值得提及该设计看来不明显的四个方面。首先，在一个进程进行系统调用时，该系统调用是在本机的CPU上被捕获并处理的，并使用操作系统表中的数据结构。

其次，因为每个操作系统都有自己的表，那么它也有自己的进程集合，通过自身调度这些进程。这里没有进程共享。如果一个用户登录到CPU 1，那么他的所有进程都在CPU 1上运行。因此，在CPU 2有负载运行而CPU 1空载的情形是会发生的。

第三，没有页面共享。会出现如下的情形：在CPU2不断地进行页面调度时CPU 1却有多余的页面。由于内存分配是固定的，所以CPU 2无法向CPU 1借用页面。

第四，也是最坏的情形，如果操作系统维护近期使用过的磁盘块的缓冲区高速缓存，每个操作系统都独自进行这种维护工作，因此，可能出现某一修改过的磁盘块同时存在于多个缓冲区高速缓存的情况，这将会导致不一致性的结果。避免这一问题的惟一途径是，取消缓冲区高速缓存。这样做并不难，但是会显著降低性能。

由于这些原因，上述模型已很少使用，尽管在早期的多处理机中它一度被采用，那时的目标是把已有的操作系统尽可能快地移植到新的多处理机上。

2. 主从多处理机

图8-8中给出的是第二种模型。在这种模型中，操作系统的一个副本及其数据表都在CPU 1上，而不是在其他所有CPU上。为了在该CPU 1上进行处理，所有的系统调用都重定向到CPU 1上。如果有剩余的CPU时间，还可以在 CPU 1上运行用户进程。这种模型称为主从模型（master-slave），因为CPU 1是主CPU，而其他的都是从属CPU。

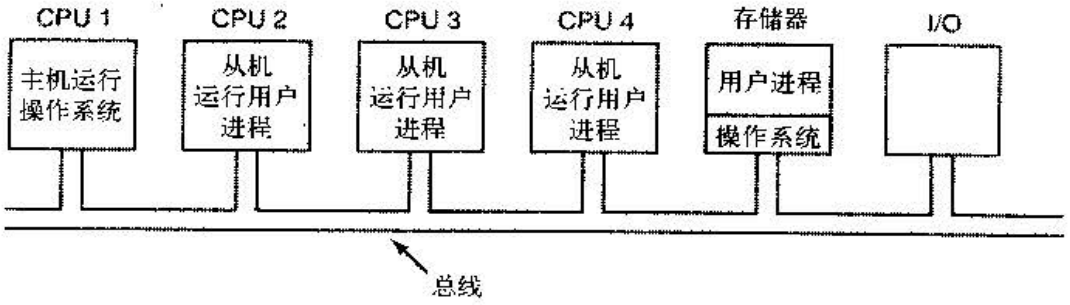


图8-8 主从多处理机模型

主从模型解决了在第一种模型中的多数问题。有单一的数据结构（如一个链表或者一组优先级链表）用来记录就绪进程。当某个CPU空闲下来时，它向CPU 1上的操作系统请求一个进程运行，并被分配一个进程。这样，就不会出现一个CPU空闲而另一个过载的情形。类似地，可在所有的进程中动态地分配页面，而且只有一个缓冲区高速缓存，所以决不会出现不一致的情形。

这个模型的问题是，如果有很多的CPU，主CPU 会变成一个瓶颈。毕竟，它要处理来自所有CPU的系统调用。如果全部时间的10%用来处理系统调用，那么10个CPU就会使主CPU饱和，而20个CPU就会使主CPU彻底过载。可见，这个模型虽然简单，而且对小型多处理机是可行的，但不能用于大型多处理机。

3. 对称多处理机

我们的第三种模型，即对称多处理机（Symmetric MultiProcessor，SMP），消除了上述的不对称性。在存储器中有操作系统的一个副本，但任何CPU都可以运行它。在有系统调用时，进行系统调用的CPU陷入内核并处理系统调用。图8-9是对SMP模式的说明。

这个模型动态地平衡进程和存储器，因为它只有一套操作系统数据表。它还消除了主CPU的瓶颈，因为不存在主CPU，但是这个模型也带来了自身的问题。特别是，当两个或更多的CPU同时运行操作系统代码时，就会出现灾难。想象有两个CPU同时选择相同的进程运行或请求同一个空闲存储器页面。处理这些问题的最简单方法是在操作系统中使用互斥信号量（锁），使整个系统成为一个大临界区。当一个CPU要运行操作系统时，它必须首先获得互斥信号量。如果互斥信号量被锁住，就得等待。按照这种方式，任何CPU都可以运行操作系统，但在任一时刻只有一个CPU可运行操作系统。