J. Parallel Distrib. Comput. 86 (2015) 29–44

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Modeling the availability of Cassandra

Carlos Pérez-Miguel [*], Alexander Mendiburu, Jose Miguel-Alonso

Intelligent Systems Group, Department of Computer Architecture and Technology, The University of the Basque Country, UPV/EHU, Paseo Manuel de Lardizabal, 1, 20018, San Sebastian-Donostia, Spain

HIGHLIGHTS

- We model Cassandra under two different failure situations.
- Our models provide information about the availability of Cassandra.
- We validate our models through experimentation with a real system.
- Several use cases of our models with real applications using Cassandra are given.

ARTICLE INFO

Article history:
Received 23 September 2014
Received in revised form
27 July 2015
Accepted 9 August 2015
Available online 17 August 2015

Keywords:
Availability
Fault tolerance
Modeling techniques
Markov processes
Peer-to-peer
Distributed hash table

ABSTRACT

Peer-to-Peer systems have been introduced as an alternative to the traditional client–server scheme. Distributed Hash Tables, a type of structured Peer-to-Peer system, have been designed for massive storage purposes. In this work we model the behavior of a DHT based system, Cassandra, with focus on its fault tolerance capabilities, and more specifically, on its availability when facing two different situations: (1) transient failures, those in which a node goes off-line for a while and returns on-line maintaining its data, and (2) memory-less failures, those in which a node goes off-line and returns with no data. First, we introduce two analytical models (one for each scenario) that provide approximations to the behavior of Cassandra under different configurations, and secondly, in order to validate our models, we complete a set of experiments over a real Cassandra cluster. Experimental results confirm the validity of the proposed models of the availability of Cassandra. We also provide some examples of how these models can be used to optimize the availability configuration of Cassandra-based applications.

1. Introduction

Since the emergence of Peer-to-Peer (P2P) systems [48], which where initially designed for file sharing, industry and academia have paid attention to them as a powerful mechanism to support other types of applications, such as massive storage systems, name indexing or voice calls over IP.

Recent works on Distributed Hash Tables (DHT) [41], a type of structured P2P storage system, have attracted attention because of the possibilities that this type of systems offers in terms of availability, scalability and fault tolerance. One of these proposals is Cassandra [26], a non-relational database system, proposed by Facebook and based on the replication model of Amazon's Dynamo [12]. This replication model is considered to be highly scalable, providing a high availability. However, we could ask ourselves how reliable these systems are. If we consider actual systems serving thousands of requests per unit of time, even small gaps of data unavailability could correspond to large amounts of lost revenue.

In this work, we study the capability of Cassandra to deal with node failures. To that extent, we have developed analytical models of the availability of Cassandra when confronting two different types of failures that resemble typical malfunction situations:

- Transient failures: those which imply the recovery of a failed node after some time, without any loss of the already stored information. However, note that these nodes will lose updates during their off-line period. This type of failures can be caused by network or power outages and node disconnections caused by users (churning).
- Memory-less failures: this would include a total crash of the hard drive of the node with total information loss, and the later recuperation by partially or totally substituting the node and repairing the data that should be stored in that node.

The models we propose are based on the stochastic modeling of replica groups using Markov chains [29]. Markov chains

[*] Corresponding author.
E-mail addresses: carlos.perezm@ehu.es (C. Pérez-Miguel), alexander.mendiburu@ehu.es (A. Mendiburu), j.miguel@ehu.es (J. Miguel-Alonso).

http://dx.doi.org/10.1016/j.jpdc.2015.08.001
0743-7315/© 2015 Elsevier Inc. All rights reserved.

are stochastic processes that obey the Markov property, which establishes that future states of a system depend only on the current state but not on the previous ones. More information about these mathematical models can be found in [34].

As Cassandra's replication model is based on Dynamo, these models could also be applicable, with some effort, to the latter one as well as to other systems based on it, such as LinkedIn Project Voldemort [42] or Basho Riak [24]. The first model we present depends only on the replication strategy of Cassandra, so it is directly applicable to these systems. The second model comprises the reparation processes implemented in Cassandra, which could be different in each of these systems, so in order to adapt the model to them, their reparation processes should be modeled properly.

In order to validate the proposed models, we have run different experiments on a real Cassandra cluster simulating failures and measuring system availability while performing sets of I/O operations. We define the system as available when it successfully replies when an operation is requested. We compare the availability predicted by our models with that measured on the real system, and conclude that our model accurately describes the availability of Cassandra.

The rest of the paper is structured as follows. In Section 2 we provide a summary of the related work about failure models in P2P networks. Section 3 presents an overview of Cassandra and its main characteristics. Section 4 explains the models that approximate Cassandra's behavior. In Section 5 we carry out a set of experiments on a real system with the aim of comparing the results with the availability provided by the theoretical models. In Section 6 we can find several use cases showing how the models can be used to analyze a variety of applications based on Cassandra. In Section 7 we present some conclusions and a discussion of future lines of work.

2. Related work

To the best of our knowledge, this is the first work modeling Cassandra's availability. However, there are similar works close to this that should be mentioned. In [10], Yahoo proposes a system for benchmarking different cloud data-storage systems, the Yahoo Cloud Serving Benchmark (YCSB). YCSB executes operations over cloud systems in order to measure the performance and scalability of these systems. As future improvements to this tool, Yahoo will focus on reliability. At the time of writing this work, no progress has been made in this area. Patil et al. propose in [31] the YCSB++ benchmark, built on top of YCSB. It provides extensions to improve performance understanding and debugging of features such as parallel testing with multiple YCSB clients, bulk-load testing or weak consistency testing. This latter characteristic permits measuring the latency of a "read after write" operation obtaining updated data. This feature is similar to the model proposed by Bailis et al. in [3], that allows characterizing the cost of eventual consistency, computing the probability of obtaining stale data Δ seconds after performing a write operation as a function of the consistency level and the replica size. Note, though, that none of these models and benchmarks are capable of computing the availability of a storage system in the presence of failures.

Authors of [15] propose a failure model of P2PMPI, a peer-to-peer implementation of the Message Passing Interface (MPI) [16], and use it to estimate the optimal degree of replication in this P2P system.

Carra et al. [7] propose a work similar to ours over KAD [40], a DHT based file sharing system without data replication. In KAD, each node is responsible for the objects it stores. Each node publishes references to its objects in other nodes to help find data. Unlike our proposal, they take into account only one type of failure, memory-less failures, and consider that nodes are periodically replaced with new ones without reparation costs. Based on the theory of reliability, they analyze the probability of finding an object published in the system, and study how this probability evolves over time. Finally, they use this model to propose improvements in KAD that could reduce the cost of maintaining the availability of the objects by reducing the amount of object references that can be created in the system.

There are several reliability analyses of different redundancy schemes in P2P networks, on which we have based this work targeting Cassandra. Houri et al. investigate in [19] the behavior of different replication and erasure code schemes (a form of forward error correction) [35] using a non-stochastic reliability analysis to estimate the probability of losing data and the cost of repairing data. In [49] we can find a statistical analysis of replication vs. erasure code schemes whose conclusion is that using erasure codes is less expensive in bandwidth terms, while it increases the Mean Time To Failure by several orders of magnitude. However, in [46] a stochastic analysis comparing erasure codes and replication concludes that simple replication is better when peer availability is low.

In [28] we can find an analysis of the impact of node churn in different erasure code schemes when the availability periods of nodes follow three different distributions: exponential, Pareto and Weibull.

When analyzing the behavior of networked systems, some works [9,43,4] assume independence between nodes. However, in [14] Ford et al. prove that failures in a data-center where nodes share switches and power units are not independent. Nevertheless, in this work we assume independence between nodes in order to reflect a wider range of situations, from volunteer desktop computing systems to corporate data-centers.

With respect to the distribution of failures and recoveries, the uptime and downtime of nodes are commonly modeled as exponentially distributed [9,43,6,38,23,11,44,50], although several works [21,39] prove that the time between disk failures is not exponentially distributed and exhibits autocorrelation and long-range dependences, so authors argue that long-tail distributions (such as Weibull) should be better approximations. Yet, in [14], Ford et al. argue that, in large and heterogeneous environments, the aggregation of large populations of disks with different ages tends to be stable, so the failure rate should be stable as well. When the failure rate is stable, there are no significant differences between both distributions. Therefore, in this work we have considered that modeling failures and recoveries using exponential distributions is a reasonable approximation.

3. An overview of the architecture of Cassandra

Cassandra is a distributed storage system based on the concept of Distributed Hash Table (DHT) [41] that uses data replication in order to improve availability. A DHT is a structured peer-to-peer system which stores {key, value} pairs. Most of the existing DHTs use a look-up algorithm which permits the system to reach any piece of data with a complexity of $O(\log m)$ hops, where m is the number of nodes in the system. This is due to the use of routing tables with partial information, usually with $O(\log m)$ entries, obtaining a good trade-off between time and space costs. However, in Cassandra, nodes are able to reach any point of the system in constant time ($O(1)$ complexity) at the expense of having routing tables of size $O(m)$ at each node.

In Cassandra, the data model is slightly different from a typical DHT. Instead of storing single values, each value is a structured tuple named Column Family. Each column of the tuple has a name, a value and a timestamp. This data model, proposed in [8] for Google's BigTable, is becoming a popular alternative to the