Ersch. in: Objects and databases : third international conference, ICOODB 2010, Frankfurt/Main, Germany, September 28 - 30, 2010; proceedings / Alan Dearle ... (eds.). - Berlin [u.a.] : Springer, 2010. - S. 25-39. - (Lecture notes in computer science ; 6348). - ISBN 978-3-642-16091-2

The Case for Object Databases in Cloud Data Management

Michael Grossniklaus

Dipartimento di Elettronica e Informazione, Politecnico di Milano
P.za Leonardo Da Vinci, 32
I-20133 Milano, Italy
grossniklaus@elet.polimi.it

Abstract. With the emergence of cloud computing, new data management requirements have surfaced. Currently, these challenges are studied exclusively in the setting of relational databases. We believe that there exist strong indicators that the full potential of cloud computing data management can only be leveraged by exploiting object database technologies. Object databases are a popular choice for analytical data management applications which are predicted to profit most from cloud computing. Furthermore, objects and relationships might be useful units to model and implement data partitions, while, at the same time, helping to reduce join processing. Finally, the service-oriented view taken by cloud computing is in its nature a close match to object models. In this position paper, we examine the challenges of cloud computing data management and show opportunities for object database technologies based on these requirements.

1 Introduction

Database management systems are used in a wide variety of applications, ranging from mobile or embedded scenarios to large-scale solutions to support data-intensive and global applications. To address the requirements of different applications, different database technologies have emerged and the consensus today is that "no size fits it all" [1]. Therefore, one of the challenges has become to match technologies to requirements. The vision of cloud computing is to solve this problem by making computing a commodity that adapts to initial application requirements, but can also evolve and gracefully scale when these requirements change.

While people from different fields have slightly different definitions of the term *Cloud Computing*[1], the common denominator of most of these definitions is to look at processing power, storage and software as commodities that are readily available from large infrastructures and, thus, no longer have to be provided by desktop computers or local servers. As a consequence, cloud computing unifies elements of distributed, grid, utility and autonomic computing to provide software,

[1] Multiple Experts Try Defining "Cloud Computing": http://tech.slashdot.org/article.pl?sid=08/07/17/2117221

platforms and infrastructure as a service. At the lowest level, Infrastructure-as-a-Service (IaaS) offers resources such as processing power or storage as a service. Examples include Amazon's Elastic Compute Cloud (EC2)[2], Sun Cloud[3] and GoGrid[4]. One level above, Platform-as-a-Service (PaaS) provides development tools to build applications based on the service provider's API. Notable solutions on this level are Microsoft's Windows Azure Platform[5] and the Google App Engine[6]. Finally, on the top-most level, Software-as-a-Service (SaaS) describes the model of deploying applications to clients on demand.

The impact of cloud computing on data management research, and on query processing in particular, has recently been studied by Abadi [2] and Gounaris [3], respectively. Abadi argues that characteristics such as scalability through parallelization, storing data on untrusted hosts, and wide geographic distribution or replication, render cloud computing unsuitable for transactional data management. These applications are typically quite write-intensive and require strict ACID guarantees. Both of these characteristics do not match well with the properties of a cloud computing environment. However, analytical data management applications that mostly query large data stores for decision support or problem solving will profit from these properties. Further, this type of application is also becoming increasingly important both in science and industry [4]. Both authors agree that the requirements of data management in cloud computing can be partially addressed by integrating existing results from database research into systems that combine features from parallel, distributed and stream database management systems. However, they also point out that, in order to deliver on the cloud computing vision, hybrid solutions that integrate other execution paradigms have to be considered to better support complex analytic and extract-transform-load tasks [5].

The direction into which cloud computing data management is evolving, makes it an ideal setting to investigate the use of *object databases* as they have become a popular choice for data-intensive analytical processing tasks. For instance, the Herschel project[7] of the European Space Agency (ESA) uses the Versant Object Database[8] to store, manage and process all data gathered by the telescope in outer space. Another example is Objectivity/DB[9] that is often selected for analytical scenarios such as the Space Situational Awareness Foundational Enterprise (SSAFE) that tracks space debris in real-time to avoid collisions in future space missions. Additionally, Objectivity has recently released a version of their product that can be deployed on cloud computing infrastructures.

[2] `http://aws.amazon.com/ec2/`
[3] `http://www.sun.com/solutions/cloudcomputing/`
[4] `http://www.gogrid.com`
[5] `http://www.microsoft.com/windowsazure/`
[6] `http://appengine.google.com/`
[7] `http://www.esa.int/herschel`
[8] `http://www.versant.com`
[9] `http://www.objectivity.com`

In this position paper, we will examine the exact requirements of cloud computing data management and, based on these challenges, demonstrate the corresponding opportunities for and benefits of object databases. We begin in Sect. 2 by introducing the three main challenges of cloud computing data management. The current state of the art in cloud computing data management is highlighted in Sect. 3. As this body of research is very vast, we have decided to point out the most influential approaches, rather than to give a comprehensive survey. In Sect. 4, we revisit the three main challenges and show how object database technologies can be leveraged to address these requirements. We conclude in Sect. 5.

2 Challenges of Cloud Data Management

The challenges of cloud computing data management can be summarized as massively parallel and widely distributed data storage and processing, integration of novel processing paradigms as well as the provision of service-based interfaces. In the following, we will examine each of these challenges in more detail.

2.1 Parallel and Distributed Data Storage and Processing

One of the promises of cloud computing is to make computing power a commodity that can easily scale based on the current requirements of the application. This elasticity is typically achieved by allocating more resources in terms of servers, processor cores or memory and storage space to an application. As a consequence, leveraging these additional resources is more complex than migrating an application to a single more powerful machine. As pointed out in [2], the workload of an application has to be parallelizable in order to profit from cloud computing. In the context of data management, this means that applications which have been designed to run on a shared-nothing architecture [6] are good candidates for cloud computing.

Replication and distribution of data are further important characteristics of cloud computing data management [2]. Apart from delivering scalable computing power, reliability and quality of service are important goals of cloud computing. In terms of data management, this translates to providing highly available and durable cloud storage solutions. On the one hand, replication on a large geographic scale serves a dual purpose in this setting. First, the possibility to transparently replicate data ensures availability and durability. Second, a global network of cloud storages also allows the computation to be moved close to the client. Distribution, on the other hand, will arise as a natural consequence of the service-oriented architecture of cloud computing. In this service-based setting, it is foreseeable that a loosely coupled form of distribution will have to be supported, where many of the traditional assumptions about the data schema, distribution and statistics are no longer guaranteed to hold.

As we will discuss in the next section, both parallelism and distribution are topics that have been studied in detail. While these results are a good starting point, their applicability in widely distributed and heterogeneous settings is

limited [3]. As a consequence, the challenge of parallelism and distribution in cloud computing data management lies in developing paradigms and technologies that are capable to address the requirement of massively parallel and widely distribution data processing and storage.

2.2 Integration of Novel Processing Paradigms

Data stream management systems [7] have introduced a processing paradigm that is different from the one of traditional database management systems. Instead of dynamically running queries over mostly static data, data stream management system register static queries over dynamic data. This property has made them a successful choice in the analysis of large volumes of rapidly changing data that arise, for example, in the real-time monitoring of complex systems. Most data stream management systems operate by evaluating the registered queries over so-called windows that extract a finite set of data out of an otherwise infinite data stream. Depending on the systems capabilities, these windows are advanced based on their size or at predefined intervals and the results can be recomputed or maintained incrementally. As applications in this domain are becoming more important, data processing in cloud computing needs to be applicable to both stored and streaming data [3].

By its very nature, cloud computing takes a very service-centric view on computing. In terms of data management, this signifies that database management systems will only be accessible through service-based interfaces. This development can already be observed on the Web today as many data sources are exposed using interfaces such as REST or SOAP, rather than a traditional database interface. The consequences of this evolution to data management and, in particular, data processing are manifold. In traditional database management systems, the performance of query processing depends on a number of factors. At query compilation-time, precise statistics about data distribution are required by the optimizer in order to determine the evaluation order of the query and chose the most appropriate physical implementations of the logical operators. At run-time, the query execution time is additionally governed by the storage layout, data clustering and access methods such as indexes. While traditional query processing is by no means trivial, the problem becomes even more complex in the setting of service-based data sources. As the data itself might not be under the control of the data management system, it is difficult to obtain complete and reliable data statistics. However, as query execution time is mainly dominated by the time it takes to invoke a service and the distribution of the data it returns [8], precise statistics are even more important. A successful cloud computing data management system will need to address these challenges in order to leverage data sources with service-based interfaces.

Finally, the paradigm of map-reduce [9] has recently been proposed to execute massively parallel data processing tasks. The name “map-reduce” stems from the fact that these systems decompose processing tasks into a map and a reduce step. Both functions are provided by the client of the system and defined based on a data set that is given as $\langle key, value \rangle$ pairs. The map function is defined as

$map(k_i, v_1) \rightarrow \texttt{list}(k_j, v_j)$, i.e. it takes an input data tuple of one domain and returns a list of output data tuples of another domain. In the first processing step, a map-reduce system applies the map function to all input in the data set. It then collects the resulting lists of output tuples and regroups them according to keys. Then the reduce function, defined as $reduce(k_2, \texttt{list}(v_2)) \rightarrow \texttt{list}(v_3)$, is applied to each group producing zero or more result values. Finally, these values are collected in one list and returned to the client. Typically, the map operation works as a kind of fork that splits the data set into smaller pieces, while the reduce operation is comparable to a join or aggregation that reassembles the final result. Therefore, the authors of [5] argue that map-reduce is more similar to extract-transform-load systems than to a database system. However, since neither map-reduce nor traditional parallel database systems can deliver the full promise of cloud computing data management, there is agreement that hybrid solutions that integrate both processing paradigms have to be built [2].

2.3 Provision of Service-Based Interfaces

In Sect. 2.2, we have reasoned that a cloud computing data management system has to be able to process tasks over service-based data sources. In this section, we make the complementary argument and motivate why cloud computing data management systems themselves have to offer a service-based interface.

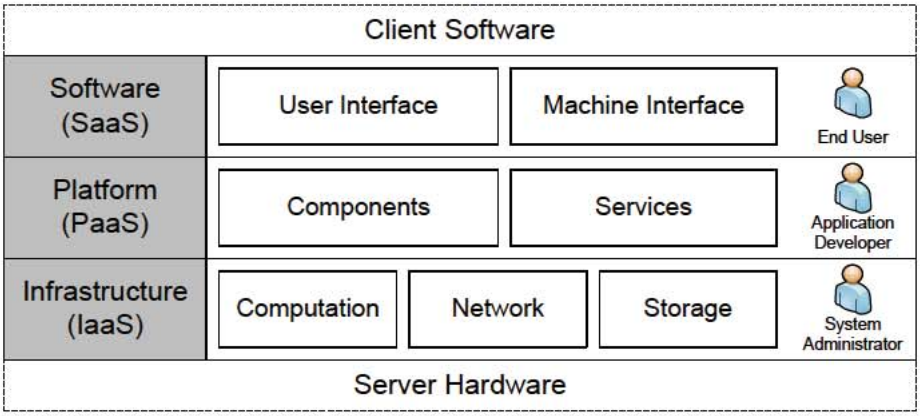

Fig. 1. Cloud computing stack, adapted from Wikipedia (`http://www.wikipedia.org`)

Figure 1 shows an overview of the cloud computing stack as it is generally agreed upon. The lowest layer, Infrastructure-as-a-Service (IaaS), provides computing resources such as processing power, network and data storage as services. The main goal of this layer is to deliver on the previously mentioned promise of providing elastic computing resources that scale gracefully in correspondence to the application's demand. Typically, this elasticity is achieved using platform

virtualization. As a consequence, scalability signifies more resources, i.e. more processor cores, more network bandwidth and more storage space, rather than migrating an application to a more powerful machine behind the scenes. The IaaS layer is built on top of the server layer that consists of hardware products that support the delivery of cloud services through technologies as multi-core processors and hardware virtualization.

The aim of the Platform-as-a-Service (PaaS) layer is to provide an integrated computing platform that facilitates the development and deployment of applications. As the PaaS layer is built on the service-based interfaces of the IaaS layer, PaaS solutions typically offer software components and high-level services that can be used to implement applications. Examples of high-level services include relational database engines, payment services or costumer relationship management. Apart from development support, some PaaS solution also offer application hosting in the sense that an application that has been implemented using the offered components and services can be deployed to the vendor's cloud computing infrastructure.

Finally, the Software-as-a-Service (SaaS) provides end-user applications as a service. The advantages of delivering software as a service over the Internet are numerous. SaaS eliminates the need for customers to install the software on their local machines, while, at the same time, always working with the most up-to-date version of the application. For the software manufacturer, the advantage of SaaS lies in better control over the licenses that are in use and prevention of illegal duplication of the application. Cloud application services are built using the service-based interfaces of the PaaS layer. Furthermore, they typically expose service-based interfaces themselves in order to enable application interoperability.

With respect to data management, we note that there are challenges on all three layers of the cloud computing stack. Cloud computing data management starts on the IaaS layer with the provision of appropriate storage management services. In particular, the challenge on this layer is to provide a service-based interface for the functionality that corresponds to the physical layer of a traditional database management system. A declarative interface to the cloud computing data management system that facilitates data definition, manipulation and querying would typically be situated on the PaaS layer. In particular, data processing functionality such as traditional query evaluation or the aforementioned novel paradigms will be realized on this layer and offered using service-based interfaces to upper layers. Finally, user interfaces such as database browsing, designing or administration applications will be provided on the SaaS layer. Apart from these generic tools, also custom end-user applications will be deployed and hosted on this layer. As a consequence, a cloud computing data management system needs to address the challenge of providing service-based interfaces at various levels. Furthermore, system components located at higher layers need to be capable of leveraging and orchestrating the services on lower layers to implement their functionality.

3 State of the Art

Since cloud computing data management is a rather new discipline, the current state of the art is still quite limited. However, several works exist that identify challenges of cloud computing data management [2,3]. The requirement of storing and processing data in a parallel and distributed setting has been addressed in several existing works. On the one hand, research on *parallel databases* has led to a good understanding of parallelism, both in architectures [6] and query processing [10]. Research on *distributed databases* [11], on the other hand, has lead to results that can be leveraged to address distributed query processing [12].

Due to the focus on analytic data management, processing tasks in cloud computing are expected to be complex and long-running. Therefore, another requirement is fault tolerance in terms of self-optimizing and self-healing systems. This requirement relates cloud computing data management to the field of *autonomic computing* [13] and corresponding approaches to query processing (e.g. [14]). Another field of interest is *adaptive query processing* [15], where adaptive execution models (e.g. [16]) have been developed that demonstrate how entire query plans can be continuously adapted to available resources. The same goal is attained by adaptive query operators (e.g. [17]) that provide adaptation within individual nodes of an otherwise static query plan.

Data stream management systems [7] (e.g. [18]) are often mentioned for their capability to process rapidly changing data sets. However, it has to be noted that their data processing paradigm is fundamentally different from traditional query processing. As discussed in [3], significant advances have been made in this area in terms of query optimization, but they do not yet extend to the widely distributed and massively parallel setting of cloud computing.

With respect to integrating traditional data processing and service-oriented computing, approaches such as query processing over Web services [19,20] or search computing [21] have also to be considered relevant. In [19], a general-purpose Web Service Management System (WSMS) is described that can optimize and execute select-project-join queries spanning multiple Web services. The authors of [20] propose the "Serena" (service-enabled) algebra and a corresponding execution environment. Serena is based on the relational algebra, extended with service calls that can either be "get" or "set" calls and are classified according to whether they have or do not have side-effects. The authors also present rewrite rules for Serena that form the basis of rudimentary optimizations. Finally, search computing [21] extends query processing over services to search engines that return ranked results. The query processor developed in the project follows a traditional database approach in the sense that declarative queries are transformed into a logical plan. This plan is then optimized and translated into an executable physical plan. The optimizer choses the best query plan using a branch-and-bound algorithm that uses heuristics for determining the plan topology and load-balancing. Similarly to a traditional databases management system, the query execution environment supports different implementations of

the logical operators such as join strategies that govern in which order the results from two search services are combined.

The field of *grid computing*, which can be considered a predecessor to cloud computing, also takes a service-centric approach on computing. Data management in grid computing has been studied intensely by several surveys (e.g. [22]) and research in this field has yielded results both in grid database systems (e.g. [23]) and grid query processing (e.g. [24]).

MapReduce [9] and related software are designed with built-in fault tolerance and capable of processing massively parallel and complex execution tasks at a large scale. Early approaches that point into the direction of integrating database functionality and map-reduce software have already been proposed, e.g. Yahoo's Pig Latin [25] or Microsoft's DryadLINQ [26], and SCOPE [27]. These approaches are only able to integrate the two paradigms at the language and not at the system level, since they layer map-reduce interfaces on top of traditional parallel databases. The approach of a hybrid architecture that supports multiple paradigms side-by-side is taken by HadoopDB [28] and Clustera [29], an integrated computation and data management system that is capable of executing database queries, workflows over Web services and map-reduce processes. In [30], a benchmark for large-scale data analysis systems is defined and implementation concepts for future hybrid systems are recommended.

In summary, numerous approaches exist that contribute to addressing these requirements. However, the core challenge of extending and integrating them into a comprehensive platform has not been fully addressed so far. Furthermore, most of these existing works were conducted in the context of the relational model. We believe that it is also imperative to study the possibility to exploit technologies from *object databases* [31] for cloud computing data management. This claim is motivated by the above-mentioned observation that analytical data management applications will benefit most from cloud computing and the fact that this class of applications is an ideal use case for object databases. Existing results from the domain of object databases that are relevant in this context include the works on object algebras (e.g. [32]), query processing (e.g. [33]), and query optimization (e.g. [34]). Further, approaches such as OMS Connect [35] have shown how features unique to object databases can be used to support multi-databases and modular peer-to-peer databases.

4 Opportunities for Object Databases

Object databases are a good match to both the type of data management applications that is anticipated to benefit most from cloud computing and the service-oriented view taken by cloud computing. As a consequence, we believe that cloud computing research needs to include these technologies. In order to make the case for object databases in cloud computing data management, we will now revisit the challenges outlined in Sect. 2 and show possible opportunities for object database technologies.

4.1 Parallel and Distributed Data Storage and Processing

As explained in Sect. 2, cloud computing takes parallel and distributed data storage and processing to a new level by requiring it to be massively parallel and widely distributed. In the following, we will first examine the case of data storage before looking at the case of data processing. Typically, parallel and distributed data storage is addressed through horizontal and vertical partitioning of the dataset.

In the setting of parallel databases that use the relational model, tables can be partition horizontally using selection predicates that segment a table into smaller ones. These table segments are then placed on different computing nodes and thus provide support for parallel processing of data manipulation operations. In contrast to the relational model, object data models provide more opportunities for application developers to define horizontal partitions. While the value-based approach is still possible, object databases can also leverage the existence of class extents or object collections for horizontal partitions. For example, Objectivity/DB and ObjectStore both support concepts that support the explicit clustering of objects into "containers", i.e. object collections that also govern the physical storage layout. Furthermore, Objectivity/DB is built around the concept of federated databases that could prove helpful in realizing such dataset partitions.

A vertical partition in a relational database management system segments a table in terms of columns, i.e. a (not necessarily strict) subset of the columns of a table are placed on different computing nodes. Object data models also provide ampler possibilities to realize vertical partitions. For example, the object-slicing technique [36] could be used to partition classes by leveraging the inheritance hierarchy and to distribute object data accordingly. Additionally, the existence of references and relationships between objects is a valuable asset to partition a dataset vertically as they can serve as natural points of decomposition [35]. As most existing object databases support binary relationships that are managed independent of the objects themselves, references can easily be traversed in both directions and thus bridging different partitions is straightforward.

In the past, several techniques (e.g. [37]) have been proposed to support horizontal, vertical and method-induced class partitioning in object databases. However, the requirements of cloud data management raise the question whether these partitioning schemes are still sufficient or whether more advanced techniques are required. For example, a recent article in InformationWeek[10] discusses the adoption of cloud computing in industry. The authors state that companies are attempting to split their data management needs between in-house and cloud computing platforms. This new form of partitioning allows transactional and analytical processes to be delegated to the appropriate computing infrastructure. We believe that also in this setting, objects and relationships are a useful unit to model, support and bridge such partitions.

[10] `http://www.informationweek.com/news/showArticle.jhtml?articleID=221901196`

According to [11], the cost of data processing in parallel and distributed databases is generally a weighted combination of disk I/O, CPU and communication cost, where communication cost is typically considered as the most important factor. Therefore, query operators that require access to data from different partitions are typically associated with high costs. For example, in the case of vertical partitions, a processing node that executes a relational join of two tables or table segments has to access both operands. If the two operands are not stored on the same node, such a join operation can be very expensive. In the setting of object databases, it is again possible to profit from the existence of, potentially bidirectional, references. In contrast to the relational join operation that computes relationships between data at query execution time, references are managed statically by the database. Therefore, they do not have to be materialized, but can simply be navigated without accessing the referenced object. The benefits of leveraging references or pointers in parallel and distributed object databases have been demonstrated in the past. For example, the authors of [38] present several parallel pointer-based join algorithms for set-valued attributes, together with an evaluation of their performance. Similarly, ParSets [39] have also been shown to increase the performance of object graph traversals through parallelization. In the case of horizontal partitions, the costly operation is the one that performs a union over the data segments that contribute to the query result. However, in contrast to the traversal of relationships, this operation is typically less costly as a query optimizer can avoid to access remote data that does not contribute to the final result. Nevertheless, we point out that most object databases already feature collection data structures including the associated collection operations such as union, intersection and difference [40].

4.2 Integration of Novel Processing Paradigms

As object databases are situated at the intersection of object-oriented programming languages and database management systems [31], they are already tightly integrated with programming languages. As a consequence, the separation between the language and the system level is less pronounced. In fact, many object databases rely on programming rather than dedicated query languages to specify processing tasks. In the following, we will examine the characteristics of object databases that facilitate the integration of the processing paradigms introduced in Sect. 2, i.e. data streams, services and map-reduce.

The integration of data stream and traditional data management is difficult because the two processing paradigms are fundamentally different. In traditional data management, various dynamic queries run over a slowly changing database, whereas in data stream management queries are statically registered and process rapidly changing streams. Regardless of the fact that early data stream management systems have been proposed over a decade ago, the processing of streaming object data has not yet been investigated. Nevertheless, object databases are a suitable candidate for the integration of these two paradigms. On the one hand, the fact that classes of objects can define methods and the object database supports their execution provides a mechanism to “register” queries. On the other

hand, the presence of events and listeners as, for example, in the Versant Object Database forms the basis for event-based processing which can be applied to realize data stream management.

To process data over service-based data sources, data management systems need to address the issues of long access times for data and uncertainty because of lacking data statistics. Furthermore, services have, in contrast to other data sources, interfaces that distinguish between ingoing and outgoing fields. As a consequence, selections and join predicates can be delegated to the services themselves whenever constants or outgoing fields of one service can be matched to ingoing fields of another. This property of service-based data processing gives rise to two types of joins, namely parallel and pipe joins [21]. In a parallel join, two services are invoked at the same time and the returned results are combined using a join predicate. This type of join corresponds to a relational join and the fact that services are invoked in parallel reduces execution time. If, however, the overlap in the output of the two services is small, it can also lead to costly and superfluous service invocations. In a pipe join, one service is invoked first and its output is used as input for the second service. While the sequential invocation of services might be less efficient, it allows the second service to be queried in a more directed way. This second type of join is similar to index-based joins or the notion of navigating object references. Therefore, object databases also provide a good basis for the integration of service-based data processing with traditional data management.

As mentioned before, map-reduce is a paradigm which provides a simple model that allows complex distributed processes to be specified. One of the advantages of map-reduce is that the base data (e.g. Web pages) can be cast into different implicit models such as bag of words, set of paragraphs, set of links, or list of links. The disadvantage of this approach is that there is no type checking during query processing since the model or type is constructed on the fly. Object databases could be used to support typing of queries by defining different object wrappers for the same base data instances. However, database and so-called extract-transform-load systems have very different architectures which makes their integration challenging. Early hybrid approaches can be classified into vertical architectures that build higher-level map-reduce interfaces on top of existing database systems and horizontal architectures where the two paradigms exist in parallel. However, as most object databases are already tightly coupled with object-oriented programming languages, they present a unique opportunity to investigate the integration of further processing paradigms. One possible approach to do so is to extend an object-oriented programming language with a domain-specific component that is handled by a compiler plug-in. In this way, DryadLINQ [26] has integrated map-reduce at the language level in the same way as LINQ has extended C# with query capabilities. Another interesting direction is to investigate object query languages that already provide operations similar to map and reduce as, for example, the algebra associated with the OM data model [41].

4.3 Provision of Service-Based Interfaces

In Sect. 2.3, we have motivated the challenge of providing service-based interfaces at all layers of the cloud computing stack as the complementary challenge to support data management over services. The limitation of relational database management systems in this context is twofold. On the one hand, services and relational data management are difficult to integrate as the two models do not align well. On the other hand, using relational systems and the software stacks that surround them in a service-oriented architecture is challenging due to the complexity of mapping service-based interfaces to the relational model. In a sense, this criticism goes back to the original impedance mismatch between object-oriented systems and relational databases [42], with the difference that it nowadays also applies to service-oriented architectures. With the the large-scale deployment of service-based data sources that is to be expected in the setting of cloud computing, the object-relational mapping overhead will grow to new dimensions, too.

As object data models and service-oriented interfaces are closely related, we are convinced that object databases have a lot to offer to cloud computing data management. The concept of orthogonal persistence, that is an essential feature of most recent object databases, is particularly relevant in this context. For example, the authors of [43] point out that the use of orthogonal persistence can already be observed in many modern systems. They speculate that the notion of orthogonal persistence could be extended in order to simplify the development of cloud applications. Instead of only abstracting from the the storage hierarchy, this extended orthogonal persistence would also abstract from replication and physical location, giving transparent access to distributed objects.

5 Conclusion

The promise of cloud computing to render computing a commodity is a promising direction that should also include data management capabilities. In this paper, we summarized the challenges that are associated with delivering cloud computing data management. We argued that data management solutions in the cloud need to be capable of storing data massively parallel and widely distributed. Further they need to integrate novel processing paradigms, such as data stream, service-based and map-reduce processing. Finally, cloud computing data management systems must themselves provide service-based interfaces in order to integrate horizontally and vertically in the cloud computing stack. While some of these challenges have previously been identified by other authors [2,3], this paper presents an integrated and extended view of the requirements of cloud computing data management. We also showed that these challenges clearly surpass the requirements that current data management systems are capable to address.

Based on an overview of the current state of the art in cloud computing data management, we argued that these challenges have, so far, only been addressed by using and extending relational technologies. As a consequence, we revisited

the requirements of cloud computing data management and identified several opportunities for object databases. Due to the unique properties of object data models and algebras, these opportunities exist in the context of all identified requirements. Finally, these opportunities will foster technological innovation in industry and, at the same time, present interesting challenges for research in the domain of object databases. We believe that, in order for cloud computing data management to be successful, it is essential to pursue both of these directions.

To conclude, we would like to clearly state that object databases are not the only technology that needs to be considered for cloud computing data management. Rather, we have made the case that object databases have a lot to offer in the context of cloud computing. As many of their concepts align well with both the cloud computing stack and novel processing paradigms, object databases are a good basis for the integration of these other technologies.

Acknowledgment

The author would like to thank Moira C. Norrie, David Maier and Alan Dearle for the discussions about the work presented in this paper and their valuable feedback on initial drafts.

References

1. Stonebraker, M., Çetintemel, U.: One Size Fits All: An Idea Whose Time Has Come and Gone. In: Proc. Intl. Conf. on Data Engineering, pp. 2–11 (2005)
2. Abadi, D.J.: Data Management in the Cloud: Limitations and Opportunities. IEEE Data Eng. Bull. 32(1), 3–12 (2009)
3. Gounaris, A.: A Vision for Next Generation Query Processors and an Associated Research Agenda. In: Proc. Intl. Conf. on Data Management in Grid and Peer-to-Peer Systems, pp. 1–11 (2009)
4. Vesset, D.: Worldwide Data Warehousing Tools 2005 Vendor Shares. Technical Report 203229, IDC (August 2005)
5. Stonebraker, M., Abadi, D.J., DeWitt, D.J., Madden, S., Paulson, E., Pavlo, A., Rasin, A.: MapReduce and Parallel DBMS: Friends or Foes? Commun. ACM 53(1), 64–71 (2010)
6. Stonebraker, M.: The Case for Shared Nothing. IEEE Data Eng. Bull. 9(1), 4–9 (1986)
7. Golab, L., Özsu, M.T.: Issues in Data Stream Management. SIGMOD Rec. 32, 5–14 (2003)
8. Braga, D., Ceri, S., Daniel, F., Martinenghi, D.: Optimization of Multi-Domain Queries on the Web. In: Proc. Intl. Conf. on Very Large Databases, Auckland, New Zealand, August 23-28, pp. 562–573 (2008)
9. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. In: Proc. Symp. on Operating Systems Design and Implementation, pp. 137–149 (2004)
10. DeWitt, D.J., Gray, J.: Parallel Database Systems: The Future of High Performance Database Systems. ACM Commun. 35(6), 85–98 (1992)

11. Özsu, M.T., Valduriez, P.: Principles of Distributed Database Systems, 2nd edn. Prentice-Hall, Englewood Cliffs (1999)
12. Kossmann, D.: The State of the Art in Distributed Query Processing. ACM Comput. Surv. 32(4), 422–469 (2000)
13. Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. Computer 36(1), 41–50 (2003)
14. Gounaris, A., Smith, J., Paton, N.W., Sakellariou, R., Fernandes, A.A., Watson, P.: Adaptive Workload Allocation in Query Processing in Autonomous Heterogeneous Environments. Distrib. Parallel Databases 25(3), 125–164 (2009)
15. Deshpande, A., Ives, Z., Raman, V.: Adaptive Query Processing. Found. Trends Databases 1(1), 1–140 (2007)
16. Avnur, R., Hellerstein, J.M.: Eddies: Continuously Adaptive Query Processing. In: Proc. ACM SIGMOD Intl. Conf. on Management of Data, pp. 261–272 (2000)
17. Luo, G., Ellmann, C.J., Haas, P.J., Naughton, J.F.: A Scalable Hash Ripple Join Algorithm. In: Proc. ACM SIGMOD Intl. Conf. on Management of Data, pp. 252–262 (2002)
18. Abadi, D.J., Ahmad, Y., Balazinska, M., Çetintemel, U., Cherniack, M., Hwang, J.H., Lindner, W., Maskey, A.S., Rasin, A., Ryvkina, E., Tatbul, N., Xing, Y., Zdonik, S.: The Design of the Borealis Stream Processing Engine. In: Proc. Intl. Conf. on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, pp. 277–289 (2005)
19. Srivastava, U., Munagala, K., Widom, J., Motwani, R.: Query Optimization over Web Services. In: Proc. Intl. Conf. on Very Large Data Bases, pp. 355–366 (2006)
20. Gripay, Y., Laforest, F., Petit, J.M.: A Simple (Yet Powerful) Algebra for Pervasive Environments. In: Proc. Intl. Conf. on Extending Database Technology, 359–370 (2010)
21. Ceri, S., Brambilla, M. (eds.): Search Computing – Challenges and Directions. Springer, Heidelberg (2010)
22. Pacitti, E., Valduriez, P., Mattoso, M.: Grid Data Management: Open Problems and New Issues. J. Grid Comput. 5(3), 273–281 (2007)
23. Antonioletti, M., Atkinson, M.P., Baxter, R., Borley, A., Hong, N.P.C., Collins, B., Hardman, N., Hume, A.C., Knox, A., Jackson, M., Krause, A., Laws, S., Magowan, J., Paton, N.W., Pearson, D., Sugden, T., Watson, P., Westhead, M.: The Design and Implementation of Grid Database Services in OGSA-DAI: Research Articles. Concurr. Comput.: Pract. Exper. 17(2-4), 357–376 (2005)
24. Lynden, S., Mukherjee, A., Hume, A.C., Fernandes, A.A.A., Paton, N.W., Sakellariou, R., Watson, P.: The Design and Implementation of OGSA-DQP: A Service-Based Distributed Query Processor. Future Gener. Comput. Syst. 25(3), 224–236 (2009)
25. Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A.: Pig Latin: A Not-So-Foreign Language for Data Processing. In: Proc. ACM SIGMOD Intl. Conf. on Management of Data, 1099–1110 (2008)
26. Yu, Y., Isard, M., Fetterly, D., Budiu, M., Erlingsson, Ú., Gunda, P.K., Currey, J.: DryadLINQ: A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language. In: Proc. Symp. on Operating Systems Design and Implementation, pp. 1–14 (2008)
27. Chaiken, R., Jenkins, B., Larson, P.Å., Ramsey, B., Skakib, D., Weaver, S., Zhou, J.: SCOPE: Easy and Efficient Parallel Processing of Massive Data Sets. In: Proc. Intl. Conf. on Very Large Databases, pp. 1265–1276 (2008)

28. Abouzeid, A., Bajda-Pawlikowski, K., Abadi, D.J., Rasin, A., Silberschatz, A.: HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. In: Proc. Intl. Conf. on Very Large Databases, pp. 922–933 (2009)
29. DeWitt, D.J., Paulson, E., Robinson, E., Naughton, J., Royalty, J., Shankar, S., Krioukov, A.: Clustera: An Integrated Computation and Data Management System. In: Proc. Intl. Conf. on Very Large Databases, pp. 28–41 (2008)
30. Pavlo, A., Paulson, E., Rasin, A., Abadi, D.J., DeWitt, D.J., Madden, S., Stonebraker, M.: A Comparison of Approaches to Large-Scale Data Analysis. In: Proc. ACM SIGMOD Intl. Conf. on Management of Data, pp. 165–178 (2009)
31. Atkinson, M.P., Bancilhon, F., DeWitt, D.J., Dittrich, K.R., Maier, D., Zdonik, S.B.: The Object-Oriented Database System Manifesto. In: Building an Object-Oriented Database System, The Story of O2, pp. 3–20. Morgan Kaufmann, San Francisco (1992)
32. Fegaras, L., Maier, D.: Towards an Effective Calculus for Object Query Languages. In: Proc. ACM SIGMOD Intl. Conf. on Management of Data, pp. 47–58 (1995)
33. Özsu, M.T., Blakeley, J.A.: Query Processing in Object-Oriented Database Systems. In: Modern Database Systems: The Object Model, Interoperability, and Beyond, pp. 146–174. ACM Press/Addison-Wesley Publishing Co. (1995)
34. Wang, Q., Maier, D., Shapiro, L.D.: The Hybrid Technique for Reference Materialization in Object Query Processing. In: Proc. Intl. Symp. on Database Engineering and Applications, pp. 37–46 (2000)
35. Norrie, M.C., Palinginis, A., Würgler, A.: OMS Connect: Supporting Multidatabase and Mobile Working through Database Connectivity. In: Proc. Intl. Conf. on Cooperative Information Systems, pp. 232–240 (1998)
36. Kuno, H.A., Ra, Y.G., Rudensteiner, E.A.: The Object-Slicing Technique: A Flexible Object Representation and its Evaluation. Technical Report CSE-TR-241-95, University of Michigan (1995)
37. Karlapalem, K., Li, Q.: A Framework for Class Partitioning in Object-Oriented Databases. Distrib. Parallel Databases 8(3), 333–366 (2000)
38. Lieuwen, D.F., DeWitt, D.J., Mahta, M.: Parallel Pointer-based Join Techniques for Object-Oriented Database. In: Proc. Intl. Conf. on Parallel and Distributed Information Systems, pp. 172–181 (1993)
39. Witt, D.J.D., Naughton, J.F., Shafer, J.C., Venkataraman, S.: Parallelizing OODBMS Traversals: A Performance Evaluation. The VLDB Journal 5(1), 3–18 (1996)
40. Cattell, R.G.G., Barry, D.K., Berler, M., Eastman, J., Jordan, D., Russell, C., Schadow, O., Stanienda, T., Velez, F. (eds.): The Object Data Standard: ODMG 3. Morgan Kaufmann, San Francisco (2000)
41. Norrie, M.C.: An Extended Entity-Relationship Approach to Data Management in Object-Oriented Systems. In: Proc. Intl. Conf. on the Entity-Relationship Approach, pp. 390–401 (1993)
42. Maier, D.: Representing Database Programs as Objects. In: Proc. Intl. Workshop on Database Programming Languages, pp. 377–386 (1987)
43. Dearle, A., Kirby, G.N.C., Morrison, R.: Orthogonal Persistence Revisited. In: Proc. Intl. Conf. on Object Databases, pp. 1–23 (2009)