第十一章、性能与可伸缩性

11.3 引入线程的开销

对于为了提升性能而引入的线程来说，并行带来的性能提升必须超过并发导致的开销。

1、上下文切换

如果可运行线程大于cpu数量，操作系统会将正在运行的线程进行调度，这将导致一次上下文切换：保存当前运行线程的执行上下文，将新调度进来的线程的执行上下文设置为当前上下文。一般消耗几微秒

2、内存同步

某个线程中的同步可能会影响其他线程的性能。同步会增加共享内存总线上的通信量，因为总线带宽有限，而且所有处理器都共享这条总线，所以如果多个线程竞争同步带宽，性能会下降

3、阻塞

线程阻塞被挂起过程中包含两次额外的上下文切换，被阻塞的线程在执行时间片还未用完就交换出去，随后获得锁或者资源可用时再次被切换回来。（由于锁竞争而导致阻塞时，线程在持有锁时存在一定的开销：当释放锁时必须告诉操作系统恢复运行阻塞的线程）

11.4 减少锁的竞争

三种方式减低锁的竞争程度：

减少锁的持有时间

降低锁的请求频率

使用带有协调机制的独占锁，这些锁允许更好的并发性

1.缩小锁的范围：在真正需要加锁的地方加锁

2.减小锁的粒度：锁分解，采用相互独立的锁保持自己的状态

3.锁分段：对锁分解的扩展，将一组独立对象的锁进行分解。concurrentHashMap利用的就是锁分段技术，包含16个锁的数组。

4.替代独占锁的方法，比如readWriteLock，静态计数器，序列发生器，或者对列表数据结构中头结点的引用

5.检测cpu利用率：vmstat，mpstat命令，或者Windows的perfmon.exe

6.减少上下文的开销：例如对于记录日志，可以将日志进行异步打印