链滴

java 性能优化之 for 循环

作者：yk

原文链接：https://ld246.com/article/1502952625036

来源网站：链滴

完成同样的功能，用不同的代码来实现，性能上可能会有比较大的差别，所以对于一些性能敏感的模来说，对代码进行一定的优化还是很有必要的。今天就来说一下Java代码优化的事情，今天主要聊一对于for（while等同理）循环的优化。

作为三大结构之一的循环，在我们编写代码的时候会经常用到。循环结构让我们操作数组、集合和其一些有规律的事物变得更加的方便，但是如果我们在实际开发当中运用不合理，可能会给程序的性能来很大的影响。所以我们还是需要掌握一些技巧来优化我们的代码的。

嵌套循环

```
1. stratTime = System.nanoTime();
2. for (int i = 0; i < 10000000; i++) {
3.     for (int j = 0; j < 10; j++) {
4.
5.     }
6. }
7. endTime = System.nanoTime();
8. System.out.println("外大内小耗时："+ (endTime - stratTime));
```

应改为：

```
1. stratTime = System.nanoTime();
2. for (int i = 0; i <10 ; i++) {
3.     for (int j = 0; j < 10000000; j++) {
4.
5.     }
6. }
7. endTime = System.nanoTime();
8. System.out.println("外小内大耗时："+(endTime - stratTime));
```

两者耗时对比：

```
1. 外大内小耗时：200192114
2. 外小内大耗时：97995997
```

由以上对比可知，优化后性能提升了一倍，嵌套循环应该遵循“外小内大”的原则，这就好比你复制多个小文件和复制几个大文件的区别。

提取与循环无关的表达式

```
1. stratTime = System.nanoTime();
2. for (int i = 0; i < 10000000; i++) {
3.     i=i    ab;
4. }
5. endTime = System.nanoTime();
```

```
6. System.out.println("未提取耗时："+(endTime - stratTime));
```

应改为：

```
1. stratTime = System.nanoTime();
2. c = a*b;
3. for (int i = 0; i < 10000000; i++) {
4.     i=i*c;
5. }
6. endTime = System.nanoTime();
7. System.out.println("已提取耗时："+(endTime - stratTime));
```

两者耗时对比：

```
1. 未提取耗时：45973050
2. 已提取耗时：1955
```

代码中a+b与我们的循环无关，所以应该把它放到外面，避免重复计算，可以看出，优化后性能提升好几个数量级，这些是不容忽视的。

消除循环终止判断时的方法调用

```
1. stratTime = System.nanoTime();
2. for (int i = 0; i < list.size(); i++) {
3.
4. }
5. endTime = System.nanoTime();
6. System.out.println("未优化list耗时："+(endTime - stratTime));
```

应改为：

```
1. stratTime = System.nanoTime();
2. int size = list.size();
3. for (int i = 0; i < size; i++) {
4.
5. }
6. endTime = System.nanoTime();
7. System.out.println("优化list耗时："+(endTime - stratTime));
```

两者耗时对比：

```
1. 未优化list耗时：27375
2. 优化list耗时：2444
```

list.size()每次循环都会被执行一次，这无疑会影响程序的性能，所以应该将其放到循环外面，用一个

量来代替，优化前后的对比也很明显。

异常捕获

```
1. stratTime = System.nanoTime();
2. for (int i = 0; i < 10000000; i++) {
3.     try {
4.     } catch (Exception e) {
5.     }
6. }
7. endTime = System.nanoTime();
8. System.out.println("在内部捕获异常耗时："+(endTime - stratTime));
```

应改为：

```
1. stratTime = System.nanoTime();
2. try {
3.     for (int i = 0; i < 10000000; i++) {
4.     }
5. } catch (Exception e) {
6.
7. }
8. endTime = System.nanoTime();
9. System.out.println("在外部捕获异常耗时："+(endTime - stratTime));
```

两者耗时对比：

```
1. 在内部捕获异常耗时：12150142
2. 在外部捕获异常耗时：1955
```

大家都知道，捕获异常是很耗资源的，所以不要讲try catch放到循环内部，优化后同样有好几个数量的提升。

性能优化的内容有很多，代码优化只是其中一小部分，我们在日常开发中应养成良好的编码习惯。接来会跟大家探讨更多关于性能优化的内容，希望大家积极交流指导。