Fusing Storage and Computing for the Domain of Business Intelligence and Analytics – Research Opportunities

Anna Queralt[1], Jonathan Marti[1], Henning Baars[2], André Brinkmann[3], Toni Cortes[1,4]

1. Barcelona Supercomputing Center, Spain
2. Universität Stuttgart
3. Johannes Gutenberg – Universität Mainz
4. Unversitat Politècnica de Catalunya, BarcelonaTech

Abstract

With the growing importance of external and shared data, the set of requirements for Business Intelligence and Analytics (BIA) is shifting. Current solutions still come with shortcomings, esp. in multi-stakeholder environments where sensitive content is exchanged. We argue that a new level in the evolution of BIA can be unlocked by tearing down the barriers between storage and computing based on upcoming storage technologies. In particular, we propose a revitalization of ideas from object-oriented databases. We present results from a joint project that aimed at delineating design options for BIA solutions built upon this idea. The paper outlines the interplay of various architectural layers from storage to business with an object-oriented platform at its core. By doing so, we mark the way for further research, outline a novel design of BIA solutions, and illustrate the consequences of low-level infrastructure innovations on the business layer.

1. Introduction and motivation

Management Support Systems have a long and rich history with continued interest both from the managerial [14] and the research side [8, 20]. During its evolution, a continuous stream of innovations has repeatedly rejuvenated the field: In the 1980s, model-driven analysis systems and data mining expanded the analytical toolset of management support. The 1990s brought the Data Warehouse (DWH) and with it the angle of integration and consistency [22] – thus leading to the term of Business Intelligence (BI) [12]. BI later also engulfed organizational aspects [12, 32]. In the 2000s, active and real-time data warehousing fostered the rise of "Operational BI" with reporting solutions embedded into operational process support [30]. The 2010s have already seen innovations based on a Cloud-based sourcing of BI services [46], as well as in-memory analytics and NoSQL-/MapReduce-environments for the ad-hoc analysis of large and poly-structured BigData sources [8].

Larger macro-economic trends (e.g. the rise of social networks, the diffusion of cyber-physical systems, and the increased relevance of interconnected global supply networks) fuel the demand for a new level of BIA with an agile, dynamic, and multi-stakeholder decision support [3]. While the mentioned technological innovations answer some of the resulting challenges, current architectures for BIA are still not up to par with all arising requirements. The adoption of Cloud-based, shared, and multi-stakeholder BIA solutions in particular is hampered by security and privacy concerns [4, 33, 46].

A potent lever for dealing with these challenges might be to overcome the prevalent separation between storage and computing processes. In the so far dominating collocated scenarios where a single user organization is in control of the BI solution, this separation is not an issue. When information needs to cross enterprise borders, however, data access is governed by and dependent on the policies of the infrastructure providers. We argue that a radical application of concepts from object orientation that fuses data and computing might lead to a viable solution. If the data is encapsulated by the functions that access, administer, and apply it, it can be moved across organizational borders without losing the behavior defined by its owner (integrity control, policy enforcement, etc.) while upholding behavioral constraints e.g. for data integration or aggregation. In order to become truly trustworthy, we propose an approach that goes down *below the application level*. New innovations on the physical storage layer can become relevant drivers of such ideas – innovations that are not yet discussed in the BIA community.

In the following, we present the results of a design research project that explored how a platform can be realized that exploits these ideas and that mapped out the connections to the BIA side. The core artifact of this project consists of two interlocked parts: First, the *design of the platform* that unlocks the potential of new storage technologies in Cloud environments based on the idea of self-contained objects. Second, a layer-based *framework* that shows how such a platform can be embedded in a BIA solution and how it is connected to new design options on various architectural layers. It also highlights relevant resulting research opportunities.

The course of the argumentation is as follows: First, we discuss the limitations of the current separation of data and computing processes for BIA and introduce our methodological approach. Afterwards, we propose design options on various Information Technology / Information Systems (IT/IS) layers and show their interrelations. After a short outline of our feasibility prototype for the platform, the paper concludes with a discussion or our results and of the implications for BIA and IS research and practice.

2. Manifestations and implications of the computing/storage-schism

The dominant paradigm in application programming is object orientation which at its core fuses computation and data. Here, the various merits of this approach – especially the encapsulation of information behind methods and flexibility through polymorphism – are immediately palpable, which is why this paradigm has become text-book knowledge [41, 44]. In a way, the methods enwrapping the data have the potential to embed it into an application context. This encompasses data access policies, data preparation, data transfer, or the relations of objects to other data. Nevertheless, object orientation usually does not transcend the middleware level. This becomes visible in multitier web architectures that assign separate tiers for managing persistence and handling business logic [13, 24], or on a more abstract level in enterprise architecture management approaches like the Zachman Framework with its Data (*What?*) and Function (*How?*) columns [52], or TOGAF which includes separate Data and Function blocks in its Architecture Content Framework [45]. For BIA, this distinction materializes in distinct data and computation layers that can be found in most industry or academic architecture blueprints. A *data layer* is responsible for providing a consistent, integrated data support (mainly realized by the DWH), while the analysis logic is handled in a separate *logic* or *analytic layer* (which often draws its data from excerpts of the DWH, the data marts). Mostly, these are complemented by administrative components (e.g. for data access policies, data lifecycle management, or metadata administration) [5, 22].

The reasons for this separation are almost trivial, given the characteristics of the prevalent infrastructures that require separate approaches to computing and persistence, particularly when dealing with the different behavior of data I/O (which in disk-based storage systems is still mostly block-wise) and data transfer (with separate physical networks for storages and applications). The machinery installed, the methods applied, and the skillsets needed differ strongly.

The introduction of in-memory databases into the realm of BIA [39] gives a glimpse on how infrastructure innovations begin to blur this line [28]. However, even with in-memory databases, the "glue" between data and computation is provided by the application. This leads to shortcomings that restrict the applicability of new cross-border (BIA) applications:

S1: Shared infrastructures and the Cloud model

The Cloud model is the latest building block of the vision to provide IT like a utility. While the benefits of this approach are well-accepted, e.g. enhancing the agility of IT provision, reaping the professionalization advantages of a Cloud provider, or shifting from capital to operating expenditures [7, 48], the diffusion of Cloud-BI is still hampered by security concerns of the potential users. This is mostly a question of data access and access policy enforcement [23, 43].

S2: Shared data

BIA applications that require the sharing and combination of data across organizational borders (e.g. in a Supply Chain scenario, in market place environments, or in franchise arrangements) add questions of data exchange policies and data integration.

S3: Data kept for various purposes and metadata

Data pools that are not tied to an ex-ante specified user group have to be particularly versatile, e.g. in open data arrangements [34]. Here, the issues from S1 and S2 become even more pressing. Despite the fact that a classical DWH is designed for application-independent use, it is still residing within the confines of a defined organization with its defined set of semantics and business rules. The wider the range of applications using the data, possibly even across industries, the higher the importance of sharing semantics without a cumbersome and error-prone transfer of metadata. Ideally, the metadata and its computational interpretation are bundled with the data.

S4: Mobility of data

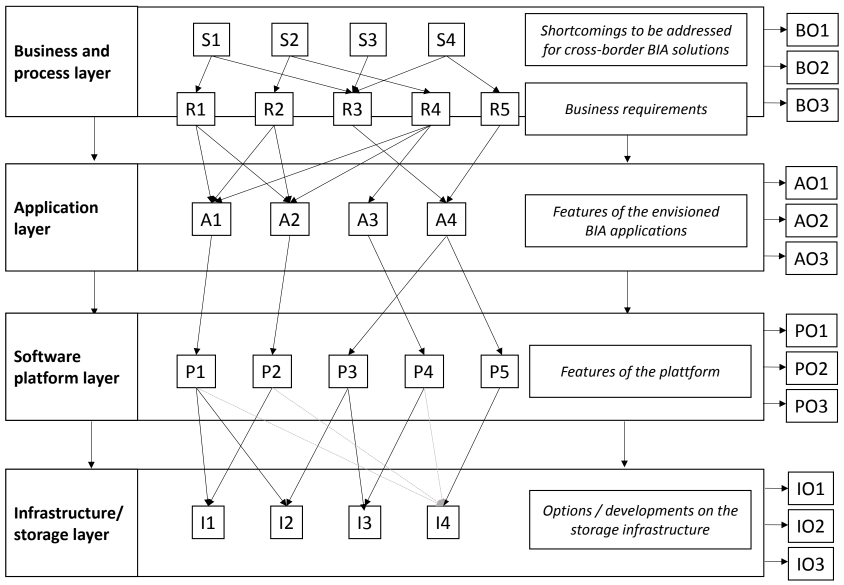
Business and process layer
S1
S2
S3
S4
Shortcomings to be addressed for cross-border BIA solutions
R1
R2
R3
R4
R5
Business requirements
BO1
BO2
BO3
Application layer
A1
A2
A3
A4
Features of the envisioned BIA applications
AO1
AO2
AO3
Software platform layer
P1
P2
P3
P4
P5
Features of the plattform
PO1
PO2
PO3
Infrastructure/ storage layer
I1
I2
I3
I4
Options / developments on the storage infrastructure
IO1
IO2
IO3

Figure 1. Structure of the argumentation and the framework

The number of devices for processing BIA data has multiplied in the last decade. Decision support functionality can come from mobile client devices, general-purpose servers, massive clusters of servers, or BIA-specific appliances. This further adds the dimension of where data is to be stored and used.

3. Methodology

The presented results are the outcome of a two years pan-European collaboration with working groups of researchers in the fields of high performance computing (HPC), Cloud computing, storage, semantic technologies, IT security, information systems, as well as with managers and developers of a BIA tool provider, and a pharmaceutical BIA user organization (plus organizations from other domains). The results of the different sub-groups were tied together and evaluated in two face-to-face-workshops in Spain, document-based collaboration, and several dozen telephone conferences in the years 2011 to 2013.

Our work derived viable designs based on the idea of merging data and computation in an object-oriented fashion based on new storage technologies. The nucleus is an object-oriented, Cloud-oriented platform built under consideration of the next generation of storage systems. As a design-oriented research endeavor, this project deliberately focused on the problem identification and object definition phases of design science research [38] with the aim of pollinating a broad range of subsequent activities and studies.

For structuring our results and for building up the eventual framework we applied a layer model condensed from Enterprise Architecture Management and Cloud Computing [29, 50]. It distinguishes between:

A *business and process layer* that addresses strategic options, supported business functions and processes, and the resulting requirements.

An *application layer* that specifies how the requirements can be dealt with on the application side under consideration of new possibilities at the platform layer. The application layer translates the business requirements into application features, which are in turn requirements for the underlying layers.

A *software platform layer* in the sense of the Platform-as-a-Service concept of a Cloud solution [29]. This layer encompasses services of networking middleware and (object-oriented) database management systems. This layer contains our platform.

The *infrastructure / storage layer* that grounds the platform in hardware design and maps the object-oriented services of the platform to innovative options for the low-level infrastructure systems.

We understand the design-oriented outcomes of our research as *design options* that address either requirements or design options of a higher layer. The design options are thereby not only shaping the solution space but also act as linking pins between different layers. They come in the form of application and platform features, and infrastructure options. In the terminology of design patterns [10], the design options represent *solutions*, while the higher level requirements/options present the *problem* and describe the *application context*. In this contribution we also highlight selected implications and impacts. This article focusses on the form and function of the resulting framework and the platform (the *what*) and will not go into the inner workings of our solutions regarding architecture, applied principles, or implementations (the *how*). This particularly excludes the discussion of the applied security techniques from Grid- and Cloud computing for enabling secure access to confidential information or object methods.

With respect to the criteria for describing a Design Science endeavor given by Gregor and Jones [16], the design options outline the *form and the function* of the framework with the embedded platform and connects the latter to their *applications*. As for our *scope,* the results are targeted towards shared, Cloud-based, and data-intensive applications and respective BIA applications in particular. The platform builds up on the *justificatory knowledge* of the body of research from the domains of BIA, Cloud Computing, security, and storage design. The *mutability* has been addressed by the discussion of further application domains (cf. section 6). The interdisciplinary telephone conferences and workshops were used for the *evaluation* of intermediary design options conceived by the sub-groups regarding relevance and feasibility. This project set-up also reflected the *iterative search process* for a holistic solution. The inclusion of the heterogeneous groups from research aimed at ensuring the rigor and novelty, while the discussion of the results with domain practitioners of the application and the tool side fed the relevance cycle in the sense of Hevner et al. [19].

In conclusion, the research was deductive by nature. The use of an example application for SCM analytics was used for purposes of concretization, illustration, and enrichment. This injected an inductive element into the approach.

Figure 1 illustrates the course of the argumentation and the structure of the resulting framework that connects the various requirements and design options: the addressed business requirements (R1-R5), the relevant application features (A1-A4), the features of the conceived platform (P1-P5), and eventually current and upcoming options on the storage / infrastructure layer (I1-I4). For each layer three open research questions are defined that we deem critical.

4. Results on the various layers

4.1 The business and process layer

During our meetings, a number of potential applications for our conception of a low-level based, object-oriented platform were explored and evaluated for their potential on the business layer. Across all domains, the possibility to *share large volumes of data among independent organizations* has been identified as a potent enabler for innovative systems.

This particularly opens options for novel solutions in the data-centric BIA area. There are already examples for shared BIA solutions e.g. in joint venture operations, horizontal business partnerships, market platforms with BIA components, or – as in our exemplary case – Supply Chain Management (SCM).

In order to facilitate the applicability of such solutions we recommend meeting the following requirements:

R1. Strictly enforce access policies so that confidential data is only accessible for those who hold the respective rights (even in a shared platform).
R2. Mutually disclose data with different degrees of granularity among the various partners.
R3. Use a scalable and globally accessible platform that can be dynamically scaled up, e.g. in case of conducting a new plan run or analysis.
R4. Exchange and combine data from the various partners.
R5. Enable high computing performance, esp. for ad-hoc and interactive analysis and planning (including the calculation of what-if scenarios).

The difference between R5 to R3 is that in R5 the focus is on the provision of specific computational performance while R3 refers to the general ability to quickly scale the solution up and down.

The pharmaceutical organization in the consortium provided use cases which were explored for purposes of results validation, concretization, and illustration. One in particular was deemed important by the company and found to be representative for the class of envisioned BIA applications: A distributed planning and budgeting system that involves multiple supply chain partners. It needs to be highlighted that the value of data sharing in SCM for planning, steering, and control have been shown in research before [2, 21].

In the given case, cost planning and budgeting have already been extended across the value chain with an exchange of data on production and delivery amounts. In one region, the data of the manufacturer has been integrated with supplier data in an external platform. Not yet realized, however, is an envisioned large-scale data integration for planning that spans all business units and regions and that involves suppliers and customers alike – mostly for not meeting R1-R2. Moreover, the current system is based on a patchwork of disparate tools with semi-manual data sharing (addressed by R4). Insufficient performance prohibits fully interactive planning scenarios as the planning process is based on terabytes of interdependent, confidential data (leading to R5). As planning is not done continuously and partners enter and leave the process, scalability is also an issue (R3). The traits of the application also show the particularities of the BIA domain with its large, interlinked data repositories, heterogeneous access patterns, and distributed user groups.

Open research questions on this layer that we see crucial include:

Objective BO1: Analyse the connections to service level agreements and to the legal and contractual framework surrounding the systems.

Objective BO2: Scrutinize the business impact for platform providers, tool developers, and consultancies and conduct a value-chain analysis.

Objective BO3: Analyse the potential to build large-scale BIA eco-systems with participating public administration organisations, small- and medium-sized enterprises, research institutions etc.

4.2 The application layer

An object-oriented platform, especially when based on the storage types introduced in Section 4.4, can be utilized in a way that applications are only allowed to disclose a desired selection and a defined degree of granularity of the data (R1, R2).

For the SCM example case that means that a planner on the supplier side will only see aggregated planning numbers from the pharmaceutical company, while the pharmaceutical planner can work with her data on the highest level of detail. As this feature is built in the platform and realized on the infrastructure layer, the partners are not dependent on trust when it comes to data sharing (R1, R2, R4).

Application feature A1: Built up on platform functionality to provide features that govern data access.

Application feature A2: Built up on platform functionality to ensure a user-dependent data aggregation.

Furthermore, the data sharing and exchange itself (R4) can be supported by specific functionality that "sticks" with the data (e.g. by supporting semantic mapping, cleansing, or metadata provision). The idea is to move routines from disjoint ETL, metadata, and database administration components to the data itself, making the data "magnetic".

Application feature A3: Couple the data with functionality that makes it "magnetic" by facilitating the integration and combination of data.

Eventually, performance, global access, and scalability are supported by the move to an inherently secure Cloud platform. The pursued design also sees the option of moving data to the infrastructure components most suitable for the required computing needs. For interactive planning scenarios this could imply a temporary transfer of the required data to high-performance analytic appliances. This addresses R3 and R5.

Application feature A4: Include functionality that ensures the data is handled by the most pertinent infrastructure components under particular utilization of Cloud infrastructures.

The chosen approach has architectural implications: A central element of standard reference BIA architectures is the distinction of data layers for various states of preprocessing (ODS, DWH, data mart). A revised architecture based on the envisioned solution could melt this down to mere logical views on a shared data platform that is distributed across the available infrastructure. Basically, this carries forward ideas from older approaches like virtual DWHs, Enterprise Information Integration, and in-memory databases [17, 39, 49]. Similarly, the implications for current middleware layers should be scrutinized (e.g. application servers used for implementing a SOA architecture).

This leads to research objectives on this layer:

Objective AO1: Design application and integration architectures that fully exploit the solution.

Objective AO2: Re-assess the role of tightly integrated, application specific components like BI appliances and of proprietary end-to-end BIA suites.

Objective AO3: Explore the semantic limits of automated data sharing, integration, and data quality assurance as well as on how to connect the new solution to proven approaches for these tasks.

4.3 The object-oriented software platform

The platform we propose mediates between the (enhanced) storage infrastructure and the applications, which subsume any software that accesses persistent data and serves it either to the final user or to upper software layers. It is based on the concept of *self-contained objects.* A self-contained object is a piece of data that is always bound to the logic required to process it (methods) and to the policies that manage its behaviour with respect to security, integrity, etc. Unlike in current practice this connection is *never* broken apart – not even for persistence. The object "lives on" on the storage or is even executed there. The platform manages the self-contained objects and serves them to the application layer as if it was in-memory data, regardless of physical form and location. The only difference (transparent to the applications) is that the object methods are executed in the storage platform, since they belong to the objects and not to the applications using the objects.

This is also the main difference from other frameworks such as Hibernate (http://hibernate.org) or Spring (http://spring.io), which persist only the data in the objects, while their methods still belong to the application space. Since our platform stores the objects as a whole, not only the data in the objects but also their methods are shared by different applications. This provides several new possibilities that are directly linked to the requirements R1-R5.

The respective platform features are:

P1: Method-based object access

By keeping data and code together, we can move the responsibility for the data to the object, so that security is kept independent of applications. The object decides under which conditions it appears in query results (e.g. depending on a location), giving full control to the data owner. This behaviour is achieved by encoding the desired policies into the methods, which are the only access points to the data. Thus, the methods of a self-contained object will contain the original code plus whatever is required to enforce the rules of the data owner. This can be done automatically if the rules are defined by the means of a domain-specific language.

Other mechanisms have been developed that are also based on the idea of attaching rules to the data for enforcing concrete behaviour, particularly access control policies. For instance, Java provides guarded objects [35], which wrap an object together with its access control policies, implemented in a separate guard object. In contrast, in our proposal policies are included in the objects themselves, thus providing transparent access even when they include policies.

Sticky policies [37] are another example of rules attached to the data, in this case applied to the protection of personal information. In this approach, an agreed trust authority regulates access to data, while in our case it would be the methods that do this job.

P2: Disclose different degrees of granularity

Following the same approach as for P1, privacy rules can be included in the objects to show data in different ways depending on who is accessing them. This is particularly interesting for datasets containing sensitive data, and enables to prevent the combination of certain kinds of information. For instance, in a sales dataset owned by a pharmaceutical retailer in the Supply Chain, an associated shipping company will be allowed to access data from individual customers to plan the delivery. In contrast, the marketing department of the pharmaceutical manufacturer might analyse the same data but is only allowed to access the objects indirectly through a collection object that returns aggregated results.

P3: Object mobility

Since security and privacy are guaranteed by the methods, objects can leave the platform without changing their behaviour. Depending on factors such as the current work load, an object can be moved to the resource where the application is running or to a third-party resource (e.g. an external cloud). This ability to respond to fluctuating workloads allows applications to take advantage of cloud elasticity benefits. In order to guarantee that objects can be safely migrated to a third-party infrastructure, a certain trust level must be fulfilled by the destination infrastructure [36]. Thus, the platform will never offload an object to a resource with a lower trust level than the one required by the object as specified by its owner.

P4: Enrichments

Some mechanism is required so that each application can not only consume the objects as they are, but also personalize how it sees these objects or how they behave. Since the shared data is protected by its associated methods, this flexibility can be achieved by allowing applications to enrich the existing objects in order to change how information is exported to applications or how this information is processed. By *enrichment* we mean adding more information to existing objects or changing their behaviour (always within the bounds defined by their owner).

Several enrichments may coexist, and may be visible only to the application that defines them, so that different views can be offered to different applications at the same time. In BIA, this can be useful to perform analytics directly on top of a source system and skip intermediate ETL, DWH, and data mart layers. There are some well-known caveats to such an approach, most notably the loss of historical data and (legal) requirements for the archival of intermediate data [17]. Note, though, that a self-contained object might administer its own history and archive.

Enrichments are also applicable to interactive planning. Understanding enrichments as models that allow to see and to manage data in different ways, one can apply different enrichments on the same data set and see how results vary depending on different parameters and/or models. This allows the application layer to calculate simulations / what-if scenarios.

Similar to our enrichment is the support for multiple views in object-oriented databases [6, 42], although it differs from our approach as these views do not include methods, and they also lack the possibility of defining new data that is not derived from the original source. In addition, views are defined by the database administrator. Other concepts related to our enrichments are the global schema in data integration, and the target schema in data exchange [11], which may extend the schema by including new data, but not code. Subject-oriented programming [18] and aspect-oriented programming [27] also consider behaviour in order to offer different perspectives on the same objects. Unlike the enrichments, though, they only address cross-cutting concerns such as error handling or caching, in order to avoid redundant or scattered code. In contrast, the purpose of enrichments is to provide different ways to see and manage the same data according to the different roles that users have.

P5: Performance and secure offloading

In order to be useful in interactive planning scenarios, simulations at the application layer require performance from the underlying layers. Performance is in part provided by the enrichments themselves, since they can encode the most appropriate data structures to perform the calculations, and store them together with the data, resulting in some kind of "materialized views" that are calculated just once and then used in all the calculations. This is useful when a subset of the data is shared by different scenarios.

A further property of our platform in this regard is the avoidance of data movements. In other words, the platform brings computation to the data, rather than the data to the applications. Especially when the amount of data is large as in the SCM use case, data does not need to move around and network latencies are avoided. This is particularly supported by storages that have computing capabilities (cf. 4.4).

The fact that objects are self-contained still allows improving performance if computing capacity is exceeded. In this case, the platform can securely and transparently offload part of the data and computation to an external cloud or to any other resource, such as an HPC cluster, while guaranteeing security and privacy as if the computation was performed within the enterprise borders. This implies an important performance gain for parallelizable tasks.

There are still open research directions to be followed in order to completely provide the features P1-P5. The most relevant ones can be summarized in the following research objectives:

Objective PO1: Define a simple declarative language to allow data providers to formalize security and privacy policies, as well as a mechanism that automatically enforces these policies (P1 and P2).

Objective PO2: Provide a limited programming language to define enrichments, together with a mechanism that automatically verifies whether an enrichment satisfies the policies (P4).

Objective PO3: Provide a secure environment to execute the methods attached to the objects when they are offloaded to third-party resources such as third-party Clouds (P3 and P5).

4.4 The infrastructure / storage layer

For the past 15 years there have been developments on the storage side that aim at fusing data and functionality on the storage side, thus directly supporting a platform as the proposed one. The platform is designed to immediately utilize these capabilities in order to store and execute self-contained objects directly on the storage.

I1: NASD

The platform feature P1 (enforcement of access policies directly at the data) has been discussed in the academic community for the first time in the context of network attached secure disks (NASD) [15]. NASD are closely coupled with the concept of object-based storage devices, where the interface between applications and storage systems is based on transferring complete objects, so that a storage system can understand semantic boundaries between stored content [31]

Developing the idea of NASD further into the direction of today's Cloud requirements and combining it with the main ideas of trusted computing helps to increase the mobility of data (cf. P3). Data can be securely shared if it is encrypted on disk and can only be accessed through interfaces on the storage device. It also becomes possible to move the data to physically dislocated storage devices with the same or higher trust level, so that they can securely leave the physical protection boundaries of an organization.

I2: Active Disks

Directly filtering data based on the accessing role (which is the foundation for P1) requires additional processing capabilities at the object-based storage devices. These processing capabilities have been introduced with active disks that use the embedded processors within disk devices to pre-process data directly at its source [40]. Having too complex interfaces and applying too slow processors, the idea has not received attention for real applications at the time of its invention, but recently received increased interest based on new storage technologies [26, 47]. Active disks with their embedded processing capabilities enable processing data without the need of transferring it to upper layers (not even to the operating system). This way, avoiding data movements is taken to the limit since data does not even need to leave the storage device. Only results are sent to upper layers. This relieves tasks from the software layers on top and avoids latencies at runtime.

Data processing at its source is also a major building block for the provision of a scalable and globally accessible platform (as requested in R3 and A4). The bandwidth of storage arrays has become so fast that moving the data from the storage systems to the servers easily saturates the associated network interfaces and floods the processing nodes. As the computing power associated with each array increases significantly, the processing and reduction of the data before delivering it becomes feasible.

Currently missing is an interface which allows applications to easily move data processing to the disks and disk arrays. Knowledge about the application is lost at the storage interface boundary. Read and write optimizations, e.g., for database applications [25], cannot be transferred to the devices. Current approaches try to move the interface into the operating system, to allow it to attach additional semantic information, and to optimize storage accesses for each technology [1, 53], but do not yet deliver the opportunity to directly transform the data.

I4: Non-volatile memory technologies

Scalability is also limited by the performance of individual drives. While the access speed to in-memory databases only incurs the latency of main memory, object-based disks are still restricted by the underlying magnetic disks and flash. Non-volatile memory technologies (NVRAM) like phase-changed RAM, MRAM, or RRAM will significantly shrink the current performance gap between standard DRAM and storage and reduce access times down to 10 ns. This helps fulfilling R5 and building an economical and *persistent* alternative to current in-memory approaches [51]. Furthermore, these technologies are well-suited to be directly coupled with logic [9] with the potential to drive the approach proposed here even further.

NVRAM will enable addressing data (objects) directly as if it was in memory. This frees both applications and the middleware on top of storage systems from the responsibility of mapping in-memory data to a persistent storage. For instance, assume an object-oriented application that deals with persistent data. As a first step, the application must transform in-memory data to the format required by the database used (either relational or NoSQL), or must explicitly write it into files. Afterwards, there is at least a second transformation to pack data into fixed size disk blocks that is done by the operating system. The same happens in the opposite direction, i.e. when retrieving data from disk.

As the discussion above shows, additional research into the following directions is necessary:

Objective S1: Leverage the concept of network attached secure disks to the requirements of agile cloud environments, so that data can be moved between different, previously untrusted entities.

Objective S2: Overcome the latencies of magnetic- and flash-based disks by including non-volatile memory technologies and to find architectural means to integrate them into the storage and BI stack.

Objective S3: Provide stronger interfaces that allow passing semantic information between storage layers, so that physical new device technologies can be integrated according to their capabilities.

5. Platform prototype

In order to evaluate the feasibility of realizing our core platform on a functional level and to gain further insights into its applicability and behaviour, we developed a working prototype. Since we (and the storage research community) are still researching on how to make the new storage technologies operable, the prototype is currently accessing more conventional storage technology, but keeping in mind the features and interfaces that the new devices will offer. Thereby we will be able to exploit the new storage technologies as soon as they are available – unleashing the full potential of our approach.

In its current form, the prototype includes a layer which maps in-memory objects to a NoSQL data store. As soon as NVRAM becomes a reality, this layer will no longer be necessary, as we do not rely on any of the data management features offered by the database and use it just to access objects through an identifier, which is how NVRAM works.

What the current prototype already does is enabling Java applications built on top to manage self-contained objects in a transparent way, i.e. as if they were only in memory. The prototype itself is also implemented in Java and runs on several back-ends with storage and compute capabilities.

The core feature of the prototype is the ability to store not only data objects, but also their structure and the methods associated with them (classes). The rest of the prototype's functionalities are based on this feature as explained in the following.

First, since object methods are stored in the platform, the execution of methods happens in the same back-end where the data is, avoiding unnecessary data transfers from storage to application.

Second, because methods are stored in the platform, data policies can be encoded within them to keep them independent from applications, thus ensuring that policies are always satisfied (R1, R2). Currently, policies have to be manually implemented in the methods, but our goal is that data providers can define them declaratively and automate their encoding (cf. PO1 and PO2). Note that the enforcement of each policy is embedded in the code of the affected methods and is checked only when required. This makes an object as heavy as the amount and complexity of the policies it has to satisfy without loosening security guarantees.

Third, the fact that classes are stored in the platform also provides users with the possibility to enrich the existing data by adding new classes, or by adding new properties and methods to existing classes. Each user can define her own enrichments and make them accessible to other users (R4, R5).

Part of the current solution has been published in [*blinded*]. In order to fully fulfil all the requirements, the platform must support offloading data and computation to potentially untrusted infrastructures (PO3). This problem has not yet been approached, but the fact that objects are the building blocks of our solution, and that data is encapsulated through methods, will allow us to do this offloading in a secure way.

6. Discussion, Limitations, and Outlook

Our results indicate that the possibility to fuse computing and storage based on active disks and non-volatile memory can have disruptive consequences in BIA on higher layers, particularly when coupled with object-oriented concepts. These range from a full revision of BIA architectures, obliterating established conceptions of DWHs, up to the enablement of new large-scale cross-border BIA solutions. The SCM decision support scenario is one upon many. The vision is to allow for analytical cross-border BIA eco-systems with unprecedented levels of security (BO3).

In our group we also explored other application domains that underscore the mutability and generalizability of the platform and the approach: Commercial pharmaceutical research, science, management of large-scale public events, news archives, and geo-data services. Further domains could include health care, public administration, or social media services. A full-scale application of our approach would also imply modified approaches to EAM.

It needs to be stressed that we see the novelty of our research and thus its core contribution less in its individual design options but in their interplay. The core limitation of our work is that we have not yet evaluated a full-fledged solution that spans all layers. This will be particularly relevant for evaluating trade-offs between performance and object mobility, for testing application-specifics, or for addressing compatibility. Besides, the discussed solutions still require research input on storage, platform, and application layer before becoming available. However, BIA tool providers, consultants, and user organizations need to be aware of those developments in order to braze for the resulting changes and to seize the opportunities early. This particularly means to tackle the business- and application-oriented questions BO1-BO3 and AO1-AO3.

What our research eventually does is to highlight how seemingly “mere technical” low-level infrastructure innovations on the storage side can have ripple-effects all the way to the business model.

7. References

[1] Appuswamy, R., Van Moolenbroek, D.C., and Tanenbaum, A.S., "Integrating Flash-Based SSDs into the Storage Stack", Proceedings of the IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST), 2012.

[2] Aviv, Y., "On the Benefits of Collaborative Forecasting Partnerships between Retailers and Manufacturers", Management Science, 53(5), 2007, pp. 777-794.

[3] Baars, H., Felden, C., Gluchowski, P., Hilbert, A., Kemper, H.-G., and Olbrich, S., "Shaping the Next Incarnation of Business Intelligence", Business & Information Systems Engineering, 6(1), 2014, pp. 11-16.

[4] Baars, H., and Kemper, H.-G., "Ubiquitous Computing– an Application Domain for Business Intelligence in the Cloud?", Proceedings of the 17th Americas Conference on Information Systems, 2011.

[5] Baars, H., and Kemper, H.G., "Management Support with Structured and Unstructured Data – an Integrated Business Intelligence Framework", Information Systems Management, 25(2), 2008, pp. 132-148.

[6] Boufaida, M., and Fouzia, B., "The Viewpoint Mechanism for Object-Oriented Databases Modelling, Distribution and Evolution", CIT. Journal of computing and information technology, 15(2), 2007, pp. 95-110.

[7] Buyya, R., Yeo, C.S., and Venugopal, S., "Market-Oriented Cloud Computing: Vision, Hype and Reality for Delivering It Services as Computing Utilities", Future Generation Computer Systems, 25(6), 2009, pp. 599-616.

[8] Chen, H., Chiang, R.H., and Storey, V.C., "Business Intelligence and Analytics: From Big Data to Big Impact", MIS Quarterly, 36(4), 2012, pp. 1165-1188.

[9] Coburn, J., Caulfield, A.M., Akel, A., Grupp, L.M., Gupta, R.K., Jhala, R., and Swanson, S., "NV-Heaps: Making Persistent Objects Fast and Safe with Next-Generation, Non-Volatile Memories", Proceedings of the 16th international intl. conf. on Architectural support for programming languages and operating systems, 2011, pp. 105-118.

[10] Erl, T., “SOA design patterns”. Prentice Hall, Upper Saddle River (NJ) et al., 2009.

[11] Fagin, R., Kolaitis, P.G., Miller, R.J., and Popa, L., "Data Exchange: Semantics and Query Answering", Theoretical Computer Science, 336(1), 2005, pp. 89-124.

[12] Foley, É., and Guillemette, M., "What Is Business Intelligence?", International Journal of Business Intelligence Research (IJBIR), 1(4), 2010, pp. 1-28.

[13] Fowler, M., Patterns of Enterprise Application Architecture, Addison-Wesley, Boston (Mass.), 2002.

[14] Gartner, Gartner Exp Worldwide Survey (for the Years 2007 to 2013). On the pages of the Gartner Group.
[15] Gibson, G.A., Nagle, D.F., Amiri, K., Chang, F.W., Feinberg, E.M., Gobioff, H., Lee, C., Ozceri, B., Riedel, E., Rochberg, D., and Zelenka, J., "File Server Scaling with Network-Attached Secure Disks", Proceedings of the ACM Intl. Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS ’97), 1997, pp. 272-284.
[16] Gregor, S., and Jones, D., "The Anatomy of a Design Theory", Journal of the Association for Information Systems, 8(5), 2007,
[17] Halevy, A.Y., Ashish, N., Bitton, D., Carey, M., Draper, D., Pollock, J., Rosenthal, A., and Sikka, V., "Enterprise Information Integration: Successes, Challenges and Controversies", Proceedings of the ACM SIGMOD International conference on Management of data, 2005, pp. 778-787.
[18] Harrison, W., and Ossher, H., "Subject-Oriented Programming: A Critique of Pure Objects", Proceedings of the eighth annual conference on Object-oriented programming systems, languages, and applications (OOPSLA '93), 1993.
[19] Hevner, A.R., March, S.T., and Park, J., "Design Science in Information Systems Research", MIS Quarterly, 28(1), 2004, pp. 75-105.
[20] Hosack, B., Hall, D., Paradice, D., and Courtney, J.F., "A Look toward the Future: Decision Support Systems Research Is Alive and Well", Journal of the Association for Information Systems, 13(5), 2012, pp. 3.
[21] Huang, G.Q., Lau, J.S.K., and Mak, K.L., "The Impacts of Sharing Production Information on Supply Chain Dynamics: A Review of the Literature", International Journal of Production Research, 41(7), 2003, pp. 1483-1517.
[22] Inmon, W.H., Building the Data Warehouse, 4th edn., Wiley, New York, 2005.
[23] Jansen, W.A., "Cloud Hooks: Security and Privacy Issues in Cloud Computing", Proceedings of the 44th Hawaii International Conference on System Sciences (HICSS), 2011.
[24] Jendrock, E., Cervera-Navarro, R., Evans, I., Haase, K., and Markito, W., "The Java EE 7 Tutorial", ORACLE, http://docs.oracle.com/javaee/7/tutorial/doc/home.htm, accessed June 2014.
[25] Kaiser, J., Margaglia, F., and Brinkmann, A., "Extending SSD Lifetime in Database Applications with Page Overwrites", Proceedings of the 6th International Systems and Storage Conference (SYSTOR’13), 2013.
[26] Kang, Y., Kee, Y.-S., Miller, E.L., and Park, C., "Enabling Cost-Effective Data Processing with Smart SSSD", 29th IEEE Symposium on Massive Storage Systems and Technologies (MSST 13), 2013.
[27] Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.-M., and Irwin, J., Aspect-Oriented Programming, Springer, Berlin, 1997.
[28] Knabke, T., and Olbrich, S., "Towards Agile BI: Applying in-Memory Technology to Data Warehouse Architectures", in (Lehner, W., Piller, G., eds.): Proceedings of the IMDM 2011, pp. 101-114.
[29] Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., and Leaf, D., "NIST Cloud Computing Reference Architecture", NIST Special Publication, NIST SP - 500-292, 2011.
[30] Marjanovic, O., "The Next Stage of Operational Business Intelligence: Creating New Challenges for Business Process Management", Proceedings of the 40th Hawaii International Conference on System Sciences (HICSS), 2007, pp. 215c-215c.
[31] Mesnier, M., Ganger, G.R., and Riedel, E., "Object-Based Storage", Communications Magazine, 41(8), 2003, pp. 84-90.
[32] Miller, G.J., Bräutigam, D., and Gerlach, S.V., Business Intelligence Competency Centers: A Team Approach to Maximizing Competitive Advantage, Wiley, Hoboken, USA., 2006.
[33] Mircea, M., Ghilic, B., and Stoica, M., "Combining Business Intelligence with Cloud Computing in Delivery to Agility in Acutal Economy", Economic Computation & Economic Cybernetics Studies & Research, 45(1), 2011.
[34] O'Riain, S., Curry, E., and Harth, A., "XBRL and Open Data for Global Financial Ecosystems: A Linked Data Approach", International Journal of Accounting Information Systems, 13(2), 2012, pp. 141-162.
[35] ORACLE, Guarded Object, http://docs.oracle.com/ javase/7/docs/api/java/security/GuardedObject.html, accessed March 2014, 2014.
[36] Pawar, P., Rajarajan, M., Nair, S.K., and Zisman, A., "Trust Model for Optimized Cloud Services": Trust Management VI – IFIP Advances in Information and Comm. Technology, 374, Springer, 2012, pp. 97-112.
[37] Pearson, S., and Mont, M.C., "Sticky Policies: An Approach for Managing Privacy across Multiple Parties", Computer, 44(9), 2011, pp. 60-68.
[38] Peffers, K., Tuunanen, T., Rothenberger, M.A., and Chatterjee, S., "A Design Science Research Methodology for Information Systems Research", Journal of Management Information Systems, 24(3), 2007, pp. 45-77.
[39] Plattner, H., "A Common Database Approach for OLTP and OLAP Using an in-Memory Column Database", Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, 2009.
[40] Riedel, E., Faloutsos, C., Gibson, G.A., and Nagle, D., "Active Disks for Large-Scale Data Processing", Computer, 34(6), 2001, pp. 68-74.
[41] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W.E., Object-Oriented Modeling and Design, Prentice Hall, Upper-Saddle River, NJ (US), 1991.
[42] Rundensteiner, E.A., "Multiview: A Methodology for Supporting Multiple Views in Object-Oriented Databases", Proceedings of the 18th International Conference on Very Large Data Bases (VLDB '92), 1992, pp. 187-198.
[43] So, K., "Cloud Computing Security Issues and Challenges", International Journal of Computer Networks, 3(5), 2011, pp. 247-255.
[44] Sommerville, I., Software Engineering, 9th ed, Pearson, London (UK) 2010.
[45] The Open Group, TOGAF Version 9.1 – the Open Group Architecture Framework (TOGAF), 2011.

[46] Thomson, W.J.J., and Van Der Walt, J.S., "Business Intelligence in the Cloud ", South African Journal of Information Management, 12(1), 2010,

[47] Tiwari, D., Boboila, S., Vazhkudai, S.S., Kim, Y., Ma, X., Desnoyers, P., and Solihin, Y., "Active Flash: Towards Energy-Efficient, in-Situ Data Analytics on Extreme-Scale Machines", Proceedings of the 11th USENIX Conference on File and Storage Technologies (FAST'13), 2013.

[48] Weinhardt, C., Aadasivam, A., Blau, B., Borissov, N., Meinl, T., Michalk, W., and Stößer, J., "Cloud Computing – a Classification, Business Models, and Research Directions", Business & Information Systems Engineering (BISE), 5(1), 2009, pp. 391-399.

[49] White, C., "Data Integration: Using ETL, EAI, and EII Tools to Create an Integrated Enterprise", Business Intelligence Journal, 10(I), 2005.

[50] Winter, R., and Fischer, R., "Essential Layers, Artifacts, and Dependencies of Enterprise Architecture", Proceedings of the 10th IEEE on International Enterprise Distributed Object Computing Conference Workshops (EDOCW '06), 2006, pp. 30-30.

[51] Wong, H.-S., Raoux, S., Kim, S., Liang, J., Reifenberg, J.P., Rajendran, B., Asheghi, M., and Goodson, K.E., "Phase Change Memory", Proceedings of the IEEE, 98(12), 2010, pp. 2201-2227.

[52] Zachman, J.A., "A Framework for Information Systems Architecture", IBM systems journal, 26(3), 1987, pp. 276-292.

[53] Zhang, Y., Arulraj, L.P., Arpaci-Dusseau, A.C., and Arpaci-Dusseau, R.H., "De-Indirection for Flash-Based Ssds with Nameless Writes", Proceedings of the 10th USENIX conf. on File and Storage Technologies. 2012.