1. 概念

CopyOnWriteArrayList为写时复制容器。对于一个共享的CopyOnWriteArrayList，一个线程要向其中添加一个数据的时候，会首先复制一个完整的副本，把数据添加到副本中，然后再将原来的CopyOnWriteArrayList的引用指向该副本。在写入数据的过程中，并不影响读，读数据的线程仍然在原CopyOnWriteArrayList中读取。

2. 适用场景

读多写少的并发场景。

思想：读写分离

3. 原理

在原有1，2，3数据的容器上添加一个元素4。

首先创建原容器的副本



图 1

然后，把数据添加到副本容器中



图 2

最后，把原容器的引用指向副本容器



图 3

问题：并发写不还是会有问题吗（创建多个副本）？------>源码

4. 源码

public boolean add(E e) {

final ReentrantLock lock = this.lock;

lock.lock();

try {

Object[] elements = getArray();

int len = elements.length;

Object[] newElements = Arrays.copyOf(elements, len + 1);

newElements[len] = e;

setArray(newElements);

return true;

} finally {

lock.unlock();

}

}

观看源码可以看到，

首先，在写入的时候会锁住避免创建多个副本；

然后，进行容器复制，新容器多一个长度；

然后，写入数据；

然后，将引用指向新的容器；

最后，释放锁。

5. 优点

（1）在读多写少的高并发场景下可以提供更好的性能。

（2）读取（迭代）的时候不需要加锁。

5. 缺点

（1）每次写入都要复制容器，需要一定的开销。