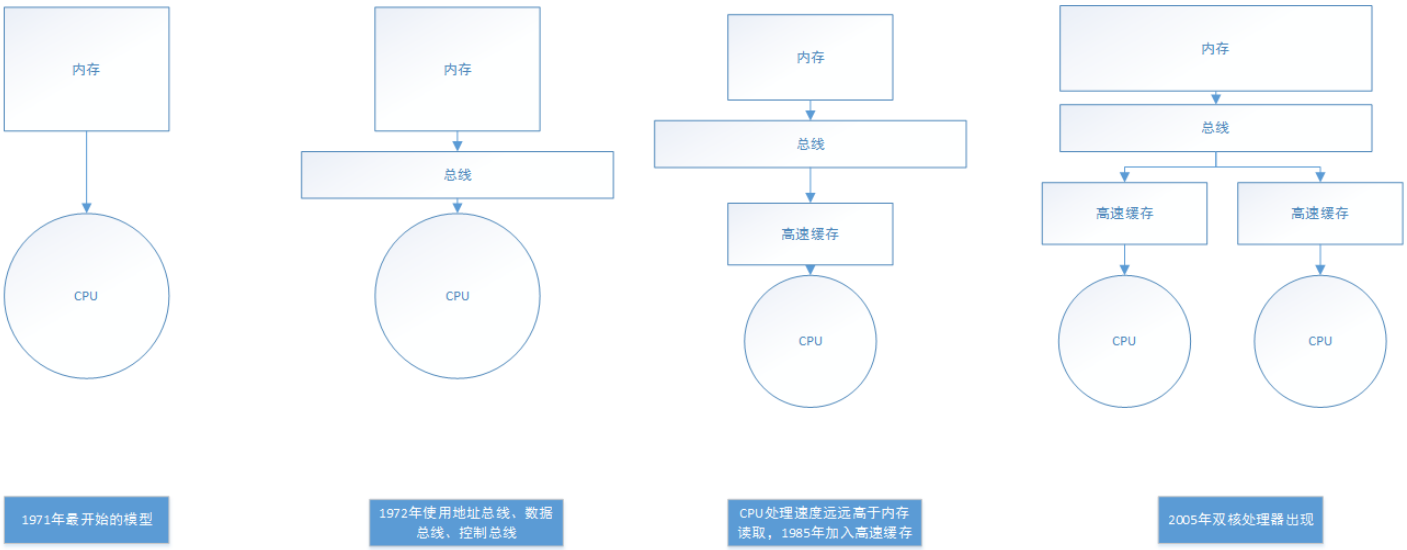
多线程基础-重排序、原子性、可见性

2019-05-30 15:02:14 3 0 0

Intel CPU发展史



总结：先设计出来单核CPU，然后加入了高速缓存，最后发展了多核CPU。

CPU发展了这么多年，你了解CPU吗？

1.CPU不会按照你编写的代码顺序执行它的指令

为什么我的程序能获得正确的结果？

某程序员的代码如下：

```
1.  int a = 1;
2.  int b = 1;
3.  a = a + 1;
4.  b = b + 1;
5.  system.out.print(a);
6.  system.out.print(b);
```

CPU可能执行的”代码”如下：

```
1.  int a = 1;
2.  a = a + 1;
3.  int b = 1;
4.  b = b + 1;
5.  system.out.print(a);
6.  system.out.print(b);
```

在保证你程序结果正确的情况下，确实允许代码执行的时候或者编译的时候对某程序猿代码进行 **重排序**

处理器或者编译器为啥要这么干？

CPU在执行第二行的时候a已经在寄存器中了，CPU担心寄存器加载b的时候会把a踢出去，一旦踢出去下次还要加载进来，不如直接计算完毕。

另外一个程序猿：我不答应-指令重排序。

```
/**
 * 战争结束后勇士与公主结婚
 * 两个线程同时执行，可造成战士和公主结婚
 */
public class WarStory {
    private String name ="战士";
    private boolean warflag = false; //战争结束标识
    //一个线程执行fighting
    public void fighting(){
        name = "勇士";
        warflag = true;
    }
    //一个线程执行married
    public void married(){
        if(warflag){
            //默认认为name就等于< 勇士 > 了。
            //然后执行< 勇士 >和 < 公主 >结婚代码。
        }
    }
}
```

在多线程下指令重排序会出现了不正确的结果。

CPU厂商：不答应指令重排序我们也要指令重排序，这是趋势。我们CPU厂商研究决定，采用折中方式解决这个冲突。

加入指令重排序的处理器在市场大卖，CPU厂商也要存活。我们可以多加几条cpu指令，让高级程序员自己根据业务告诉cpu是否要进行指令重排序，在CPU厂商这里不统一搞了。

2.来自同一段代码的多个线程只有程序员明白、CPU并不知情。

一个线程的时间片快到了，处理器会将线程现场数据压入数据段寄存器，下次线程获取时间片后，现场恢复。

为什么我的程序在多线程下出现了线程安全问题？

```java
/**
 * 自增操作
 */
public class IncData {
    private int i = 0;

    //第一种写法
    //多线程情况下造成两个线程本应该执行两次相加操作结果却和单线程下执行一次结果相同
    public void incremte(){
        int j = i;//时间片快到了处理器将赋值好的j压入堆栈
        i = j +1;//时间片到了处理器将堆栈数据j恢复到寄存器中-现场恢复
    }

    //第二种写法
    public void inc(){
        i++;
    }
}
```

程序员：我觉得这种CPU就是个傻子，CPU厂商你们能不能造的智能点、毕竟来自同一段代码！

CPU厂商：emmmm…. CPU也许不知道程序本身是被单线程调用的还是多线程调用才是问题根源吧。在单线程下不是很快嘛。再提供一个折中方案，我们给几个指令，程序员自己标记是否需要把该类设计成线程安全的吧。

3.CPU厂商再次坑人

2005年双核CPU问世。

两个CPU通信能力很差

```java
/**
 * 不安全的类
 */
public class UnsafeClass {

    private long count = 0;
    //一直使用cpu1执行a线程，a线程只调用get方法
    public long getCount(){return count;};
    //一直使用cpu2执行b线程，b线程只调用set方法
    public void setCount(long count){this.count = count;}
    //在b线程中set了10次，a线程才get出来。

}
```

这种通讯能力用啥实现的？

CPU不会通讯，主要是每个CPU对应的高速缓存不能被其他CPU访问导致的。CPU与CPU之间通讯只能将数据放在内存中。

缓存状态

状态	描述
M	修改状态,一个非失效状态的缓存被修改
E	独享状态，该高速缓存与内存值一致、其他高速缓存与内存中的值不一致
S	共享状态,至少两个高速缓存存储的数据与内存中存储的一致
I	失效状态,该高速缓存与内存中存储的值不一致

MESI协议

`	M	E	S	I
M	x	x	x	o
E	x	x	x	o
S	x	x	o	o
I	o	o	o	o

1、异步方式通知另一个缓存改变状态
2、共享状态的两个缓存不能同时被两个CPU修改。

总结：异步会导致状态修改后，完全不必等到别的缓存状态改变，只要自己发出通知就可以做下一指令。另一个缓存收到通知后会将通知存放到队列中，cpu不会遍历该队列。