Research on A Tunable Consistency Model of Distributed Database

2 Tunable Consistency Model

2.1 Overall design framework of the model

Commonly used systems divide business applications into three layers: User Interface Layer, Business Logic Layer, Data access Layer. In our model ,we adds a request scheduling layer between the interface layer and the business logic layer.



Fig. 2.1 The comparison of the structure before and after adding the tunable consistency strategy.

The request schedules layer added in the model are the core architectures that achieve tunable consistency. The request scheduling layer is a consistent control layer and is an implementation of an adaptive consistency model. This layer receives the data from the client and performs the consistency level processing. This layer processes the data from the business logic layer and outputs it according to the consistency level.



Fig.2.2 System Module diagram

As shown in Figure 2.2, the request scheduling layer in this paper designed a consistency level processing module, request distribution module, priority request processing queue, request scheduling module and request processing module.

The consistency level processing module is divided into a monitoring module and an adaptive consistency module. The monitoring module is responsible for collecting the relevant information from the Cassandra storage cluster through the Cassandra Node tool and passing the information to the adaptive consistency module. The adaptive consistency module uses the replica factor number adaptive algorithm. The adaptive consistency module calculates how many replications of the request are processed to achieve the requirement for consistency of the application, resulting in a system consistency level.

The request distribution module calculates the threshold for real-time high, medium, and low queues based on the consistency level of each request obtained by concurrency. And the received requests are placed into the three queues according to the threshold.

The priority request processing queue consists of three queues. These queues determine which requests are stored in the queue based on the working mechanism of the requesting distribution module. The requests in the queue are processed in the order of the high queue, the middle queue, and the low queue.

The request processing module reads the request from the three queues according to the priority and passes the request's corresponding consistency level to the Cassandra Java Client in the YCSB module. While the module is responsible for passing the read data to the interface layer.

The Cassandra Java Client is responsible for reading the number of replicationtions that are consistent with the request consistency level from the data access layer.

2.2 Realization of the model

The general idea of the tunable consistency model can be shown as Alg.1.

Algorithm 2: Tunable consistency model

QueHigh = new List();

QueMiddle = new List();

QueLow = new List();

Xn = Calconsistence(Tp, app\_atale\_rate, λr, λw); //根据Alg.2得到一致性级别

Th = Calthreshold(); //阈值计算

PushQue(Th,Xn);

If(time >Δ T1){//防饿死

Pop(QueMiddle,N1);

}else if(time>Δ T2){//防饿死

Pop(QueLow,N2);

}else{//正常处理队列

DealQue();}

Xn is the number of replication, Tp is the time required to write or update the request to propagate to all copies, app\_atale\_rate is need of the application for consistency, λr and λw are the parameters of the exponential distribution followed by the random variable read time and write time, Th is the threshold, N is the number of replications, Δ T is the interval of the initialization time.

2.2.1 Consistency level processing

In the adaptive consistency module, we use the replica factor number adaptive algorithm to get the consistency level of the request. The goal of using the replica factor algorithm is to be able to dynamically handle the consistency requirements of different transaction requests while the system is running, in order to be able to balance the consistency, performance and usability. Therefore, the replica factor number adaptive algorithm not only consider the application requirements, but also consider the system's storage status. The replica factor number adaptive algorithm uses stale read rate to precisely define the consistency requirements of the application. Stale read rate is a probability, which represents the next read request is abnormal. The main algorithm of Consistency level processing is as follows:

Algorithm 2: Consistency level processing

if app\_stale\_rate >=θstale then

Choose eventuall consistency (Consistency Level = One)

else

Compute Xn the number of always consistent replicas necessary to have

app\_stale\_rate >=θstale

Choose consistency level based on Xn

end if

θstale is the stale read rate.

App\_stale\_rate describe the application's need for consistency. When the application requires strong consistency, app\_stale\_rate=0; When the application requires a general level of consistency, app\_stale\_rate=0; When the application requires strong consistency, app\_stale\_rate=100.

The process of arriving is often seen as a Poisson distribution, which is used in both literatures [50] and [51] to describe transactional access. The formula of stale read rate is as follows:



Xn =1 represents the system to meet the ONE consistency strategy. Xiw is the sum of all requests. It obeys the gamma distribution.

[50] Wada H, Fekete A, Zhao L,Lee K. Data consistency properties and the trade-offs in commercial cloud storage: the consumers[C]. 5th Biennial Conference on Innovative Data Systems Research, Asilomar,CA,USA, 2011: 134-143.

[51] Tai A T, Meyer J F. Performability Management in Distributed Database Systems: An Adaptive Concurrency Control Protocol[C]. 4th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, Washington,DC,USA, 1996: 212-216.