

02 | 内存池：如何提升内存分配的效率？

2020-04-27 陶辉

系统性能调优必知必会

[进入课程 >](#)



讲述：陶辉

时长 13:30 大小 10.82M



你好，我是陶辉。

上一讲我们提到，高频地命中 CPU 缓存可以提升性能。这一讲我们把关注点从 CPU 转移到内存，看看如何提升内存分配的效率。

或许有同学会认为，我又不写底层框架，内存分配也依赖虚拟机，并不需要应用开发者了解。如果你也这么认为，我们不妨看看这个例子：在 Linux 系统中，用 Xmx 设置 JVM 的最大堆内存为 8GB，但在近百个并发线程下，观察到 Java 进程占用了 14GB 的内存。☆
什么会这样呢？

这是因为，绝大部分高级语言都是用 C 语言编写的，包括 Java，申请内存必须经过 C 库，而 C 库通过预分配更大的空间作为内存池，来加快后续申请内存的速度。这样，预分配的 6GB 的 C 库内存池就与 JVM 中预分配的 8G 内存池叠加在一起，造成了 Java 进程的内存占用超出了预期。

掌握内存池的特性，既可以避免写程序时内存占用过大，导致服务器性能下降或者进程 OOM（Out Of Memory，内存溢出）被系统杀死，还可以加快内存分配的速度。在系统空闲时申请内存花费不了多少时间，但是对于分布式环境下繁忙的多线程服务，获取内存的时间会上升几十倍。

另一方面，内存池是非常底层的技术，当我们理解它后，可以更换适合应用场景的内存池。在多种编程语言共存的分布式系统中，内存池有很广泛的应用，优化内存池带来的任何微小的性能提升，都将被分布式集群巨大的主机规模放大，从而带来整体上非常可观的收益。

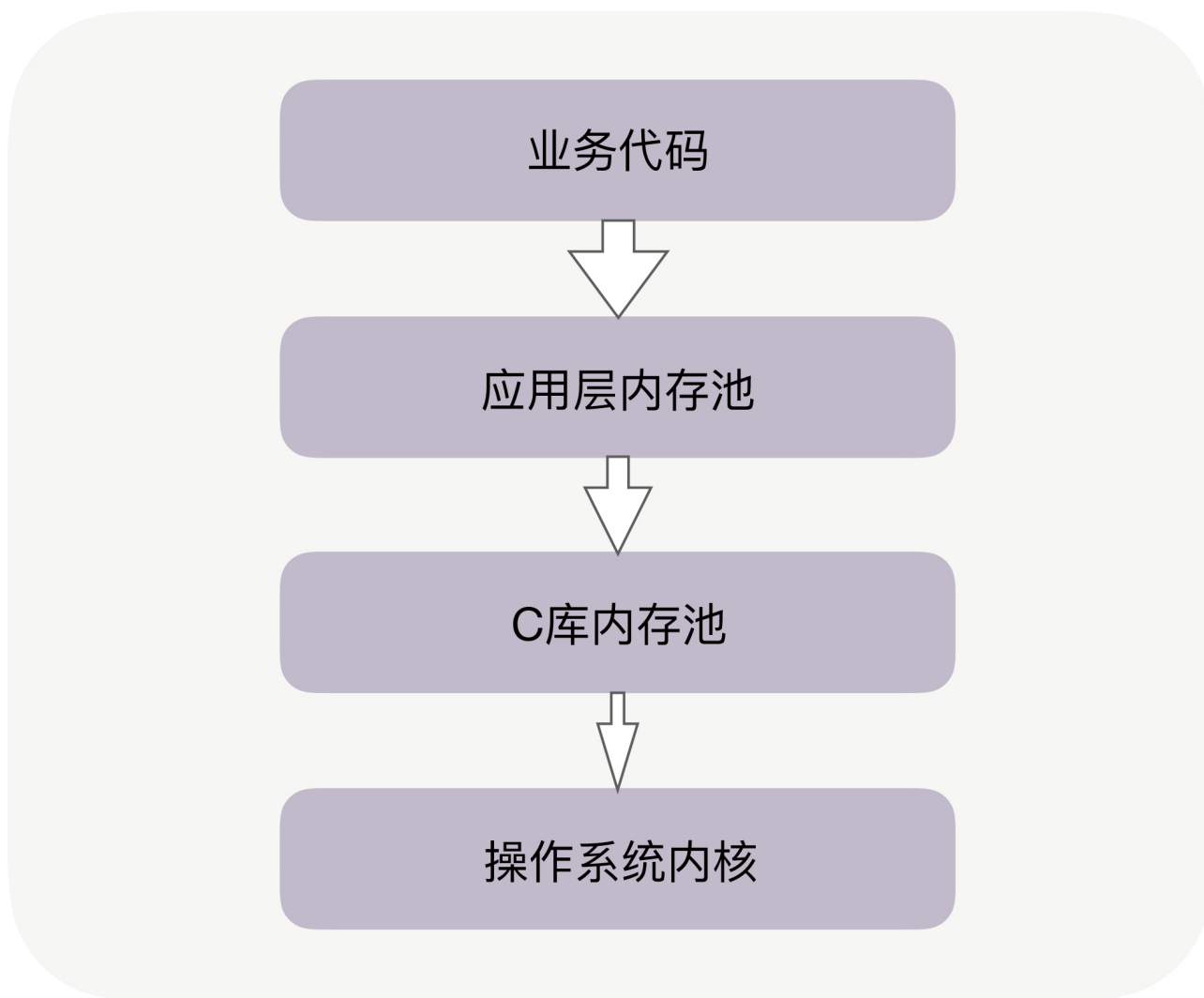
接下来，我们就通过对内存池的学习，看看如何提升内存分配的效率。

隐藏的内存池

实际上，在你的业务代码与系统内核间，往往有两层内存池容易被忽略，尤其是其中的 C 库内存池。

当代码申请内存时，首先会到达应用层内存池，如果应用层内存池有足够的可用内存，就会直接返回给业务代码，否则，它会向更底层的 C 库内存池申请内存。比如，如果你在 Apache、Nginx 等服务之上做模块开发，这些服务中就有独立的内存池。当然，Java 中也有内存池，当通过启动参数 Xmx 指定 JVM 的堆内存为 8GB 时，就设定了 JVM 堆内存池的大小。

你可能听说过 Google 的 TCMalloc 和 FaceBook 的 JEMalloc，它们也是 C 库内存池。当 C 库内存池无法满足内存申请时，才会向操作系统内核申请分配内存。如下图所示：



回到文章开头的问题，Java 已经有了应用层内存池，为什么还会受到 C 库内存池的影响呢？这是因为，除了 JVM 负责管理的堆内存外，Java 还拥有一些堆外内存，由于它不使用 JVM 的垃圾回收机制，所以更稳定、持久，处理 IO 的速度也更快。这些堆外内存就会由 C 库内存池负责分配，这是 Java 受到 C 库内存池影响的原因。

其实不只是 Java，几乎所有程序都在使用 C 库内存池分配出的内存。C 库内存池影响着系统下依赖它的所有进程。我们就以 Linux 系统的默认 C 库内存池 Ptmalloc2 来具体分析，看看它到底对性能发挥着怎样的作用。

C 库内存池工作时，会预分配比你申请的字节数更大的空间作为内存池。比如说，当主进程下申请 1 字节的内存时，Ptmalloc2 会预分配 132K 字节的内存（Ptmalloc2 中叫 Main Arena），应用代码再申请内存时，会从这已经申请到的 132KB 中继续分配。

如下所示（你可以在 [这里](#) 找到示例程序，注意地址的单位是 16 进制）：

```
1 # cat /proc/2891/maps | grep heap
2 01643000-01664000 rw-p 00000000 00:00 0 [heap]
```

当我们释放这 1 字节时，Ptmalloc2 也不会把内存归还给操作系统。Ptmalloc2 认为，与其把这 1 字节释放给操作系统，不如先缓存着放进内存池里，仍然当作用户态内存留下来，进程再次申请 1 字节的内存时就可以直接复用，这样速度快了很多。

你可能会想，132KB 不多呀？为什么这一讲开头提到的 Java 进程，会被分配了几个 GB 的内存池呢？这是因为**多线程与单线程的预分配策略并不相同**。

每个**子线程预分配的内存是 64MB**（Ptmalloc2 中被称为 Thread Arena，32 位系统下为 1MB，64 位系统下为 64MB）。如果有 100 个线程，就将有 6GB 的内存都会被内存池占用。当然，并不是设置了 1000 个线程，就会预分配 60GB 的内存，子线程内存池最多只能到 8 倍的 CPU 核数，比如在 32 核的服务器上，最多只会有 256 个子线程内存池，但这也非常夸张了，16GB（64MB * 256 = 16GB）的内存将一直被 Ptmalloc2 占用。

回到本文开头的问题，Linux 下的 JVM 编译时默认使用了 Ptmalloc2 内存池，因此每个线程都预分配了 64MB 的内存，这造成含有上百个 Java 线程的 JVM 多使用了 6GB 的内存。在多数情况下，这些预分配出来的内存池，可以提升后续内存分配的性能。

然而，Java 中的 JVM 内存池已经管理了绝大部分内存，确实不能接受莫名多出来 6GB 的内存，那该怎么办呢？既然我们知道了 Ptmalloc2 内存池的存在，就有两种解决办法。

首先可以调整 Ptmalloc2 的工作方式。**通过设置 `MALLOC_ARENA_MAX` 环境变量，可以限制线程内存池的最大数量**，当然，线程内存池的数量减少后，会影响 Ptmalloc2 分配内存的速度。不过由于 Java 主要使用 JVM 内存池来管理对象，这点影响并不重要。

其次可以更换掉 Ptmalloc2 内存池，选择一个预分配内存更少的内存池，比如 Google 的 TCMalloc。

这并不是说 Google 出品的 TCMalloc 性能更好，而是在特定的场景中的选择不同。而且，盲目地选择 TCMalloc 很可能会降低性能，否则 Linux 系统早把默认的内存池改为 TCMalloc 了。

TCMalloc 和 Ptmalloc2 是目前最主流的两个内存池，接下来我带你通过对比 TCMalloc 与 Ptmalloc2 内存池，看看到底该如何选择内存池。

选择 Ptmalloc2 还是 TCMalloc?

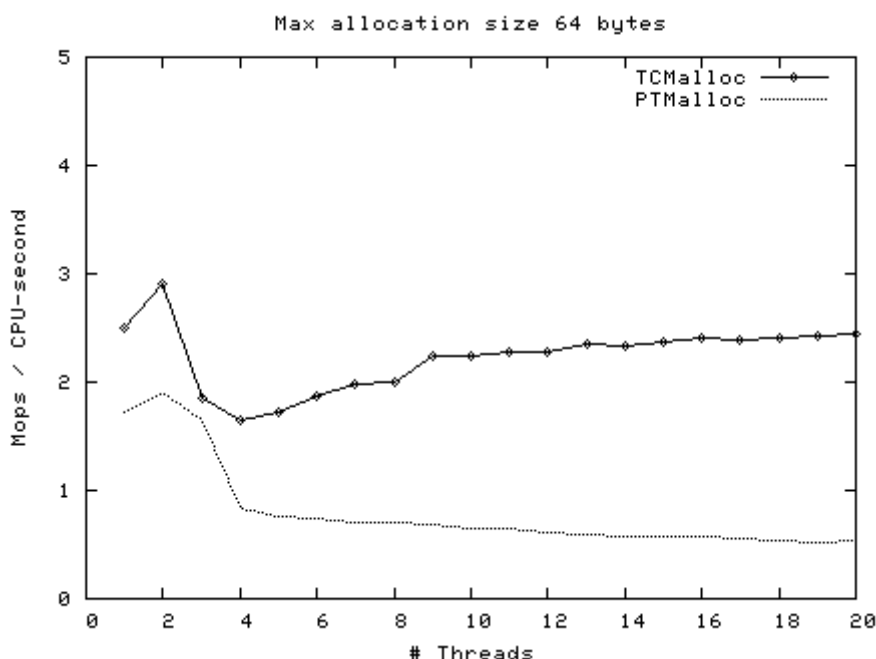
先来看 TCMalloc 适用的场景，它对多线程下小内存的分配特别友好。

比如，在 2GHz 的 CPU 上分配、释放 256K 字节的内存，Ptmalloc2 耗时 32 纳秒，而 TCMalloc 仅耗时 10 纳秒（测试代码参见[这里](#)）。差距超过了 3 倍，为什么呢？这是因为，Ptmalloc2 假定，如果线程 A 申请并释放了的内存，线程 B 可能也会申请类似的内存，所以它允许内存池在线程间复用以提升性能。

因此，每次分配内存，Ptmalloc2 一定要加锁，才能解决共享资源的互斥问题。然而，加锁的消耗并不小。如果你监控分配速度的话，会发现单线程服务调整为 100 个线程，Ptmalloc2 申请内存的速度会变慢 10 倍。TCMalloc 针对小内存做了很多优化，每个线程独立分配内存，无须加锁，所以速度更快！

而且，线程数越多，Ptmalloc2 出现锁竞争的概率就越高。比如我们用 40 个线程做同样的测试，TCMalloc 只是从 10 纳秒上升到 25 纳秒，只增长了 1.5 倍，而 Ptmalloc2 则从 32 纳秒上升到 137 纳秒，增长了 3 倍以上。

下图是 TCMalloc 作者给出的性能测试数据，可以看到线程数越多，二者的速度差距越大。所以，当应用场景涉及大量的并发线程时，换成 TCMalloc 库也更有优势！



那么, 为什么 Glibc 不把默认的 Ptmalloc2 内存池换成 TCMalloc 呢? **因为 Ptmalloc2 更擅长大内存的分配。**

比如, 单线程下分配 257K 字节的内存, Ptmalloc2 的耗时不变仍然是 32 纳秒, 但 TCMalloc 就由 10 纳秒上升到 64 纳秒, 增长了 5 倍以上! **现在 TCMalloc 反过来比 Ptmalloc2 慢了 1 倍!** 这是因为 TCMalloc 特意针对小内存做了优化。

多少字节叫小内存呢? TCMalloc 把内存分为 3 个档次, 小于等于 256KB 的称为小内存, 从 256KB 到 1M 称为中等内存, 大于 1MB 的叫做大内存。TCMalloc 对中等内存、大内存的分配速度很慢, 比如我们用单线程分配 2M 的内存, Ptmalloc2 耗时仍然稳定在 32 纳秒, 但 TCMalloc 已经上升到 86 纳秒, 增长了 7 倍以上。

所以, **如果主要分配 256KB 以下的内存, 特别是在多线程环境下, 应当选择 TCMalloc; 否则应使用 Ptmalloc2, 它的通用性更好。**


从堆还是栈上分配内存?

不知道你发现没有, 刚刚讨论的内存池中分配出的都是堆内存, 如果你把在堆中分配的对象改为在栈上分配, 速度还会再快上 1 倍 (具体测试代码可以在 [这里](#)找到) ! 为什么?

可能有同学还不清楚堆和栈内存是如何分配的, 我先简单介绍一下。

如果你使用的是静态类型语言, 那么, 不使用 new 关键字分配的对象大都是在栈中的。比如:

```
1 C/C++/Java语言: int a = 10;
```

 复制代码

否则, 通过 new 或者 malloc 关键字分配的对象则是在堆中的:

```
1 C语言: int * a = (int*) malloc(sizeof(int));  
2 C++语言: int * a = new int;
```

 复制代码

```
3 Java语言: int a = new Integer(10);
```

另外，对于动态类型语言，无论是否使用 `new` 关键字，内存都是从堆中分配的。

了解了这一点之后，我们再来看看，为什么从栈中分配内存会更快。

这是因为，由于每个线程都有独立的栈，所以分配内存时不需要加锁保护，而且栈上对象的尺寸在编译阶段就已经写入可执行文件了，执行效率更高！性能至上的 Golang 语言就是按照这个逻辑设计的，即使你用 `new` 关键字分配了堆内存，但编译器如果认为在栈中分配不影响功能语义时，会自动改为在栈中分配。

当然，在栈中分配内存也有缺点，它有功能上的限制。一是，栈内存生命周期有限，它会随着函数调用结束后自动释放。在堆中分配的内存，并不随着分配时所在函数调用的结束而释放，它的生命周期足够使用；二是，栈的容量有限，如 CentOS 7 中是 8MB 字节，如果你申请的内存超过限制会造成栈溢出错误（比如，递归函数调用很容易造成这种问题），而堆则没有容量限制。

所以，当我们分配内存时，如果在满足功能的情况下，可以在栈中分配的话，就选择栈。

小结

最后我们对这一讲做个小结。

进程申请内存的速度，以及总内存空间都受到内存池的影响。知道这些隐藏内存池的存在，是提升分配内存效率的前提。

隐藏着的 C 库内存池，对进程的内存开销有很大的影响。当进程的占用空间超出预期时，你需要清楚你正在使用的是哪种内存池，它对每个线程预分配了多大的空间。

不同的 C 库内存池，都有它们最适合的应用场景，例如 `TCMalloc` 对多线程下的小内存分配特别友好，而 `Ptmalloc2` 则对各类尺寸的内存申请都有稳定的表现，更加通用。

内存池管理着堆内存，它的分配速度比不上在栈中分配内存。只是栈中分配的内存受到生命周期和容量大小的限制，应用场景更为有限。然而，如果有可能的话，尽量在栈中分配内

存，它比内存池中的堆内存分配速度快很多！

OK，今天我们从内存分配的角度聊了分布式系统性能提升的内容，希望学习过今天的内容后，你知道如何最快速地申请到内存，了解你正在使用的内存池，并清楚它对进程最终内存大小的影响。即使对第三方组件，我们也可以通过 `LP_PRELOAD` 环境变量，在程序启动时更换最适合的 C 库内存池（Linux 中通过 `LD_PRELOAD` 修改动态库来更换内存池，参见 [🔗 示例代码](#)）。

内存分配时间虽然不起眼，但时刻用最快的方法申请内存，正是高手与初学者的区别，相似算法的性能差距就体现在这些编码细节上，希望你能够重视它。

思考题

最后，留给你一个思考题。分配对象时，除了分配内存，还需要初始化对象的数据结构。内存池对于初始化对象有什么帮助吗？欢迎你在留言区与大家一起探讨。

感谢阅读，如果你觉得这篇文章对你有一些启发，也欢迎把它分享给你的朋友。

系统性能调优必知必会

深入底层直击性能问题本质

陶辉

智链达 CTO、前阿里云高级技术专家



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

精选留言 (11)

写留言



忆水寒

2020-04-27

- 1、思考题：内存池中可以利用享元模式将常用的对象一直保留着，减少重复申请导致的性能的顺耗。
- 2、最后一段话“内存分配时间虽然不起眼，但时刻用最快的方法申请内存，正是高手与初学者的区别。”说的很好，是的，真正的高手应该能够从算法到底层都能优化。

展开 v

作者回复: 忆水寒同学说得对，享元模式这个词用得也很好！享元模式有广泛的应用，不只在应用层，在内核中也被广泛使用。



3

5



ermaot

2020-04-27

解决了我很多疑惑。比如mysql很多人建议把内存分配换成tcmalloc，就是因为mysql要支持大量并发，适合tcmalloc的应用场景。没有对比就没有发现，两库一比，知识点就出来了

作者回复: 很高兴能帮到你，我跟编辑一直担心这个会不会讲得太深了呢，其实很多性能优先的组件都可以用得上



2

2



奥特曼

2020-04-28

老师好，我有3个问题：

-----问题1-----

讲义里面提到64位的环境下，一个子线程创建会有64M的内存申请，我最开始理解的这64M是这个子线程独有的。

在后面又提到“Ptmalloc2 假定，如果线程 A 申请并释放了的内存，线程 B 可能也会申...

展开 v

1

1



weing

2020-04-28

alloc_address.cpp 代码中，分配的是1MB，还是8MB？感觉应该是1MB才对呢

```
printf("接下来分配8MB内存\n");  
getchar();  
addr = (char*) malloc(1*1024*1024);  
printf("分配了8MB内存\n");
```

展开 ∨

作者回复: 不好意思，因为最初我用8MB测试，后面改回1MB，但忘了改printf提示语句了。我现在已经把它更新过来了



1



aoe

2020-04-28

1. 原来Java堆的内存空间是通过C库内存池申请！
2. 第一次知道内存分配器的存在：Ptmalloc2、TCMalloc
3. 在栈中申请内存比堆中快是因为不需要加锁。

收获惊呆了！

展开 ∨

作者回复: 又添aoe大招了^_^



1



baggio

2020-04-28

正好有一个服务堆外内存占用很大，可以试试调整一下malloc



alan

2020-04-28

老师好，这节课真好，第一次了解到内存池也是有层次的。我遇到一个问题想请教一下：我有一个和数据库交互的groovy程序，运行起来后会占用很大内存，启动时，将Xmx设置为多少，该程序的内存占用就不会超过Xmx指定的上限。比如，Xmx=10g，程序就稳定占10g内存，但如果不限制的话，最高见过占用30G左右。这个您觉得有什么可能的原因吗？

展开 ∨

1





张智凯

2020-04-28

学到了一个新的名词享元模式，内存池是堆上分配的内存，可以在内存池里维护常用对象的cache，这样新生成对象时就可以直接拿来复用，初始化时可以减少初始化的成员变量



梦醒人间

2020-04-28

TCMalloc 对中等内存、大内存的分配速度很慢，比如我们用单线程分配 2M 的内存，Pt malloc2 耗时仍然稳定在 32 纳秒，但 TCMalloc 已经上升到 86 纳秒，增长了 7 倍以上。

老师你好，这块能展开说一下吗？对中等内存、大内存，为什么 TCMalloc 慢，而 Ptmalloc2 快呢？

展开 ▾



我来也

2020-04-27

以前写c程序，会习惯性的使用bzero或memset初始化结构体数据。因为申请的里面可能有历史数据，使用不当会有莫名其妙的问题。

现在的golang所有未赋值的变量都会是默认值。

（也许有人会觉得go这个操作有点多余。）

...

展开 ▾



kofssl

2020-04-27

确实写的很清晰，前面处理三方组件内存问题时，开始以为内存泄漏了，占用内存一点一点上去，就不释放，后来查代码也都没有明显的错误，最后是通过同事提到的内存碎片解决的，就用到了jemalloc的替换方式，同事是高手，你也是，哈哈

展开 ▾

作者回复: 呵呵，谢谢kofssl的夸奖。在你构建出分布式系统的完整执行路径后，相信面临其他疑难杂症时都会有明确的方向，可以google上自行找答案！



