Low Cost and High Concurrency ID Maker in Distributed Environment

Kan Yao [1], Hu-Xuan Ni [2], Yuan Wang [3], Jing Tuo [4], Yun-Feng Li[5], Jun-Jie Ding[6]

[1]*State Grid Hubei Electric Power Research Institute, Wuhan, China*
[2]*State Grid Hubei Electric Power Research Institute, Wuhan, China*
[3]*State Grid Hubei Electric Power Research Institute, Wuhan, China*
[4]*State Grid Hubei Electric Power Research Institute, Wuhan, China*
[5]*State Grid Hubei Electric Power Research Institute, Wuhan, China*
[6]*State Grid Hubei Electric Power Research Institute, Wuhan, China*
[1] 351784661@qq.com, [2]312929323@qq.com, [3]365054779@qq.com, [4]77191928@qq.com, [5]460651211@qq.com, [6] 10390449@qq.com

Abstract: In many practical computer engineering projects, will use the ID as a unique identifier. Many methods to generate ID, but it is not easy to choose a cost-effective solution. Through the improved ID method to generate common defects, ID manufacturing is to build a high performance and low cost, for rapid generation of distributed only under the environment of ID.

1. Introduction

Architecture of a web system, the system is the only ID is often encountered problems, in order to adapt to the needs of different scenarios and performance requirements, the common method of generating ID has the following:

1.1 Relational Self Growth Field Generation ID

This is the way, the most common use of the database, the database only. The code is relatively simple, each increment to lock field, expensive. ID digital natural sorting, paging and sorting of the results is very helpful. But the drawbacks in the performance is not up to the requirements of the situation, it is difficult to expand in a single data or read and write separate or more from a major case, only one main library can be generated, the risk of a single point of failure.

1.2 UUID

The use of machine information and time stamp, a theory on the generation of the only global ID. by numberandletter, string length generally in about 32. Has the advantages of rapid, simple. But the disadvantages are obvious: no ranking, cannot guarantee the increasing trend. With UUID as the primary storage, query efficiency low. The storage space is relatively large, if it is a massive database, you need to consider the problem of storage. The amount of data transmission is also great, UUID and reading is not intuitive.

1.3 REDIS

REDIS is a memory database, ID increment operation more quickly, when the traditional relational database since the increased pressure becomes larger, sometimes will take REDIS to generate ID. but when there is no REDIS system, also need the introduction of new components, increasing the complexity of the system. To encoding and the workload is relatively. But there is a single point of failure and the cost of network communication, redis is usually used in the form of a list in the data cache, landing the advantage is not obvious, the need for additional open disk synchronous refresh function, in order to guarantee the redis after the restart of ID only.

1.4 U Se Zookeeper To Generate Unique ID

Zookeeper mainly through the znode data version to generate a serial number, can generate 32 bit and 64 bit data version number, the client can use the version number as a unique serial number. This method is mainly rely on the zookeeper to generate a unique ID, and multistep call API, if the competition is high, need to consider the use of.Zookeeper distributed lock in a production environment requires at least three machines, in order to ensure that a process can be down. And each ID needs network communication, each request time at the MS level.

In summary, the common method of manufacturing ID will frequently produce expensive network communications, network cost. Relational database and redis are single point failure problem obviously. While

DOI: 10.1051/itmconf/20171203003

the use of zookeeper cluster, in order to ensure the high availability of cluster server, you need at least three or more expensive machines, in order to solve the above problems., reduce network communication, improve the clustering performance, then defines a new ID machine, the following named id_maker.

2. System Architecture

Wherever In order to reduce the network communication as much as possible, we can get a batch of ID after the cache to the corresponding machine, so as to avoid the distribution of ID to store ID machine request. Then when the ID is fast enough, then get ahead of another batch of ID, added to the original cache architecture as shown Are in figure1below

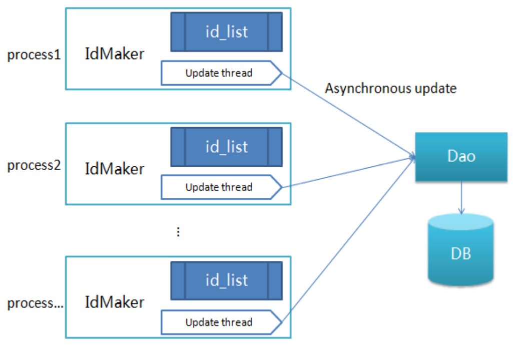


Figure1. nnotation Dao: Data Access Object DB: Database.

I id_list is used to store the list of Id blocks obtained from Dao, most of which are discontinuous between blocks and blocks

There is a unified update thread behind each use process, if the number of id_list in ID is less than 10% of each allocation depth of the configuration, the new block will be filled

E Dao is used to assign each type of manufacturer from the DB global unique ID list block, mainly responsible for the implementation of SQL (UPDATE Fid=Fid Falloc_size WHERE Fsect=xxx;)It's features like Michi Ko, each of the types used have a pool, each ID will be filled with a large piece of water level, when the water level is insufficient to update the water level

3. Unified Configuration

As shown in figure 2:

Type: used to distinguish between different business needs

Used id: inform current usage of Id

Block size: each time the size of the block, the configuration determines the height of the water level of each fill. The smaller the value of the business can be configured smaller, equivalent to the appropriate increase in the value of the configuration can be large

Initial size: there will be an initial water level item at the time of the first increase in configuration. When used for initial data entry, do not repeat the existing data

4. Interface Definition and Realization

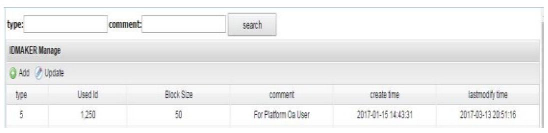
4.1 Interface Definition Code Is As Follows

Public abstract class IdMaker {public abstract long createId() throws IdException; public int createIdIntSafe() throws IdException;}

Public class IdMakerFactory { public IdMaker getIdMaker(int type);}

4.2 Realization

Pull the ID from the local buffer ID, if the number of the buffer pool is found to be 1/10 of the original, then start the update, fill. If there is no buffer in itself, then directly from the DB (as shown in figure 3):

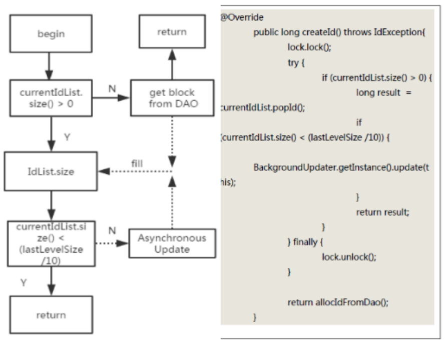

type: comment: search

IDMAKER Manage

Add Update

type	Used Id	Block Size	comment	create time	lastmodify time
5	1,250	50	For Platform Oa User	2017-01-15 14:43:31	2017-03-13 20:51:16

Figure2. configuration



```
@Override
        public long createId() throws IdException{
                lock.lock();
                try {
                        if (currentIdList.size() > 0) {
                                long result  =
currentIdList.popId();
                                if
(currentIdList.size() < (lastLevelSize /10)) {

        BackgroundUpdater.getInstance().update(t
his);
                                }
                                return result;
                        }
                } finally {
                        lock.unlock();
                }

                return allocIdFromDao();
        }
```

Figure3.

4.3 Database Table Design:

Field name	Field type	Meaning
Fsect	INT	the type of business
Fid	BIGINT	current depth
Falloc_size	INT	each allocation size
Fdesc	VARCHAR(128)	business description
Fcreate_timestamp	INT	create time
Flastmodify_timestamp	INT	last modified time

CREATE TABLE tallocid_bysect (Fsect INT UNSIGNED NOT NULL, // the type of business
Fid BIGINT UNSIGNED NOT NULL DEFAULT 1000// current depth
Falloc_size INT UNSIGNED NOT NULL DEFAULT 500 //each allocation size
Fdesc VARCHAR (128) NOT NULL DEFAULT '', //

business description
Fcreate_timestamp INT NOT NULL DEFAULT 0 // create time
Flastmodify_timestamp INT NOT NULL DEFAULT 0//last modified time
PRIMARY KEY (Fsect)

5. Conclusions

Easy to use, only need to configure different types, can support the needs of a variety of different types of online ID generated.

High availability in MySQL crash case, each machine can be configured with large enough buffer pool. Combined with the rapid MySQL downtime monitoring system, operation and maintenance can be involved in.

Less input resources, only a single MySQL can be carried on the theory of infinite number of types of business infinite number of distribution (depending on the frequency of CPU)

Production speed, no network communication pressure. Every time to get ID is an integer of 1 operation, time-consuming basic for 0

References

[1] [1]Distributed Systems: An Algorithmic Approach, Second Edition (Chapman & Hall/CRC Computer and Information Science Series)

[2] [2]Big Data: A Revolution That Will Transform How We Live, Work, and Think2013-03

[3] [3] Java programming ideas (fourth edition) paperback - June 1, 2007 Shi Er (author), Chen Haopeng (translator)