Ein Cloud-basiertes räumliches Decision Support System für die Herausforderungen der Energiewende

Golo Klossek
Hochschule Regensburg
golo.klossek
@stud.hs-regensburg.de

Stefanie Scherzinger
Hochschule Regensburg
stefanie.scherzinger
@hs-regensburg.de

Michael Sterner
Hochschule Regensburg
michael.sterner
@hs-regensburg.de

KURZFASSUNG

Die Energiewende in Deutschland wirft sehr konkrete Fragestellungen auf: Welche Standorte eignen sich für Windkraftwerke, wo können Solaranlagen wirtschaftlich betrieben werden? Dahinter verbergen sich rechenintensive Datenverarbeitungsschritte, auszuführen auf *Big Data* aus mehreren Datenquellen, in entsprechend heterogenen Formaten. Diese Arbeit stellt exemplarisch eine konkrete Fragestellung und ihre Beantwortung als MapReduce Algorithmus vor. Wir konzipieren eine geeignete, Cluster-basierte Infrastruktur für ein neues *Spatial Decision Support System* und legen die Notwendigkeit einer deklarativen, domänenspezifischen Anfragesprache dar.

Allgemeine Begriffe

Measurement, Performance, Languages.

Stichworte

Cloud-Computing, MapReduce, Energiewende.

1. EINLEITUNG

Der Beschluss der Bundesregierung, zum Jahr 2022 aus der Kernenergie auszusteigen und deren Anteil am Strom-Mix durch erneuerbare Energien zu ersetzen, fordert einen rasanten Ausbau der erneuerbaren Energien. Entscheidend für den Bau neuer Windkraft- und Solaranlagen sind vor allem die zu erzielenden Gewinne und die Sicherheit der Investitionen. Somit sind präzise Ertragsprognosen von großer Bedeutung. Unterschiedliche Standorte sind zu vergleichen, die Ausrichtung der Windkraftanlagen zueinander in den Windparks ist sorgfältig zu planen. Als Entscheidungsgrundlage dienen hierzu vor allem historische Wetterdaten. Für die Kalkulation des Ertrags von Windkraftanlagen muss effizient auf die Datenbasis zugegriffen werden können. Diese erstreckt sich über große Zeiträume, da das Windaufkommen nicht nur jährlich schwankt, sondern auch dekadenweise variiert [3, 9].

25^{th} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 28.05.2013 - 31.05.2013, Ilmenau, Germany.

Die Standortfindung etwa für Bankfilialen und die Zonierung, also das Ausweisen geographischer Flächen für die Landwirtschaft, sind klassische Fragestellungen für räumliche Entscheidungsunterstützungssysteme [6].

Die Herausforderungen an solch ein *Spatial Decision Support System* im Kontext der Energiewende sind vielfältig:

1. Verarbeitung heterogener Datenformate.
2. Skalierbare Anfragebearbeitung auf *Big Data*.
3. Eine elastische Infrastruktur, die mit der Erschließung neuer Datenquellen ausgebaut werden kann.
4. Eine deklarative, domänenspezifische Anfragesprache für komplexe *ad-hoc* Anfragen.

Wir begründen kurz die Eckpunkte dieses Anforderungsprofils im Einzelnen. Dabei vertreten wir den Standpunkt, dass existierende Entscheidungsunterstützungssysteme auf Basis relationaler Datenbanken diese nicht in allen Punkten erfüllen können.

(1) Historische Wetterdaten sind zum Teil öffentlich zugänglich, werden aber auch von kommerziellen Anbietern bezogen. Prominente Vertreter sind das *National Center for Atmospheric Research* [12] in Boulder Colorado, der *Deutsche Wetterdienst* [7] und die *Satel-Light* [14] Datenbank der Europäischen Union. Hinzu kommen Messwerte der hochschuleigenen experimentellen Windkraft- und Solaranlagen. Die Vielzahl der Quellen und somit der Formate führen zu den klassischen Problemen der Datenintegration.

(2) Daten in hoher zeitlicher Auflösung, die über Jahrzehnte hinweg erhoben werden, verursachen Datenvolumina im *Big Data* Bereich. Der Deutsche Wetterdienst allein verwaltet ein Datenarchiv von 5 Petabyte [7]. Bei solchen Größenordnung haben sich NoSQL Datenbanken gegenüber relationalen Datenbanken bewährt [4].

(3) Wir stellen die Infrastruktur für ein interdisziplinäres Team der *Regensburg School of Energy and Resources* mit mehreren im Aufbau befindlichen Projekten bereit. Um den wachsenden Anforderungen unserer Nutzer gerecht werden zu können, muss das System elastisch auf neue Datenquellen und neue Nutzergruppen angepasst werden können.

(4) Unsere Nutzer sind überwiegend IT-affin, doch nicht erfahren in der Entwicklung komplexer verteilter Systeme. Mit einer domänenspezifischen Anfragesprache wollen die Autoren dieses Artikels die intuitive Nutzbarkeit des Systems gewährleisten.

Unter diesen Gesichtspunkten konzipieren wir unser System als Hadoop-Rechencluster [1, 5]. Damit sind die Ska-

lierbarkeit auf große Datenmengen (2) und die horizontale Skalierbarkeit der Hardware gegeben (3). Da auf historische Daten ausschließlich lesend zugegriffen wird, bietet sich der MapReduce Ansatz geradezu an. Zudem erlaubt Hadoop das Verarbeiten unstrukturierter, heterogener Daten (1). Der Entwurf einer eigenen Anfragesprache (4) stellt dabei eine spannende und konzeptionelle Herausforderung dar, weil hierfür ein tiefes Verständnis für die Fragestellungen der Nutzer erforderlich ist.

Struktur. Die folgenden Kapitel liefern Details zu unserem Vorhaben. In Kapitel 2 beschreiben wir eine konkrete Fragestellung bei der Standortfindung von Windkraftwerken. In Kapitel 3 stellen wir unsere Lösung als MapReduce Algorithmus dar. Kapitel 4 skizziert unsere Infrastruktur. Im 6. Kapitel wird auf verwandte Arbeiten eingegangen. Das letzte Kapitel gibt eine Zusammenfassung unserer Arbeit und zeigt deren Perspektive auf.

2. WINDPOTENTIALANALYSE

Ein aktuelles Forschungsprojekt der Hochschule Regensburg beschäftigt sich mit der Potentialanalyse von Windkraftanlagen. Hier werden die wirtschaftlichen Aspekte, die für das Errichten neuer Windkraftanlagen entscheidend sind, untersucht. Mithilfe der prognostizierten Volllaststunden einer Windkraftanlage kann eine Aussage über die Rentabilität getroffen werden. Diese ist bestimmt durch die Leistungskennlinie der Windkraftanlage und letztlich durch die zu erwartenden Windgeschwindigkeiten.

Abbildung 1 (aus [9]) skizziert die spezifische Leistungskennlinie einer Windkraftanlage in vier Phasen:

I) Erst ab einer gewissen Windgeschwindigkeit beginnt die Anlage Strom zu produzieren.

II) Die Leistung steigt über den wichtigsten Arbeitsbereich in der dritten Potenz zur Windgeschwindigkeit an, bis die Nennleistung der Anlage erreicht ist.

III) Die Ausgangsleistung wird auf die Nennleistung der Anlage begrenzt. Ausschlaggebend für die Höhe der Nennleistung ist die Auslegungsgröße des Generators.

IV) Die Windkraftanlage schaltet sich bei zu hohen Windgeschwindigkeiten ab, um eine mechanische Überbelastung zu verhindern.

Wie Abbildung 1 verdeutlicht, ist zum Errechnen der abgegeben Arbeit einer Windkraftanlage eine genaue Kenntnis der stochastischen Verteilung der Windgeschwindigkeit [1] notwendig. Mithilfe entsprechender Histogramme können somit potentielle Standorte für neue Windkraftanlagen verglichen, und Anlagen mit geeigneter Leistungskennlinie passend für den spezifischen Standort ausgewählt werden.

Als Datenbasis eignen sich etwa die Wetterdaten des Forschungsinstitut des *National Center for Atmospheric Research* [12] und die des Deutschen Wetterdienstes [7].

Insbesondere im Binnenland ist eine hohe räumliche Auflösung der meteorologischen Daten wichtig. Aufgrund der

[1]Wir verwenden die Begriffe Windgeschwindigkeit und Windstärke synonym. Streng genommen wird die Windgeschwindigkeit als Vektor dargestellt, während die Windstärke als skalare Größe erfasst wird. Dabei kann die Windstärke aus der Windgeschwindigkeit errechnet werden.

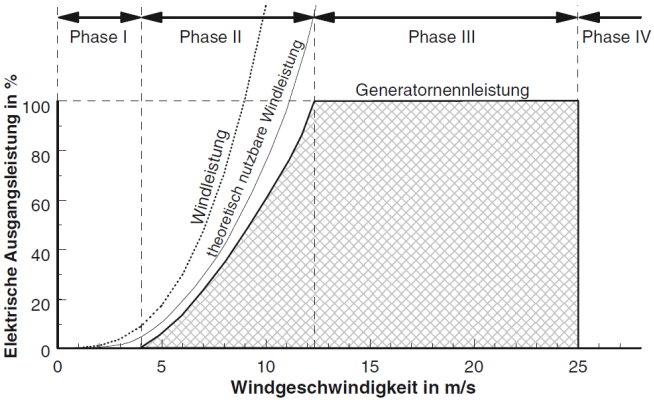

Abbildung 1: Aussagen über die Leistung in Abhängigkeit zur Windgeschwindigkeit (aus [9]).

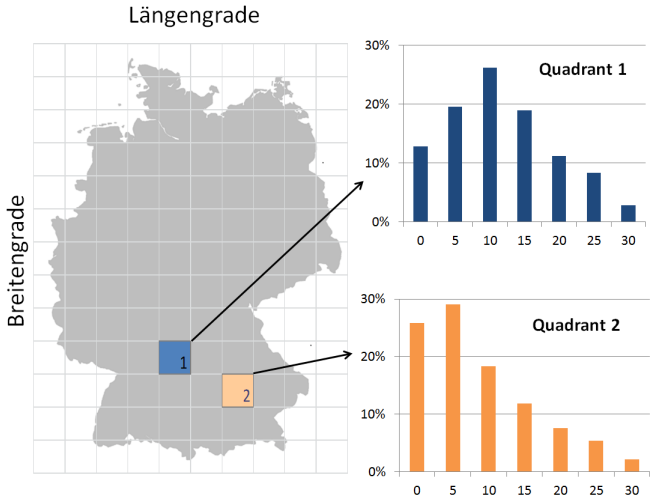

Abbildung 2: Histogramme über die Windstärkeverteilung.

Orographie variieren die Windgeschwindigkeiten schon bei kurzen Distanzen stark.

Abbildung 2 skizziert die resultierende Aufgabenstellung: Geographische Flächen werden kleinräumig unterteilt, was die Abbildung aus Gründen der Anschaulichkeit stark vereinfacht darstellt. Für jeden Quadranten, bestimmt durch Längen- und Breitengrad, interessiert die Häufigkeitsverteilung der Windstärken (dargestellt als Histogramm).

Je nach Fragestellung wird von unterschiedlichen Zeiträumen und unterschiedlicher Granularität der Quadranten ausgegangen. Aufgrund der schieren Größe der Datenbasis ist hier ein massiv paralleler Rechenansatz gefordert, wenn über eine Vielzahl von Quadranten hinweg Histogramme berechnet werden sollen.

3. MASSIV PARALLELE HISTOGRAMMBERECHNUNG

Im Folgenden stellen wir einen MapReduce Algorithmus zur parallelen Berechnung von Windgeschwindigkeitsverteilungen vor. Wir betrachten dabei die Plattform Apache Ha-

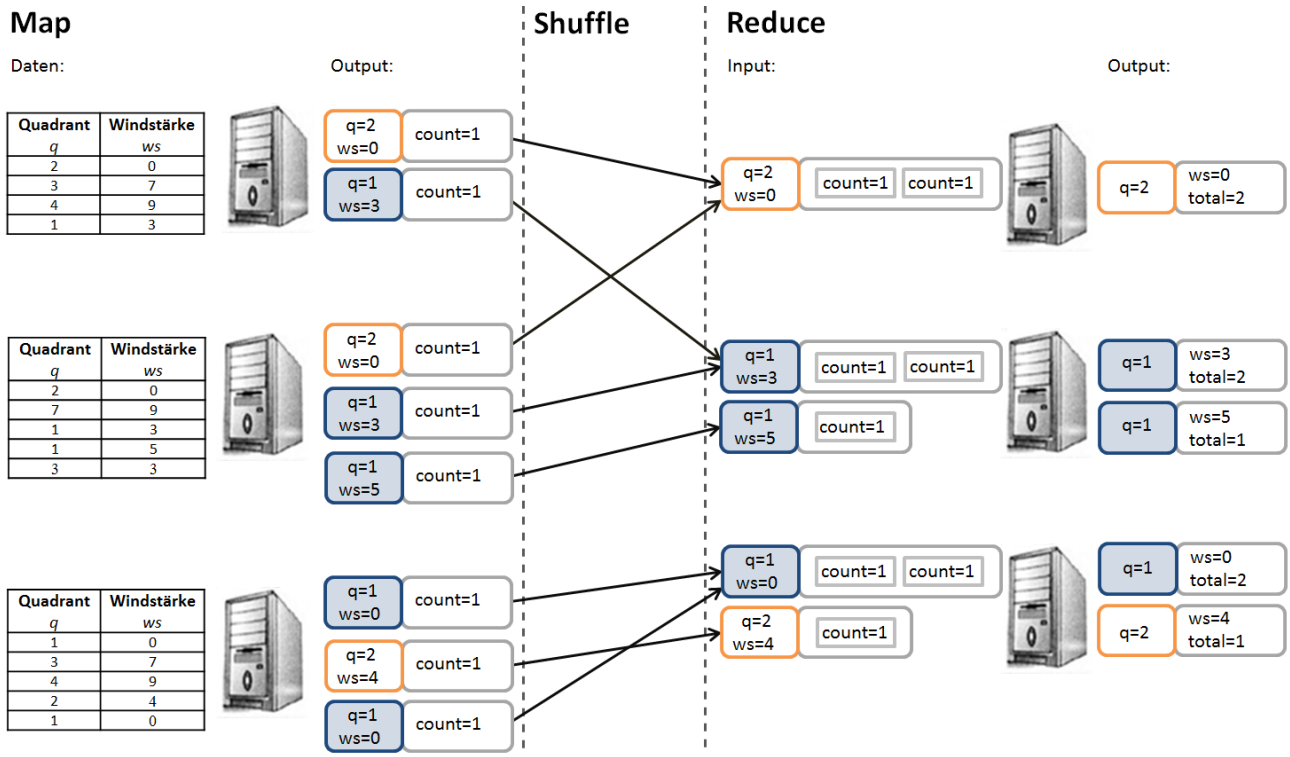

Abbildung 3: Erste MapReduce-Sequenz zur Berechnung der absoluten Häufigkeiten.

doop [1], eine quelloffene MapReduce Implementierung [5].

Hadoop ist dafür ausgelegt, mit großen Datenmengen umzugehen. Ein intuitives Programmierparadigma erlaubt es, massiv parallele Datenverarbeitungsschritte zu spezifizieren. Die Plattform partitioniert die Eingabe in kleinere Datenblöcke und verteilt diese redundant auf dem *Hadoop Distributed File System* [15]. Dadurch wird eine hohe Datensicherheit gewährleistet. Als logische Basiseinheit arbeitet Hadoop mit einfachen Schlüssel/Werte Paaren. Somit können selbst unstrukturierte oder nur schwach strukturierte Daten *ad hoc* verarbeitet werden.

MapReduce Programme werden in drei Phasen ausgeführt.

1. In der ersten Phase wird auf den partitionierten Eingabedaten eine Map-Funktion parallel ausgeführt. Diese Map-Funktion transformiert einfache Schlüssel/Werte Paare in eine Liste von neuen Schlüssel/Werte Paaren.

2. Die anschließende Shuffle-Phase verteilt die entstandenen Tupel so um, dass nun alle Paare mit dem gleichen Schlüssel an demselben Rechner vorliegen.

3. Die Reduce-Phase berechnet meist eine Aggregatfunktion auf allen Tupeln mit demselben Schlüssel.

Die Signaturen der Map- und Reduce-Funktion werden üblicherweise wie folgt beschrieben [11]:

$$\begin{aligned} \textbf{Map:} \quad & (k1, v1) \rightarrow \text{list}(k2, v2) \\ \textbf{Reduce:} \quad & (k2, \text{list}(v2)) \rightarrow \text{list}(k3, v3) \end{aligned}$$

Wir erläutern nun unseren MapReduce Algorithmus zum Erstellen von Histogrammen der Windgeschwindigkeitsverteilungen. Im Sinne einer anschaulichen Darstellung abstrahieren wir von dem tatsächlichen Eingabeformat und beschränken uns auf nur eine Datenquelle. Die Eingabetupel enthalten einen Zeitstempel, den Längen- und Breitengrad als Ortsangabe und diverse Messwerte.

Wir nehmen vereinfachend an, dass die Ortsangabe bereits in eine Quadranten-ID übersetzt ist. Diese Vereinfachung erlaubt eine übersichtlichere Darstellung, gleichzeitig ist die Klassifikation der Datensätze nach Quadranten einfach umzusetzen. Zudem ignorieren wir alle Messwerte bis auf die Windstärke. Tabelle 1 zeigt exemplarisch einige Datensätze, die wir in unserem laufenden Beispiel verarbeiten.

Wir betonen an dieser Stelle, dass diese vereinfachenden Annahmen nur der Anschaulichkeit dienen und keine Einschränkung unseres Systems darstellen.

Quadrant *q*	Windstärke *ws*
2	0
3	7
4	9
1	3
...	...

Tabelle 1: Tabellarisch dargestellte Eingabedaten.

Wir schalten zwei MapReduce-Sequenzen in Reihe:

- Die erste Sequenz ermittelt, wie oft in einem Quadranten eine konkrete Windstärke aufgetreten ist.
- Die zweite Sequenz fasst die berechneten Tupel zu Histogrammen zusammen.

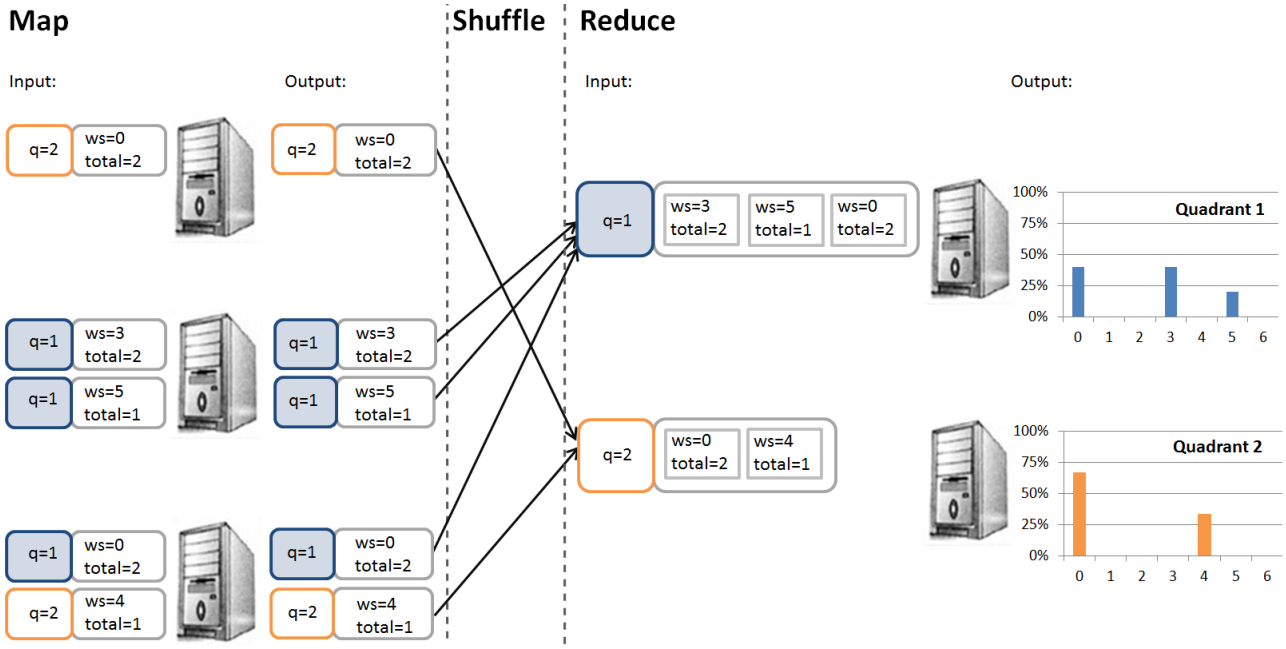

Abbildung 4: Zweite MapReduce-Sequenz zur Berechnung der Histogramme.

```
def map(Datei d, Liste<Quadrant, Windstärke> L) :
    foreach (q, ws) in L do
      if (q ∈ Q)
        int count = 1;
        emit ((q, ws), count);
      fi
    od
```

Abbildung 5: Map-Funktion der ersten Sequenz.

```
def reduce((Quadrant q, Windstärke ws), Liste<Integer> L) :
    int total = 0;
    foreach count in L do
      total += count;
    od
    emit (q, (ws, total));
```

Abbildung 6: Reduce-Funktion der ersten Sequenz.

Durch das Aneinanderreihen von MapReduce-Sequenzen werden ganz im Sinne des Prinzips „teile und herrsche“ mit sehr einfachen und gut parallelisierbaren Rechenschritten komplexe Transformationen spezifiziert. Wir betrachten nun beide Sequenzen im Detail.

3.1 Sequenz 1: Absolute Häufigkeiten

Die erste Sequenz erinnert an das „WordCount“ -Beispiel, das klassische Einsteigerbeispiel für MapReduce Programmierung [11]. Die Eingabe der Map-Funktion ist der Name einer Datei und deren Inhalt, nämlich eine Liste der Quadranten und der darin gemessenen Windstärke. Wir nehmen an, dass nur eine ausgewählte Menge von Quadranten Q interessiert, etwa um mögliche Standorte von Windkraftanlagen im Regensburger Raum zu untersuchen.

Abbildung 5 zeigt die Map-Funktion in Pseudocode. Die Anweisung *emit* produziert ein neues Ausgabetupel. In der Shuffle-Phase werden die produzierten Tupel nach der Schlüsselkomponente aus Quadrant und Windstärke umverteilt. Die Reduce-Funktion in Abbildung 6 berechnet nun die Häufigkeit der einzelnen Windstärkewerte pro Quadrant.

Beispiel 1. Abbildung 3 visualisiert die erste Sequenz anhand konkreter Eingabedaten. Die Map-Funktion selektiert nur Tupel aus den Quadranten 1 und 2 (d.h. Q = {1, 2}). Die Shuffle-Phase reorganisiert die Tupel so, dass anschließend alle Tupel mit den gleichen Werten für Quadrant und Windstärke bei demselben Rechner vorliegen. Hadoop fasst dabei die count-Werte bereits zu einer Liste zusammen.

Die Reduce-Funktion produziert daraus Tupel mit dem Quadranten als Schlüssel. Der Wert setzt sich aus der Windstärke und ihrer absoluten Häufigkeit zusammen. □

3.2 Sequenz 2: Histogramm-Berechnung

Die Ausgabe der ersten Sequenz wird nun weiter verarbeitet. Die Map-Funktion der zweiten Sequenz ist schlicht die Identitätsfunktion. Die Shuffle-Phase gruppiert die Tupel nach dem Quadranten. Somit findet die finale Erstellung der Histogramme in der Reduce-Funktion statt.

Beispiel 2. Abbildung 4 zeigt für das laufende Beispiel die Verarbeitungsschritte der zweiten Sequenz. □

4. ARCHITEKTURBESCHREIBUNG

Unsere Vision eines Cloud-basierten *Spatial Decision Support Systems* für die Fragestellungen der Energiewende fußt fest auf MapReduce Technologie.

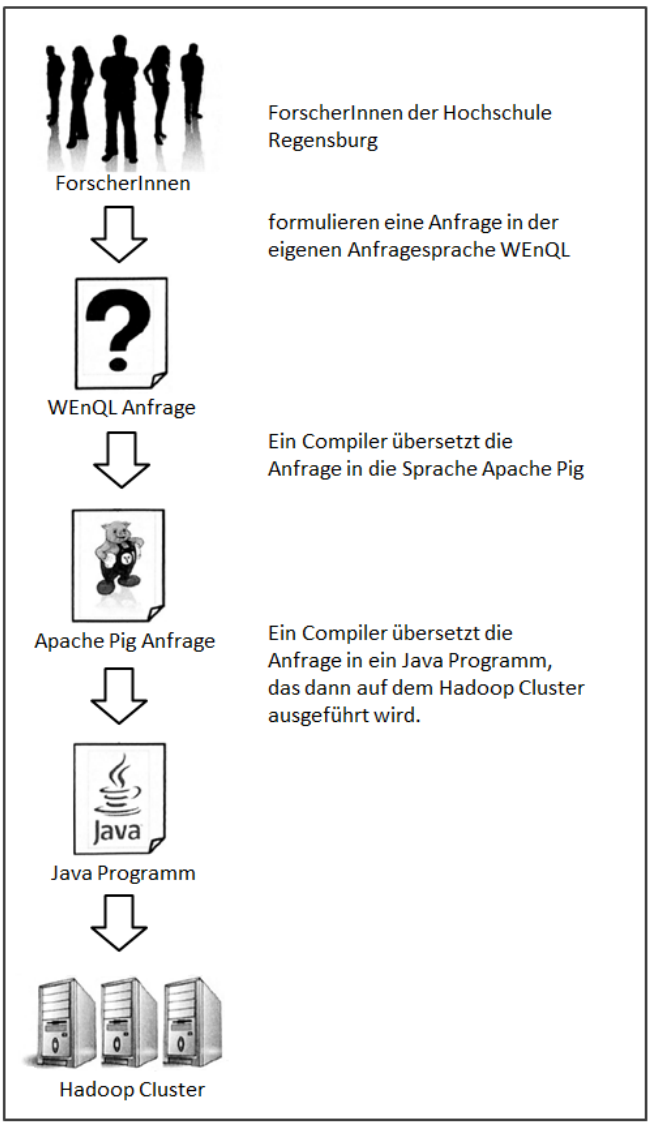

Abbildung 7: Architektur des Cloud-basierten Spatial Decision Support Systems.

Das Projektvorhaben geht dabei über den bloßen Einsatz von Cluster-Computing hinaus. Das Ziel ist der Entwurf einer domänenspezifischen Anfragesprache *WEnQL*, die *„Wetter und Energie Query Language“*. Diese wird in interdisziplinärer Zusammenarbeit mit dem Forschungsinstitut *Regensburg School of Energy and Resources* entwickelt. Mit ihr sollen auch MapReduce Laien in der Lage sein, Algorithmen auf dem Cluster laufen zu lassen.

Abbildung 7 illustriert die Vision: Unsere Nutzer formulieren eine deklarative Anfrage in WenQL, etwa um die Wind geschwindigkeits-Histogramme einer Region zu berechnen. Ein eigener Compiler übersetzt die WenQL Anfrage in das gängige Anfrageformat Apache Pig [8, 13], das wiederum nach Java übersetzt wird. Das Java Programm wird anschließend kompiliert und auf dem Hadoop Cluster ausgeführt.

Die Übersetzung in zwei Schritten hat den Vorteil, dass das Programm in der Zwischensprache Pig von Experten hinsichtlich Performanz und Korrektheit analysiert werden kann. Hadoop Laien hingegen brauchen sich mit diesen Interna nicht zu belasten. Aktuell erarbeiten wir einen Katalog konkreter Fragestellungen der Energiewirtschaft, um gängige Query-Bausteine für WenQL zu identifizieren.

5. VERWANDTE ARBEITEN

In diversen Forschungsgebieten der Informatik finden sich Berührungspunkte mit unserer Arbeit. Viele Forschungsprojekte, die sich mit der *Smart Grid* Technologie beschäftigen, setzten auf distributive Systeme zum Bearbeiten ihrer Daten [4, 17]. Ähnlich wie in unserem Projekt wird diese Entscheidung aufgrund der großen Datenmengen getroffen, welche aus unterschiedlichen Quellen stammen. Wettereinflüsse auf Kraftwerke, Börsenstrompreise, Auslastung von Stromnetzen und das Stromverbrauchsverhalten von Millionen von Nutzern müssen verglichen werden. Smart Grid Analysen unterscheiden sich von unserem Projekt darin, dass wir nur auf historische Daten zurückgreifen und somit keine Echtzeitanforderungen an das System stellen.

Fragestellungen wie Standortbestimmung und Zonierung haben eine lange Tradition in der Entwicklung dedizierter *Spatial Decision Support* Systeme [6]. Deren Architekturen fußen üblicherweise auf relationalen Datenbanken zur Datenhaltung. Mit der Herausforderung, auf *Big Data* zu skalieren, und mit heterogenen Datenquellen zu arbeiten, besitzen *NoSQL* Systeme wie Hadoop und das Hadoop Dateisystem hingegen eindeutige Vorteile in der Datenhaltung und Anfragebearbeitung.

Die Definition deklarativer Anfragesprachen für MapReduce Algorithmen ist ein sehr aktives Forschungsgebiet. Am relevantesten für uns sind SQL-ähnliche Anfragesprachen, wie etwa im Hive Projekt umgesetzt [2, 16]. Allerdings wird SQL von unseren Anwendern in der Regel nicht beherrscht. Daher planen wir, eine eigene Anfragesprache zu definieren, die möglichst intuitiv für unsere Anwender zu erlernen ist.

6. ZUSAMMENFASSUNG UND AUSBLICK

Der Bedarf für eine neue, *BigData*-fähige Generation von räumlichen Entscheidungsunterstützungssystemen für diverse Fragestellungen der Energiewende ist gegeben.

In dieser Arbeit stellen wir unsere Vision eines Cloud-basierten *Spatial Decision Support Systems* vor. Anhand des Beispiels der Windpotentialanalyse zeigen wir die Einsatzfähigkeit von MapReduce Algorithmen für strategische Fragestellungen in der Energieforschung.

Wir sind zuversichtlich, ein breites Spektrum essentieller Entscheidungen unterstützen zu können. Eine Weiterführung unserer Fallstudie ist die Ausrichtung von Windkrafträdern innerhalb eines Windparks. Dafür ist die dominierende Windrichtung entscheidend, um Windkrafträder günstig zueinander und zur Hauptwindrichtung auszurichten. Ein einzelnes Windkraftwerk kann zwar die Gondel um 360° drehen, um die Rotoren gegen den Wind auszurichten. Die Anordnung der Türme zueinander im Park ist allerdings fixiert. Bei einem ungünstigen Layout der Türme können somit Windschatteneffekte die Rendite nachhaltig schmälern. Abbildung 8 (aus [10]) visualisiert die Windstärke und Windrichtung als Entscheidungsgrundlage. Unser MapReduce Algorithmus aus Kapitel 3 kann dementsprechend erweitert werden.

Darüber hinaus eruieren wir derzeit die Standortbestim-

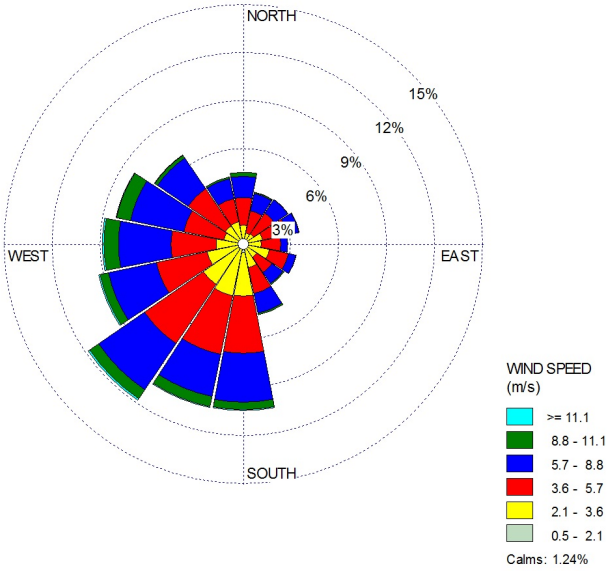

Abbildung 8: Aussagen über Windstärke und Windrichtung (aus [10]).

mung von Solaranlagen, und noch komplexer, den strategischen Einsatz von Energiespeichern, um etwa Windstillen oder Nachtphasen ausgleichen zu können.

Mit den Fähigkeiten unseres künftigen *Decision Support Systems*, der Skalierbarkeit auf sehr großen Datenmengen, dem flexible Umgang mit heterogenen Datenformaten und nicht zuletzt mit einer domänenspezifischen Anfragesprache wollen wir unseren Beitrag zu einer klimafreundlichen und nachhaltigen Energieversorgung leisten.

7. DANKSAGUNG

Diese Arbeit ist ein Projekt der *Regensburg School of Energy and Resources*, eine interdisziplinäre Einrichtung der Hochschule Regensburg und des *Technologie- und Wissenschaftsnetzwerkes Oberpfalz*.

8. LITERATUR

[1] *Apache Hadoop*. http://hadoop.apache.org/, 2013.

[2] *Apache Hive*. http://hive.apache.org/, 2013.

[3] O. Brückl. *Meteorologische Grundlagen der Windenergienutzung*. Vorlesungsskript: Windenergie. Hochschule Regensburg, 2012.

[4] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4, 2008.

[5] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008.

[6] P. J. Densham. Spatial decision support systems. *Geographical information systems: Principles and applications*, 1:403–412, 1991.

[7] *Deutscher Wetterdienst*. http://www.dwd.de/, 2013.

[8] A. Gates. *Programming Pig*. O'Reilly Media, 2011.

[9] M. Kaltschmitt, W. Streicher, and A. Wiese. *Erneuerbare Energien Systemtechnik, Wirtschaftlichkeit, Umweltaspekte*. Springer, 2006.

[10] *Lakes Environmental Software*. http://www.weblakes.com/, 2013.

[11] C. Lam. *Hadoop in Action*. Manning Publications, 2010.

[12] *National Center for Atmospheric Research (NCAR)*. http://ncar.ucar.edu/„ 2013.

[13] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig latin: A not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1099–1110. ACM, 2008.

[14] *Satel-Light*. http://www.satel-light.com/, 2013.

[15] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The Hadoop Distributed File System. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1–10, 2010.

[16] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy. Hive: A warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment*, 2(2):1626–1629, 2009.

[17] D. Wang and L. Xiao. Storage and Query of Condition Monitoring Data in Smart Grid Based on Hadoop. In *Computational and Information Sciences (ICCIS), 2012 Fourth International Conference*, pages 377–380. IEEE, 2012.