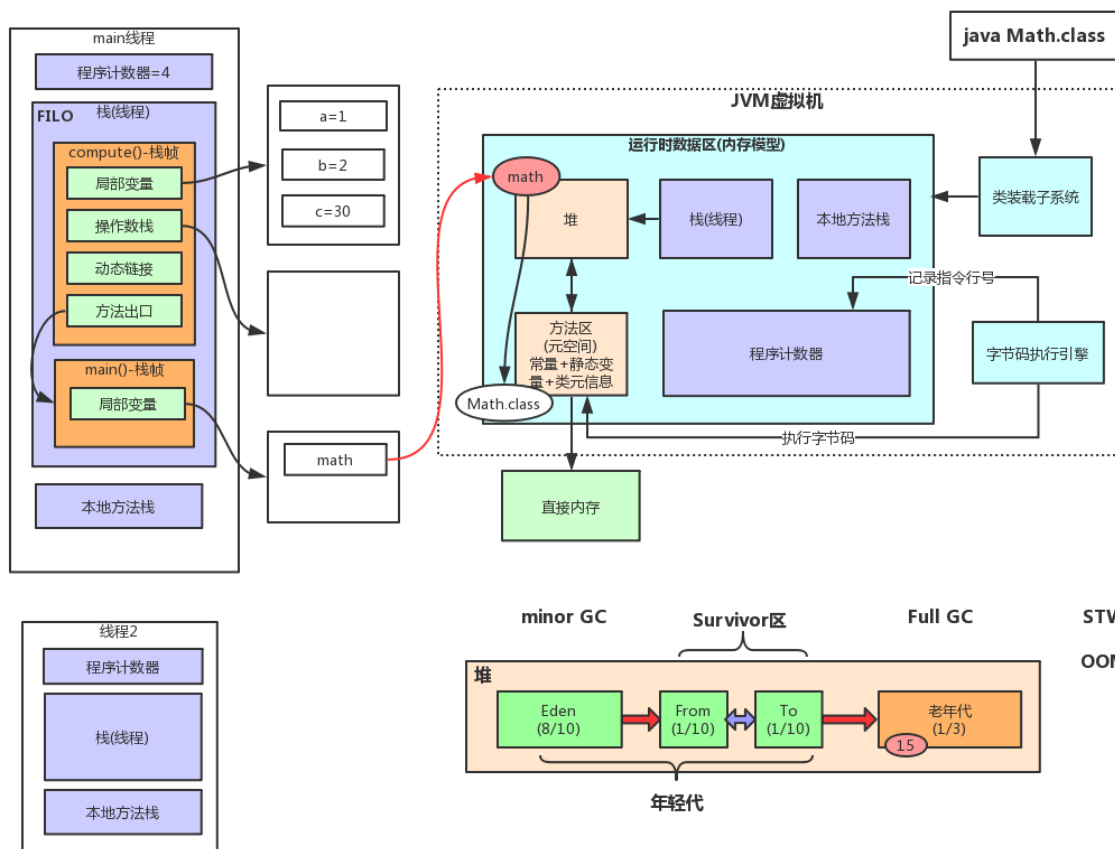
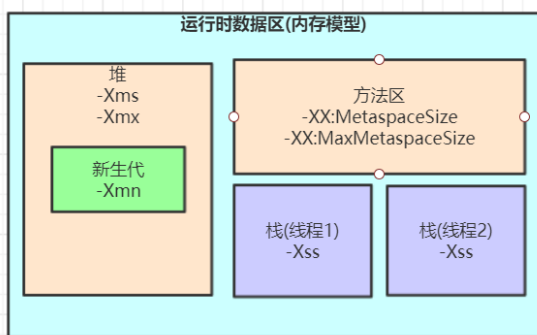


一、JVM整体结构及内存模型



二、JVM内存参数设置



Spring Boot程序的JVM参数设置格式(Tomcat启动直接加在bin目录下catalina.sh文件里):

```
1 java -Xms2048M -Xmx2048M -Xmn1024M -Xss512K -XX:MetaspaceSize=256M -XX:MaxMetaspaceSize=256M -jar microservice-eureka-server.jar
```

StackOverflowError示例:

```
1 // JVM设置 -Xss128k(默认1M)
2 public class StackOverflowTest {
3
4     static int count = 0;
5
6     static void redo() {
7         count++;
8         redo();
9     }
10 }
```

```

10
11 public static void main(String[] args) {
12     try {
13         redo();
14     } catch (Throwable t) {
15         t.printStackTrace();
16         System.out.println(count);
17     }
18 }
19 }
20
21 运行结果:
22 java.lang.StackOverflowError
23   at com.tuling.jvm.StackOverflowTest.redo(StackOverflowTest.java:12)
24   at com.tuling.jvm.StackOverflowTest.redo(StackOverflowTest.java:13)
25   at com.tuling.jvm.StackOverflowTest.redo(StackOverflowTest.java:13)
26   .....

```

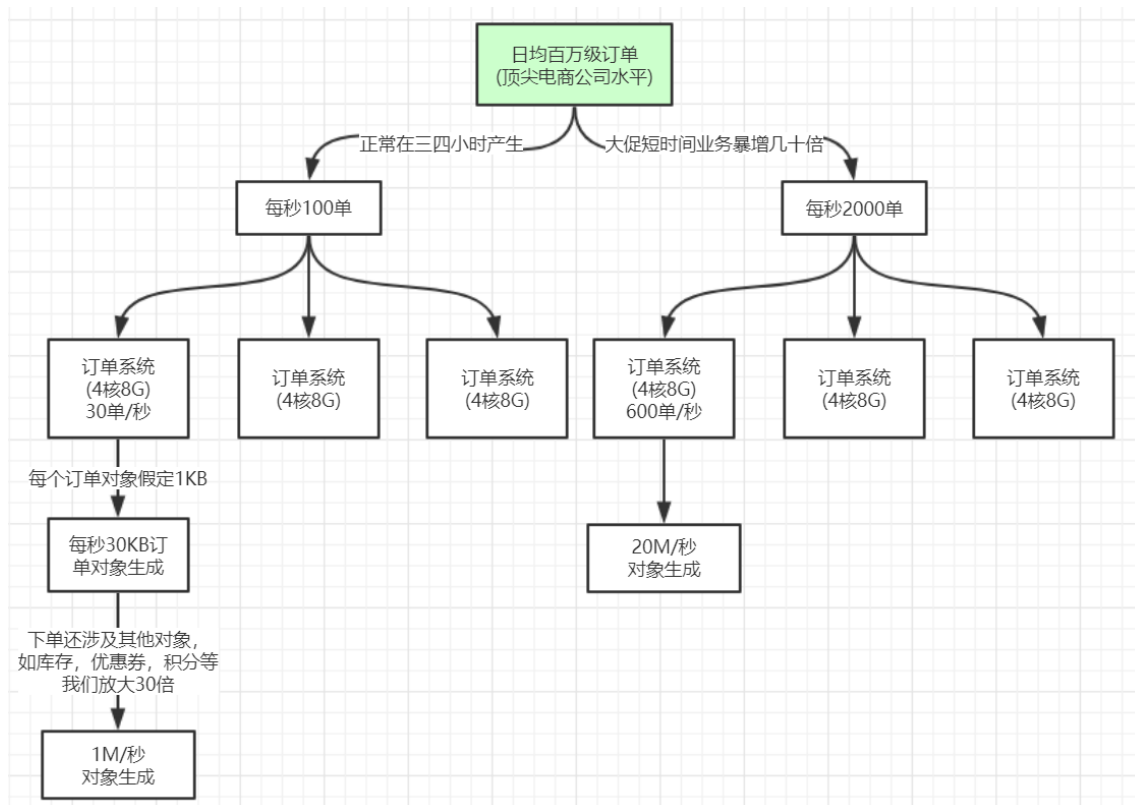
结论:

-Xss设置越小count值越小, 说明一个线程栈里能分配的栈帧就越少, 但是对JVM整体来说能开启的线程数会更多

JVM内存参数大小该如何设置?

JVM参数大小设置并没有固定标准, 需要根据实际项目情况分析, 给大家举个例子

日均百万级订单交易系统如何设置JVM参数



一天百万级订单这个绝对是现在顶尖电商公司交易量级, 对于这种量级的系统我们该如何设置JVM参数了?

我们可以试着估算下, 其实日均百万订单主要也就是集中在当日的几个小时生成的, 我们假设是三小时, 也就是每秒大概生成100单左右。

这种系统我们一般至少要三四台机器去支撑, 假设我们部署了四台机器, 也就是每台每秒钟大概处理完成25单左右, 往上毛估每秒处理30单吧。

也就是每秒大概有30个订单对象在堆空间的新生代内生成，一个订单对象的大小跟里面的字段多少及类型有关，比如int类型的订单id和用户id等字段，double类型的订单金额等，int类型占用4字节，double类型占用8字节，粗略估计下一个订单对象大概**1KB**左右，也就是说每秒会有**30KB**的订单对象分配在新生代内。

真实的订单交易系统肯定还有大量的其他业务对象，比如购物车、优惠券、积分、用户信息、物流信息等等，实际每秒分配在新生代内的对象大小应该要再**扩大几十倍**，我们假设30倍，也就是每秒订单系统会往新生代内分配近**1M**的对象数据，这些数据一般在订单提交完的操作做完之后基本都会成为垃圾对象。

我们一般线上服务器的配置用得较多的就是**双核4G或4核8G**，如果我们用双核4G的机器，因为服务器操作系统包括一些后台服务本身可能就要占用1G多内存，也就是说给JVM进程最多分配2G多点内存，刨开给方法区和虚拟机栈分配的内存，那么堆内存可能也就能分配到1G多点，对应的新生代内存最后可能就几百M，那么意味着没过**几百秒**新生代就会被垃圾对象撑满而出发minor gc，这么频繁的gc对系统的性能还是有一定影响的。

如果我们选择4核8G的服务器，就可以给JVM进程分配四五个G的内存空间，那么堆内存可以分到三四个G左右，于是可以给新生代至少分配2G，这样算下差不多需要半小时到一小时才能把新生代放满触发minor gc，这就大大降低了minor gc的频率，所以一般我们线上服务器用得较多的还是4核8G的服务器配置。

如果系统业务量继续增长那么可以水平扩容增加更多的机器，比如五台甚至十台机器，这样每台机器的JVM处理请求可以保证在合适范围，不至于压力过大导致大量的gc。

有的同学可能有疑问说双核4G的服务器好像也够用啊，无非就是minor gc频率稍微高一点呀，不是说minor gc对系统的影响不是特别大吗，我成本有限，只能用这样的服务器啊。

其实如果系统业务量比较平稳也能凑合用，如果经常业务量可能有个几倍甚至几十倍的增长，比如时不时的搞个促销秒杀活动什么的，那我们思考下会不会有什么问题。

假设业务量暴增几十倍，在不增加机器的前提下，整个系统每秒要生成几千个订单，之前每秒往新生代里分配的1M对象数据可能增长到**几十M**，而且因为系统压力骤增，一个订单的生成不一定能在1秒内完成，可能要几秒甚至几十秒，那么就有很多对象会在新生代里存活几十秒之后才会变为垃圾对象，如果新生代只分配了几百M，意味着一二十秒就会触发一次minor gc，那么很有可能部分对象就会被挪到老年代，这些对象到了老年代后因为对应的业务操作执行完毕，马上又变为了垃圾对象，随着系统不断运行，被挪到老年代的对象会越来越多，最终可能又会导致full gc，full gc对系统的性能影响还是比较大的。

如果我们用的是4核8G的服务器，新生代分配到2G以上的水平，那么至少也要几百秒才会放满新生代触发minor gc，那些在新生代即便存活几十秒的对象在minor gc触发的时候大部分已经变为垃圾对象了，都可以被及时回收，基本不会被挪到老年代，这样可以大大减少老年代的full gc次数。

