1 介绍

jemalloc最初是Jason Evans为FreeBSD开发的新一代内存分配器, 用来替代原来的phkmalloc, 最早投入使用是在2005年. 到目前为止, 除了原版jemalloc, 还有很多变种被用在各种项目里. Google在android5.0里将bionic中的默认分配器从dlmalloc替换为jemalloc, 也是看中了其强大的多核 多线程 分配能力.

同经典分配器, 如dlmalloc相比, jemalloc在基本思路和实现上存在明显的差别. 比如, dlmalloc在分配策略上倾向于先dss后mmap的方式, 为的是快速向前分配, 但jemalloc则完全相反. 而实现上也放弃了经典的boundary tag. 这些设计牺牲了局部分配速度和回收效率, 但在更大的空间和时间范围内却获得更好的分配效果.

更重要的是, 相对经典分配器, **jemalloc最大的优势还是其强大的多核/多线程分配能力**. 以现代计算机硬件架构来说, 最大的瓶颈已经不再是内存容量或cpu速度, 而是多核/多线程下的lock contention(锁竞争). 因为无论CPU核心数量如何多, 通常情况下内存只有一份. 可以说, 如果内存足够大, CPU的核心数量越多, 程序线程数越多, jemalloc的分配速度越快. 而这一点是经典分配器所无法达到的.

这篇文章基于android5.x中的jemalloc3.6.0. 最新的版本为4.x, 获取最新代码请至,
https://github.com/jemalloc/jemalloc/releases

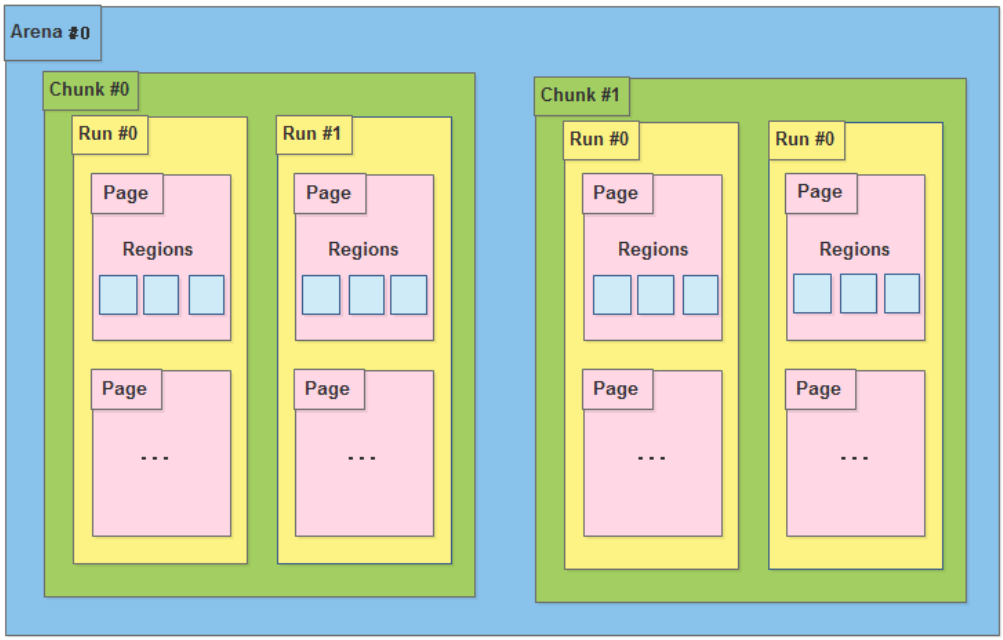
2 基础结构

相对于dlmalloc, jemalloc引入了更多更复杂的分配结构, 如arena, chunk, bin, run, region, tcache等等. 其中有些类似dlmalloc, 但更多的具有不同含义, 本节将对它们做一一介绍.

2.1 概览

首先, 先给出一个整体的概念. jemalloc对内存划分按照如下**由高到低**的顺序:

1. 内存是由一定数量的arenas进行管理.
2. 一个arena被分割成若干chunks, 后者主要负责记录bookkeeping（记录信息）.
3. chunk内部又包含着若干runs, 作为分配小块内存的基本单元.
4. run由pages组成, 最终被划分成一定数量的regions,
5. 对于small size的分配请求来说, 这些region就相当于user memory.



可以把user使用jemalloc进行内存分配的过程类比从电商购物:

电商向批发商批发大量整箱的货物，然后或进行拆分零售，或整块出售。整箱的内存称为chunk，对于巨大件内存订单，则直接出售chunk，对于小件和大件订单，则把chunk进一步拆分成run。

其中chunk的大小为4MB（可调）且为4MB对齐；run大小为page（默认大小为4KB）的整数倍。

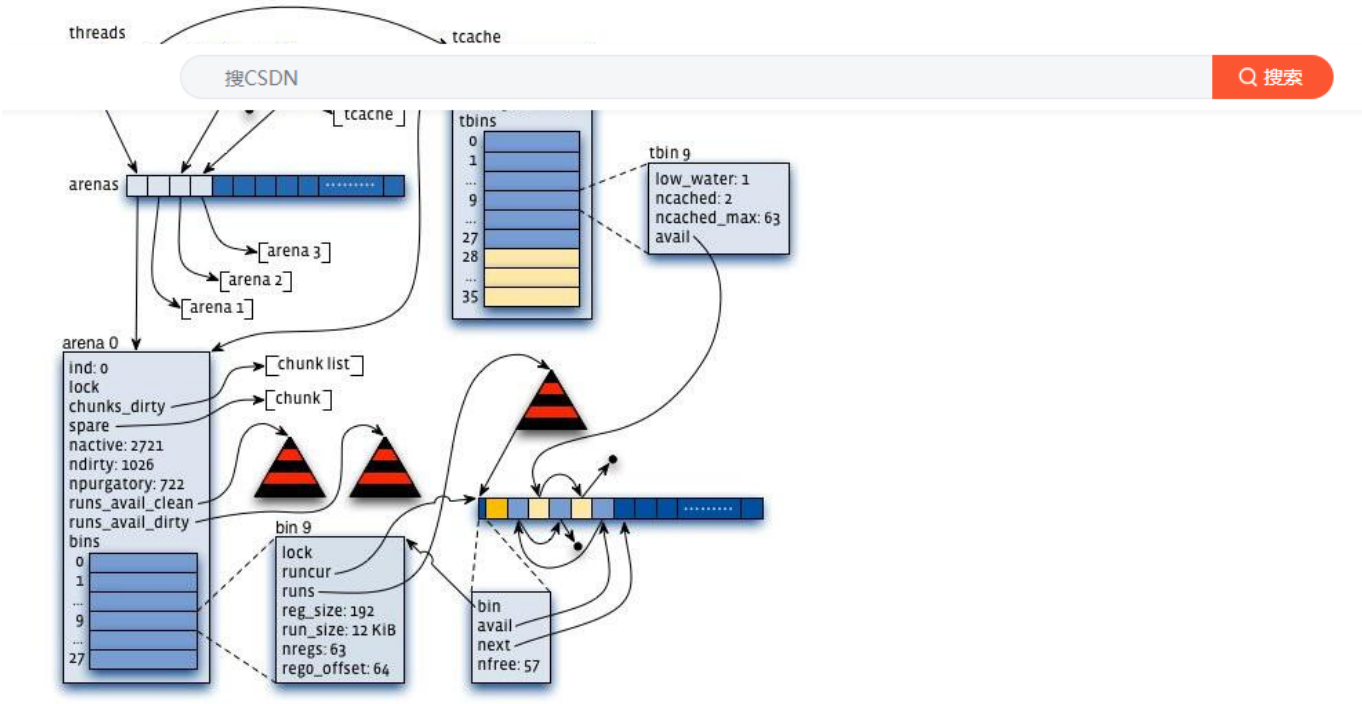
对于小件订单，并不是直接出售run，run更像是一个小件内存的仓库，并只满足同一大小的内存订单，其中每个可出售的小内存称为regions。

jemalloc可出售的小件内存大小范围是1～57344字节，为了减少内存分配碎片和快速找到合适大小的内存进行出售，需要对小件订单再进一步细分，将1～57344字节区间拆分成(拆分的算法是二分法)44个run仓库，每次出售时按照订单大小（对齐后）找到对应大小的run仓库，并从该run仓库分配regions。eg,小于8字节的内存申请，直接分配8字节空间，17～32字节内存申请，直接分配32字节空间。

如果订购的物品是小件（eg，一块橡皮、一本书或是一个微波炉等），那么直接从同城仓库送出；如果订购的物品时大件（eg,电视机、空调等），那么需要从区域（eg，华东区仓库）仓库送出；如果订购的物件是一个巨大件（eg,汽车、轮船等），需要从全国仓库送出。

同城仓库相当于tcache—线程独有的内存仓库；区域仓库相当于arena—多个线程共享的内存仓库；全国仓库类比全局变量指向的内存仓库，所有线程共用。

完整的jemalloc架构图：



从结构图中可以看出jemalloc远比dlmalloc复杂，等阅读完整个文章后再回来看下该图，会理解更深刻一些。