2. 伪共享

在某些关键方式上DSM系统与多处理机类似。在这两种系统中，当引用非本地存储器字时，从该字所在的机器上取包含该字的一块内存，并放到进行引用的（分别是内存储器或高速缓存）相关机器上。一个重要的设计问题是应该调取多大一块。在多处理机中，其高速缓存块的大小通常是32字节或64字节，这是为了避免占用总线传输的时间过长。在DSM系统中，块的单位必须是页面大小的整数倍（因为MMU以页面方式工作），不过可以是1个、2个、4个或更多个页面。事实上，这样做就模拟了一个更大尺寸的页面。

对于DSM而言，较大的页面大小有优点也有缺点。其最大的优点是，因为网络传输的启动时间是相当长的，所以传递4096字节并不比传输1024个字节多花费多少时间。在有大量的地址空间需要移动时，通过采用大单位的数据传输，通常可减少传输的次数。这个特性是非常重要的，因为许多程序表现出引用上的局部性，其含义是如果一个程序引用了某页中的一个字，很可能在不久的将来它还会引用同一个页面中其他字。

另一方面，大页面的传输造成网络长期占用，阻塞了其他进程引起的故障。还有，过大的有效页面引起了另一个问题，称为伪共享（false sharing），如图8-23所示。图8-23中一个页面中含有两个无关的共享变量A和B。进程1大量使用A，进行读写操作。类似地，进程2经常使用B。在这种情形下，含有这两个变量的页面将在两台机器中来回地传送。

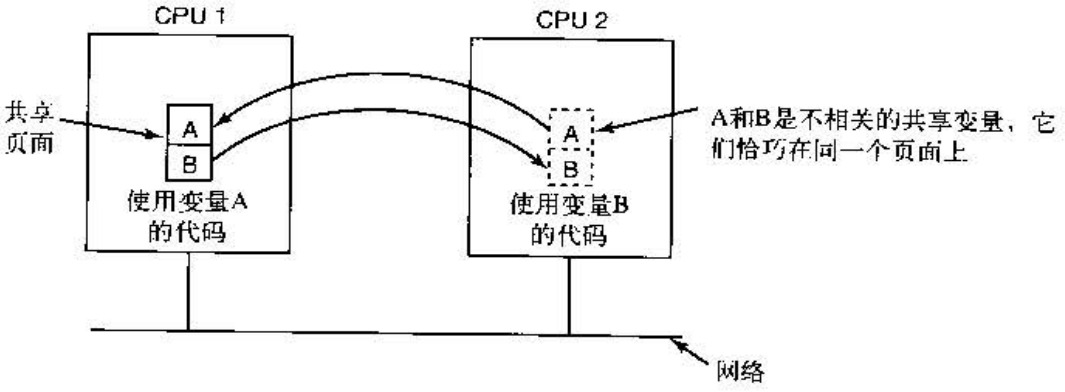

图8-23 含有两个无关变量的页面的伪共享

这里的问题是，尽管这些变量是无关的，但它们碰巧在同一个页面内，所以当某个进程使用其中一个变量时，它也得到另一个。有效页面越大，发生伪共享的可能性也越高；相反，有效页面越小，发生伪共享的可能性也越少。在普通的虚拟内存系统中不存在类似的现象。

理解这个问题并把变量放在相应的地址空间中的高明编译器能够帮助减少伪共享并改善性能。但是，说起来容易做起来难。而且，如果伪共享中节点1使用某个数组中的一个元素，而节点2使用同一数组中的另一个元素，那么即使再高明的编译器也没有办法消除这个问题。

3. 实现顺序一致性

如果不对可写页面进行复制，那么实现一致性是没有问题的。每个可写页面只对应有一个副本，在需要时动态地来回移动。由于并不是总能提前了解哪些页面是可写的，所以在许多DSM系统中，当一个进程试图读一个远程页面时，则复制一个本地副本，在本地和远程各自对应的MMU中建立只读副本。只要所有的引用都做读操作，那么一切正常。

但是，如果有一个进程试图在一个被复制的页面上写入，潜在的一致性问题就会出现，因为只修改一个副本却不管其他副本的做法是不能接受的。这种情形与在多处理机中一个CPU试图修改存在于多个高速缓存中的一个字的情况有类似之处。在多处理机中的解决方案是，要进行写的CPU首先将一个信号放到总线上，通知所有其他的CPU丢弃该高速缓存块的副本。这里的DSM系统以同样的方式工作。在对一个共享页面进行写入之前，先向所有持有该页面副本的CPU发出一条消息，通知它们解除映射并丢弃该页面。在其所有解除映射等工作完成之后，该CPU便可以进行写操作了。