4-13 | 缓存的过期时间设计

缓存雪崩：

大量的 key 同一时间段过期，导致请求打入数据库层，将 DB 压垮。

用户中台的单个用户查询操作，要记得去设置过期时间。

```Java
public UserDTO getByUserId(Long userId) {
    if (userId == null) {
        return null;
    }
    String key =
userProviderCacheKeyBuilder.buildUserInfoKey(userId);
    UserDTO userDTO = redisTemplate.opsForValue().get(key);
    if (userDTO != null) {
        return userDTO;
    }
    userDTO =
ConvertBeanUtils.convert(userMapper.selectById(userId),
UserDTO.class);
    if (userDTO != null) {
        redisTemplate.opsForValue().set(key, userDTO,30,
TimeUnit.MINUTES);
    }
    return userDTO;
}
```

批量 key 设置过期时间，建议批量的 key 的过期时间不要设置相同，随机性多一些。

```Java
public Map<Long, UserDTO> batchQueryUserInfo(List<Long>
userIdList) {
    if (CollectionUtils.isEmpty(userIdList)) {
        return Maps.newHashMap();
    }
    userIdList = userIdList.stream().filter(id -> id >
10000).collect(Collectors.toList());
```

```
    if (CollectionUtils.isEmpty(userIdList)) {
        return Maps.newHashMap();
    }
    // redis
    List<String> keyList = new ArrayList<>();
    userIdList.forEach(userId -> {

keyList.add(userProviderCacheKeyBuilder.buildUserInfoKey(userId));
    });
    List<UserDTO> userDTOList =
redisTemplate.opsForValue().multiGet(keyList).stream().filter(x ->
x != null).collect(Collectors.toList());
    if (!CollectionUtils.isEmpty(userDTOList) &&
userDTOList.size() == userIdList.size()) {
        return
userDTOList.stream().collect(Collectors.toMap(UserDTO::getUserId,
x -> x));
    }
    List<Long> userIdInCacheList =
userDTOList.stream().map(UserDTO::getUserId).collect(Collectors.to
List());
    List<Long> userIdNotInCacheList = userIdList.stream().filter(x
-> !userIdInCacheList.contains(x)).collect(Collectors.toList());
    // 多线程查询 替换了union all
    Map<Long, List<Long>> userIdMap =
userIdNotInCacheList.stream().collect(Collectors.groupingBy(userId
-> userId % 100));
    List<UserDTO> dbQueryResult = new CopyOnWriteArrayList<>();
    userIdMap.values().parallelStream().forEach(queryUserIdList ->
{

dbQueryResult.addAll(ConvertBeanUtils.convertList(userMapper.selec
tBatchIds(queryUserIdList), UserDTO.class));
    });
    if (!CollectionUtils.isEmpty(dbQueryResult)) {
        Map<String, UserDTO> saveCacheMap =
dbQueryResult.stream().collect(Collectors.toMap(userDto ->
userProviderCacheKeyBuilder.buildUserInfoKey(userDto.getUserId()),
x -> x));
        redisTemplate.opsForValue().multiSet(saveCacheMap);
        //对命令执行批量过期设置操作
        redisTemplate.executePipelined(new
SessionCallback<Object>() {
            @Override
            public <K, V> Object execute(RedisOperations<K, V>
operations) throws DataAccessException {
                for (String redisKey : saveCacheMap.keySet()) {
                    operations.expire((K) redisKey,
createRandomTime(), TimeUnit.SECONDS);
                }
                return null;
            }
        });
        userDTOList.addAll(dbQueryResult);
    }
    return
userDTOList.stream().collect(Collectors.toMap(UserDTO::getUserId,
x -> x));
}

/**
 * 创建随机的过期时间 用于redis设置key过期
 *
 * @return
 */
private int createRandomTime() {
    int randomNumSecond =
ThreadLocalRandom.current().nextInt(100);
    return randomNumSecond + 30 * 60;
}
```