**实习内容**

--字节

First of all, our data comes from the foreign version of some apps, such as tiktok. Then there will be a big data data stream, which will contain real-time data and offline data. Our team is responsible for a piece of data service in the data stream, that is, after getting the upstream data, we will use some HSQL tasks to perform some dimensional aggregation and then give the data that is aggregated to the downstreams that related to us.

Most of the things I have been worked with is about offline data. I had create many HSQL tasks include hive2hive or hive2es. These HSQL tasks also use resource middleware, that is, some services written in java with spring, and python. I have written some services myself, such as getting some mapping relationship of some fields from es and integrate the result to HSQL tasks. Finally, advertisers can see the effect of their advertisements on the advertising board in different dimensions, such as the the rate of clicks and conversions of advertisements in Japan.

Another piece of work is about cross-domain transformation of data. Because the U.S. government restricts the outflow of U.S. data, we need to build a new computer room in the U.S. to allow data to flow out of the U.S with some limitations. Therefore, in that computer room, we need to deploy services similar to our original computer room, and there will also be some task deployment of cross-computer room data exchange.

There are also some other work such as setting task monitoring and timeout retry time. I also built some test environment to allow tasks to be tested and result to be validated before they go online.

-- 商汤

Used interface of enterprise WeChat and enterprise Didi to realize that SenseTime employees can modify their workstation information in the enterprise WeChat and we can monitor the employee's enterprise Didi order information.

Developed the company's visitor reservation application, through which external visitors can make reservations to visit our company and reserve parking spaces. I have developed functions such as visitor modification and deletion of reservations. When the visitor reach the first floor of the company, they will scan the code to verify their identity. To be specific, they will receive the verification code stored in redis, which is implemented using the Alibaba Cloud service interface. At the same time, in order to ensure the concurrency safety when reserving a parking space, for example, in order to ensure that two people fail to reserve the remaining one parking space at the same time, some lock mechanisms are used to achieve it.

**项目**

1. **登录注册功能:选用密码MD5加密保证密码的安全性。选用kaptcha生成验证码，使用邮件完成注册，Redis优 化验证码保存的方式解决分布式session问题**。

**Login and registration function: Use MD5 encryption to ensure the security of password and Kaptcha to generate the verification code. Redis optimizes the way to save the verification code and solve the distributed session problem.**

**MD5**: After the password registered by the user, it will be appended with a random string. Then it will be encrypted by MD5 function, and finally stored in the database as the password, and the user's salt will be set as that randomly generated string. In the stage of verification, the password entered by the user combined with the user's salt that encrypted with MD5 will be compared with that in the database. If they are the same, user can login in successfully.

**Why MD5 is irreversible**: it is a hash function and uses the hash algorithm. Some information of the original text is lost in the calculation process

**Kaptcha**: ready-made verification code tool

Import jar package from Maven, and then we write kaptcha configuration class, then we can generate random characters and pictures

1. **发布帖子与敏感词过滤:发帖构建Trie数据结构，实现对发表帖子评论的敏感词过滤。**

**Post and sensitive word filtering: Post and build trie data structure to filter the sensitive words of post comments.**

Algorithm: Create a sensitive word multi-branches tree, record the first character (without repetition) of each sensitive word into the child node of the root node, and mark the last character of the sensitive word. When checking the sensitive word, one pointer points to the position of the sensitive word character, two pointers point to the string (the text of the sensitive word needs to be filtered), one points to the head of the sensitive word (instead of moving one bit back), and one points to the character to be matched

1. **点赞关注功能:利用Redis实现对帖子、评论点赞功能来提升性能，同时利用Redis的zset并结合Redis实现对其他用户关注的功能。**

**Like and focus function: Utilized redis to implement the feature of liking posts and comments to improve application performance. At the same time, used Zset of redis and redis to implement the feature of following other users.**

1. **异步消息系统:鉴于点赞关注后的系统通知实时性不需要特别高，因此选用了kafka实现异步的发送系统通知。**

**Asynchronous message system: since the real-time performance of system notification after liking does not need to be high, so I use Kafka to send system notification asynchronously.**

过程：发消息给消费者，消费者添加数据库

Send a message to the consumer and the consumer adds the data to database

为什么用MQ：

1. 在触发事件之后，此方法就可继续向下执行，消费者会新建一个线程进行操作，可以提高性能，处理效率高（**异步**）

After triggering the event, this method can continue to execute downward. The consumer will create a new thread to operate, which can improve performance and ensure high processing efficiency since it is **asynchronous**

2.当此贴是个热帖时：评论的人非常高，可以用消息队列攒着慢慢处理（**削峰**）

When this post is a hot post, which implies the number of comments is very high, so we can save it in the message queue and deal with it slowly, which is the concept of **peak shaving**

1. **搜索功能:选用ElasticSearch实现对帖子的搜索功能，以及结果的高亮显示。 **

**Search function: Utilized elasticsearch to realize the search function of Posts and highlight the results.**

为什么用es

Because it is a distributed real-time file storage, each field can be saved to the index so that it can be retrieved later. And the search engine can be extended to hundreds of servers to handle Pb (unit of memory) level data and highly integrated services. Also, applications can interact through simple restful API, clients in various languages and command line

为什么不用数据库搜索：

The database can also provide search, but it finds data through specific column, but what users enter when they search may not contained in a column, the input may contain multiple columns. The database cannot achieve full-text matching through word segmentation, but the es search engine can

倒排索引Inverted index 分片slice 节点node 副本replication

1. **定时热帖排行:鉴于热帖排行功能需要每隔段时间从数据库查询最热门的帖子显示，quartz实现定时任务的参数是存储到数据库中的，不会出现重复代码的问题，因此利用Spring Quartz完成热门帖子的分数计算模块。**

**Timing hot post ranking: since the hot post ranking function needs to query the most popular posts from the database at regular intervals, and the parameters of quartz to realize the scheduled task are stored in the database, there will not have problem of duplicate code, I use spring Quartz to complete the score calculation module of popular posts.**

1. **多级缓存:考虑到热门帖子数量不大，直接存储在服务器本地上并不会给服务器带来过重的存储负担，而且 热门帖子的改变并不是很频繁，适合缓存存储，因此选用Caffeine本地缓存存储。利用本地缓存+Redis+DB的 形式避免缓存雪崩，提高系统的可用性。**

**Multi level cache: considering the small number of popular posts, directly storing them on the server will not bring excessive storage burden to the server, and the changes of popular posts are not very frequent, which is suitable for cache storage. Therefore, caffeine local cache storage is selected. Use the form of local cache + redis + DB to avoid cache avalanche and improve the availability of the system.**

**Redis**

持久化Persistence 事务transaction 原子性Atomicity

主从复制Master-slave replication 丰富rich 宕机Downtime

硬盘hard disk 缓存 cache 内存memory

高并发High concurrency 复用模型Reuse model 动态扩容 Dynamic capacity expansion

过期策略Expiration Policies 淘汰策略Elimination strategy

悲观锁Pessimistic lock 乐观锁Optimistic lock

缓存穿透Cache penetration 缓存击穿Cache breakdown 缓存雪崩Cache avalanche

**Kafka**

**高吞吐量**High throughput **TB级的数据**terabyte data

**消息持久化**：将消息传入硬盘里，并且通过硬盘顺序读取提高读取速度

message persistence: send messages to the hard disk and read them sequentially through the hard disk to improve the reading speed

**高扩展**：集群不够了就加一台服务器

High expansion: add a server when the cluster is not enough

**高可靠**：通过分布式集群部署实现，有副本机制，每个partition的数据都会同步到其他机器上，形成自己的副本

High reliability: through distributed cluster deployment, there is a copy mechanism. The data of each partition will be synchronized to other machines to form its own copy

**发布订阅模式:**Publish subscribe mode

**基础知识**

**Java**

子类subclass 父类parent class 修饰decorate

封装Encapsulation 继承inheritance 多态polymorphism 向上转型Upward transformation

**JVM**

**新生代**young generation **老年代**old generation

1. 标记清除法Mark and sweep 2.标记整理法Concurrent mark sweep (CMS) garbage collection

3.分代收集法Generational Collection

a.Serial garbage collection

This algorithm uses mark-copy for the Young Generation and mark-sweep-compact for the Old Generation. It works on a single thread. When executing, it freezes all other threads until garbage collection operations have concluded.

b.Parallel garbage collection

Simimar to serial GC, It uses mark-copy in the Young Generation and mark-sweep-compact in the Old Generation. Multiple concurrent threads are used for marking and copying / compacting phases.

**内存泄漏**memory leak 没关连接

a type of [resource leak](https://en.wikipedia.org/wiki/Resource_leak" \o "Resource leak) that occurs when a [computer program](https://en.wikipedia.org/wiki/Computer_program" \o "Computer program) incorrectly manages [memory allocations](https://en.wikipedia.org/wiki/Memory_allocation" \o "Memory allocation) in a way that memory which is no longer needed is not released

**内存溢出**memory overflow 一直创建object

means that the program does not have enough memory space for it to use when applying for memory, and it appears out of the memory; For example, an integer is applied, but the number that has long to save is the memory overflow.

**栈溢出**StackOverflowError

无限递归 infinite recursion

**反射**reflection

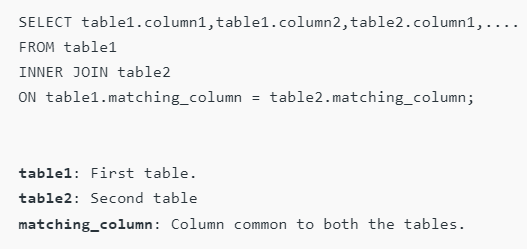
**线程同步的实现/机制**Implementation / mechanism of thread synchronization

**单例模式** singleton pattern 工厂模式Factory pattern 代理模式 agency pattern

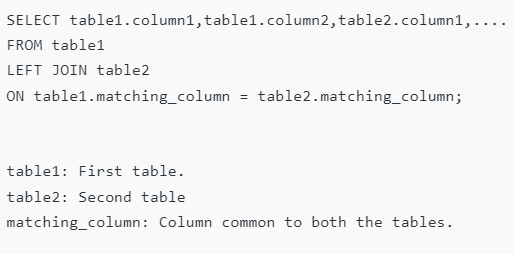
观察者模式 observer pattern

**数据库**

**INNER JOIN**: selects all rows from both the tables as long as the condition satisfies.



**LEFT JOIN**:returns all the rows of the table on the left side of the join and matching rows for the table on the right side of join. The rows for which there is no matching row on right side, the result-set will contain null. LEFT JOIN



**RIGHT JOIN**: returns all the rows of the table on the right side of the join and matching rows for the table on the left side of join. The rows for which there is no matching row on left side, the result-set will contain null.

**FULL JOIN**: creates the result-set by combining result of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both the tables. The rows for which there is no matching, the result-set will contain NULL values.

**关系型数据库**relational database

**数据库事务Database transaction**: a logical unit in the execution of DBMS, which is composed of a sequence of database operations

**目的Purpose**: 1 It provides a method to restore the operation sequence from failure to normality, while allowing the database to maintain consistency under abnormal conditions

2. Provide an isolation when multiple applications access the database concurrently, so that the operation of the program will not interfere with other operations

**4个特征 features**:

1 Atomicity (all operations are either successful or cancelled) Implemented by redo log

2. Consistency (database integrity is not destroyed before and after the transaction)Implemented by rollback log

3. Isolation (multiple transactions are executed concurrently, and the execution of one transaction does not affect other things) Execution of transaction, that is, it is invisible to other transactions before the transaction is committed) Realized by database lock

4. Persistence (after the transaction is committed, its result is permanent) is realized by redo log

**隔离等级Isolation level**:

1 Read uncommitted (one transaction can read the execution results of other uncommitted transactions)

2. Non-repeatable reading (one transaction can only read the content submitted by other transactions)

3. Repeatable reading (MySQL level) ensures that when a transaction repeatedly reads data, it can read repeatedly The value of the obtained data row is unchanged

4. Serialization (transactions are queued one by one, and other transactions can be executed only after the transaction is committed The execution efficiency is very poor, which may lead to lock competition

**3个要阻止的现象 phenomena to be prevented**:

**脏读Dirty read**: a transaction reads data written by another transaction but has not yet committed

**不可重复读unrepeatable read**: the contents of the two queries are different because other transactions have been modified during the period

**幻读phantom**: when the transaction is re queried according to the previous conditions, the number of result sets is inconsistent, and then the extra row is Phantom line, that is, the data inserted by other transactions is read

**MySQL Solves phantom**:

**1 Mvcc** (snapshot read / current read): save a snapshot of historical data, and the modified data of other transactions is invisible to the current transaction.

Snapshot: select \* from where

Current read: update

**2. Next key lock:** lock the gap between the current data row and the upper and lower rows,

Check which data lines are locked to ensure that the data read within this range is consistent, but it does not Not exactly equal to the isolation level of serialization

**操作系统**

**进程process**: The execution of a program that allows you to perform the appropriate actions specified in a program. It can be defined as an execution unit where a program runs.

**线程thread** :an execution unit that is part of a process and  a unit of execution in concurrent programming

不同点：

1.Process means a program is in execution, whereas thread means a segment of a process.

2.A Process is not Lightweight, but Threads are.

3.A Process takes more time than threads on creation, termination and context switching

4.A Process is mostly isolated, whereas Threads share memory and data.

**协程Synergetic proces**s

描述符descriptor