7.11.

	Durchschnittliche Laufzeit
Postgres-XL	975s
PostgreSQL	969s

Tabelle 7.11: Vergleich der Testergebnisse der Verarbeitung

Die erwartete Steigerung der Leistung gegenüber PostgreSQL konnte nicht bewiesen werden. Zwar ist die Laufzeit bei Nutzung aller Coordinator höher als bei Verwendung einer PostgreSQL Instanz, aber nicht so hoch, dass sich ein sechsfacher Hardwareaufwand lohnt.

8 Fazit

In diesem letzten Kapitel wird das ausgewählte Framework Postgres-XL erneut einer Nutzwertanalyse unterzogen, da die vorhergehenden Kapitel neue Erkenntnisse hervorbrachten. Im Anschluss wird diese Arbeit zusammengefasst und die Ergebnisse dieser für Agri Con gewertet. Diese Arbeit und dieses Kapitel enden mit einem Abschnitt zum Ausblick, in welchem die zukünftige Nutzung der gewonnenen Erkenntnisse allgemein und bei Agri Con erläutert wird.

8.1 Nutzwertanalyse

Aufbauend auf die Definition der Nutzwertanalyse in Abschnitt 5.1 wird diese erweitert, um die Ergebnisse aus den Tests in Kapitel 7 zu berücksichtigen.

Metrik	Gewichtung in %
Interoperabilität	20
Funktionsumfang	20
Dokumentation	15
Zeitverhalten	40
Modifizierbarkeit	5

Tabelle 8.1: Neuer Wertungsmaßstab der einzelnen Metriken

Nach Erfassung der Testergebnisse der Leistung, wird in diesem Schritt das Zeitverhalten zusätzlich in einer erneuten Nutzwertanalyse bewertet. Die Interoperabilität wird weiterhin mit zwölf Punkten bzw. 100% Erfüllung bewertet, da die Ergebnisse der Funktionstests beweisen, dass beide Schnittstellen verwendet werden können.

Die Erkenntnisse, welche durch die Durchführung der Funktions- und Leistungstests entstanden sind, bedingen eine Änderung der Punkte für den Funktionsumfang. Verschneidungsfunktionen sind wie in FT05 gezeigt vorhanden und einsetzbar, weshalb der Wert dafür auf vier erhöht wird. Vermindert wird dagegen der Wert der Parallelität auf eins, da Funktionen des Lasttestes der Verarbeitung nicht gleichzeitig genutzt werden konnten. Dies ergibt einen Wert von 53.

Der Wert der Dokumentation wird um eins auf acht vermindert, da fehlender Funktionsumfang schlecht oder gar nicht dokumentiert ist. Dies ist aber eine wesentliche Information für die Auswahl und Verwendung eines Frameworks. Funktionsumfang in Dokumentation wird mit eins bewertet.

Das Zeitverhalten wurde mit den Lasttests in Kapitel 7.3 ermittelt. Die Laufzeit der Aggregation mit Postgres-XL liegt unter der mit PostgreSQL. Dagegen ist die Verarbeitungsleistung gleich. Entsprechend der Bewertungsfunktion auf Seite 29 ergibt das die Bewertung drei und eins. Das Zeitverhalten wird mit zwei gewertet.

Hinsichtlich der Modifizierbarkeit gab es keine neuen Erkenntnisse, somit ändert sich dessen Bewertung nicht.

Metrik	erreichter Wert	Erfüllung in %	Kommentar	gewichteter Teilnutzen
Interoperabilität	12	100	Analog des Ist-Standes.	20
Funktionsumfang	53	87	Mindestabdeckung erfüllt, jedoch sind Geostatistik und Versionierung nicht vorhanden. Außerdem keine parallele Nutzung einer Funktion.	17
Dokumentation	8	69	Dokumentation zu PostGIS ist sehr gut, zu Postgres-XL grob mit Mängeln bei fehlenden Funktionalitäten. Mindestabdeckung ist erfüllt.	10
Zeitverhalten	2	67	Bei Nutzung aller Coordinator besseres Zeitverhalten als PostgreSQL, jedoch bestehen hohe Kosten in der Hardwareanschaffung.	27
Modifizierbarkeit	5	100	Vollständige Abdeckung vorhanden. Möglichkeiten sind in SQL gegeben.	5

Tabelle 8.2: Neue Nutzwertanalyse von Postgres-XL

Entsprechend Tabelle 8.2 ergibt sich ein Nutzwert von 79.

8.2 Zusammenfassung

Die Untersuchung quelloffener verteilter geografischer Informationssysteme zur Verarbeitung agrartechnischer Kennzahlen am Beispiel des aktuellen Standes bei Agri Con GmbH wird in diesem Abschnitt zusammengefasst.

Diese Arbeit begann mit den Grundlagen, welche für das Verständnis der darauf folgenden Ausführungen und das angewandte Vorgehen notwendig zu klären sind. Dazu zählen Begriffe zu *DBMS*, räumliche Datenverarbeitung und Alternativen zum relationalen Datenbankmodell. Darin wurden im dritten Abschnitt Frameworks vorgestellt, welche in den anderen Kapiteln relevant sind. Dazu zählen Postgres-XL, Rasdaman und GeoMesa.

Nach Sicherstellung der theoretischen Grundlagen folgte in Kapitel 3 die Darlegung und Begründung der Methodik dieser Arbeit. Diese Darlegung fand anhand einer Unterteilung des Themas in vier Unteraufgaben statt. Dabei waren die Vorgehen zur Softwareauswahl und Leistungsbestimmung Schwerpunkte. Es wurde die Nutzwertanalyse und Funktions- sowie Leistungstests als geeignete Mittel heraus gearbeitet.

Kapitel 4 legte dar, worauf und wie diese Methodik angewandt wird. Das heißt, dass der Anwendungsfall mit Anforderungen festgelegt wurde. Die Anforderungen wurden wissenschaftlich in Form von Softwarequalität festgehalten und mit Qualitätsmetriken messbar gemacht. Die Softwarequalität wurde umfassend beschrieben und für die Untersuchung relevante Kriterien heraus gearbeitet. Außerdem wurden Funktions- und Lasttests für den Anwendungsfall skizziert. Das Kapitel endet mit einer Übersicht über relevante Literatur zum Thema dieser Arbeit. Darin wird deutlich, dass keine thematisch vergleichbare Arbeit existiert und Teilprobleme in anderen Arbeiten zu finden sind.

Die erste Hälfte der Aufgabenstellung wird im Kapitel 5 gelöst. Nach Definition der Nutzwertanalyse wurden Frameworks an Hand ihrer Spezifikation und der Nutzwertanalyse bewertet und Postgres-XL mit einem Nutzwert von 86 als geeignet ausgewählt. Die Frameworks GeoMesa und Rasdaman erhielten dabei die Wertung 56 bzw. 51.

Postgres-XL erfuhr im darauf folgenden Kapitel eine Untersuchung hinsichtlich der allgemeinen Verwendung und der Möglichkeiten für den Einsatz bei Agri Con. So wurde das Vorgehen der Installation, die Nutzung der Schnittstellen und die Möglichkeiten der Verarbeitung erörtert, wobei die Schnittstellen und die Verwendung analog zu den Schnittstellen und der Verwendung von PostgreSQL sind. Der Einsatz bei Agri Con wurde heraus gearbeitet und eine tief greifende Integration in den Ist-Stand als notwendig ermittelt. Doch auf Grund von fehlender Funktionalität ist die Integration von Postgres-XL im Rahmen dieser Arbeit nicht möglich. Für das weitere Vorgehen wurde die Anpassung

des Ist-Standes an Postgres-XL für die Untersuchung mit Funktions- und Leistungstests beschrieben.

Kapitel 7 enthält die Definition der Testumgebung sowie die Definition und die Ergebnisse der Funktions- und Leistungstests. Als Testumgebung diente ein IBM Server, auf welchem mit virtualisierten Maschinen Postgres-XL und PostgreSQL installiert und miteinander vergleichbar konfiguriert wurden. Die Funktionstests validierten die Funktionalität von Postgres-XL, wobei einzig der Test FT06 fehlschlug, da Speicher Befehle in SQL Funktionen Fehler verursachten und mit R berechneten Werte nicht validiert werden konnten. Die Leistungsfähigkeit bezüglich der Aggregation und Verarbeitung von Daten wurde mit den Leistungstests ermittelt. Diese Ermittlung fand mit Postgres-XL und PostgreSQL statt, um relative Aussagen treffen zu können. In der Präambel fand eine Auseinandersetzung mit den Begriffen Leistung, Lastmessung, Auswertung, Datenverteilung und Verbesserungen in Postgres-XL, der Lastverteilung und der Skalierung statt. Eine Zusammenfassung der Testergebnisse ergab geringere Laufzeiten von bis zu 16% bezüglich der Aggregation mit Postgres-XL. Die Verarbeitungsleistung unterscheidet sich um weniger als 1%. Diese Steigerung ging mit sechsfachen Hardwareaufwand einher.

8.3 Wertung

Die Definition der Anforderungen kann für zukünftige Validierungen des Ist-Standes und zu analysierender Frameworks verwendet werden. Lücken der Funktionalität in Postgres-XL bedingen Ergänzungen der Anforderungen bezüglich allgemeiner Funktionen von *DBMS* zur umfassenden Bewertung eines Frameworks. So ist Ordnungsmäßigkeit gesondert zu bewerten. Die Berücksichtigung dieses Qualitätskriteriums hat eine Verminderung des Nutzwertes von Postgres-XL zur Folge. Diese Berücksichtigung allgemeiner Qualitäten muss für eine umfassende und allgemeine Untersuchung eines solchen Frameworks verwendet werden. Obwohl Postgres-XL nicht empfohlen werden kann, macht der hohe Nutzwert folgendes deutlich: Die Nutzwertanalyse ist eine Abstraktion der Bewertung von Softwarequalität. Dabei wird eine festgelegte Menge von Qualitäten untersucht, der Umfang der Untersuchung ist somit eingegrenzt. Außerdem führen fehlende essentielle Qualitätskriterien nicht zu einer null Wertung des untersuchten Systems. Der hohe Wert spricht nichts desto trotz für eine stetige Berücksichtigung von Postgres-XL in solchen Untersuchungen, da es sich wegen des Umfanges und den Möglichkeiten um ein vielversprechendes Framework handelt.

Die in der Literaturrecherche aufgedeckte Lücke an wissenschaftlichen Dokumenten zu diesem Thema ist negativ zu bewerten. Diese Arbeit schließt diese Lücke nicht, da die Untersuchung für ein Anwendungsszenario und nicht allgemein stattfand, da als Grundlage zur Softwareauswahl eine unbestätigte Liste an relevanten Frameworks verwendet wurde und da die Leistungstests die Skalierbarkeit nicht ausreichend bestimmten, um die Skalierbarkeit für andere Knotenzahlen zu schätzen.

Die Übersicht und Bewertung der wichtigsten verteilten *GIS* ist trotz fehlender Validierung von allgemeinem Interesse und zukünftig zu aktualisieren.

Die Softwareauswahl und -bewertung erfolgte nachvollziehbar, weshalb diese Arbeit als Handlungsempfehlung für ähnliche Anwendungsszenarien verwendet werden kann. Wird die Nutzwertanalyse um Ordnungsmäßigkeit erweitert, kann bei der Bewertung die Durchführung der Funktionstests entfallen.

Die durch eine genauere Untersuchung aufgedeckten Lücken in der Funktionalität von Postgres-XL sprechen gegen einen produktiven Einsatz des Frameworks im zu Grunde gelegten Anwendungsfall. Es ist in der verwendeten Version 9.2.34 nicht ausgereift. Darin werden keine Trigger und Sub-Transaktionen unterstützt, der Transaktionstyp internal ist in Prozeduren nicht verwendbar, Sequenzen unterscheiden sich auch bei Verteilung einer Tabelle per Replikation und die Verwendung von R Funktionen ist problematisch. Lagert man die virtuellen Maschinen auf eigenständige physische Maschinen aus, muss mit zu berücksichtigenden Kosten für den Netzwerkverkehr und höherer Lese - und Schreibgeschwindigkeit des Festspeichers pro Knoten gerechnet werden. In beiden Fällen ist der Leistungsgewinn durch Einsatz eines Clusters zu gering, als das sich ein sechsfacher Hardwareaufwand lohnt. Ein Nutzen ergibt sich mit Postgres-XL, wenn Daten in wenigen Relationen verteilt gespeichert und gelesen werden müssen. Besonders bei Datenmengen, welche die Größe von Festplatten übersteigen. Bei Agri Con wäre der Bereich Docu dafür geeignet. So könnten alle Positionsdaten der Maschinen aller Betriebe unveränderlich abgelegt und mit kurzen Laufzeiten aggregiert werden.

Für Agri Con ist das Ergebnis, dass der Ist-Stand mit der eingesetzten Technologie für diese umfangreichen Anforderungen am besten geeignet ist, sofern kostenlose Frameworks berücksichtigt werden. Änderungen am Datenbankschema, an den Kostenwerten der Funktionen, an der Konfiguration des Query Planers und den Datenbank nahen Anwendungen erhöhen die Leistungsfähigkeit der PostgreSQL Installation um die gewünschte Verminderungen im Laufzeitverhalten zu erwirken. Der Einsatz von anderen Frameworks ist dagegen für Teilaufgaben sinnvoll. Beispielsweise würde sich Rasdaman bei wesentlicher Nutzung von Rasterdaten, welche unabhängig vom Datenbankschema sind, zur Speicherung, Verarbeitung und Bereitstellung eignen.

8.4 Ausblick

Agri Con wird Änderung an der PostgreSQL Installation vornehmen, um die gewünschten Effekte zu erzielen. Die in dieser Arbeit durchgeführte Untersuchung wird mittelfristig nicht erneut durchgeführt. Bei Änderung des Szenarios werden Untersuchungen spezieller Frameworks für diese Änderungen erhoben.

Werden sich die Anforderungen bezüglich des Zeitverhaltens erhöhen, ist momentan die Erstellung eines Clusters mit PostgreSQL Instanzen das gewünschte Vorgehen. Dabei wird das Datenbankschema aufgeteilt und in zwei PostgreSQL Instanzen integriert. Es wird von einer annähernden Verdopplung der zusammengefassten Leistungsfähigkeit ausgegangen. Neben der Veränderung des Schemas sind umfangreiche Änderungen in Programmen der Agri Con durchzuführen.

Weiter- und Neuentwicklungen bedingen eine ständige Aktualisierung der Übersicht verteilter GIS. Zwar ist langfristig mit einem zu PostgreSQL mit PostGIS konkurrierenden Framework zu rechnen, aber das Forschungsziel der Bewertung solcher Frameworks ist von stetigem Interesse.

A Anhang

A.1 GeoTools Funktionalitäten

Übersicht

[Geo15] listet die wichtigsten Funktionalitäten wie folgt auf:

- *A clean data access API supporting feature access, transaction support and locking between threads*
 - *Access GIS data in many file formats and spatial databases*
 - *Coordinate reference system and transformation support*
 - *Work with an extensive range of map projections*
 - *filter and analyze data in terms of spatial and non-spatial attributes*
- *A stateless, low memory renderer, particularly useful in server-side environments.*
 - *compose and display maps with complex styling*
 - *vendor extensions for fine control of text labels and color blending*
- *Powerful schema assisted parsing technology using XML Schema to bind to GML content*

The parsing / encoding technology is provided with bindings for many OGC standards including GML, Filter, KML, SLD, and SE.

Unterstützte Formate sind nach der selben Quelle:

- *raster formats and data access*

arcscde, arcgrid, geotiff, grassraster, gtopo30, image (JPEG, TIFF, GIF, PNG), imageio-ext-gdal, imagemosaic, imagepyramid, JP2K, matlab
- *Database “jdbc-ng” support*

db2, h2, mysql, oracle, postgis, spatialite, sqlserver

A Anhang

- *Vector formats and data access*
app-schema, arcsde, csv, dxf, edigeo, excel, geojson, org, property, shapefile, wfs
- *XML Bindings*
Java data structures and bindings provided for the following: xsd-core (xml simple types), fes, filter, gml2, gml3, kml, ows, sld, wcs, wfs, wms, wps, vpf. Additional Geometry, Filter and Style parser/encoders available for DOM and SAX applications.

A.2 Systembewertung

A.2.1 GeoMesa

Interoperabilität

PostgreSQL - 7 Scala kann mit JDBC auf PostgreSQL zugreifen.

UMN MapServer - 0 *UMN MapServer* bietet Accumulo nicht als Quelle an und GeoMesa besitzt keine *OGC* konformen Dienste wie beispielsweise WMS.

Die Wertung für Interoperabilität ist somit 7 mit einer Erfüllung von 58%.

Funktionsumfang

Parallele Verarbeitung - 2 Verteilte Datenhaltung durch Accumulo auf *HDFS* und verteiltes sowie paralleles Rechnen mit beispielsweise Spark möglich. [Fox14]

Geografische Datentypen - 12 Vollständige Datentypen aus Simple Feature Access vorhanden. [Fox14]

Umrechnungsfunktionen - 10 Datenverarbeitung direkt in Spark mit GeoTools möglich. [OSG15a]

Gruppierungsfunktionen - 7 Funktionale Verarbeitung mit Scala immanent.

Verschneidungsfunktionen - 3 *JTS* stellt difference, union und symmetric difference zur Verfügung. [Viv, S.29 ff.]

Overlayfunktionen - 2 *JTS* stellt relate und overlay zur Verfügung.

Geostatistik - 0 Keine eingebaute Funktionalität.

Filterfunktionen - 10 Räumliche Filterung ist mit GeoTools möglich [Geo15]

Schemaversionierung - 0 Accumulo erlaubt entsprechend des BigTable Ansatzes ein dynamisches Datenbankschema, jedoch ohne Versionierung. Einzig erzeugte Datentypen, bestehend aus Simple Features, können in GeoMesa als Konstrukt persistiert werden.

Daraus ergibt sich ein Wert von 48, was 79% des maximal zu erreichenden Wertes ist.

Dokumentation

Installation - 1 Knappe Hinweise für GeoMesa auf [Eic14] vorhanden, dagegen ausführliche Anleitungen für Accumulo auf [Sof14b].

Zeitverhalten - 0 Keine Dokumentation vorhanden.

Funktionsumfang - 1 Konkrete Funktionalität von GeoMesa nur grob auf [Loc14b] angedeutet. MapReduce mit Accumulo ist ausführlich beschrieben. [Sof14b]

Interoperabilität - 1 Nicht explizit bei GeoMesa angegeben, aber Anbindungsmöglichkeiten mit Scala bzw. Java sind im allgemeinen ausführlich dokumentiert.

Best Practise - 0 Keine Dokumentation vorhanden.

Anpassbarkeit - 1 In [Loc14b] sind einige Anregungen zu finden. Beispielsweise die Erzeugung eigener Schemabestandteile. [Loc14c]

Das Qualitätskriterium Dokumentation wird für GeoMesa mit dem Wert 4, bzw. der Erfüllung von 31%, belegt.

Modifizierbarkeit

Verwendung eigener Datentypen - 0 Es sind eigene Schemas aber keine Datentypen erstellbar. [Loc14c]

Erstellung eigener Schnittstellen - 1 Indirekt über JDBC und ODBC möglich.

Erstellung eigener Funktionen - 1 Durch verschiedenste Frameworks zur Datenverarbeitung wie Spark beliebige Funktionen erstellbar.

Verwendung der Programmiersprachen Scala oder R - 1 GeoMesa ist in Scala geschrieben und kann mit dieser verwendet werden. R kann über das Tool SparkR und beim Einsatz von Hadoop über RHadoop verwendet werden.

Anlegen eigener Berechnungsvorgängen zur späteren Abarbeitung - 1 Mit einer Vielzahl von Tools möglich, bspw. Spark, *Storm*, Pig und *Cascading*.

Hier ist die Wertung 4 von 5 Punkten und damit 80%.

A.2.2 Postgres-XL

Interoperabilität

PostgreSQL - 7 PostgreSQL ist Bestandteil von Postgres-XL wobei die Datentypen vollständig verfügbar sind.

UMN MapServer - 5 Mit der Erweiterung PostGIS direkt als Quelle für *UMN MapServer* angebbbar. [Min14]

Die Wertung für Interoperabilität ist somit 12 mit einer Erfüllung von 100%.

Funktionsumfang

Parallele Verarbeitung - 2 Verteilte Datenhaltung mit partitioning der Daten und verschränkte parallele Datenverarbeitung mit *MPP* möglich. [Tra15a]

Geografische Datentypen - 14 Vollständige Datentypen aus Simple Feature Access sowie PostGIS raster vorhanden. [OSG15b]

Umrechnungsfunktionen - 10 Direkter Funktionsaufruf zur Umrechnung von und in beliebige EPSG Codes. [OSGa]

Gruppierungsfunktionen - 10 SQL in PostgreSQL mit der Erweiterung PostGIS erlaubt beliebige Querys mit geografischen Daten. [OSG15b]

Verschneidungsfunktionen - 3 Funktionsübersicht zeigt intersection, difference und symmetric difference. [OSGb]

Overlayfunktionen - 2 Funktionsübersicht zeigt relation und intersects. [OSGb]

Geostatistik - 2 Interpolation nur von Linie zu Punkt mit PostGIS möglich. Jedoch kann mit R oder C++ beliebige Geostatistik mit vorhandenen und eigenen Funktionen durchgeführt werden.

Filterfunktionen - 10 In SQL mit mehreren Funktionen möglich. [OSGb]

Schemaversionierung - 0 Nicht eingebaut. Mit eigenen Skripten nachrüstbar.

Daraus ergibt sich ein Wert von 53, was 87% des maximal zu erreichenden Wertes ist.

Dokumentation

Installation - 1 Knapp auf [\[Trab\]](#) beschrieben.

Zeitverhalten - 0 Keine Angaben.

Funktionsumfang - 2 Es existiert eine Übersicht zur Verwaltung eines Postgres-XL Clusters. Dazu ist die allgemeine Dokumentation zu PostgreSQL und PostGIS verfügbar. [\[Trac\]](#)

Interoperabilität - 2 Verweis auf Dokumentation von PostgreSQL und PostGIS sowie API auf [\[Traa\]](#) vorhanden.

Best Practise - 1 Einige Hinweise auf [\[Trac\]](#) vorhanden.

Anpassbarkeit - 3 [\[Trad\]](#) dokumentiert Erweiterung mit SQL, tcl, Perl und Python.

Das Qualitätskriterium Dokumentation wird für Postgres-XL mit dem Wert 9, bzw. der Erfüllung von 69%, belegt.

Modifizierbarkeit

Verwendung eigener Datentypen - 1 Mit PostgreSQL eigene Datentypen erstellbar.

Erstellung eigener Schnittstellen - 1 Für eigene Programme mit JDBC oder ODBC Daten verwendbar.

Erstellung eigener Funktionen - 1 Ebenso mit SQL möglich.

Verwendung der Programmiersprachen Scala oder R - 1 R kann direkt in SQL Funktionen eingebettet werden. Scala ist mit JDBC verwendbar.

Anlegen eigener Berechnungsvorgängen zur späteren Abarbeitung - 1 Hier sind Trigger und selbstständige Programme mit JDBC Nutzung zu nennen.

Hier ist die Wertung 5 von 5 Punkten und damit 100%.

A.2.3 Rasdaman

Interoperabilität

PostgreSQL - 7 PostgreSQL wird nach [Ras14a] unterstützt.

UMN MapServer - 0 Rasdaman bietet einzig *Web Coverage Service* und *Web Processing Service* Dienste an, diese können jedoch vom *UMN MapServer* der aktuellen Version 6.4 nicht verwendet werden.

Die Wertung für Interoperabilität ist somit 7 mit einer Erfüllung von 58%.

Funktionsumfang

Parallele Verarbeitung - 1 Parallele Server Instanzen verwendbar. In der kostenlosen Version keine Query Optimierung für mehrere Kerne und Instanzen vorhanden. [Ras14a]

Geografische Datentypen - 2 Raster und Punkte sind für die räumliche Datenverarbeitung vorhanden. Dazu sind Arrays mit beliebig vielen Dimensionen verwendbar. [Ras14c]

Umrechnungsfunktionen - 0 Nur mit externer Bibliothek *Geospatial Data Abstraction Library (GDAL)* für zwei-dimensionale Arrays möglich. [Ear14]

Gruppierungsfunktionen - 0 Laut Dokumentation der Funktionen keine Gruppierung möglich. [Ras15a]

Verschneidungsfunktionen - 1 Dagegen sind einfache Array Operationen vorhanden.

Overlayfunktionen - 1 Ditto.

Geostatistik - 0 Keine eingebaute Funktionalität.

Filterfunktionen - 5 Operationen für Array-Verarbeitung vorhanden.

Schemaversionierung - 0 Keine eingebaute Funktionalität.

Daraus ergibt sich ein Wert von 10, was 16% des maximal zu erreichenden Wertes ist.

Dokumentation

Installation - 1 [Ras15b] ist eigenes Installationsdokument.

Zeitverhalten - 0 Keine Dokumentation vorhanden.

Funktionsumfang - 2 Ist grob unter [Ras14a] beschrieben und detailliert in [Ras15a] aufgeführt.

Interoperabilität - 3 Interoperabilität mit PostgreSQL und API unter [Ras15a] verfügbar.

Best Practise - 1 Einzig Hinweise verfügbar. [Ras14b]

Anpassbarkeit - 1 Kein eigenständiges Dokument vorhanden, erschließt sich aber aus genannten Quellen.

Das Qualitätskriterium Dokumentation wird für Postgres-XL mit dem Wert 8, bzw. der Erfüllung von 62%, belegt.

Modifizierbarkeit

Verwendung eigener Datentypen - 0 Keine eigenen Datentypen erstellbar. Einzig die Verwendung von selbst definierten Arrays ist verfügbar.

Erstellung eigener Schnittstellen - 1 Über *JDBC/Open Database Connectivity (ODBC)* in Java und C++ möglich.

Erstellung eigener Funktionen - 1 In der Abfragesprache rasql nicht möglich, dagegen mit API.

Verwendung der Programmiersprachen Scala oder R - 1 Scala mit API verwendbar.

Anlegen eigener Berechnungsvorgängen zur späteren Abarbeitung - 0 Nicht vorgesehen.

Hier ist die Wertung 3 von 5 Punkten und damit 60%.

A.3 Postgres-XL Installationsskript

LISTING A.1: Postgres-XL installation

```
sudo apt-get install bison flex libreadline6 libreadline6-dev \  
libedit-dev make zlib1g libghc-zlib-dev jade  
./configure  
make  
sudo make install  
PATH=$PATH:/usr/local/pgsql/bin  
export PATH
```

A.4 Pl/R Installationsskript

LISTING A.2: Pl/R installation

```
#install R:
#add source: deb http://ftp5.gwdg.de/pub/misc/cran/bin/linux/
    ubuntu \
    trusty/
sudo apt-get update
sudo apt-get install r-base

#install Pl/R:
#copy plr src to postgres-xl/src/contrib
#cd there
make
sudo make install
psql -h localhost template1
create extension plr;
```

A.5 PostGIS Installationsskript

LISTING A.3: PostGIS installation

```
tar -xvzf postgis-2.1.5.tar.gz
cd postgis-2.1.5/
sudo apt-get install libproj0 libproj-dev libgeos-dev libxml2 \
    libxml2-dev libgdal-dev
./configure --with-pgconfig=/usr/local/pgsql/bin/pg_config
make
sudo make install
psql -h localhost --port=6661 -d agrodb -f \
    /usr/local/pgsql/share/extension/postgis--2.1.5.sql
psql -h localhost --port=6661 -d agrodb
alter table spatial_ref_sys distribute by replication;
\q
```

A.6 pgxcctl Konfigurationsskript

LISTING A.4: Konfigurationsdatei pgxc-ctl

```
#!/usr/bin/env bash
pgxcInstallDir=$HOME/pgxc-ctl
#----- OVERALL -----
pgxcOwner=$USER
pgxcUser=$pgxcOwner
tmpDir=/tmp
localTmpDir=$tmpDir
configBackup=n
configBackupHost=pgxc-linker
configBackupDir=$HOME/pgxc-backup
configBackupFile=pgxc_ctl.bak
#----- GIM -----
#----- GIM Master -----
#----- Overall -----
gtmName=gtm
gtmMasterServer=node0
gtmMasterPort=20001
gtmMasterDir=$HOME/data_gtm
#----- Configuration -----
gtmExtraConfig="log_min_messages = INFO"
gtmMasterSpecificExtraConfig=none
#----- GIM Slave -----
#----- Overall -----
gtmSlave=n
gtmSlaveName=gtmSlave
gtmSlaveServer=node12
gtmSlavePort=20001
gtmSlaveDir=$HOME/pgxc/nodes/gtm
#----- Configuration -----
gtmSlaveSpecificExtraConfig=none
#----- GIM Proxy -----
#----- Shortcuts -----
gtmProxyDir=$HOME/data_gtm_proxy
#----- Overall -----
gtmProxy=y
gtmProxyNames=(gtm_pxy1 gtm_pxy2 gtm_pxy3 gtm_pxy4 gtm_pxy5 gtm_pxy6)
gtmProxyServers=(node1 node2 node3 node4 node5 node6)
gtmProxyPorts=(20001 20001 20001 20001 20001 20001)
gtmProxyDirs=( $gtmProxyDir $gtmProxyDir $gtmProxyDir $gtmProxyDir $gtmProxyDir $gtmProxyDir )
#----- Configuration -----
gtmPxyExtraConfig=none
gtmPxySpecificExtraConfig=(none none none none none none)
#----- Coordinators -----
#----- shortcuts -----
coordMasterDir=$HOME/data_coordinator
coordSlaveDir=$HOME/pgxc/nodes/coord_slave
coordArchLogDir=$HOME/pgxc/nodes/coord_archlog
#----- Overall -----
coordNames=(coord1 coord2 coord3 coord4 coord5 coord6)
coordPorts=(20004 20004 20004 20004 20004 20004)
poolerPorts=(20010 20010 20010 20010 20010 20010)
coordPgHbaEntries=(192.168.0.0/24 192.168.99.0/24)
#coordPgHbaEntries=(::1/128)
#----- Master -----
coordMasterServers=(node1 node2 node3 node4 node5 node6)
coordMasterDirs=( $coordMasterDir $coordMasterDir $coordMasterDir $coordMasterDir $coordMasterDir $coordMasterDir )
coordMaxWALsernder=0
coordMaxWALSenders=( $coordMaxWALsernder $coordMaxWALsernder $coordMaxWALsernder $coordMaxWALsernder $coordMaxWALsernder
                    $coordMaxWALsernder $coordMaxWALsernder )
#----- Slave -----
coordSlave=n
coordSlaveSync=y
coordSlaveServers=(node07 node08 node09 node06)
coordSlavePorts=(20004 20005 20004 20005)
coordSlavePoolerPorts=(20010 20011 20010 20011)
coordSlaveDirs=( $coordSlaveDir $coordSlaveDir $coordSlaveDir $coordSlaveDir )
coordArchLogDirs=( $coordArchLogDir $coordArchLogDir $coordArchLogDir $coordArchLogDir )
#----- Configuration files -----
coordExtraConfig=coordinator.conf
coordSpecificExtraConfig=(none none none none none none)
```

A Anhang

```
coordExtraPgHba=none
coordSpecificExtraPgHba=(none none none none none none)
#----- Additional Slaves -----

coordAdditionalSlaves=n
coordAdditionalSlaveSet=(cad1)
cad1_Sync=n
cad1_Servers=(node08 node09 node06 node07)      # Hosts
cad1_dir=$HOME/pgxc/nodes/coord_slave_cad1
cad1_Dirs=($cad1_dir $cad1_dir $cad1_dir $cad1_dir)
cad1_ArchLogDir=$HOME/pgxc/nodes/coord_archlog_cad1
cad1_ArchLogDirs=($cad1_ArchLogDir $cad1_ArchLogDir $cad1_ArchLogDir $cad1_ArchLogDir)
#----- Datanodes -----
#----- Shortcuts -----
datanodeMasterDir=$HOME/data_node1
datanodeMasterDir2=$HOME/data_node2
datanodeSlaveDir=$HOME/pgxc/nodes/dn_slave
datanodeArchLogDir=$HOME/data_node1/pg_log_arch
datanodeArchLogDir2=$HOME/data_node2/pg_log_arch
#----- Overall -----
#primaryDatanode=datanode1
primaryDatanode=datanode1
datanodeNames=(datanode1 datanode2 datanode3 datanode4 datanode5 datanode6 datanode7 datanode8 datanode9 datanode10
                datanode11 datanode12)
datanodePorts=(20008 20009 20008 20009 20008 20009 20008 20009 20008 20009 20008 20009)
datanodePoolerPorts=(20012 20013 20012 20013 20012 20013 20012 20013 20012 20013 20012 20013)
datanodePgHbaEntries=(192.168.0.0/24 192.168.99.0/24)
#datanodePgHbaEntries=(::1/128)
#----- Master -----
datanodeMasterServers=(node1 node1 node2 node2 node3 node3 node4 node4 node5 node5 node6 node6)
datanodeMasterDirs=($datanodeMasterDir $datanodeMasterDir2 $datanodeMasterDir $datanodeMasterDir2
                    $datanodeMasterDir $datanodeMasterDir2 $datanodeMasterDir $datanodeMasterDir2 $datanodeMasterDir
                    $datanodeMasterDir2 $datanodeMasterDir $datanodeMasterDir2)
datanodeMaxWalSender=0
datanodeMaxWalSenders=($datanodeMaxWalSender $datanodeMaxWalSender $datanodeMaxWalSender $datanodeMaxWalSender
                       $datanodeMaxWalSender $datanodeMaxWalSender $datanodeMaxWalSender $datanodeMaxWalSender $datanodeMaxWalSender
                       $datanodeMaxWalSender $datanodeMaxWalSender $datanodeMaxWalSender)
#----- Slave -----
datanodeSlave=n
datanodeSlaveServers=(node07 node08 node09 node06)
datanodeSlavePorts=(20008 20009 20008 20009)
datanodeSlavePoolerPorts=(20012 20013 20012 20013)
datanodeSlaveSync=y
datanodeSlaveDirs=($datanodeSlaveDir $datanodeSlaveDir $datanodeSlaveDir $datanodeSlaveDir)
datanodeArchLogDirs=( $datanodeArchLogDir $datanodeArchLogDir2 $datanodeArchLogDir $datanodeArchLogDir2
                     $datanodeArchLogDir $datanodeArchLogDir2 $datanodeArchLogDir $datanodeArchLogDir2 $datanodeArchLogDir
                     $datanodeArchLogDir2 $datanodeArchLogDir $datanodeArchLogDir2 )
#----- Configuration files -----
datanodeExtraConfig=datanode.conf
datanodeSpecificExtraConfig=(none none none none none none none none none none none none)
datanodeExtraPgHba=none
datanodeSpecificExtraPgHba=(none none none none none none none none none none none none)
#----- Additional Slaves -----
datanodeAdditionalSlaves=n
#----- WAL archives -----
walArchive=n
#=====

datanode.conf
shared_buffers = 1GB #overall buffersize
temp_buffers = 8MB #maximum number of temporary buffers used by each database session
checkpoint_segments = 24 #number of writable wal files before checkpoint
max_locks_per_transaction = 200
autovacuum_max_workers = 3 #number of workers
max_prepared_transactions = 60 #should be same as max_connections
wal_buffers = 16MB #see checkpoint_segments
work_mem = 5MB
max_connections = 60

coordinator.conf
log_destination = 'stderr'
logging_collector = on
log_directory = 'pg_log'
listen_addresses = '*'
max_connections = 60
shared_buffers = 1400MB #overall buffersize
temp_buffers = 8MB #maximum number of temporary buffers used by each database session
```

A Anhang

```
checkpoint_segments = 24 #number of writable wal files before checkpoint
max_locks_per_transaction = 200
autovacuum_max_workers = 3 #number of workers
max_prepared_transactions = 60 #should be same as max_connections
wal_buffers = 16MB #see checkpoint_segments
work_mem = 1MB
```

A.7 Testdokument Funktionstests

Testfall:	Vorhandene Schnittstelle zu PostgreSQL.
Beschreibung:	Ein direkter Datenaustausch muss mit PostgreSQL möglich sein. Dabei sollen Datenbankkonfigurationen und Daten übertragen und verwendet werden können.
Testdaten:	Beliebige Einträge aus farm.farms. Das Datenbankschema wird vor den Daten übernommen.
Sollergebnis:	Die Schemata sind mit SQL übertragbar und die Daten werden ohne Fehler und ohne Umwandlung eingefügt.
Ist Ergebnis:	Schema ist nach Triggern, Primär- und Fremdschlüssel anzupassen, anschließend lassen sich die Daten direkt übertragen.
Bestanden:	Ja

FT01

Testfall:	Vorhandene Schnittstelle zu <i>UMN MapServer</i> .
Beschreibung:	Das System muss im <i>UMN MapServer</i> als Datenquelle nutzbar sein.
Testdaten:	Mapfile zur Anzeige beispielhafter Schläge aus farm.fields der farmid 1038. Darin muss ein Ausschnitt mit drei Schlägen zu sehen sein.
Sollergebnis:	Alle Schläge werden angezeigt.
Ist Ergebnis:	Die ausgewählten Schläge werden als Bild ausgegeben.
Bestanden:	Ja

FT02

Testfall:	Datenaustausch nach PostGIS Format.
Beschreibung:	Daten sind in einem PostGIS Format zu übertragen und zu speichern.
Testdaten:	Beliebige Einträge aus farm.fields und n.sensorlogs. Das Datenbankschema wird zunächst ohne die Daten übernommen.
Sollergebnis:	Einträge aus den Tabellen werden ohne Umwandlung direkt in die Datenbank geschrieben.
Ist Ergebnis:	Datenübertragung mit dblink und PostGIS Datenformaten ohne Fehler durchgeführt.
Bestanden:	Ja

FT03

Testfall:	Umwandlung zwischen Koordinatenreferenzsystemen mit EPSG Codes 4326 und 3857.
Beschreibung:	Vorhandene räumliche Daten werden zwischen 4326 nach 3857 umgewandelt.
Testdaten:	Beliebige Einträge aus nutrients.samples.
Sollergebnis:	Einträge werden mit st_transform von 4326 nach 3857 und anders herum umgewandelt.
Ist Ergebnis:	Funktion ist nutzbar und liefert die korrekten Ergebnisse.
Bestanden:	Ja

FT04

Testfall:	Verschneidung von räumlichen Daten.
Beschreibung:	Überlagernde Vektordaten werden miteinander verschnitten.
Testdaten:	Ausgewählte Schläge und Teilschläge aus farm.fields.
Sollergebnis:	Intersection, Union, Difference und Symmetric Difference ist durchführbar und liefert das korrekte Ergebnis.
Ist Ergebnis:	Die Verschneidung erfolgt korrekt und kann dargestellt werden.
Bestanden:	Ja

FT05

Testfall:	Interpolation von Punktdaten mit dem Spezialfall Kriging.
Beschreibung:	Punkte aus nutrients.samples werden über die spezielle Kriging Bibliothek der Agri Con interpoliert.
Testdaten:	Ausschnitt aus nutrients.samples eines Betriebes und eines Grundnährstoffes.
Sollergebnis:	Vektordaten mit gewichteten Werten des Grundnährstoffes entsprechend des Algorithmus. Karten sollen mit denen des Ist-Standes übereinstimmen.
Ist Ergebnis:	Berechnungen konnten ausgeführt werden, jedoch waren sie nur nebenwirkungsfrei verwendbar, weshalb das Ergebnis nicht validiert werden kann.
Bestanden:	Nein

FT06

Testfall:	Räumliche Filterung.
Beschreibung:	Räumliche Daten können räumlich gefiltert werden, dass heißt die Aggregation filtert anhand räumlicher Eigenschaften wie den Koordinaten.
Testdaten:	Schläge aus farm.fields und Userlayer aus common.userlayers eines Betriebes.
Sollergebnis:	Zu einem Userlayer sollen nur die entsprechend dessen Ausdehnung enthaltenen Schläge geliefert werden.
Ist Ergebnis:	Die Funktionen st_intersects() und st_disjoint() filtern korrekt entsprechend der Ausdehnung des Userlayer, bewiesen durch Gesamtzahl der Schläge, enthaltene und nicht enthaltene Schläge.
Bestanden:	Ja

FT07

A.8 Mapfile Aggregation Punktdaten

LISTING A.5: Mapfile Aggregation Punktdaten

```
MAP
    NAME "WMSmap"
    STATUS ON
    SIZE 256 256
    EXTENT 1560098.91160313 6610084.3680505 1561696.27498181 6611681.73142918
    #Idee: mit UMN CLient entsprechend userlayern ausschnitte erzeugen und diese verwenden viewids
    = [8357,4491,4701,4720,4696]
    UNITS METERS
    IMAGETYPE 'png'

    CONFIG "MS_ERRORFILE" "/tmp/ms_error.txt"
    DEBUG 5
    CONFIG "MS_TEMPPATH" "/tmp/"

    OUTPUTFORMAT
        NAME "png"
```

A Anhang

```
MIMETYPE "image/png"
DRIVER "AGG/PNG"
EXTENSION "png"
IMAGEMODE RGBA
TRANSPARENT TRUE
END # OUTPUTFORMAT

WEB
  IMAGEPATH "/tmp/"
  IMAGEURL "tmp/"
  METADATA
    WMS_TITLE "WMSmap"
    WMS_ONLINERESOURCE "http://localhost:84/fcgi-bin/mapserv?map=WMSmap.map&"
    WMS_SRS "EPSG:3857"
    WMS_ENABLE_REQUEST "GetMap GetFeatureInfo GetCapabilities GetLegendGraphic"
  END
END

PROJECTION #EPSG:3857
  "proj=merc"
  "a=6378137"
  "b=6378137"
  "lat_ts=0.0"
  "lon_0=0.0"
  "x_0=0.0"
  "y_0=0"
  "k=1.0"
  "units=m"
  "nadgrids=@null"
  "wktext"
  "no_defs"
END

SYMBOL
  NAME "sld_mark_symbol_circle_filled"
  TYPE ELLIPSE
  FILLED TRUE
  POINTS
    1 1
  END
END

LEGEND
  IMAGECOLOR 255 255 255
  OUTLINECOLOR 80 80 80
  KEYSIZE 30 20
  STATUS off
  TRANSPARENT on
END

QUERYMAP
  SIZE -1 -1
  STATUS OFF
  STYLE HILITE
END # QUERYMAP

LAYER
  NAME "NSensorlogs"
  TYPE Point
  VALIDATION
    'shash' '[\w\d\s\|/+={ }]{28}'
  END
  CLASSITEM "applraten"
  LABELITEM "applraten"
  CONNECTION "user=postgres password=postgres dbname=agrodbs2 host=192.168.0.114 port=20001"
  CONNECTIONTYPE postgis
  PROCESSING "CLOSE_CONNECTION=DEFER"
  DATA "geom from (Select nsensorlogs.id,nsensorlogs.geom,nsensorlogs.fileid,nsensorlogs.date,
    nsensorlogs.altitude,nsensorlogs.marker1,nsensorlogs.marker2,nsensorlogs.marker3,nsensorlogs.
    s1,nsensorlogs.s2,nsensorlogs.sn,nsensorlogs.applraten,nsensorlogs.applfactor,nsensorlogs.
    opmode,nsensorlogs.spraten,nsensorlogs.s1l,nsensorlogs.s2l,nsensorlogs.s1r,nsensorlogs.s2r,
    nsensorlogs.bi,nsensorlogs.farmid,nsensorlogs.applratenclass from n.nsensorlogs where fileid =
    35457) layersselect using unique id using srid=4326"
  #fileid entfernen
  PROJECTION
    #"init=epsg:4326"
    "proj=latlong"
```

A Anhang

```
"ellps=WGS84"
"datum=WGS84"
"no_defs"

END
STATUS On
#TRANSFORM true
UNITS meters
METADATA
    "wms_title" "NSensorlogs"
    #"wms_srs" "EPSG:3857"
    "wms_extent" "4.49157642122284 16.16967511640222 46.741611471444564 55.196005052343054"
    "wms_enable_request" "GetMap GetFeatureInfo GetCapabilities GetLegendGraphic"

END

CLASS
    NAME "< 18"
    EXPRESSION ([applraten] < 18)
    STYLE
        COLOR 247 251 255
        OUTLINECOLOR 0 0 0
        SIZE 6
        SYMBOL "sld_mark_symbol_circle_filled"
    END # STYLE
END # CLASS

CLASS
    NAME "18 - 36"
    EXPRESSION ( ([applraten] >= 18) And ([applraten] < 36))
    STYLE
        COLOR 218 232 245
        OUTLINECOLOR 0 0 0
        SIZE 6
        SYMBOL "sld_mark_symbol_circle_filled"
    END # STYLE
END # CLASS

CLASS
    NAME "36 - 54"
    EXPRESSION ( ([applraten] >= 36) And ([applraten] < 54))
    STYLE
        COLOR 186 214 235
        OUTLINECOLOR 0 0 0
        SIZE 6
        SYMBOL "sld_mark_symbol_circle_filled"
    END # STYLE
END # CLASS

CLASS
    NAME "54 - 73"
    EXPRESSION ( ([applraten] >= 54) And ([applraten] < 73))
    STYLE
        COLOR 136 190 220
        OUTLINECOLOR 0 0 0
        SIZE 6
        SYMBOL "sld_mark_symbol_circle_filled"
    END # STYLE
END # CLASS

CLASS
    NAME "73 - 91"
    EXPRESSION ( ([applraten] >= 73) And ([applraten] < 91))
    STYLE
        COLOR 83 158 204
        OUTLINECOLOR 0 0 0
        SIZE 6
        SYMBOL "sld_mark_symbol_circle_filled"
    END # STYLE
END # CLASS

CLASS
    NAME "91 - 109"
    EXPRESSION ( ([applraten] >= 91) And ([applraten] < 109))
    STYLE
        COLOR 42 122 185
        OUTLINECOLOR 0 0 0
        SIZE 6
        SYMBOL "sld_mark_symbol_circle_filled"
    END # STYLE
END # CLASS

CLASS
    NAME "109 - 127"
    EXPRESSION ( ([applraten] >= 109) And ([applraten] < 127))
```

A Anhang

```
STYLE
  COLOR 11 85 159
  OUTLINECOLOR 0 0 0
  SIZE 6
  SYMBOL "sld_mark_symbol_circle_filled"
END # STYLE
END # CLASS
CLASS
  NAME "> 145"
  EXPRESSION ([applraten] >= 127)
  STYLE
    COLOR 8 48 107
    OUTLINECOLOR 0 0 0
    SIZE 6
    SYMBOL "sld_mark_symbol_circle_filled"
  END # STYLE
END # CLASS
END #Layer

END
```

Literaturverzeichnis

- [Ahl13] AHLERS, Jöhrn: *Untersuchung von Techniken verteilter Datenbanksysteme zur Speicherung und Abfrage von räumlichen Daten*, Jade Hochschule Oldenburg, Master Thesis, Mai 2013
- [Baa12] BAAS, Bart: *NoSQL spatial Neo4j versus PostGIS*, TU Delf, Universität Utrecht, Master Thesis, Mai 2012. – Geographical Information Management and Applications
- [Bal98] BALZERT, Helmut: *Lehrbuch der Software-Technik. Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung..* Bd. Bd. 2. Heidelberg : Spektrum Akad. Verl., 1998. – XX, 769 S. <http://www.bsz-bw.de/cgi-bin/ekz.cgi?SWB06262739>. – ISBN 3827400651
- [BC94] BONHAM-CARTER, Graeme: *Geographic information systems for geoscientists: Modelling with GIS*. 1. ed. Kidlington [u.a.] : Pergamon, 1994. – XVII, 398 S.. – ISBN 0080418678
- [Ben13] BENSBERG, Frank: *Nutzwertanalyse*. <http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/is-management/Management-von-Anwendungssystemen/Beschaffung-von-Anwendungssoftware/Nutzwertanalyse>. Version: 8 2013. – abgerufen am 11.1.2015
- [Bor07] BORTHAKUR, Dhruva: *The Hadoop Distributed File System, Architecture and Design* / The Apache Software Foundation. 2007. – Forschungsbericht
- [CCR13] CCRINC: *GeoMesa LocationTech DC*. <http://de.slideshare.net/CCRinc/location-techdc-talk2-28465214>. Version: 11 2013. – abgerufen am 30.3.2015
- [Com98] COMMUNITY RESEARCH AND DEVELOPMENT INFORMATION SERVICE: *RASDAMAN Report Summary*. http://cordis.europa.eu/result/rcn/20754_en.html. Version: 7 1998. – abgerufen am 19.2.2015

- [Ear14] EARTHSERVER: *Rasdaman GDAL driver*. <http://earthserver.eu/trac/wiki/DocumentationGDALrasdaman>. Version: 8 2014. – abgerufen am 19.2.2015
- [Edl11] EDLICH, Stefan (Hrsg.): *NoSQL : Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken*. 2., aktualisierte und erw. Aufl. München : Hanser, 2011. – XVIII, 392 S. <http://deposit.d-nb.de/cgi-bin/dokserv?id=3791928&prov=M&dok%5Fvar=1&dok%5Fext=htm>. – ISBN 9783446427532
- [Eic14] EICHELBERGER, Chris: *GeoMesa Quick Start*. <http://www.geomesa.org/2014/05/28/geomesa-quickstart/>. Version: 5 2014. – abgerufen am 2.12.2014
- [Eva09] EVANS, Eric: *NOSQL 2009*. http://blog.sym-link.com/2009/05/12/nosql_2009.html. Version: 9 2009
- [FC06] FAY CHANG, Sanjay Ghemawat Wilson C. Hsieh Deborah A. Wallach Mike Burrows Tushar Chandra Andrew Fikes Robert E. G. Jeffrey Dean D. Jeffrey Dean: *Bigtable: A Distributed Storage System for Structured Data* / Google, Inc. 2006. – Forschungsbericht
- [Fox14] FOX, Anthony: *GeoMesa: Scaling up Geospatial Analysis*. http://www.eclipse.org/community/eclipse_newsletter/2014/march/article3.php. Version: 3 2014
- [Gau] GAUL, Wolfgang: *Analytic Hierarchy Process (AHP)*. <http://marketing.wiwi.uni-karlsruhe.de/institut/viror/kaiman/kaiman/ahp/index.xml.html>. – abgerufen am 24.2.2015
- [Geo15] GEOTOOLS: *GeoTools*. <http://docs.geotools.org/latest/userguide/geotools.html>. Version: 2015. – abgerufen am 21.1.2015
- [Han95] HANSEN, Olav: *Leistungsanalyse paralleler Programme*. Heidelberg [u.a.] : Spektrum, 1995. – 200 S.. – ISBN 3860257072
- [Hoh96] HOHENSTEIN, Uwe (Hrsg.): *Objektorientierte Datenbanksysteme : ODMG-Standard, Produkte, Systembewertung, Benchmarks, Tuning*. Braunschweig : Vieweg, 1996. – VII, 269 S.. – ISBN 3528055014
- [Hä13] HÄBERLEIN, Dan: *Migration und Extrahierung von Datensätzen mittels spaltenorientierten Datenbanken am Beispiel von Apache HBase*, Universität Leipzig, Bachelor Thesis, September 2013. – Wirtschaftswissenschaftliche Fakultät

- [JD04] JEFFREY DEAN, Sanjay G.: MapReduce, Simplified Data Processing on Large Clusters / Google, Inc. 2004. – Forschungsbericht
- [Jun12] JUNGHANN, Kurt: *Analyse der OpenSource Mapserver GeoServer und Map-Server für die Nutzung mit Agriport Mobile*, Hochschule Mittweida (FH), Bachelor Thesis, Oktober 2012. – Fakultät für Mathematik, Naturwissenschaften und Informatik
- [Kud07] KUDRASS, Thomas (Hrsg.): *Taschenbuch Datenbanken : Mit ... 28 Tabellen*. München : Fachbuchverl. Leipzig im Carl-Hanser-Verl., 2007. – 582 S. <http://swbplus.bsz-bw.de/bsz260569755inh.htm>. – ISBN 3446409440
- [Lan13] LANGE, Norbert d.: *Geoinformatik in Theorie und Praxis*. 3. Springer, 2013
- [Loc] LOCATIONTECH: *About*. <https://www.locationtech.org/about>. – abgerufen am 19.2.2015
- [Loc14a] LOCATIONTECH: *Ingest and Export with GeoMesa Tools*. <http://www.geomesa.org/geomesa-tools-ingest-export/>. Version: 9 2014. – abgerufen am 7.12.2014
- [Loc14b] LOCATIONTECH: *Tutorials*. <http://www.geomesa.org/tutorials/>. Version: 12 2014. – abgerufen am 7.12.2014
- [Loc14c] LOCATIONTECH: *Tutorials*. <http://www.geomesa.org/2014/10/09/geomesa-tools-features/>. Version: 10 2014. – abgerufen am 7.12.2014
- [Lud07] LUDEWIG, Jochen ; LICHTER, Horst (Hrsg.): *Software-Engineering : Grundlagen, Menschen, Prozesse, Techniken*. 1. Aufl. Heidelberg : dpunkt-Verl., 2007. – XXI, 618 S. <http://bvbr.bib-bvb.de:8991/F?func=service&doc%5Flibrary=BVB01&doc%5Fnumber=012989845&line%5Fnumber=0002&func%5Fcode=DB%5FRECORDS&service%5Ftype=MEDIA>. – ISBN 3898642682
- [mas13] MASONS: *Supersized PostgreSQL: Postgres-XL for Scale-Out OLTP and Big Data Analytics*. http://de.slideshare.net/mason_s/postgres-xl-scaling. Version: 6 2013. – abgerufen am 30.3.2015
- [Min14] MINNESOTA, University of: *LAYER*. <http://mapserver.org/mapfile/layer.html>. Version: 2014. – abgerufen am 17.2.2015
- [OSGa] OSGEO: *UpdateGeometrySRID*. <http://postgis.net/docs/manual-2.1/UpdateGeometrySRID.html>. – abgerufen am 12.2.2015

- [OSGb] OSGEO: *Using OpenGIS Standards*. <http://postgis.net/docs/manual-2.1/reference.html>. – abgerufen am 8.2.2015
- [OSG14] OSGEO LIVE: *Rasdaman*. http://live.osgeo.org/de/overview/rasdaman_overview.html. Version: 2014
- [OSG15a] OSGEO: *Geometry CRS Tutorial*. <http://docs.geotools.org/latest/tutorials/geometry/geometrycrs.html#coordinate-reference-system>. Version: 2 2015. – abgerufen am 12.2.2015
- [OSG15b] OSGEO: *Using OpenGIS Standards*. http://postgis.net/docs/manual-2.1/using_postgis_dbmanagement.html. Version: 2 2015. – abgerufen am 8.2.2015
- [Ras12] RASDAMAN: *News-Archiv*. <http://www.rasdaman.com/News/archive.php>. Version: 2 2012. – abgerufen am 19.2.2015
- [Ras14a] RASDAMAN: *Features*. <http://www.rasdaman.org/wiki/Features>. Version: 12 2014. – abgerufen am 19.2.2015
- [Ras14b] RASDAMAN: *Installation and Administration Guide*. http://www.rasdaman.org/browser/manuals_and_examples/manuals/doc-guides/inst-guide.pdf. Version: 12 2014. – abgerufen am 19.2.2015
- [Ras14c] RASDAMAN: *Introduction*. <http://www.rasdaman.org/wiki/Introduction>. Version: 4 2014. – abgerufen am 19.2.2015
- [Ras14d] RASDAMAN: *Welcome to rasdaman – the World's Leading Array Database*. <http://www.rasdaman.org/>. Version: 2014
- [Ras15a] RASDAMAN: *Query Language Guide*. http://www.rasdaman.org/browser/manuals_and_examples/manuals/doc-guides/ql-guide.pdf. Version: 2 2015. – abgerufen am 19.2.2015
- [Ras15b] RASDAMAN: *Rasdaman Documentation*. <http://www.rasdaman.org/wiki/Documentation>. Version: 2 2015. – abgerufen am 19.2.2015
- [Ras15c] RASDAMAN: *Welcome to rasdaman*. <http://www.rasdaman.org/>. Version: 2 2015. – abgerufen am 2.02.2015

- [Ros01] ROSNER, Hans-Joachim: *Verarbeitung geographischer Daten: Methodische Bausteine zu Statistik und GIS*. 3., erg. und verb. Aufl. Tübingen : Selbstverl. des Geographischen Inst. der Univ. Tübingen, 2001. – VI, 110 S.. – ISBN 388121061X
- [Sof14a] SOFTWARE FOUNDATION APACHE: *Apache Accumulo Notable Features*. https://accumulo.apache.org/notable_features.html. Version: 2014. – abgerufen am 21.1.2015
- [Sof14b] SOFTWARE FOUNDATION APACHE: *Apache Accumulo User Manual Version 1.6*. https://accumulo.apache.org/1.6/accumulo_user_manual.html. Version: 5 2014. – abgerufen am 13.1.2015
- [Swa95] SWAN, Andrew R H. ; SANDILANDS, Michael (Hrsg.): *Introduction to geological data analysis*. 1. publ. Oxford [u.a.] : Blackwell Science, 1995. – XIV, 446 S. <http://digitool.hbz-nrw.de:1801/webclient/DeliveryManager?pid=2668143&custom%5Fatt%5F2=simple%5Fviewer>. – ISBN 0632032243
- [Thu12] THURM, Benjamin: *Einsatz von NoSQL-Datenbanksystemen für Geodaten*, Hochschule für Technik und Wirtschaft Dresden, Bachelor Thesis, Februar 2012. – Geoinformation und Vermessungswesen
- [Traa] TRANSLATTICE: *Client Interfaces*. <http://files.postgres-xl.org/documentation/client-interfaces.html>. – abgerufen am 8.2.2015
- [Trab] TRANSLATTICE: *Installation from Source Code*. <http://files.postgres-xl.org/documentation/installation.html>. – abgerufen am 8.2.2015
- [Trac] TRANSLATTICE: *Postgres-XL 9.2 Documentation*. <http://files.postgres-xl.org/documentation/>. – abgerufen am 8.2.2015
- [Trad] TRANSLATTICE: *Server Programming*. <http://files.postgres-xl.org/documentation/server-programming.html>. – abgerufen am 8.2.2015
- [Tra13] TRANSLATTICE: *TransLattice Acquires StormDB to Enhance TransLattice Elastic Database*. http://www.translattice.com/pr/TransLattice_Acquires_StormDB_to_Enhance_TransLattice_Elastic_Database.shtml. Version: 10 2013. – abgerufen am 19.2.2015
- [Tra15a] TRANSLATTICE: *Overview*. <http://www.postgres-xl.org/overview/>. Version: 2 2015. – abgerufen am 8.2.2015

- [Tra15b] TRANSLATTICE: *What is Postgres-XL?* <http://files.postgres-xl.org/documentation/intro-what-is.html>. Version: 2 2015. – abgerufen am 8.2.2015
- [Viv] VIVID SOLUTIONS: *JTS Topology Suite Technical Specification*. <http://www.vividsolutions.com/jts/bin/JTS%20Technical%20Specs.pdf>. – abgerufen am 3.3.2015
- [Wik15a] WIKIPEDIA: *Wikipedia article traffic statistics*. <http://stats.grok.se/en/latest90/spatial%20database>. Version: 2 2015. – abgerufen am 1.2.2015
- [Wik15b] WIKIPEDIA - LETZTE ÄNDERUNG TBOONX: *Spatial database*. https://en.wikipedia.org/wiki/Spatial_database. Version: 2 2015. – abgerufen am 1.2.2015
- [YZ11] YAN ZHOU, Yeting Z. Qing Zhu Z. Qing Zhu: Spatial Data Dynamic Balancing Distribution Method Based on the Minimum Spatial Proximity for Parallel Spatial Database / College of Automation University of Electric Science and Technology of China, State Key Laboratory of Information Engineering in Surveying Mapping and Remote Sensing,. 2011. – Forschungsbericht
- [YZ12] YUNQIN ZHONG, Tieying Zhang Zhenhua Li Jinyun Fang Guihai C. Jizhong Han H. Jizhong Han: Towards Parallel Spatial Query Processing for Big Spatial Data / Peking University, Beijing, China. 2012. – Forschungsbericht

Eidesstatliche Erklärung

Ich versichere, dass die Masterarbeit mit dem Titel „Untersuchung quelloffener verteilter geografischer Informationssysteme zur Verarbeitung agrartechnischer Kennzahlen“ nicht anderweitig als Prüfungsleistung verwendet wurde und diese Masterarbeit noch nicht veröffentlicht worden ist. Die hier vorgelegte Masterarbeit habe ich selbstständig und ohne fremde Hilfe abgefasst. Ich habe keine anderen Quellen und Hilfsmittel als die angegebenen benutzt. Diesen Werken wörtlich oder sinngemäß entnommene Stellen habe ich als solche gekennzeichnet.

Leipzig, 28. Mai 2015

Unterschrift