Towards Granular Data Placement Strategies for Cloud Platforms

Johannes Lorey
Hasso-Plattner-Institut
Prof.-Dr.-Helmert-Straße 2-3
D-14482 Potsdam, Germany
Email: johannes.lorey@hpi.uni-potsdam.de

Felix Naumann
Hasso-Plattner-Institut
Prof.-Dr.-Helmert-Straße 2-3
D-14482 Potsdam, Germany
Email: felix.naumann@hpi.uni-potsdam.de

***Abstract*—Traditional data placement strategies in the context of Information Lifecycle Management (ILM) are applicable only to on-site storage systems. In contrast to this approach, Cloud storage provides a novel possibility to reduce or entirely eliminate capital expenditures for hardware. As a unique solution to buffer short-term resource demand peaks, Cloud infrastructures can be combined with on-site systems to support efficient placement of data.**

The algorithms underlying this optimization must consider not only the workload as a whole, but rather variable-sized subworkloads to determine an optimal placement. As a means to identify these subworkloads, we introduce a multi-dimensional granularization approach. Based on different granules of metadata information, we propose a flexible hybrid data placement system incorporating both on-site and Cloud resources.

I. Introduction

Due to the ever-increasing amount of digital information and the continuous effort to reduce capital expenditures for hardware, being able to dynamically determine the current value of a single datum is a key factor in choosing its optimal storage location. This decision process is generally referred to as Information Lifecycle Management. ILM strategies usually consider on-site hardware resources only, such as hard disk drives or magnetic tapes [1] [2]. These resources can be classified hierarchically into different categories based upon their technical and economical characteristics. Generally speaking, higher-level storage is more costly (in terms of money per storage unit) and more easily accessible, while lower-level storage is cheap and mostly used as a back-up solution. Thus, purchasing and operating a hard disk drive will result in higher cost than employing a magnetic tape with identical size. On the other hand, a record that is accessed frequently should reside on a faster hard disk drive instead of a slower tape in order to ensure time-critical operations can be executed quickly. During its lifetime, a single datum is typically moved from higher-level to lower-level storage depending on several factors, such as its access frequency or its current importance [3].

One major drawback of this traditional set-up is its lack to flexibly handle varying demand in storage resources. This shortcoming is typically bypassed with higher capital expenditures, i.e., throwing more hardware at the problem. Clearly, this approach cannot be regarded as a sophisticated solution. Instead, there is a demand for an alternative storage platform to flexibly buffer demand peaks without high up-front cost.

Here, Cloud Computing and Cloud storage offer a novel approach for ILM. As two of the main characteristics of the Cloud are its rapid elasticity and the associated metering of resource consumption, there are little to no capital expenditures associated with using Cloud services [4]. On the other hand, long-term storage necessities may economically justify purchasing on-site hardware instead of renting resources in the Cloud.

As Cloud Computing has been a major technological trend for only a short time, the decision between using on-site or Cloud resources has usually been a binary one for companies up to now. Established organizations usually have the resources required for their daily operations at hand and thus do not sense the need to outsource infrastructure. On the contrary, security and reliability concerns might even prevent them from doing so [5]. For start-up businesses on the other hand, commercial Cloud Computing offerings usually satisfy most resource demands and provide the entire hardware and software stack for them to offer services over the Internet.

However, there has been only little research on integrating both on-site and Cloud resources. Most of the ongoing projects aim at emulating certain aspects of commercial providers [6] or face the challenge of establishing a hybrid system by storing identical copies of data on-site and in the Cloud [7]. Generally, data is either regarded in its atomic form or in its entirety. Other work focuses on the technical aspects of combining local and remote storage facilities [8], but not on the actual placement decisions. Only recently have there been advances to include Cloud resources for ILM systems, but these have been rather limited to certain scenarios [9].

Cloud storage services are based on the same concepts as distributed databases. Thus, the elementary challenges faced for highly-distributed systems are also present in a Cloud environment, such as concurrency control, reliability, or consistency [10]. It has been recognized by the research community that the old paradigms established for local databases, such as ACID, are not always suitable for or even necessary in a distributed or Cloud environment. Instead, new requirements have been proposed, such as predictability and flexibility [11]. Again, this somewhat impedes the combination of Cloud and on-site resources.

We propose a hybrid approach, both in terms of storage capabilities and data organization. In the next section, we introduce a classification based on certain criteria inherent to data stores and different applications. Afterwards, we present a formal granular view on data and workloads. These granules can in turn be used to associate different granules to different storage systems, which is covered in the next to last section. In the conclusion, we summarize this work and present an outlook on future research.

II. Classification of Data Management Systems

As mentioned above, the worldwide distribution of data has sparked a rethinking of requirements for data stores. Hence, traditional taxonomies of databases that regarded relational database management (RDBMS) systems only are not sufficient anymore. Based upon recent advances in this field, we use an empirical approach to propose a new classification that incorporates the requirements of modern distributed applications. This helps us to relate the different features of data stores to the characteristics of individual workloads.

When trying to categorize data management systems, there are multiple characteristics that may be taken into account. These include technical features, security and safety aspects, and economical considerations. Within the scope of this work, the latter three shall be omitted and for the sake of illustration only a limited amount of technical dimensions will be addressed, namely:

- Scalability,
- Transaction Policy and
- Storage Model

With respect to these three dimensions, we propose Figure 1 as a classification device for data management systems.

In Figure 1, a traditional RDBMS would most likely be placed around point $(\underline{R}, \underline{A}, \underline{L})$, whereas the individual axis values for most Cloud data stores move further outward and the according systems could be placed closer to the point $(\underline{K}, \underline{B}, \underline{C})$. Figure 1 shall be referred to as the data placement cuboid.

In terms of nomenclature, there are a few things to notice:

- The dimension of **Scalability** refers to both the degree of distribution of a platform and the flexibility to quickly provide new resources on demand. While a *Parallel* set-up usually requires similar machines in terms of hardware and software configuration, this is usually not required for a truly *Distributed* or *Cloud* system. However, the latter two might be hard to differentiate as most private distributed infrastructures could easily be transformed into a private Cloud. This is indicated by the dotted line. Nevertheless, we assume that the first three storage systems are on-site while Cloud storage is always provided as a remote service. Hence, in contrast to the other three facilities, Cloud storage is rented while *Local*, *Parallel*, or *Distributed* storage typically is purchased [12].
- For the **Storage Model**, *Row/Column* describes the way data is retained in conventional RDBMS. A *Key/Value* store associates a datum with a unique ID, but does not support schema definitions. Thus, all data is stored as a String, as is the case with, e.g., Amazon's SimpleDB [13]. On the other hand, a *Hybrid* model refers to any system that combines non-traditional approaches with some of the features found in RDBMS. An example of a hybrid system is Google's Bigtable, which allows managing a flexible number of rows based on a dynamic schema and is described as a "sparse, distributed multi-dimensional sorted map" [14].
- The **Transaction Policy** describes how potentially conflicting operations are handled by the system. *ACID* comprises a well-known set of transaction properties and has been implemented in virtually every RDBMS. *BASE*, on the other hand, was introduced only recently [15] and is an acronym for "**B**asically **A**vailable, **S**oft state, **E**ventual consistency". It should be noted that most distributed data stores rely on *BASE* for transaction management. This circumstance is essentially derived from the CAP theorem [16]. However, there may be relaxations of certain, but not all ACID properties, as indicated by the dotted line [17].

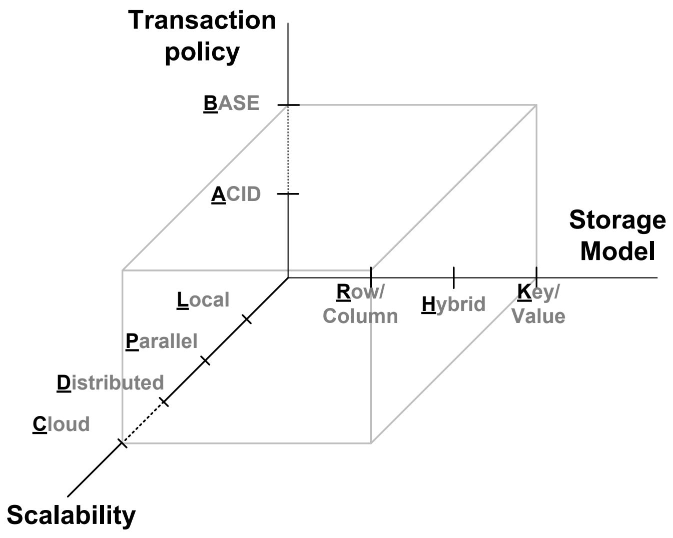

Fig. 1. Classification of data management systems.

Currently, data placement is rather infrastructure-centric, i.e., available systems are evaluated regarding the three dimensions described above and the one most fitting is selected as a data store. We argue that this approach lacks both flexibility and agility and propose a data-centric view, where certain metainformation about data is considered to formalize data placement strategies. The strategy will then determine the most appropriate storage location not for the entire data as a whole, but rather for subsets of individual pieces of data at different points in time. This hybrid approach encompasses integration of multiple data storage systems, both on-site and in the Cloud. The next two sections illustrate how granules for data placement strategies are composed and how the hybrid framework is conceived, respectively.

III. Granular data management

The traditional approach of establishing data placement strategies in the context of ILM tends to handle all involved entities isolated from one another [1]. For example, pieces of data are only considered in their most atomic form (e.g., as an individual record or binary file). Also, the lifecycle of a datum is usually partitioned into fixed-size time frames. Moreover, data access often is monitored globally only, i.e., with no regard to individual user or user group access profiles. This stems from the fact that data storage systems were conventionally ordered hierarchically according to their access speed and respective cost per storage unit.

The introduction of Cloud Computing justifies a new look on granularization of the three dimensions data, time, and users, as Cloud services offer new characteristics in terms of storage cost and accessibility as described above.

First, we give a formal definition of a simple data placement problem considering the amount of data only, the users accessing it and the nature of these accesses. We consider two kinds of access operations, non-conflicting (such as reads) and potentially conflicting (such as updates).

Definition 1 (Data access operations)**.** *Let D denote a set of atomic pieces of data with respect to some application (e.g., one value in an n-tuple or a binary file in a file system). Let U denote a set of individual users accessing this data. For $i, j \in \mathbb{N}, 0 \leq i \leq |D|$ and $0 \leq j \leq |U|$, we define a non-conflicting access operation of a datum $d_i \in D$ by a user $u_j \in U$ as a relationship*

$$d_i \dashrightarrow u_j \text{ (e.g., a datum is read by a user)}$$

We define a potentially conflicting access operation of a datum $d_i \in D$ by a user $u_j \in U$ as a relationship

$$d_i \dashleftarrow u_j \text{ (e.g., a datum is written by a user)}$$

Notice that this definition does not exclude that a datum is accessed by more than one user or that a user accesses more than one datum. However, if for the same datum both operations are invoked by a user, we assume a potentially conflicting access.

Based on Definition 1, one could establish a simple data placement decision considering single users, pieces of data, and potential conflicts. Consider the following example: for three pieces of data d_1, d_2, d_3 and two users u_1, u_2 the following access information has been monitored

$$d_1 \dashrightarrow u_1 \;;\; d_2 \dashrightarrow u_1 \;;\; d_3 \dashleftarrow u_1 \;;\; d_3 \dashleftarrow u_2.$$

The placement strategy could advise to store d_1, d_2 on the local hard disk of user u_1 as they are accessed by her alone. For d_3, on the other hand, a shared storage is needed as this datum is accessed by both users. Moreover, as the access operations are potentially conflicting, the shared data store needs to provide some mechanisms to ensure consistency.

We now formalize a more granular view on data inspired by the ideas introduced in [18].

Definition 2 (Data granules)**.** *Let D be defined as above. Instead of focusing on singular pieces of data, we now consider granules of data, such as a subset of values in an n-tuple or a number of files with distinct properties. For $i, j \in \mathbb{N}, 0 \leq i \leq |D| =: \theta, 0 \leq j \leq (|D|)_i = (\theta)_i$ a granule of data $d_{i,j}$ is defined as*

$i = 1$	$d_{1,1} = \{d_1\}$; $d_{1,2} = \{d_2\}$; ... ; $d_{1,(\theta)_i} = d_{1,\theta} = \{d_\theta\}$
$i = 2$	$d_{2,1} = \{d_1, d_2\}$; $d_{2,2} = \{d_1, d_3\}$; ... ; $d_{2,(\theta)_2} = \{d_{\theta-1}, d_\theta\}$
$i = 3$	$d_{3,1} = \{d_1, d_2, d_3\}$; $d_{3,2} = \{d_1, d_2, d_4\}$; ... ; $d_{3,(\theta)_3} = \{d_{\theta-2}, d_{\theta-1}, d_\theta\}$
$\vdots$	$\vdots$
$i = \theta$	$d_{\theta,1} = d_{\theta,(\theta)_\theta} = \{d_1, \ldots, d_\theta\}$

*Similarly, a granule of users can be defined as $u_{k,l}$, $k, l \in \mathbb{N}, 0 \leq k \leq |U|, 0 \leq l \leq (|U|)_k$. With respect to $d_{i,j}$, we refer to i as the **Level of Granularity** (LoG) and constitute that $d_{i,j}$ has a higher Level of Granularity than $d_{i-1,j}$, whereas $d_{i,j}$ has the same Level of Granularity as $u_{i,l}$. As the sole granule of LoG $\theta + 1$ basically contains the same non-trivial elements as the one of LoG θ, these two granules have the same LoG.*

While a fine-grained user granule, such as $u_{1,1}$, refers to a single user, a granule with higher LoG, e.g., $u_{2,1}$ might denote a team or department. The access operations $\dashrightarrow$ and $\dashleftarrow$ can intuitively be defined on granules with the same LoG: $d_{i,j} \dashrightarrow u_{i,l}$ and $d_{i,j} \dashleftarrow u_{i,l}$. To improve readability and simplify illustration, in the scope of this work we apply $\dashrightarrow$ and $\dashleftarrow$ to granules of same LoG only, such as $d_{3,1}$ and $u_{3,2}$, but not $d_{3,1}$ and $u_{2,1}$. However, the same principles apply to granules of different LoG. Moreover, as a granule combination may contain both non-conflicting and potentially conflicting access operations, we assume that the potentially conflicting ones overrules the non-conflicting ones: if the access of a data granule $d_{i,j}$ by a user granule $u_{i,k}$ contains both operations $\dashrightarrow$ and $\dashleftarrow$, we always indicate $d_{i,j} \dashleftarrow u_{i,k}$.

The axis labelings of Figure 2 visualize the granularization of individual data and user entities. It should be noted that for the sake of illustration not all possible granulates are depicted. The colored squares in Figure 2 indicate different types of operations where a green square represents a non-conflicting access and a red square identifies a potentially conflicting access on a datum.

Using different levels of granularity enables a novel view on data placement strategies. Referring back to the example

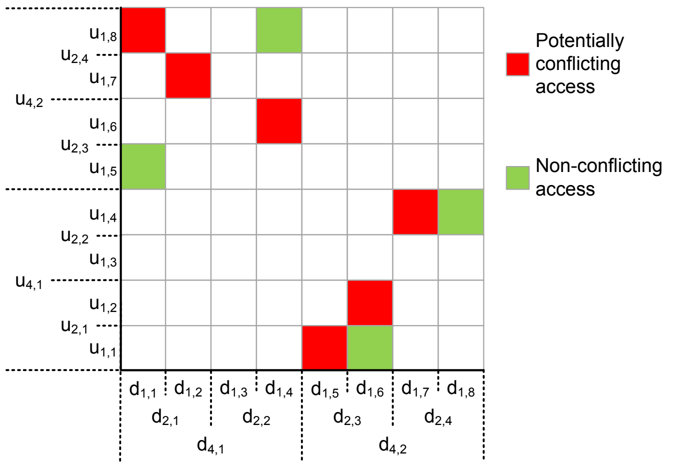

Fig. 2. Granular data access indicating variations in user group size and data amount.

illustrated in Figure 2, a low LoG of 1 might indicate that granules $d_{1,7}$ and $d_{1,8}$ should be stored on the local storage of user $u_{1,4}$, as she is the only one accessing them. However, when considering an LoG of 2, it might be deduced that $d_{2,3}$ is closely related to $u_{2,1}$. Hence, if $u_{2,1}$ refers to some sort of user group, e.g., a department in a company, $d_{2,3}$ should be stored on the department's file server. Notice that the granules here are treated as black boxes, i.e., there is no way to identify single building blocks of a granule such as individual users in $u_{2,1}$. Therefore, it cannot be deduced that the granule $d_{2,3}$ can or should be stored on an individual user's hard drive. Increasing the Level of Granularity to 3 allows for yet another data placement strategy that was not observable earlier, i.e., the correlation between $d_{4,1}$ and $u_{4,2}$.

While in general granules will be treated as black boxes, in certain scenarios it might be useful to identify the atomic elements composing a granule. Hence, we define the function ϕ that takes in a granule of any LoG and extracts the set of elementary entities that the input granule is composed of, e.g., $\phi(d_{2,1}) = \{d_{1,1}, d_{1,2}\}$.

Definition 3 (Decomposition of granules)**.** *Let D be defined as above. For an arbitrary granule $d_{i,j}$ we define the decomposition function ϕ as*

$$\phi(d_{i,j}) \mapsto \{d_k\} \, | \forall_k d_k \in d_{i,j}$$

As the goal of this work is to establish a framework suitable for Information Lifecycle Management, the dimension of time T is an important variable. In accordance with Definition 2 we apply the same approach to T and derive an analogous granularization $t_{i,j}$. This expands the two-dimensional illustration of Figure 2 to the three-dimensional view depicted in Figure 3. For the sake of readability, we omit the labelings of the two-dimensional components introduced earlier. The observant reader will notice that for $t_{1,8}$ the correlation between users and data is identical to that of Figure 2. However, by adding a third dimension, entirely new indications can be recognized, such as one between $t_{2,4}$, $d_{2,1}$, and $u_{2,4}$ (notice, that it is concealed in the figure).

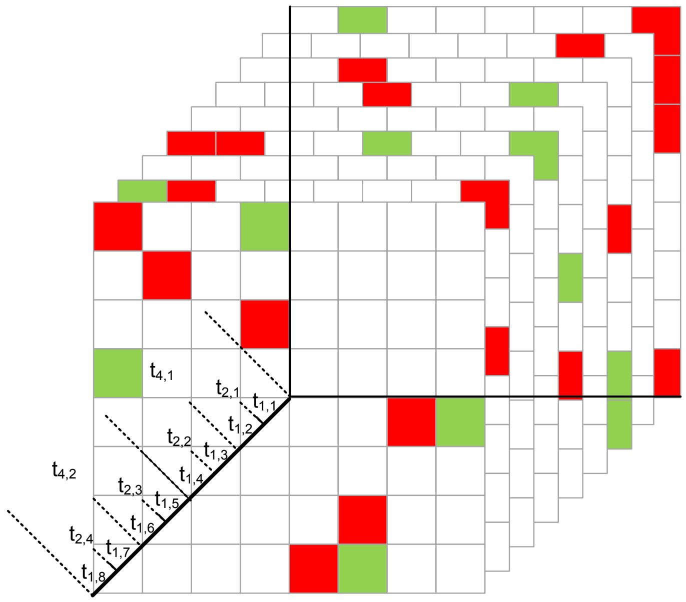

Fig. 3. Granular data access indicating variations in user group size, time, and data amount.

We incorporate Definition 1 into Definition 2 to reference this additional dimension and establish according operations.

Definition 4 (Time granules and granular access)**.** *Let $D, U, d_{i,j}, u_{i,k}$ be defined as above. Let T denote a set of atomic time frames t, where the length of each individual t is fixed (e.g., a second). Similarly to Definition 2, for $0 \leq l \leq (|T|)_i$ a granule of time $t_{i,l}$ is defined as*

$i = 1$	$t_{1,1} = \{t_1\}$; $t_{1,2} = \{t_2\}$; ...
$i = 2$	$t_{2,1} = \{t_1, t_2\}$; $t_{2,2} = \{t_1, t_3\}$; ...
$\vdots$	$\vdots$

Based on the operations introduced in Definition 1, a user granule access on a granule of data within a certain time granule can be formalized as

$$d_{i,j} \overset{t_{i,l}}{\dashrightarrow} u_{i,k}$$

for non-conflicting access and

$$d_{i,j} \overset{t_{i,l}}{\dashleftarrow} u_{i,k}$$

for potentially conflicting access. Again, $\dashleftarrow$ overrules $\dashrightarrow$ if both operations are present within a granule combination.

Using the concepts of Definition 4, we are now able to express user data access pattern over time on different levels of granularity. This helps to establish general policies for data placement in the context of ILM.

IV. Granular ILM in a Hybrid Storage Environment

In Section II, we introduced one approach to classify different storage systems according to a number of their technical features. Section III established a formal notation for data access granularity. We now integrate these two aspects in order to manage the placement of entire data workloads. We first formulate the concept of a workload.

Definition 5 (Workloads and granular workloads)**.** *A workload $W = (D, U, T, \mathcal{A})$ is characterized by the data D to store, the set of users U accessing this data, the overall available time frame T for the workload, and the set of access operations $\mathcal{A}$. Here, the elements of $\mathcal{A}$ represent the operations introduced in Definition 1 and extended in Definition 4. We define sets of all possible granules D^i in D, U^i in U, and T^i in T based on a fixed Level of Granularity i as*

$$D^i \subseteq \left\{d_{i,1}, \dots, d_{i,(|D|)_i}\right\}$$
$$U^i \subseteq \left\{u_{i,1}, \dots, u_{i,(|U|)_i}\right\}$$
$$T^i \subseteq \left\{t_{i,1}, \dots, t_{i,(|T|)_i}\right\}$$

so that the following conditions are satisfied:

$$\left\{d_1, \dots, d_{|D|}\right\} \subseteq \bigcup_{d_{i,j} \in D^i} \phi(d_{i,j})$$

$$\left\{u_1, \dots, u_{|U|}\right\} \subseteq \bigcup_{u_{i,k} \in U^i} \phi(u_{i,k})$$

$$\left\{t_1, \dots, u_{|T|}\right\} \subseteq \bigcup_{t_{i,l} \in T^i} \phi(t_{i,l})$$

We refer to the entire set of operations based on the granularization D^i, U^i, and T^i:

$$\forall_{d_{i,j} \in D^i, u_{i,k} \in U^i, t_{i,l} \in T^i} \left\{d_{i,j} \overset{t_{i,l}}{\dashrightarrow} u_{i,k}\right\} \cup$$
$$\forall_{d_{i,j} \in D^i, u_{i,k} \in U^i, t_{i,l} \in T^i} \left\{d_{i,j} \overset{t_{i,l}}{\dashleftarrow} u_{i,k}\right\}$$

as $\mathcal{A}^i$.

Using this granular approach, W can be expressed in terms of LoG i as

$$W^i = (D^i, U^i, T^i, \mathcal{A}^i)$$

We illustrate the granularization of a workload using an example. Consider the granules of data $d_{3,1}$, $d_{3,2}$ and the granules of users $u_{3,1}$ and $u_{3,2}$ as depicted in Figure 4. Note that for simplicity purposes, additional granules of LoG are not portrayed, neither is the third dimension time. Let us assume, that the displayed relationship between data and users is maintained over $t_{3,1}$, but there are no access operations outside of $t_{3,1}$. A granular workload W^3 is then for example characterized by

$$W^3 = \Big(\left\{d_{3,1}, d_{3,2}\right\}, \left\{u_{3,1}, u_{3,2}\right\}, \left\{t_{3,1}, t_{3,2}\right\},$$
$$\left\{d_{3,1} \overset{t_{3,1}}{\dashrightarrow} u_{3,1} \ , \ d_{3,2} \overset{t_{3,1}}{\dashleftarrow} u_{3,1}\right\}\Big)$$

Note, that in order to satisfy the conditions established in Definition 5, all atomic user, data, and time objects need to be included in the granular workload, even if they are not associated with any other atomic elements. In the example, $u_{1,4} \in U$ never accesses any datum, but needs to be contained in U^3. Thus, $u_{3,2}$ is part of the granular workload.

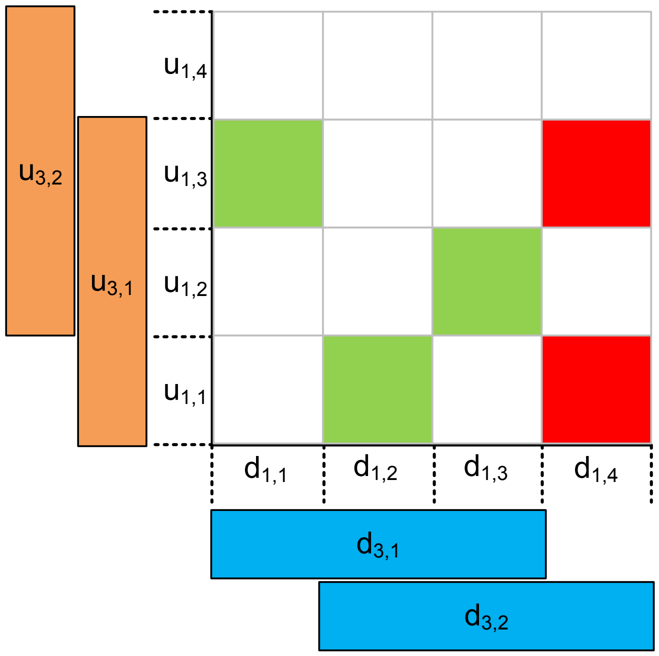

Fig. 4. An example of a two-dimensional granular workload with LoG 3.

We use the data placement cuboid introduced in Section II as a means to relate different dimensions of storage systems to the individual features $D^i, U^i, T^i, \mathcal{A}^i$ depending on the LoG i. We assume that one or more granularizations of LoG i are determined based on observing noticeable patterns in $\mathcal{A}^i$. Note, that we point out the most eminent correlations only, even though others might be possible. Potential formal policies depend on more factors than covered in this work, such as actual cost of hardware and Cloud service or the type of application in regard. However, individual considerations below are applicable in all scenarios.

A. Scalability

The choice between Cloud storage and on-site storage, i.e., local, parallel, or distributed storage, depends on D^i, U^i, and T^i. High-level granules may require too much disk space to handle them efficiently on Local or Parallel storage or retaining them there might be too expensive. Hence, these granules need to be outsourced to distributed or Cloud storage infrastructures that satisfy the desired scale demands. This also holds for higher-level U^i, as the wider a system is distributed, the better it usually can handle a large number of users. On the other hand, T^i also influences whether Cloud or on-site storage is used. Typically, on-site storage hardware is purchased while Cloud storage is rented for a limited time. Therefore, for higher-level T^i the cost of rent becomes higher than the cost of purchase, thus on-site storage might be preferable [19].

In terms of combining both on-site and Cloud resources, considering $\mathcal{A}^i$ supports placement policies as well. If certain granules of data are accessed very infrequently, i.e., in a few $t_{i,l}$, the granules need to be stored on on-site hardware for these specific $t_{i,l}$ only. In general, it is appropriate to outsource the data granules during all other time granules. Also, the composition of the individual granules $u_{i,k}$ in U^i, i.e., $\phi(u_{i,k})$ might suggest storing data on highly distributed storage such as a Cloud infrastructure. For example, this applies when the atomic elements of $u_{i,k}$ reside in different physical locations and do not have access to a common storage facility.

B. Storage Model

The Storage Model again depends on D^i, but also on $\mathcal{A}^i$. We assume that key/value stores are able to retrieve single pieces of data faster, but are not optimized to handle aggregates of data. On the other hand, relational row/column stores are designed that way, i.e., they offer built-in functions such as grouping. Thus, for higher-level D^i, row/column stores might be more appropriate. Considering $\mathcal{A}^i$, a lower-level granularity indicates a lack of access patterns on data and therefore a high number of atomic operations. For these, key/value stores are more appropriate [20].

C. Transaction Policy

In terms of Transaction Policy, we need to consider foremost the potentially conflicting access operations in $\mathcal{A}^i$ as well as the granularity of U^i. A high relative incidence of potentially conflicting access operations in relation to the overall number of accesses and high-level granules of users may yield potential inconsistencies within the data. For example, for U^1 and $\mathcal{A}^1$ this incidence is either 0 or 1, for U^2 and $\mathcal{A}^2$ it may take values of 0, .25, .5, .75, or 1 and so forth. To cope with varying degrees of inconsistencies, a threshold on this incidence can be introduced. If the incidence rate is above the threshold, a data store providing ACID guarantees is mandated, while otherwise BASE policies might be sufficient. Generally speaking, a high Level of Granularity in U^i is more prone to access conflicts and therefore potential inconsistencies. However, depending on the nature of the application associated with the workload, this circumstance might be tolerable [17].

In future work, we plan to formalize the fuzzy strategies outlined above and establish an optimization problem for a given workload. Based on the technical and economical features of all systems in question, we can determine both a granularization most suitable for the entire workload and the optimal storage location for each of the resulting granular subworkloads.

V. Conclusion

We have presented an approach to categorize different storage systems based on various feature dimensions. Subsequently, we established a formal definition of a granular workload based on data, user, and time granules. Finally, we related the degree of granularity to the individual features of the classification system.

The approach introduced in this work may serve as the foundation for a sophisticated data placement framework in the context of Information Lifecycle Management. In addition to the concepts presented here, the specific costs associated with a particular infrastructure need to be considered for real-world systems. Moreover, we are working on implementing a highly flexible data store incorporating multiple Cloud providers as well as commercial and open-source database software to allow seamless and transparent data migration. Here, we will integrate the policies mentioned above.

References

[1] C. Brooks, G. Chiapparini, W. Feyants, P. Galgali, and V. F. Jose, *ILM Library: Techniques With Tivoli Storage And IBM Totalstorage Products*. IBM Redbooks, 2006. [Online]. Available: http://www.redbooks.ibm.com/redbooks/pdfs/sg247030.pdf

[2] Oracle. (2007, June) Information Lifecycle Management for Business Data. http://www.oracle.com/technology/deploy/ilm/pdf/ILM_for_Business_11g.pdf.

[3] P. P. Tallon, "Understanding the Dynamics of Information Management Costs," *Communications of the ACM*, vol. 53, no. 5, pp. 121–125, 2010.

[4] Peter Mell and Tim Grance, *The NIST Definition of Cloud Computing*, Information Technology Laboratory Std., July 2009.

[5] J. Staten, S. Yates, F. E. Gillett, W. Saleh, and R. A. Dines, "Is Cloud Computing Ready For The Enterprise?" Forrester Research, 400 Technology Square Cambridge, MA 02139 USA, Tech. Rep., March 2008.

[6] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-Source Cloud-Computing System," in *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 124–131.

[7] Borja Sotomayor, Ruben S. Montero, Ignacio M. Llorente, and Ian Foster, "Virtual Infrastructure Management in Private and Hybrid Clouds," *IEEE Internet Computing*, vol. 13, pp. 14–22, 2009.

[8] Heeseung Jo, Youngjin, Kwon, Hwanju Kim, Euiseong Seo, Joonwon-Lee, and Seungryoul Maeng, "SSD-HDD-Hybrid Virtual Disk Consolidated Environments," in *Proceedings of the 4th workshop in Virtualization in High-Performance Cloud Computing (VHPC)*, August 2009.

[9] M. Meisinger, C. Farcas, E. Farcas, C. Alexander, M. Arrott, J. D. L. Beaujardire, P. Hubbard, R. Mendelssohn, and R. Signell, "Serving Ocean Model Data on the Cloud," in *Proceedings of OCEANS 2009 MTS/IEEE*, 2009.

[10] M. T. Özsu and P. Valduriez, *Principles of distributed database systems (2nd ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1999.

[11] D. Florescu and D. Kossmann, "Rethinking cost and performance of database systems," *SIGMOD Rec.*, vol. 38, no. 1, pp. 43–48, 2009.

[12] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," Tech. Rep., February 2009.

[13] J. Murty, *Programming Amazon Web Services*. O'Reilly, 2008.

[14] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: a distributed storage system for structured data," in *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2006.

[15] D. Pritchett, "BASE: An Acid Alternative," *Queue*, vol. 6, no. 3, pp. 48–55, 2008.

[16] Eric A. Brewer, "Towards robust distributed systems," in *Proceedings of the 19th annual ACM symposium on Principles of distributed computing*, 2000.

[17] W. Vogels, "Eventually consistent," *Communications of the ACM*, vol. 52, no. 1, pp. 40–44, 2009.

[18] T. Y. Lin, Y. Y. Yao, and L. A. Zadeh, *Data mining, rough sets and granular computing*. Heidelberg, Germany, Germany: Physica-Verlag GmbH, 2002.

[19] Roy Campbell, Indranil Gupta, Michael Heath, Steven Y. Ko, Michael Kozuch, Marcel Kunze, Thomas Kwan, Kevin Lai, Hing Yan Lee, Martha Lyons, Dejan Milojicic, and Yeng Chai Soh, "Open Cirrus$^{\text{TM}}$Cloud Computing Testbed: Federated Data Centers for Open Source Systems and Services Research," Intel, Tech. Rep., 2009.

[20] Pat Helland, "Life beyond Distributed Transactions: an Apostate's Opinion," in *3rd Conference on Innovative Data Systems Research*, 2007, pp. 132–141.