A Generalized MapReduce Approach for Efficient mining of Large data Sets in the GRID

Matthias Röhm, Matthias Grabert and Franz Schweiggert
Institute of Applied Information Processing
Ulm University
Ulm, Germany
matthias.roehm@uni-ulm.de, matthias.grabert@uni-ulm.de, franz.schweiggert@uni-ulm.de

***Abstract*—The growing computerization in modern academic and industrial sectors is generating huge volumes of electronic data. Data mining is considered the technology to extract knowledge from these data. With an ever increasing amount of data and complexity of modern data mining applications, the demand for resources is rising tremendously. Grid and Cloud technologies promise to meet the requirements of heterogeneous, large-scale and distributed data mining applications. The DataMiningGrid system was developed to address some of these issues and provide high performance and scalability, sophisticated support for different types of users, flexible extensibility features, and support of relevant standards. While the DataMiningGrid, like most of the related grid systems, focused on compute-intensive applications, Google's MapReduce paradigm and Cloud-Computing brought up new solutions for efficient data analysis. Based on the DataMiningGrid, we developed the DataMiningGrid-Divide&Conquer system that combines these important technologies into a general-purpose data mining system suited for the different aspects of today's data analysis challenges. The system forms the core of the Fleet Data Acquisition Miner for analyzing the data generated by the Daimler fuel cell vehicle fleet.**

***Keywords*-Data mining, Grid, MapReduce.**

I. Introduction

Increasing data volumes in many industrial and academic sectors are fueling the need for novel data analysis solutions. The effective and efficient management and transformation of these data into information and knowledge is considered a key requirement for success in knowledge-driven sectors. Data mining [1] is the key methodology to address these information needs through automated extraction of potentially useful information from large volumes of data. In the last decade there have been multitudes of efforts to scale data mining algorithms for solving more complex tasks, including peer-to-peer data mining, distributed data stream mining and parallel data mining.

Recently, data mining research and development has put a focus on highly data-intensive applications. Google's publications on MapReduce [2][3], a special incarnation of the Divide&Conquer paradigm, inspired many projects working on large data sets. MapReduce frameworks like Hadoop simplify the development and deployment of peta-scale data mining applications leveraging thousands of machines. MapReduce frameworks are highly scalable because they avoid data movement and rather send the algorithms to the data. In contrast to other data mining environments these frameworks restrict themselves to a certain programming model, loosing some of the functionality provided by fully featured queuing systems.

Another branch of modern distributed data mining is motivated by the sharing of heterogeneous, geographic distributed resources from multiple administrative domains to support global organizations. This field of active research and development is generally referred to as data mining in grid computing environments. The DataMiningGrid [4] project addresses the requirements of modern data mining application scenarios arising in grid environments, in particular those which involve sophisticated resource sharing. The DataMiningGrid system is a service-oriented, scalable, high performance computing system that supports grid interoperability standards and technology. It meets the needs of a wide range of users, who may flexibly and easily grid-enable existing data mining applications and develop new grip-based approaches. The DataMiningGrid, like most of the related grid systems, focused on compute-intensive applications leading to an architecture build on three components: (1) Specialized storage servers to store data and programs. (2) Compute clusters composed of multiple compute nodes for running the algorithms. (3) Grid management servers for managing the storage and compute resources connected to them.
In such an environment data is stored on dedicated storage servers and has to be transferred to the compute nodes prior to execution. Though different scheduling algorithms have been proposed to optimize the relation between data transfer and execution time, for data-intensive applications, data should not be moved at all [5].

To bring the advantages of the MapReduce paradigm into worldwide, heterogeneous computing environments we developed the DataMiningGrid-Divide&Conquer (DMG-DC) system based on the concepts and services of the DataMiningGrid project. This article is organized as follows:
First, we briefly revise MapReduce frameworks and introduce the more general Divide&Conquer paradigm for

data-intensive applications. Then we describe the DataMiningGrid and its successor, the DMG-DC system. We also introduce a real-world data mining application based on the DMG-DC: The Fleet Data Acquisition Miner (FDA-Miner) for analyzing the data generated by the Daimler fuel cell vehicle fleet. Finally, we present system evaluation results from the FDA-Miner and discuss related technologies.

II. MapReduce and Divide&Conquer

The tremendous amount of data generated in modern science and business applications require new strategies for storing and analyzing. As the amount of data increases, data can not be efficiently stored on a single storage server but has to be distributed to multiple machines. Google's MapReduce and its open-source implementations provide frameworks to mine these distributed data sets.

The name MapReduce refers to the map and reduce functions of functional programming languages. In the context of a MapReduce framework, all applications consist of a map and a reduce function [3]. The map function reads a key/value pair and produces a set of new key/value pairs. In an intermediate step all pairs are grouped by their key values. A key and its values are presented to the reduce function which produces a list of result values.
It can easily be shown, that these functions produce the same result when applied to the whole data set or to the parts of the data set.

MapReduce frameworks build an environment for executing these map and reduce functions on a cluster. Data is split up into small chunks and stored in a distributed file system comprised of multiple standard machines acting as storage *and* compute nodes [2]. A special manager node keeps track of all data chunks and their locations in the cluster. A master process manages the execution and minimizes data movement by executing the functions on the nodes containing the data to be mined. The master identifies the nodes to use for execution by asking the distributed file system manager for the location of the data chunks. If multiple copies of a chunk are available the master schedules the execution to the least used node.

Executing the functions on the nodes that contain the data is the key to the high performance and scalability of MapReduce frameworks. As not data, but algorithms are transferred, MapReduce frameworks are perfectly suited for Clouds because they do not require information about server location and network bandwidth as traditional systems need for data scheduling.

Restricting themselves to only two functions, MapReduce frameworks are easy to program and simple to set up. However, not all data-intensive applications can be decomposed into map and reduce functions. Especially the integration of existing data mining programs is sometimes impossible.

MapReduce can be viewed as a special form of the Divide&Conquer paradigm, where a problem is split into smaller sub problems that are easier to solve. This more general paradigm does not impose any restrictions on the functions or the number of processing steps. Applied to data-intensive applications, Divide&Conquer may be defined as follows: An arbitrary function f is executed in parallel on the selected subsets d of the data D:

$$f : D \rightarrow R, \quad f(d) = r_d, \ \forall d \in D$$

The results R may be processed by another function g,

$$g : R \times R \times ... \rightarrow S$$

generating the results S, which again may be processed by another function.
A series of such execution steps can be represented by a direct acyclic graph, where each node is a function and the vertices symbolize the data flow between the functions.

A data-intensive application based on this Divide&Conquer definition requires a distributed computer system providing:
(1) A distributed file system or a data registry to locate the subsets of the data. (2) A scheduler to execute the functions on the nodes containing the data subsets. (3) A workflow manager to coordinate the execution of each function in the direct acyclic graph.
Due to their specialized approach, the components of current MapReduce frameworks can not simply be reused to build a Divide&Conquer system, especially when such a system should be integrated in an environment comprised of heterogeneous, geographic distributed resources from multiple administrative domains. Therefore the flexible DataMiningGrid system was enhanced to natively support Divide&Conquer jobs.

III. Data Mining in the Grid: DataMiningGrid

In general, a grid-enabled data mining system should support the seamless and efficient sharing of data, data mining application programs, processing units and storage devices. As data mining is used by a wide variety of users and organizations such a system should not only address the technical issues but also pay attention to the unique constraints and requirements of data mining users and applications. In the DataMiningGrid project, use case scenarios from a wide range of application areas were analyzed to identify the key requirements of grid-based data mining that can be summarized as follows:
A grid-based data mining environment should offer benefits like increased performance, high scalability to serve more users and more demanding applications, possibilities for creation of novel data mining applications and improved exploitation of existing hardware and software resources.
Grid-enabling data mining applications should not require modification of their source code. The system should not

be restricted to specific data mining programs, tools, techniques, algorithms or application domains and should support various types of data sources, including database management systems (relational and XML) and data sets stored in flat files and directories.
To support the different user groups, intricate technological details of the grid should be hidden from domain-oriented users, but at the same time users with a deep knowledge of grid and data mining technology should be able to define, configure and parameterize details of the data mining application and the grid environment.

In order to address these requirements, the DataMiningGrid system was designed according to three principles: services-oriented architecture (SOA), standardization and open technology. The early adoption of two important distributed computing standards, the Open Grid Service Architecture (OGSA) and the Web Services Resource Framework (WSRF) were essential for succeeding projects, like the one presented in this article. The OGSA is a distributed interaction and computing architecture based on the concept of a grid computing service, assuring interoperability on heterogeneous systems so that different types of resources can communicate and share information. The WSRF refers to a collection of standards which endorse the SOA and proposes a standard way of associating grid resources with web services to build stateful web services required by the OGSA.

Following these principles, the DataMiningGrid project implemented various components based on existing open technology: Data management, security mechanisms, execution management and other services commonly needed in grid systems are provided by the Globus Toolkit 4 (GT 4) grid middleware.

Three higher-level components for data, information and execution management form the core of the DataMiningGrid system. The *data components* offer several data transformation and transportation capabilities to support typical data operations for data mining applications. The *Information Service* collects and manages all information about the data mining programs available in the system. The *Resource Broker* is responsible for matching available resources to job requests, global scheduling of the matched jobs and executing, managing and monitoring of jobs, including data stage in and out operations.

The main user interface is the Triana workflow environment. In combination with special data mining units, Triana enables users to build complex grid-based data mining applications.

The DataMiningGrid Application Description Schema (ADS) is the link between all the DataMiningGrid components. The ADS covers the complete life-cycle of a data mining program and is used for discovering, configuring and executing data mining programs.

Although the necessity to address data-intensive applications was recognized in the DataMiningGrid project, due to time constraints, the project focused more on compute-intensive applications. Hence, three functionalities needed for Divide&Conquer jobs are not available in the DataMiningGrid system:

1) The Resource Broker [6], like other grid resource brokers, is only able to schedule jobs to whole clusters. As a consequence, jobs can not be placed directly on certain machines inside a cluster, as required by Divide&Conquer jobs.
2) There is no specialized data registry that could be used for scheduling data-intensive applications.
3) There is no server-side workflow execution component to coordinate the steps of Divide&Conquer jobs.

The following section describes the changes made to the DataMiningGrid components and the new features of the DMG-DC system to support Divide&Conquer jobs in grid environments.

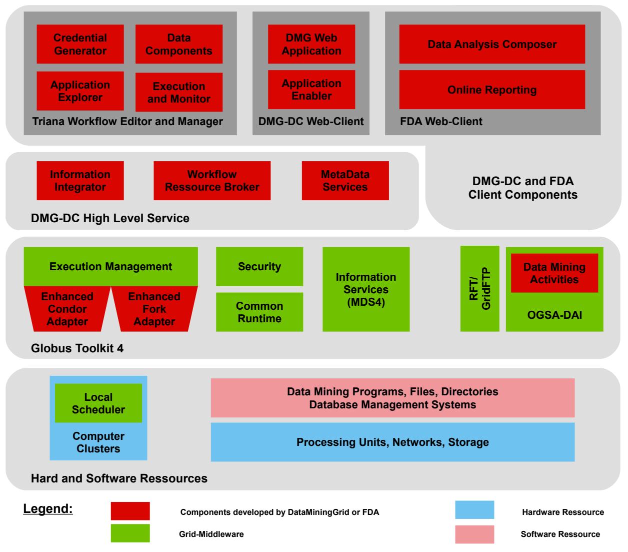


Figure 1. The DMG-DC architecture

IV. DMG-DC

The DMG-DC system is designed to support the different aspects of today's data analysis challenges. Based on the DataMiningGrid project, which already implemented many features needed for grid-based data mining, the DMG-DC development focused on the functionality to support extremely data-intensive applications. The flexible and extendable design of the DataMiningGrid system made it easy to integrate the missing functionality.

Consequently, the architecture of the DMG-DC, depicted in Figure 1, does not differ significantly from the DataMiningGrid [4], except for the new and redesigned components: The *Data Registry Service* (DRS) and the *Workflow Resource Broker* (WRB).

A. Data Registry Service

A data registry is a central component for executing Divide&Conquer jobs in a grid, as it provides the locations of all data sets to the Resource Broker. Without this information the Resource Broker would not be able to schedule jobs to the nodes containing the data to be mined. The developed distributed registry consists of a number of WSRF-compliant Data Registry Services that store user-defined metadata describing the data sets available in the grid. In contrast to the distributed file system of a MapReduce framework, the DRS only stores information about the data, leaving the actual storage to database management or file systems. Therefore, the Divide&Conquer mechanism can be applied to data stored in any storage system and is not limited to a specific distributed file system.

The DRS stores metadata in user-defined categories, which specify a list of logical and physical attributes describing the data. Logical attributes typically hold information about the content or creation process of the data set, whereas physical attributes include storage location, size or data format information. When a new data set is registered with a category, a logical and a physical object is created with unique object names. These objects contain the logical/physical attributes of that data set and are used to model replication: A logical object may reference any number of physical objects.

A single DRS may store the metadata information of all data sets in the grid. To improve reliability and performance several DRS may run on different sites in the grid. Multiple DRS automatically form a peer-to-peer network, forwarding client search requests and category information to the appropriate DRS.

A distinctive feature of the DRS, compared to other grid data registries like the Globus Toolkit Replica Location Service, is its advanced search mechanism enabling clients to search for data using multiple attributes within a single query.

B. Workflow Resource Broker

The new requirements arising from Divide&Conquer jobs led to the development of the DMG-DC Workflow Resource Broker. The WRB was designed not only to support Divide&Conquer jobs but also include all features of the DataMiningGrid Resource Broker [6]. The two main new features of the WRB are the workflow execution manager and the advanced job scheduler, able to place jobs on specific nodes inside a cluster.

As depicted in Figure 2, the WRB consist of 5 components communicating through well-defined interfaces:
Clients connect to the *workflow manager* to submit workflows, monitor and manage workflow execution. A workflow consists of one or more jobs, each described by an ADS instance, and dependencies between these jobs. The workflow manager is responsible for executing all jobs as

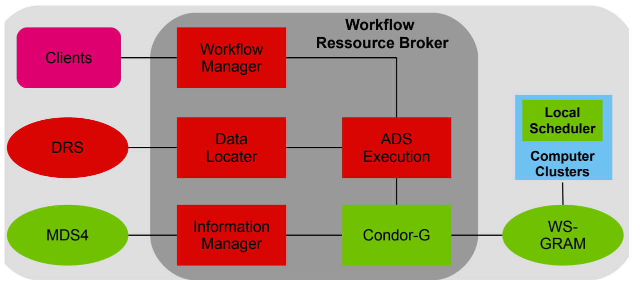


Figure 2. The components of the Workflow Resource Broker.

specified in the workflow. To start the execution of a single job, the workflow manager sends the corresponding ADS instance to the *ADS execution* component. The execution component analyzes the ADS instance and connects to the *data locator* to get the locations of all data sets specified in the ADS instance. The data locator acts as an interface to different data registries, although currently only DRS is supported. The execution component combines the data locations with the ADS definitions and generates a Condor-G job description. The job description contains all scheduling information necessary for placing Divide&Conquer jobs on nodes providing the selected data sets. *Condor-G* [7] is a powerful grid task broker providing advanced scheduling, execution and managing capabilities and an uniform interface to different grid execution management systems. A key feature of Condor-G is its ClassAd mechanism for describing and matching of jobs and compute resources. The flexible Condor-G ClassAd mechanism enables the ADS execution component to define a scheduling policy, resource and job parameters, so that Divide&Conquer jobs can be placed on a specific node in a cluster of the grid. The *information manager* collects all resource information necessary for this scheduling from the Globus Toolkit Monitoring and Discovery System (MDS4) and delivers it to Condor-G.
When receiving a job, Condor-G matches the job with all available resources in the grid and submits the job to the Globus Toolkit execution management service (WS-GRAM) providing the best match. Divide&Conquer jobs contain a special element that advises the WS-GRAM to start this job on the specified node(s), even if they are inside a cluster.

V. FDA-MINER

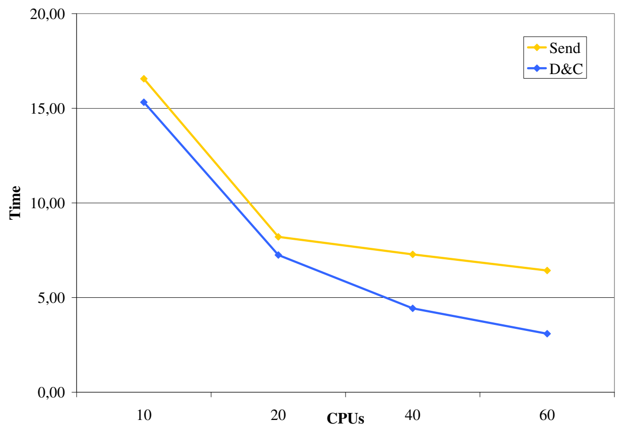
The presented DMG-DC system forms the basis of the Fleet Data Acquisition Miner (FDA-Miner) for analyzing the data generated by the Daimler fuel cell vehicle fleet. Daimler AG has been involved in fuel cell technology for more than 15 years and has released the largest fleet of zero emission fuel cell vehicles in the world with more than 100 vehicles [8]. The purpose of these operations is to test these vehicles in the hands of selected customers in everyday operations under varying climatic conditions, traffic conditions and driving styles in different locations worldwide. In order to gain the most experience for future fuel cell vehicle development, a fleet data acquisition system has been developed which continuously records all relevant parameters of vehicle operation, such as the fuel cell voltage, current and temperatures. The enormous amount of worldwide distributed data produced by the fleet - over 4 million kilometers have been recorded - and the need for compute-intensive data analysis methods were the key drivers behind the development of the DMG-DC system [9].

The FDA-Miner provides a user friendly web-based data analysis application for mining the fuel cell data. In addition to specialized visualization and reporting features, it offers a flexible front end to configure customized data mining tasks. The application uses the services of the DMG-DC to retrieve information about the available data and analysis programs. For each user defined task, the application creates the appropriate ADS instances and workflow definitions and submits a Divide&Conquer job to the WRB.

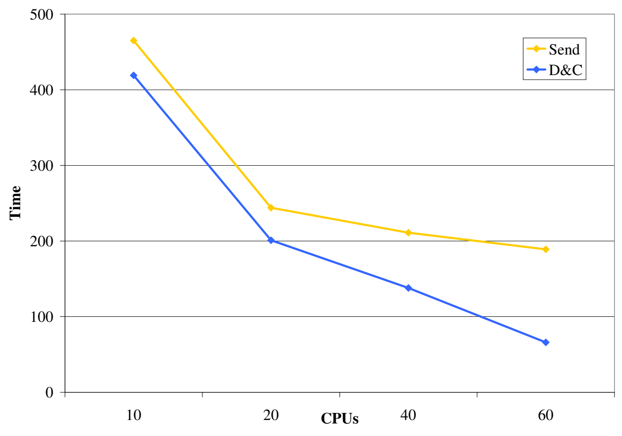
The FDA-Miner programming toolbox supports users implementing specialized Divide&Conquer data mining algorithms. The toolbox provides Perl and C modules to read the fuel cell data sets and templates for parallel data processing and combination steps.

VI. Evaluation

The FDA-Miner has been already heavily used in production and successfully computed thousands of Divide&Conquer and other jobs. The following evaluation therefore focuses on the advantages of the Divide&Conquer functionality of the DMG-DC system compared to traditional grid systems, like the DataMiningGrid, with dedicated storage servers. The evaluation set-up consisted of 9 dual quad core machines with direct attached storage connected over a 1 GBit Ethernet network. To measure the performance of the DMG-DC Divide&Conquer functionality, a subset of the fuel cell data was randomly distributed over all 9 machines and each file was placed on at least two machines. Traditional grid systems were represented by a representative scenario with 1 storage server serving the data to 8 compute nodes.
A typical FDA-Miner data analysis job, filtering the data and computing various statistical properties, was executed on both set-ups. Figure 3 shows the overall time for performing this job while varying the number of CPUs and the size of the data set. The results demonstrate that the DMG-DC (blue curve), like MapReduce systems, scales well for common data analysis jobs. Traditional grid systems (orange curve) on the other hand have to send the data to the compute nodes first. Depending on the network bandwidth the transfer time may, for simple data filtering operations, even exceed the time for data processing. Scaling of these systems is also limited by the number of concurrent connections the storage server can handle without dropping network throughput. In the presented evaluation set-up the storage server can only



(a) 11694 data sets



(b) 256530 data sets

Figure 3. Execution time of a job in Send and Divide&Conquer mode.

deliver enough throughput to server about 20 CPUs and therefore does not scale well above that point.

VII. Related work

Recently, various systems and approaches to grid-based data mining and MapReduce have been reported in the literature. Some of those, that are particularly relevant to the DMG-DC system, are briefly reviewed here.

GridMiner [10] is designed to support data mining and online-analytical processing in distributed computing environments. GridMiner implements a number of common data mining algorithms, some as parallel versions, and supports various text mining tasks. Two major differences between GridMiner and DMG-DC are the Divide&Conquer functionality and that the latter complies with the recent trend towards WSRF.

Knowledge Grid (K-Grid) [11] is a service-oriented system providing grid-based data mining tools and services. The K- grid system can be used for a wide range of data mining and related tasks such as data management and knowledge representation. The system architecture is organized into a High-level K-Grid Services and a Core-level K-Grid Services layer, which are built on top of a Basic grid Services layer. K-Grid incorporates some interesting features for distributed data mining but no Divide&Conquer or similar functionality is available at the moment.

Hadoop [12] is the most well known open source implementation of Google's MapReduce paradigm. Hadoop's MapReduce framework is build on top of the Hadoop distributed file system (HDFS) containing all data to be mined. The map and reduce function are typically written in Java, but even executables can be integrated via a streaming mechanism. As MapReduce frameworks like Hadoop do not offer the functionality to execute compute-intensive applications (MPI, PVM) on the cluster, making them unsuitable for a general-purpose data mining system. Hadoop On Demand and Oracle Grid Engine try to overcome these limitations by running Hadoop on top of a cluster queuing system, thus adding another layer of complexity. Still, both reserve the nodes to use for MapReduce exclusively, making them unusable by other jobs. Hadoop and similar MapReduce frameworks simplify the development and deployment of data-intensive applications on local clusters and cloud resources . But, in contrast to the DMG-DC system, these frameworks are currently not suited for large-scale, heterogeneous environments with multiple independent virtual organizations.

VIII. Conclusion

In this article we introduced Divide&Conquer, a generalized MapReduce paradigm, for data-intensive applications. The developed DMG-DC system provides the functionality to run diverse data mining applications, including Divide&Conquer, in a worldwide, heterogeneous grid environment. As not data, but algorithms are transferred, Cloud resources can be used to scale the system on demand. The FDA-Miner, a real world data analysis application, uses the distinct features of the DMG-DC to efficiently mine the data of the Daimler fuel cell vehicle fleet. The FDA-Miner evaluation results highlight the advantages of the DMG-DC compared to traditional grid systems.

Future work may include the integration of more powerful data management systems like the Storage Resource Broker and a generalized version of the FDA-Miner programming toolbox.

References

[1] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "The kdd process for extracting useful knowledge from volumes of data," *Commun. ACM*, vol. 39, no. 11, pp. 27–34, 1996.

[2] S. Ghemawat, H. Gobioff, and S. T. Leung, "The google file system," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 29–43, 2003.

[3] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," 2004, pp. 137–150. [Online]. Available: http://www.usenix.org/events/osdi04/tech/dean.html

[4] V. Stankovski, M. Swain, V. Kravtsov, T. Niessen, D. Wegener, M. Röhm, J. Trnkoczy, M. May, J. Franke, A. Schuster, and W. Dubitzky, "Digging deep into the data mine with datamininggrid," *IEEE Internet Computing*, vol. 12, no. 6, pp. 69–76, 2008.

[5] K. Ranganathan and I. Foster, "Decoupling computation and data scheduling in distributed data-intensive applications," in *HPDC '02: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*. IEEE Computer Society, 2002, pp. 352–358.

[6] V. Kravtsov, T. Niessen, V. Stankovski, and A. Schuster, "Service-based resource brokering for grid-based data mining," in *Proceedings of The 2006 International Conference on Grid Computing and Applications*, Las-Vegas, USA, 2006.

[7] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke, "Condor-g: A computation management agent for multi-institutional grids," *Cluster Computing*, vol. 5, no. 3, pp. 237–246, July 2002.

[8] J. Friedrich, R. Schamm, C. Nitsche, J. Keller, B. Rehfus, T. Frisch, and M. Röhm, "Advanced on-/offboard diagnostics for a fuel cell vehicle fleet," in *Society of Automotive Engineers SAE World Congress 2008*, 2008.

[9] M. Röhm, J. Keller, and T. Hrycej, "Data mining fuel cell fleet data for stack degradation analysis," in *Fuel Cell Seminar, San Antonio*, 2007.

[10] B. Peter and W. Alexander, "Grid-aware approach to data statistics, data understanding and data preprocessing," *International Journal of High performance Computing and Networking*, vol. 1, no. 6, pp. 15–24, 2009.

[11] A. Congiusta, D. Talia, and P. Trunfio, "Distributed data mining services leveraging wsrf," *Future Generation Computer Systems*, vol. 23, no. 1, pp. 34–41, 2007.

[12] T. White, *Hadoop: The Definitive Guide*, 1st ed. O'Reilly Media, 2009.