1. 基于总线的多处理机

基于总线的多处理机是由若干个 CPU 组成的，它们都连接到一个公共的总线上，并且共享一个存储器模块。典型的总线有 32 根或 64 根地址线，32 根或 64 根数据线，还有约 32 根或更多的控制线，它们都是并行操作的。为了要读取存储器中的一个字（coherent），CPU 首先将它想要读取的字的地址放到总线的地址线上，然后给控制线发送一个适当的信号表明它想要进行读操作。存储器进行响应把该地址的内容送到数据线上以使 CPU 可以对它进行读取。写操作也以相似的方式进行。

由于存储器只有一个，当 CPU A 给存储器写入一个字，一微秒后 CPU B 在读出该字的内容时会得到 A 刚写入的那个值。具有这种特性的存储器被称为是相关的（coherent）。这种实现方案的问题在于，只要有仅仅 4 到 5 个 CPU 时，总线就会经常过载，引起性能也会急剧下降。解决的办法是在 CPU 和总线之间增加一个高速缓冲存储器（cache memory），如图 1 所示。缓冲存储器保留着最近刚存取过的字。所有的内存访问请求都要经过它。如果请求的字在缓冲存储器中，缓冲存储器就会直接响应 CPU，而不产生总线请求。如果缓冲存储器足够大的话，那么成功的可能性，称为命中率，将是很高的。而且每个 CPU 的总线通信量也会急剧下降，系统中也就能够容纳更多的 CPU。通常，缓冲存储器的大小从 64 K～1 M，命中率经常可以达到 90%或更高。

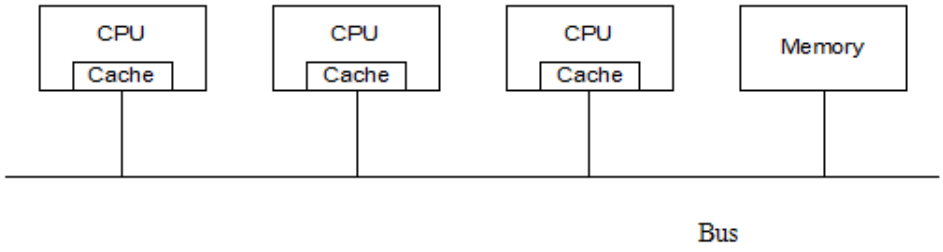

图 1 基于总线的多处理机

然而，缓冲存储器的引入也带来了一个严重的问题。假定有两个 CPU ，A 和 B，它们把同一个字分别读入自己的缓冲存储器内。在 A 改写这个字后，当 B 再读该字时，从它的缓冲存储器中得到的是原来的旧值，而不是 A 刚写入的值。为此，许多研究者研究了这个问题，也找到了许多解决办法。其中的一个方法是把缓冲存储器设计成无论何时在对其写入一个字时，该字就能同时写入存储器中。这种缓冲存储器，并不令人惊讶，叫作写直达缓冲存储器（write-through cache）。在此设计中，缓冲存储器读命中不会引起任何总线负载，但是缓冲器读失败和所有的写命中和写失败都会引起总线负载。

另外，所有的缓冲存储器都不断监听总线。当一个缓冲存储器发现它所存储的某个地址的字在存储器中正被进行写操作时，它要么从它的存储器中去掉那个字，要么把该存储器字更改为新值。这样的缓冲存储器称为监听缓冲存储器（snoopy cache），因为它总是在对总线进行“监听”（窃听）。一个包含监听通写缓冲存储器的设计是相关的，并且对于编程人员来说也是不可见的。几乎所有基于总线的多处理机都使用这种或者与之十分类似的体系结构。

2. 交换型多处理机

要建立一个拥有 64 个以上处理机的多处理机系统就需要采用一种不同的方法把这些 CPU 和存储器连接起来。一种可能性就是把存储器分成许多存储模块，用十字交叉开关把它们与 CPU 相连。如图 2 所示,在图中我们可以看出，每个 CPU 和每个存储器模块都有一个外部接口同这个交换开关相连。在每个交叉点上都有一个微小的能够用硬件打开或关闭的电子交叉点开关。当 CPU 想要访问一个特定的存储器模块时，连接它们的交叉点就暂时闭合，允许访问。许多 CPU 能够同时访问存储器是十字交叉开关的优点，尽管如果两个 CPU 想要同时访问同一个存储器模块，其中的一个必须等待另外一个访问结束之后才能够进行。

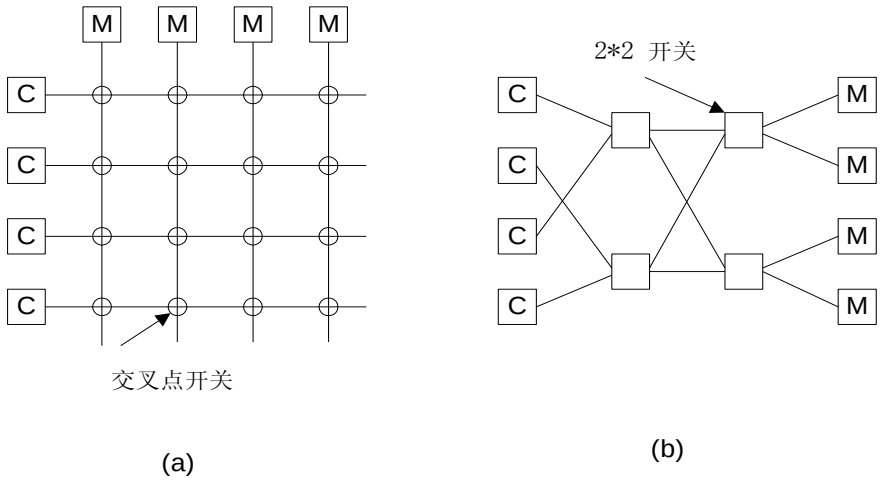

图 2

(a)交叉开关；(b) Omega 网络(图中 C 表示 CPU, M 表示 Memory)

交叉开关的不足之处在于当有 n 个 CPU 和 n 个存储器模块时，需要 n^2 个交叉开关。如果 n 值很大的话，那么这个数字就会非常大。为了避免这种情况的发生，人们已经找出了替代的交换网络，它只需要较少的开关。图 2（b）中的 Omega 网络就是替代这种网络的一个实例。在这个网络中包括 4 个（2×2）交换开关，每个交换开关都有两个输入和两个输出，而且任何一个输入都可以和任何一个输出相连接。如果我们仔细研究上图就会发现，只要正确的设置开关，任何一个

CPU 都可以访问所有的存储器。而这些交换开关的设定工作可以在纳秒级或更少的时间内完成。

在通常的情况下，如果有 n 个 CPU 和 n 个存储器模块，Omega 网络就要 $\log_2 n$ 个交换开关级，每级包括 $n/2$ 个交换开关，总共 $(n \log_2 n) / 2$ 个交换开关。虽然当 n 值很大时，它要比 n^2 好，但实际数量仍然相当大。

另外还有一个问题就是时延。例如，当 n=1024 时，从 CPU 到存储器有 10 个交换开关级，而还有另外的 10 个交换开关级用于返回所请求的字。假定 CPU 是现代的 RISC 芯片，其速度是 100 MIPS，也就是执行一条指令的时间是 10 ns。如果一个存储器请求及返回结果必须在 10 ns 内通过所有的 20 个交换开关级（10 个用于传送出请求而另外 10 个用于传送返回结果），那么交换开关的交换时间必须为 500 微微秒（0.5 ns）。根据前面的计算，整个多处理机系统需要 5120 个 500 微微秒级的交换开关，这是十分昂贵的。

有人曾经尝试通过采用分层系统减少成本。每个 CPU 都同一些存储器相连接。每个 CPU 可以快速地访问它自己的本地存储器，但是访问其他 CPU 的存储器就会慢许多。这种设计产生了称为 NUMA（非一致存储访问 NonUniform Memory Access）的机器。尽管 NUMA 机器比基于 Omega 网络的机器有更好的平均访问时间，但是它也有新的复杂的问题，为了使大部分访问集中在本地的存储器中，程序和数据的放置位置就十分关键了。

3. 基于总线的多计算机

另一方面，建立一个多计算机系统（即，非共享存储器）是容易的。每个 CPU 都与它自身的存储器直接相连。唯一遗留的问题是 CPU 间应如何通信。很明显，这里也需要一些互连方案，但是由于它只用于 CPU 和 CPU 之间的通信，通信量要比当互连网络用于 CPU 和存储器之间的通信量低几个数量级。

在图 3 中，可以看到一个基于总线的多计算机。它的拓扑结构尽管看起来与基于总线的多处理机很相似，但是因为其上的通信量很少，所以并不需要高速的总线。事实上，它可能是更低速的 LAN（相比于 300 Mbps 和需要背板总线支持的速率而言，它典型的速率只有 10～100 Mbps）。因此，图 3 更通常的是一个 LAN 上工作站的集合，而不是插入高速总线的 CPU 卡的集合（尽管后一种结构一定是一种可能的设计）。

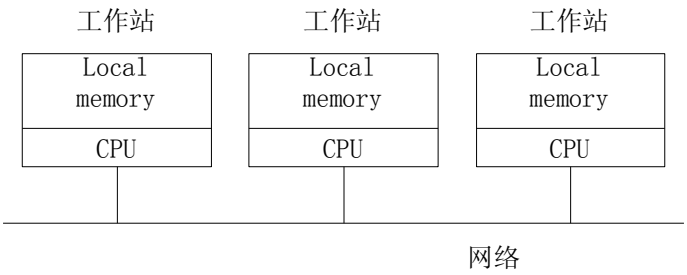

图 3 局域网上由多台工作站组成的多计算机系统

4. 交换型多计算机

最后一个类别是交换型多计算机。虽然人们已经提出并建立了各种互连网络，但是它们都有一个特性，就是每个 CPU 都直接的而且排他的访问自己的私有存储器。图 4 给出了两个流行的拓扑结构：网格和超立方体。网格结构很容易理解，它被布置在印刷电路板上。它们最适用于解决固有特性为两维的问题，例如图像理论或视觉（即机器人的眼睛或分析照片）。

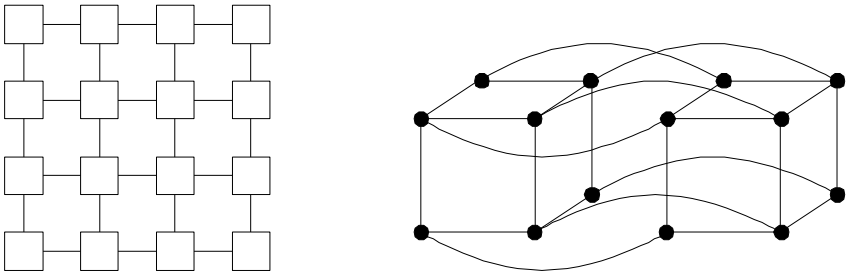

图 4 两个流行的拓扑结构

（a）网格　（b）超立方体

超立方体(hypercube)是一种 n 维立方体。图 4(b)中的超立方体是四维的。它可被想象成两个普通的立方体，每个立方体有 8 个顶点和 12 条边。每个顶点都代表一个 CPU，每条边代表两个 CPU 间的连接。两个立方体中相应的顶点是相连的。要把超立方体扩展为五维，就要在图形中再加另外两个互连的立方体，连接两部分中相应的边，其他的情况也可以此类推。对于一个 n 维的超立方体，每个 CPU 都有 n 个连接与其他 CPU 相连。因此，布线的复杂性按 CPU 数量的对数增加。因为只有最相邻的CPU是互连的，许多信息在到达终点前必须进行 n 次跳跃。然而，最长的可能路径也是以 CPU 数量的对数增加的，而不像网格中按 CPU 数量的平方根增加。有 1024 个 CPU 的超立方体在几年前就已经在商业上应用了，而具有 16,384 个 CPU 的超立方体也即将可以进行商业应用。