Distributed Datastores: Towards Probabilistic Approach for Estimation of Reliability

Kyrylo Rukkas and Galyna Zholtkevych

V.N. Karazin Kharkiv National University
4, Svobody Sqr., 61022, Kharkiv, Ukraine
galynazholtkevych1991@gmail.com

Abstract. This paper focuses on the contradiction that follows from Brewer's Conjecture for distributed datastores: the need to deliver qualitative data to the user requires a guarantee of consistency, availability and stability of the system at the same time, but Brewer's Conjecture claims that this is impossible. To overcome this contradiction in the paper it is suggested to estimate statistically violation of these guarantees. To implement this idea the interdependencies between the guarantees and indicators of information quality are considered, different kinds of models specifying the general architecture and behaviour of datastores are described, and, finally, the basic metrics of guarantee ensuring are defined. This consideration allows us to formulate several problems that have both theoretical aspects and engineering applications for the improvement of the technology of distributed data processing.

Keywords: distributed datastore, CAP-theorem, information quality, statistic metrics, simulation
Key Terms: DistributedDataWarehousing, ConcurrentComputation

1 Introduction

Nowadays, any kind of human activity requires an infrastructure to support efficiently the corresponding process of decision making. A modern answer to such a requirement is an information system that is an integrated set of components to collect, store and process data, to deliver information, knowledge, and digital products. Today the development trend of Information and Communication Technology is a wide use of networking technologies. Therefore a typical modern information system is a distributed data processing system with a distributed datastore.

Now it is generally accepted that a distributed datastore should guarantee the following properties: consistency (C), availability (A), and partition tolerance (P). They are discussed in papers [5] and [1], but this discussion is too implicit. These works had been critically reviewed in [2]. This criticism is based on the fact that nobody had so far given explicit and rigorous definitions of these guarantees. Taking into account this remark we accept the following understanding as the origin point of our research:

consistency means that all replicas of any data unit contain the same data at the same time point;

availability means that a datastore eventually returns a response (either a success report or a failure notification) on each request;

and finally, ***partition tolerance*** characterizes the ability of a datastore to continue to operate despite arbitrary message losses or failure of part of the system (sometimes to refer to this ability the more general concept **fault-tolerance** is used).

In the ideal case a distributed datastore should provide these guarantees. In contrast to relational datastores like SQL databases that satisfy ACID properties and ensure the system safety, non-relational datastores do not provide complete safety for the information system. As known Brewer's conjecture [5], being called sometimes CAP theorem, says that it is impossible to maintain simultaneously all three CAP-guarantees in asynchronous or partially synchronous network and maintain safety in this way. Taking into account that synchronization of a distributed datastore decreases essentially system throughput we study consequences of Brewer's conjecture for asynchronous distributed datastores.

A lot of research had been wasted at the consideration of CAP-guarantees. As it is impossible to implement all three of them, we suggest a new approach: to characterise stochastically that each of these requirements is fulfilled or not (see Sec. 5) and do the best for a consumer - provide him with mechanisms used in different datastores for restoring the validity of the CAP-guarantees.

This supposition leads us to the need to develop a simulation framework for supporting numerical experiments to study these mechanisms.

As we said above, the principal objective for an information system design process is to provide a consumer with qualitative information just in time. Therefore we should understand how information quality (IQ) and the CAP-guarantees are connected. In order to clarify this interconnection we give some model of information quality and determine its indicators that depend on providing the CAP-guarantees in Sec. 2.

In Sec. 4 we present the conceptual model of a distributed datastore proposed as a background for the framework that should be developed. In Sec. 3 we discuss briefly the models for maintaining the consistency property in distributed systems. And finally, in Sec. 5 we give rigorous definitions for stochastic criteria of the CAP-guarantees providing, which is based on the presented conceptual model.

Further to avoid cumbersome formulations we shall say "CAP-properties" instead of "ensuring CAP-guarantees".

2 Interrelations between IQ model and CAP-properties

The problem to identify Information Quality (IQ) model was widely described and discussed in [3], [8], [4]. On our opinion, this discussion had been compactly summarized in [7]. The information represented in these sources correlates with

data quality standards in [6]. Below we give the list of IQ indicators according to this paper:

accessibility is the indicator that characterises the extent of availability and fast retrievability of information;

appropriate amount of information is used to denote if the scope of information is appropriate for the task at hand;

believability means that the information is considered as true and credible;

completeness evaluates whether information is sufficiently broad and deep to solve the task at hand;

concise representation characterises the compactness of data representation;

consistent representation means that all the information processes operate with the same data;

ease-of-manipulation envisages that information is easy manipulated and applied to different tasks;

free-of-error means that information is correct and reliable;

interpretability characterises that information is in appropriate languages, symbols and units, and the definitions are clear;

objectivity denotes that information is considered as unbiased and impartial;

relevancy envisages that information is applicable and helpful for the task at hand;

reputation determines that information should be highly regarded in the terms of its source or content;

security is satisfied if the access to information is restricted appropriately in accordance with the access rights;

timeliness is fulfilled if information is up-to-date for the task at hand;

understandability means that information is well-comprehended;

value-added assumes that information is beneficial and provides advantages from its use.

These indicators are used to define different views of an information system. We stress that the consideration is given from the next points of view: product and service quality. These quality indicators are also classified taking into account different views, namely, system specifications and consumer expectations. Therefore they had been grouped in a two-dimensional Table 1 presented below. (See more complete description of this classification in [7]).

Focusing on our subject, we consider only those of indicators, that depend on CAP-properties. Let us explain the reasons that has motivated us to take exactly these indicators of IQ model.

Obviously, by the definition the consistency has the direct impact on the consistent representation indicator. Also, consistency ensures the correct information, thus the free-of-error indicator directly depends on it. Furthermore, if the consistency is fulfilled, there may be a lot of replicas in the system, that is decreasing the concise representation indicator. And one can simply see that the consistency definition involves up-to-date information and then timeliness immediately depends on the consistency. Consistency does not have an influence

Table 1. Information Quality Model

	Conforms to Specifications	**Meets or Exceeds Consumer Expectations**
Product Quality	Sound Information: – Free-of-Error – Concise Representation – Completeness – Consistent representation	Useful Information – Appropriate Amount – Relevancy – Understandability – Interpretability – Objectivity
Service Quality	Dependable Information: – Timeliness – Security	Usable Information: – Believability – Accessibility – Ease-of-Manipulation – Reputation – Value Added

on the accessibility, because if the consistent information is received, it does not necessarily mean that it was retrieved quickly and easily.

Now we tell about the availability interrelations. Firstly, reasoning from the availability definition (see Sec. 1) the accessibility directly depends on the availability measurement. Secondly, the free-of-error indicator definition involves reliable data that is delivered just-in-time, so it also has a direct dependence on the availability. Evidently, it does not impact on the consistent representation and concise representation indicators; also it is not tightly connected with the timeliness indicator. Further in this paper we shall determine availability more strictly: we shall denote the availability as the meantime between a request and the response on it. And following from that definition, with improving availability the speed of retrieving data is increased, but it does not mean that these data are up-to-date.

The last group of interrelations is about the partition tolerance. It has an impact on the consistent representation, free-of-error, accessibility and timeliness quality indicators. The thing is that the perfect partition tolerance fulfillment ensures the successful response from a distributed datastore. In its turn, the successful response must always contain consistent, correct, reliable and up-to-date information. Otherwise the system answer is counted as an error message. Therefore greater probability to get the successful response gives higher values of indicators mentioned above.

Evidently, it does not have any influence on the concise representation in contrast to the consistency.

We summarize this connection in Table 2.

The endorsement of some interrelations and the discovering their behaviour can be obtained by the further experiments. The rest of indicators of IQ model

Table 2. Interrelations between IQ Indicators and CAP-requirements

	Consistency	**Availability**	**Partition Tolerance**
Consistent representation	+		+
Free-of-Error	+	+	+
Concise representation	+		
Accessibility		+	+
Timeliness	+		+

depend on the quality of the information system that provide information dataunits and we are not interested in them.

3 Brief overview of Models for Distributed Systems

The distributed datastores are various, therefore it is necessary to have tools for their analysis. The simplest classification is related to the ratio of read and write operations quantity. This classification should be reasonable from the point of view of three pillars that maintain all distributed systems: consistency, availability, partition tolerance. Hence the list will be as follows:

- Systems with domination of read operations (decision support or retrieval systems);
- Systems with domination of write operations (online transaction processing systems);
- Systems without explicit domination of read or write operations (business systems).

This classification is based on the quantity of read and write operations. It is important to know for us, because it may require different mechanisms for each type of system.

In this section we tell about the way to verify consistency and maintain the probability of its fulfillment following [10, Chapter 7]. An appropriate way to increase this probability is replication – making copies of new data units at each node and their updating.

Traditionally, consistency is discussed in the context of read and write operations in distributed systems. Following the book mentioned above there are two consistency models for different distributed systems: data-centric and client-centric ones. They are applied for different types of distributed systems. The first one, data-centric consistency model is a model for wide usage: online transaction and business systems mentioned above. It involves many types of consistency, such as continuous, sequential, causal and combined one, called grouping operations. The protocols for these consistency types are more complex than protocols in the second model. That is because data-centric model should be usable for systems where a lot of write operations may occur and spoil all the data units.

For instance, imagine if two employees in a company use the same datastore. They need to modify some file. And, unfortunately, it turns off that two employees download the file from the data storage and modify it in a different way and in different places. First one upload the file back and later the second one does the same. But the problem is that the file is modified in different places and there will be some conflicts. This is the simplest example, but if this datastore is distributed on many servers, there are a lot of copies of files in the data storage and more people use the same distributed system, it causes a complete mess in the storage. Therefore these protocols for controlling consistency should prevent such errors when a lot of write operations may occur.

The second one, client-centric model, is used for retrieval systems, where mostly read operations occur, but write ones are rare. Thus it is very costly and not necessary needed to use complex protocols. That is why for this model there exist such a type of the consistency as eventual consistency that ensures such a guarantee for the distributed system that eventually all the data units become same and consistency is fulfilled. The protocols for the client-centric model are also invented (see more in [10, Chapter 7]).

So as soon as we described different types of distributed datastores and established the problem to use different models and protocols for datastores that we focused on, we can go to the next section, where we specify the architecture and important behaviour elements for a distributed datastore.

4 Conceptual and Behavioral Model of Distributed Datastore

Above we described the general accepted model of the information quality that determines indicators which are needed for our research purpose, described models that are used for distributed datastore to satisfy consistency guarantees. But in order to come to the estimation step for the distributed datastore guarantees, obviously, we should also discuss the model of a distributed datastore.

Therefore in this section we represent the model of such systems in two views: structural and behavioural. Below there is given the structural one (Fig. 1).

The main component of our system is a distributed datastore. By definition it is a set of nodes (servers) connected by links between each other. Each node may have one or more neighbours. That is why this entity is composed of nodes and links. Obviously, each link is two-sided and a node entity may have many incidences.

Every node stores data units in replicas. If a node receives a new data unit it finds the data unit with the same identifier and replaces the old replica.

Nodes are classified into ready, busy and dead ones. If a node is busy or dead it ignores all the messages, so in this case the request will be lost. That is why the behaviour in this case is trivial.

The behaviour of a ready node is represented below, in the activity diagram (see Fig. 2).

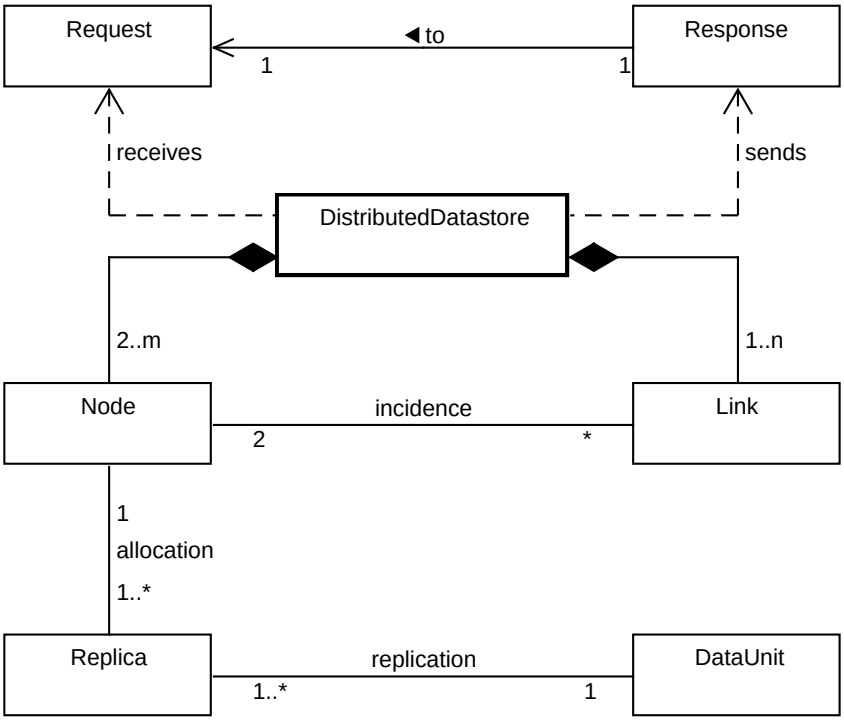

Fig. 1. Conceptual model

To present the behaviour in clear and understandable way, we separate more complex method of node *handle request* from the general behavioural view and show it in Fig. 3.

5 Distributed Datastores: Basic Prerequisites and Metrics

As said above the leading idea of this paper is to suggest an approach for estimating probabilistic metrics of CAP-properties.

It is generally accepted (see [9]) that a distributed datastore is a computer network where information is stored on more than one node, often in a replicated fashion.

Moreover, it is important to mention that a researcher has the possibility to observe datastore events in the sequence according to physical time while each

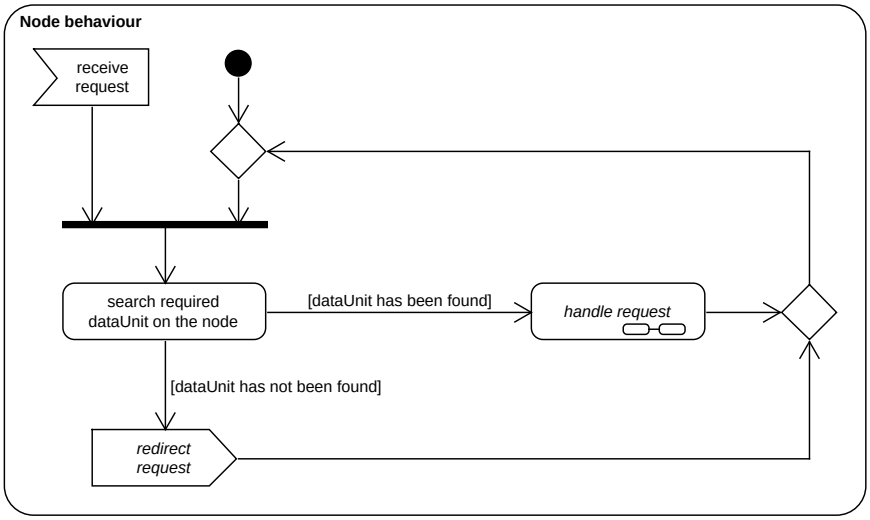

Fig. 2. Behaviour of node

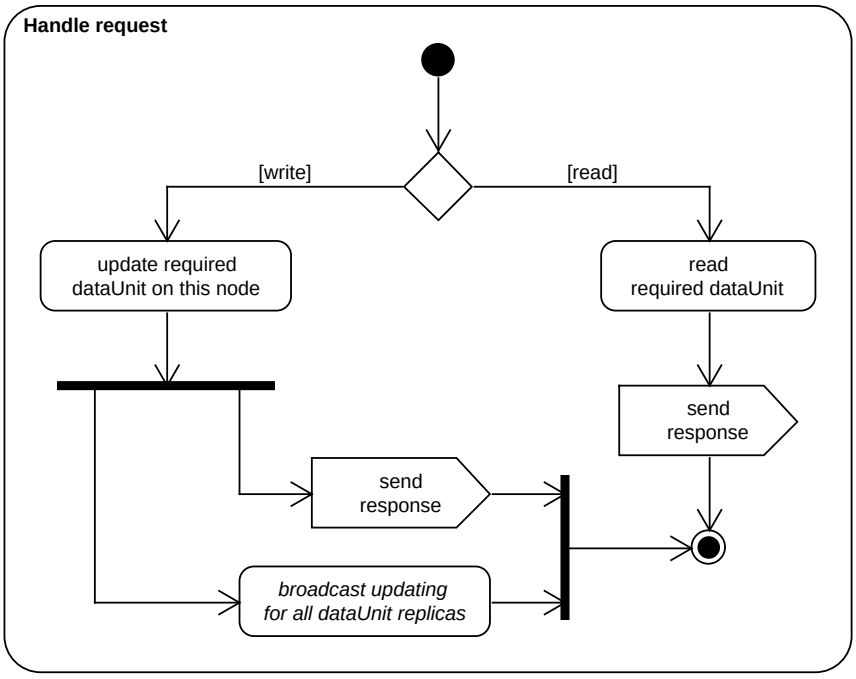

Fig. 3. Behaviour of handle request method

datastore node considers the sequence of events with respect to its local time only. Also we assume that datastores at study meet the eventual consistency requirement [10]. It means that after an isolated read request for any data unit all its replicas eventually have the same value.

Let us represent a short case study in order to understand the motivation to apply the probabilistic approach in asynchronous distributed datastore.

Let us suppose that there is the unreliable node in a distributed system that fails and recovers over over some time. At the initialization moment when nodes are established each node has the probabilistic distribution of recovering time and time of failure. In a distributed system links that bind nodes may also fail. To be able to calculate the probability of partition tolerance fulfillment we need to take into account these distributions in our research.

For now we are ready to describe how we apply the probabilistic approach in our studying by giving the rigorous definitions of CAP-properties. To do this, we describe a distributed datastore using the following mathematical model.

Our model is a tuple (N, L, ∂, D, r), where

N	is a finite set, whose elements correspond to nodes of a distributed datastore;
L	is a finite set, whose elements correspond to links of a distributed datastore;
$\partial : L \to 2^N$	is a mapping that associates each link with two nodes that it connects;
D	is a finite set, whose elements correspond to stored data units;
$r : D \to 2^N$	is a mapping that associates each data unit d with a subset of nodes that store its replica.

Thus, firstly, let us introduce the consistency metric. Taking into account that the consistency is the property when for each data unit its replicas have the same value, we shall consider the probability that all data units at distributed datastore are consistent at a certain time moment. Therefore we define the consistency metric in the following manner.

Definition 1 (Consistency metric). *Let $\sigma_t \in \{0, 1\}$ be the random variable that represents one of two events at time point $t \geq 0$:*

$\sigma_t = 0$	*corresponds to the event "there exists a data unit that is not consistent at the time point t" and*
$\sigma_t = 1$	*corresponds to the event "all data units are consistent at the time point t".*

Then the consistency metric $C(t)$ at time point t is defined by the formula

$$C(t) = \Pr(\sigma_t = 1)\,. \tag{1}$$

The second metric estimates the availability guarantee. We propose to measure the extent of availability as meantime between two events: the request has been received by the datastore and the corresponding response [1] has been sent by the datastore. More, formally:

[1] This response is either a successful report on request or an error message.

Definition 2. *Let τ_t be the time interval between a request receiving at the time point t and the corresponding response. Then the mean response time is defined by the formula*

$$T(t) = \mathbf{E}[\,\tau_t\,]\,. \tag{2}$$

Finally, to estimate the third guarantee, called the partition tolerance, we consider the ability of a datastore to survive network partitions, so that the performance of the datastore does not suffer. This definition is more complicated. Let us consider some time point t. At this instant some nodes are alive, but other ones failed. We denote by N_t^a the set of alive nodes ($N_t^a \subset N$). Similarly, we denote by L_t^a the set of alive links that connect alive nodes.

Definition 3. *We shall say that a data unit $d \in D$ is reachable from a node $n \in N_t^a$ at time point t if there exists a path in the graph (N_t^a, L_t^a) from n to some $n' \in N_t^a \bigcap r(d)$.*

Now we can introduce the metric for the partition tolerance using the previous definition.

Definition 4. *Let $\zeta_t \in \{0, 1\}$ be the random variable that represents one of two events at time point $t \geq 0$:*

- $\zeta_t = 0$ *corresponds to the event "there exists a data unit that is not reachable from some alive node at time point t" and*
- $\zeta_t = 1$ *corresponds to the event "all data units are reachable from any alive node at time point t".*

Then the partition tolerance metric $P(t)$ at time point t is defined by the formula

$$P(t) = \Pr(\zeta_t = 1)\,. \tag{3}$$

6 Conclusion

In this paper we have started studying the problem of the quality for distributed datastores. We have proposed the approach to measure the quality properties of a datastore. Therefore we have described CAP-properties and have built the metric system for CAP-properties estimation. We have described the indicators of the information quality and have found interrelations between the information quality and distributed datastores' properties basing on [7].

In order to have a view of the datastore and be able to work with it we have built its structural and behavioural model and based on this knowledge, we specified probabilistic characteristics for CAP-properties measurement.

Thus, the following steps for the problem set in the paper have been done:

- Formulation of understanding the idea: what are CAP-guarantees for distributed datastores indeed;
- Description of the information quality indicators;
- Investigating the connection between CAP-guarantees and information quality model;
- Building the structural and behavioural models for a distributed datastore;

– Forming "CAP-metrics" as the main idea for studying the quality of distributed datastores.

These steps give us possibilities to study CAP-properties fulfillment using the following background: the fault-tolerance mechanisms for asynchronous systems, concurrent programming, algorithms of data propagation in distributed systems, probably, some issues of internet strategy, Queueing Theory, Mathematical Statistics. Fault-tolerance protocols can be used to invent the algorithms for maintaining the partition tolerance guarantee. Obviously, a researcher should have the good background in Graph Theory, Probability Theory and Concurrent Programming in asynchronous distributed systems. Also, as we need to represent experimental results, it is necessary to imitate the model of a distributed datastore.

Summing up now we might set following problems:

– To simulate the model of a distributed datastore;
– To study different mechanisms to estimate the degree of ensuring each guarantee applying specified metrics;
– To analyse discovered mechanisms and their composition;
– To integrate these mechanisms with simulated asynchronous distributed datastore;
– To estimate theoretically the total time complexity for all provided mechanisms;
– To carry out the experimental estimation.

Acknowledgment. Authors express a deep gratefulness to Grygoriy Zholtkevych and Iryna Zaretska for their help and their criticism.

References

1. Brewer, E.: CAP Twelve Years Later: How the "Rules" Have Changed. Computer, IEEE Computer Society. Vol. 45, No. 2 (2012)
2. Burgess, M.: Deconstructing the 'CAP theorem' for CM and DevOps `http://markburgess.org/blog_cap.html` (2012)
3. CRG. Information Quality Survey: Administrators Guide. Cambridge Research Group, Cambridge, MA (1997)
4. English, L.P.: Information Quality Applied: Best Practices for Improving Business Information, Processes and Systems. Wiley (2009)
5. Gilbert, S., Lynch, N.: Brewers Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. ACM SIGACT News. Vol. 33, No. 2, pp. 51–59 (2002)
6. ISO 8000-1:2011. Data Quality. International Organisation for Standartization (2011).
7. Kahn, B.K., Strong, D.M., Wang, R.Y.: Information Quality Benchmarks: Product and Service Performance. Comm. ACM. Vol. 45, No. 4 (2002)
8. Kovac, R., Lee, Y.W., Pipino, L.L.: Total data quality management: the case of IRI. Conf. on Information Quality (Cambridge, MA), pp. 63–79 (1997)

9. Pessach, Ya.: Distributed Storage: Concepts, Algorithms, and Implementations. CreateSpace Independent Publishing Platform (2013)
10. Tanenbaum, A.S., Van Steen, M.: Distributed systems. Principles and Paradigms (2nd Edition). Prentice-Hall, Inc. (2006)