来源 Linux 文档

对于需要处理大量内存的关键性能计算应用程序，可以在 `libhugetlbfs` 或 `hugetlbfs` 上运行。透明大页(Transparent Hugepage)支持是一种替代方法，能够自动提升和降级页面大小(尽可能分配为大页)来实现大页虚拟内存，同时没有 `hugetlbfs` 的缺点。

应用程序运行更快的因素有两个:

- 采用2M虚拟内存页的单页错误比采用4KB内存页的单页错误概率低512倍(未命中错误)
- TLB未命中将运行更快(特别是使用嵌套页表进行虚拟化，以及在裸金属物理主机上没有虚拟化的情况)，单个TLB会映射大量的虚拟内存，从而减少TLB未命中次数

只有KVM和Linux guest可以通过虚拟化和嵌套页表来映射更大的TLB，这是因为TLB miss会运行更快。

透明大页设计

- `优雅地回退` : 没有透明大页的 `mm` 组件感知到需要回退，就会将大型 `pmd` 映射分解为 `ptes` 表，并且如有必要，拆分成一个透明大页。这些组件可以持续处理常规页面或者常规 `pte` 映射
- 如果由于内存碎片导致内存大页分配失败，则常规页面(`4KiB`)可以优雅地分配并混入相同的 `mva` 而不会有任何故障或重大延迟，也不需要userland通知
- 如果一些任务推出并且有更多的大页可以使用(要么通过伙伴buddy要么通过VM立即完成)，则guest物理内存由常规内存页面重新定位到大页上(使用 `khugepaged`)
- 透明大页不需要内存预留(不像静态大页)，只要有可能就使用大页(这里唯一可能保留的是 `kernelcore=` 来避免不可移动的页面碎片化，不过这种调整不是针对透明大页的支持，而是通用的适合所有动态高阶内存分配的核心特性)

透明大页可以最大限度利用空闲内存，如果通过允许所有未使用的内存用作缓存。透明大页不需要保留，以避免大页从用户空间看到分配失败。

在某些情况下，使用系统范围的大页会导致应用程序分配更多内存资源: 应用程序会映射一个大区域但是只使用了1个字节内存，此时采用2MB大页分配而不是4K页面是没有收益的。这就是为何在系统范围内禁止使用内存大页，而只是在关键映射区域上使用 `madvise` (MADV_HUGEPAGE)

使用

对于开发者而言，申请页面时仍然是申请4 kb大小的页面，但操作系统会自动将这些请求合并成2MB大小的透明大页，应用程序不需要知道或关心底层使用了大页。