**操作系统提高内存利用率的历程**

早期的计算机内存很小，常常会出现内存大小不够的情况。于是人们就引入了覆盖技术，以解决程序大小超过物理内存总和的问题。它针对于同一程序中模块调用其他模块的情况，让那些不能被同时访问的的模块共享同一个覆盖区。覆盖技术能解决内存大小不够的问题，但是同时由于它的覆盖结构必须由程序员声明，也增加了很大的编程负担。并且它浪费了大量的内存空间，导致内存利用率很低。

当内存中有多个程序的时候，就引入了交换技术。在内存紧张时，系统将内存中某些进程暂时换出到外存，进程中的PCB保留在内存，并且插入到挂起队列中，而外存中某些具备运行条件的进程换入内存。可以优先换出阻塞进程、优先级低的进程等。使得内存利用率有一定的提升，并且从逻辑上扩充了内存。

为了减少内部碎片，进一步提高内存利用率，就涉及到内存空间的分配与回收。

单一连续分配中，内存被分为系统区和用户区，系统区存放操作系统相关数据。但是由于用户区中只能存一道用户程序，因此会产生内部碎片，内存利用率依旧很低。

20世纪60年代出现了支持多道程序的系统，为了能在内存中装入多道程序，且这些程序间互不干扰，于是将用户空间划分为若干个固定大小的分区，每个分区只装入一道作业。这就是固定分区分配，它分为分区大小相等和分区大小不等两种。这种分配方法实现简单且不会产生外部碎片，但是会产生很多内部碎片，内存利用率低。

为了解决这些问题，就使用了动态分区分配的方法。在进程装入内存时，根据进程的大小动态建立分区。该方法相较于固定分区分配能有效的提高内存利用率，但是同时也会产生新的问题。在进行回收后会产生离散的空闲分区，内部碎片依旧存在，同时若空闲分区太小也会产生外部碎片。虽然外部碎片可以通过“紧凑”来解决，但是时间代价过大不适合经常使用。

以上三种都属于连续分配方式，连续分配都要求进程占用一整段的连续区域，会产生内部和外部碎片。因此就提出了离散分配方式，包括基本分页式存储管理、基本分段式存储管理、基本段页式存储管理。可以看做是将固定分配方式改为非连续分配版本，将进程分页/分段，离散的存入各内存块中。如果在这些方法的基础上再加入覆盖、交换技术，就能使利用率有所提升，并且逻辑上扩充内存容量。

连续和非连续都属于传统存储管理，他们有一些共同的特征和缺点。比如：很多暂时用不到的数据会长期占用内存，导致内存利用率不高；作业必须一次性全部装入内存运行，容易导致大作业无法运行、多道程序并发度下降。

现代操作系统中，提高内存利用率主要是通过虚拟存储器实现的。它就是在传统存储的基础上增加了一些功能：程序装入时，将很快会用到的部分装入内存，暂时用不到的部分留在外存；程序执行时，若访问信息不在内存，则再由操作系统将信息从外存调入内存，即请求调页功能；若内存不够，就将暂时用不到的信息换出到外存，即页面置换功能。这样能使用户看来有一个比实际内存大得多的内存，这就是虚拟内存，逻辑上扩充了内存容量。因此，这样使得处于内存中的信息都是短时间用得上的，调到外存的都是短时间用不上的，有效提高了内存利用率。