第一篇　分布式数据库系统基础

第一章　分布式处理系统

1.1 引言

分布式处理系统代表了数据处理领域中发展最快的一个分支，它迎合了大学和研究环境、商品化厂家以及军事方面各种用户的需要。大家都对"分布处理"系统提出了性能方面的种种要求，下面仅列出其中的一部分：

高系统性能、快速响应、高吞吐能力；

高可用性；

高可靠性；

降低网络费用；

故障软化能力、故障时降级使用；

易于模块化、逐步增长和结构灵活；

资源共享；

自动负载均分；

工作负载变化时的高适应性；

硬件和软件组成部分能逐步替换或升级；

易于扩充容量与功能；

易于适应新的功能要求；

对瞬时过载的良好响应，等等。

根据多处理机系统的当前水平是很难达到上述这些目标的。为此，分布式处理系统必须要有某些新的东西，从而开辟了广阔的研究领域。

本章将讨论仍处于研究阶段的这种新型系统的基本特性。

在一个系统中至少有四个物理组成部分才可以"分布"开来：硬件或逻辑线路，数据，处理本身以及控制(例如操作系统)。有一种说法认为只要有上述四种之一是分布的话，该系统就是分布式系统。但是，只根据系统的某些部分的物理分布来下定义是注定要失败的。正确的定义还必须包含这些分布部分互相作用的概念。存在物理分布而不认为是分布处理系统的例子如有的系统组成只是把输入/输出处理硬件和功能在物理上进行分布。同样，如果处理硬件没有分布则处理本身一定也没分布，并且反过来，把硬件分布而处理不分布也是难以想象的。

我们这里对分布处理系统的定义指的是能实现上面所列主要要求的"新"的系统，因而可以认为是一种"研究与发展"的定义。分布处理系统的这个定义有五个组成部分：

1. 通用资源部分(包括物理资源和逻辑资源这两方面)的**重复性**，可以在动态基础上把它们分配给具体的任务。物理资源是否匀质（homogeneity）在这里不是本质的。

2. 系统的这些物理部件和逻辑部件在物理上是分布的，并且通过通讯网络进行交互作用。

3. 有一高级的或高层的操作系统来对分布的各组成部分进行统一而综合的控制。各个处理机都有自己本地的操作系统，而且可以是各不相同的。

4. 系统透明性，即可以只用名字来请求种种服务，而不需要指出服务者。

5. 协作性自治 (cooperative autonomy)，表示了物理资源和逻辑资源的操作和交互作用的特征。

在许多系统中只是在不同程度上具有这些性质和工作特性，因而只能提供前面所列优点之中的一部分。只有把全部这些准则结合起来，才能够唯一地定义分布式处理系统。

1.2 重复性

在系统中，如果提供服务的可分配资源没有重复性的话，那么在整个系统中就不能有处理的分布。前面定义中"通用"这个词常常用来描述这些"可分配的"资源，例如通用计算机系统，或通用处理机等。系统要能在任何给定时间动态地重构那些资源以提供指定的服务。资源的这种重构或重分配必须能不影响不直接涉及的那些资源的工作。如果系统的目的是提供一种需要通用处理机的服务的话，那么该系统必须要有多个重复的通用处理机。如果系统的目的是满足某种别的功能(例如事务处理或在线控制)的话，那么该系统可以满足所有这五个准则，而可分配的硬件资源可以在广义上认为是专用的。但是，在该系统内它们是通用的和可分配的。一般情况下，系统同时具有通用和专用(固定功能)两种成分，或许专用成分中的某些部分是供专门使用的(不可分配的)。

一个极端情况是系统中每种资源只有一个，它不满足上述定义的第一项准则；而另一极端是每种资源都有重复，在这两个极端之间可以有许多的变型，这些结构中有的资源只有一个，而有些资源却有重复多个。因为在不完全重复的系统内，可自由地分配的多个重复资源的管理几乎与完全重复的情况一样困难，而且所提供的好处也几乎相同，所以下面的一般讨论同时包括了这两类系统。

为了达到高可用性、整个系统的可靠性和故障软化的目的，多个资源的可用性和有效地利用它们的能力是很重要的。这些特性也直接有助于灵活性、适应性和逐步扩充的能力。

1.3 物理分布与互相通讯

虽然可以有许多方法来定义一个网络，可以考虑它的许多不同的方面，但是在这里最重要的一个问题是信息的传送。两个物理资源之间协同工作的一个最好的例子就是通过网络进行信息的物理传送。这种传送遵从一种协议，其中通讯双方必须协作才能成功地完成传送。这种传送机构与门控式传送不同，后者主控者完全有权来迫使受控者物理上接受一报文。但在双方协作的协议中，目的地可以回送一指示信息（例如"未准备好"、"忙"、或"未确认"）来物理上拒绝接受这个报文。信息传送的门控或主从控制方法会妨碍自治式工作(后面会讨论)，而所有资源之间的高度自治对于系统的高度可用性是很重要的。

为了支持进程的自治操作概念，必须把"网络"相互通讯的概念从物理资源推广到逻

辑资源的信息传送，例如逻辑资源之间的状态、服务请求和同步等。

双方协作协议的最重要一个特性就是它允许任何资源，不管是物理资源或逻辑资源，根据它对自己状态的了解来拒绝或接受一个报文的传送。网络的这个定义对于互连路径的长度没有任何限制，它们甚至可以是在单个集成电路芯片中两个资源之间的极短连线。这个定义的本质性特征是利用了一种双方协作的协议。

1.4 系统工作的统一性

为了把一分布处理系统中的各物理的和逻辑的组成部分集成为一功能性的整体，必须要实现一高层的操作系统。各个处理机可以有自己的操作系统，但是还要有一组完善定义的策略来管理整个系统的集成操作。实现这些策略的机构可以是与每个个别的操作系统相同的，也可以只是本地策略的逻辑扩充。在高层操作系统和本地操作系统之间必须不存在任何强烈的（strong）层次结构，因为那样会违反自治操作的准则。此外，本地操作系统不一定要是匀质的，虽然存在各种各样的接口肯定会使系统设计问题复杂化。

正如有人指出的那样，在一群独立工作的计算机和一群计算机集成成工作平稳而价格可取的分布式处理系统之间的主要差别是在系统软件上。

一个分布式系统具有多个控制地点和多个处理活动。为了满足我们定义中的全部五个准则，必须同时存在这些重复性以及控制地点的动态变化。此外，控制地点或处理活动在物理上的约束必须减至最小。对于每个处理机来说很重要的操作系统的核心必须减至最小，并且整个系统控制的地点在运行时必须是动态的。固定性约束的程度必须减至最小，并且系统必须没有任何"临界的路径"或"临界的部件"，例如对一资源的任何一个拷贝的固定性约束以包含全局的状态信息。在固定指派承担系统控制功能的一资源故障或过载时系统的运行性能不得受到严重影响或极大地降级。对于处理活动也有类似的要求，虽然有些情况下专用的资源可能要求固定束缚于一种处理功能（例如数据库或某一 I/O 设备的专门传送）。

这个高层操作系统必须具有的其他特征为：处理机之间的有效传送带宽比一般简单地通讯的计算机之间的带宽高；一处理机从该网络断开时仍能继续工作并有效地利用其本地的资源，并且能容易地重新连接和结合入系统而不影响正在进行的其他操作；能支持在"细微"级上的进程和处理机的交互作用（当需要时）。

这个高层操作系统不管是有一组独立的代码还是仅为一种设计思想都能把可用的资源从一简单的硬件集合转变为一能有条理地进行工作的系统。显然，高层操作系统的设计是整个系统设计中的关键，它提出了一系列的问题要求进行广泛的深入研究来解决它们。后面还会谈到其中的一些问题。

1.5 系统透明性

如果任一系统能够提供在引言中所列的能力的话，那它提供给用户的界面（或接口）必须是一种服务而不是服务者（服务器）。用户必须能用指定要做什么事的方法来请求一个动作，而不要求说明提供此服务的物理部件或逻辑部件。

分布式系统的一个重要特性是它的存在对用户是完全透明的，除非用户为了某些特殊的效率问题而自愿去了解它。用户应该能够象与单一个集中式系统进行通讯那样来开

发程序和处理数据库的操作。事实上，分布式系统的用户接口必须要甚至比当前的集中式系统的还要简单。这主要是因为用户是与高层操作系统进行通讯的；而这个控制软件的一个功能是处理系统中提供的所有各种命令语言和数据定义。非均质网络系统的初期工作已经强烈地推动更好的(甚至是标准化的)操作系统命令语言的研究。由于分布系统的组成对用户完全透明，所以系统的状态信息也都看不到，因而用户不可能知道当前哪些资源可用，或最好用哪个资源来执行此任务。因此，用户必须能用服务的名字而不用指出服务者来请求服务。如果有的应用以指定的资源来服务较好的话，则用户也应能请求这种形式的服务。

分布式系统用户接口的性质决定了这种系统最重要特性之一，即由一分布处理系统提供的服务也可以由一单处理机系统来提供，只要存在必要的硬件并可如此组织的话。但是，这样一种单处理机系统，不能够提供分布处理系统中象可靠性、适应性和模块化等的其他优点。

1.6 协作自治性

协作自治性（cooperative autonomy）是最后一个、或许是最重要的一个要素。分布处理系统必须设计得所有组成部分或资源(物理的或逻辑的)的操作都有极高程度的自治性。在物理级上，这可以利用网络的传输协议来实现，其中报文的传输需要发送者和接收者两方面的协调动作。在逻辑级上，在进程之间也必须存在同样程度的协作。再者，任何资源即使在它接受了该物理报文以后仍要能拒绝该服务的请求。这是由于在此系统内部控制是没有任何层次结构的。

但是，这不是无政府的混乱状况。所有的组成部分遵循一"总计划"（master plan），它反映在高层操作系统的思想之中。这种工作模式应称为协作自治性而不是简单的自治性。为了获得前面列出的许多好处，在所有组成部分之间的高度自治是极重要的，只有满足了定义中的全部五个准则的要求，在系统操作和组成部分交互作用方面才能具有这一特性。

1.7 某些排除在外的系统

在新的系统设计中已经进行了大量的工作来获得引言中所列的部分优点，但是极少有系统在满足所有这些准则方面得到较大的进展。

我们定义中的大多数准则是否满足可用在一特定方向上与一阈值相交来表示。这个定义不是一组二进制的判据，通过研究被此定义排除的一些系统能够对这些准则和它们的阈值有更好的了解。

例如，它排除了在一台主机内部的分布。某些现代处理机系统在结构上的特点是包含了独立的I/O通道，有人认为这些通道是协作性的分布式处理机，因为它包含了独立的I/O处理机、运算逻辑处理机和可能有的诊断处理机。这样一种分类方法很少用处而且也未被广泛接受。显然，在这类系统组织中各个组成部分固定地束缚于某些任务。

控制与主机进行通讯的前端处理机肯定不属于这里定义的分布式系统。虽然它可能满足了某些准则，但它也是专用于一种功能而且不能自由地进行分配。

在硬件和软件控制中有很多主从式的例子。关键之点是传送信息的接收者不能决定

它是否要接受这次传送并按此进行动作。在硬件控制中，这个概念叫做门控式传送（gated transfer）。在软件控制系统中，在多计算机操作系统和基本的多处理机操作系统中经常会遇到主从式关系的问题。但是，由于不是执行协作性协议，都不属于分布处理系统。

随着硬件价格的不断下降，人们对新的多处理机系统的兴趣日益增加，这种系统组织中包含了象向量乘法器、浮点运算部件、或快速富里哀变换部件等专用的功能部件。在操作的一般概念上，这种专门功能的处理与主从关系只稍有不同而已。主要差别在于主从控制关系也从我们的定义中排除了许多包含多个通用处理部件的硬件系统。对于这些结构造成术语上混淆的原因是这些专门的服务常常是由一通用部件（例如可编程微处理机）来提供的。这个功能部件可以用一微程序来使其“专门化”，或者它完全是通用的，不过在一专门的功能地位中使用而已，例如在一大型系统中使用一通用小型机来控制输入/输出。这一类被排除的系统的区别性特点是资源专用于某一个或某一组功能。只要在控制它自己的活动时，它就工作在主从模式，这就违反了自由分配和自治性这两条准则。

单台宿主处理机带有许多收集和发送数据的远程终端并不认为是一分布处理系统，即使这些终端是智能的或还可做某些编辑和格式化的工作。这个结论除了某些推销或广告人员以外是普遍公认的。

甚至在一复杂的网络互连结构中存在多个宿主机也不一定使得系统成为分布式的。从信息交换的观点看它可以是分布式的；但是从整个操作和控制的观点来看，它一般仍是集中式的。这些系统在硬件出故障时并没有动态重新分配任务的能力。

在广告中常常把智能终端系统叫做分布处理系统。但是，必须仔细地研究具有智能终端或本地处理机的系统的工作才能判断其处理在什么程度上真正是分布的。这种系统由接至一本地处理机的若干终端组成，该本地处理机还有磁盘、磁带等外部存贮能力。它提供智能式的数据录入——通过运行本地处理机中的程序来进行现场编辑和类似的功能。它能进行共享的文件访问，但只限于本地的文件。它可与一主处理机进行通讯，不过这时的本地处理机要模仿一“笨”的终端来使用一般的协议。最后，它还能进行远程作业录入。但是没有表明控制功能有任何的分布，因为工作的分布是固定的，本地的终端不能来影响它。

一个终端带有驻留的文本编辑器（不管它是由硬件还是由软件提供的）不是分布处理系统的例子。为了满足定义的要求，这个终端必须首先要能做某些真正的工作，其次要能知道自己不能完成所分配的任务时把它传递给另一合适的服务部件。当本级的利用比较充分而简单地把工作卸载给一更高级别的能力就正是过渡至完全分布处理的开始。如果这个终端能够不要人的干预自动地协调几个并发的远程作业，在不同地点给每个作业提供不同类型的服务的话，那么它就更接近于一分布系统。当本地的控制系统能够根据本地负载和能力的分析来决定某项工作应该在本地完成或者传递给系统中其他部分时就达到了这个准则的阈值。分布式处理肯定不只是相当于“把设备搬至一系统的周围以便在数据源处获得和处理数据。

智能终端在分布处理系统的发展中或许确实起了一定的作用。它能方便于“无痛苦”地过渡至对于硬件、数据存贮和控制来说更为分散的组织形式。其实现的方法就是在建立高层系统联系和完整的全局功能以前给本地系统增加一些特点和作某些改进以增加本地的功能。

1.8 表征分布的维数

从系统实现的观点来看，可以使用三维来表征一个系统的分散程度，也就可以用它来定义分布式系统。这三维分别代表硬件组成、控制点组织和数据库组织上的分散程度。

沿着硬件组成的方向可以指出若干点来，按增加分散度的次序，它们分别如下：

1. 具有一个控制器、一个运算器（ALU）和一个中央存贮器的单一中央处理机。

2. 多个执行部件：系统含有一个控制器、多个相同的运算器（例如 Illiac IV 中的处理单元），以及或许还有多个独立的中央存贮器。

3. 独立的专用功能部件：系统含有一个通用控制器和多个 ALU 或处理部件，在处理部件中某些可以是专用部件，例如通道或 I/O 处理机、快速富里哀变换处理机、向量部件、或浮点运算部件。与这种专用功能部件相联系的可以有一些附加的控制部件，但是它们是固定的或能力有限的控制部件。与此同时，附加的 ALU 中有一些可以是相同的通用部件。

4. 多处理机：系统由多个控制器、多个 ALU（通用部件）以及或许有多个独立的中央存储器组成，但是只有一个合作的输入/输出系统。

5. 多计算机：每个计算机包含一通用的 CPU（控制器、ALU 和中央存储器）以及输入/输出系统。

除了最后两种组成以外，五个一般准则排除了其他所有组织形式归为分布式系统这一类的可能性。

与此相似，控制组织也是一个坐标，按照分散度增加的次序可分出下列各点：

1. 单个固定控制点，或者是物理的或者是概念上的。

2. 固定的主从关系：可以有多级的主从关系，而从属者之间的关系可以是非对称的。重要之点是这种主从关系是固定的，直到由完全是外部的动作来修改以前保持不变。

3. 动态的主从关系，可由软件来修改。

4. 多个（或者重复的）系统控制地点完全自治地工作。一个例子是两个独立的计算机仅在 I/O 级通过传送完整的文件来交互作用。

5. 在执行一任务时多个控制点协同工作，这个任务已被分成若干子任务。

6. 在执行一任务时重复的、相同的控制点协同工作。

7. 在执行一任务时多个控制点（不一定是匀质的）完全协同工作。

前四种排除在分布式处理之外。

沿着数据库轴上的各点的特征和次序就不那么简单，这些点的正确次序也不明显。我们在这里给出一种次序；但是，分布式数据库的各种组织的使用和维护目前尚未充分了解，本书的后面部分将在这方面进行较深入的探讨。

数据库本身有两个成份可被分布，即文件和对那些文件进行编目的目录。这两个之中哪一个都可被分布而不管另一个怎样，虽然某些组合很明显是无意义的或不实际的。除了分布的问题以外，还有一个文件与（或）目录的重复性问题。在我们图中轴上各点是已经真正实现的或已在实现上作了周密考虑的一些概念：

1. 集中式数据库，在文件及目录上只有单一拷贝放在外存储器中。

2. 同上，不过是放在主存中。

3. 分布式文件而有单一个集中式目录，且无本地目录。所有的访问都必须通过该单一个中央目录来处理。

4. 重复的数据库，在每个处理节点有一份全部文件的完整拷贝以及一份系统的目录。

5. 分片的数据库（partitioned data base），每个节点最直接需要的数据就放在该节点，而在其他节点则视需要存放其拷贝，但在主节点上则存放全部文件的一份完整的重复拷贝。在这种安排下，在一本地节点的任何事务处理发生例外情况时（即数据不在本地时）将由该主节点来进行处理。

6. 分片的数据库，本地数据（文件和目录）都保存在处理的节点，不在主节点重复存放。但是，主节点中保持一完整的目录，可以为引起本地例外情况的事务指引路由。

7. 分片的数据库，但没有任何主文件或主目录——对于一指定文件的位置来说没有任何单一的信息源。

上述头上三种数据库组织表现了一个公共的特性，即它们对于数据库的任何访问来说都只有单一个入口点——单一个集中式目录。显然，这种情况直接影响到系统的脆弱性（vulnerability）、可靠性和吞吐能力。

另一种表征数据库的方法是把数据的命名和存放作为不同的维。这也可能有几种组合，例如一公共名字空间（文件目录）和数据本身的分布式存储，与此相反的是在单一的共享存储设备上保持各完全分开的名字空间。

在前面给出的定义中，并考虑到这三维上排除的区域，就可以在一三维空间中表示出分布式系统所占的位置，如图 1-1 所示。

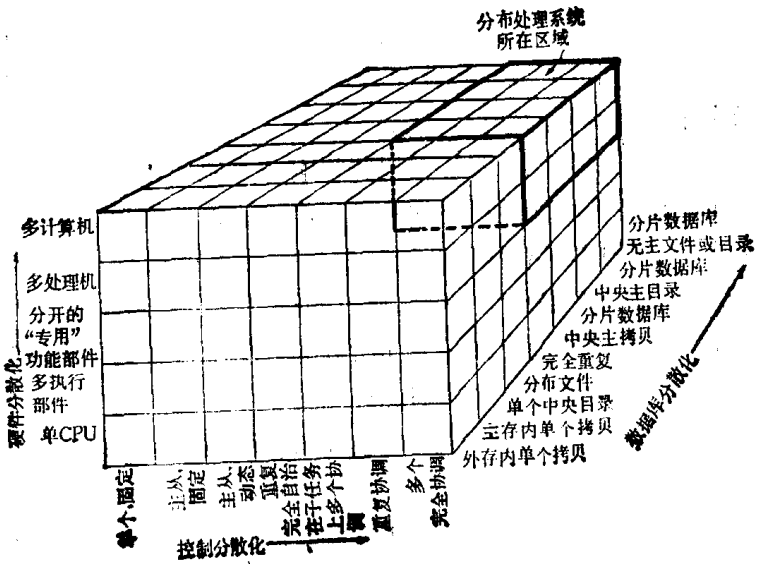


图 1.1　表征分布性的三维图

1.9　高层操作系统

高层操作系统是分布处理系统的一个关键因素，它的设计必须要考虑几个特性和问题。

操作系统的经典设计假定有大量系统的信息可用。虽然有关用户提出的工作的信息

是否完整和有效是有疑问的，但是一般假定操作系统能取到其工作环境的完整而准确的信息。但是，在分布处理系统中就不是这种情况；这时永远也得不到关于该系统的完整的信息。资源提供一个服务，但是它们可能有意无意地不让外界来检查其信息。

在分布系统中，在收集关于系统组成部分的状态信息时总有一定的时间延迟。这些时间延迟的不一致是极为重要的。在一般的集中式处理机中，操作系统可以请求状态信息，而保证被询问的部件不会改变状态而等待根据这些状态信息所作的决定，因为只有这一个询问的操作系统能发出命令。但在分布处理系统中，发生的时间滞后可能很显著；结果，由于自治的部件按自己的路径向前进行，所以可能发送出不准确的(过期的)信息。如果你曾使用过输入/输出设备处理程序的话，那你肯定常常要问所获得的信息是否准确。因此，对于分布处理系统来说，必须要设计成甚至在错误或不准确状态信息情况下也仍能工作。

关于系统信息有效性方面进一步复杂化的原因是提供给各系统控制器的信息可能不相同。这种差别可能是时间延迟和不同控制器信息屏蔽上的差别造成的。在时间和空间这两方面缺乏唯一性这一点具有非常重要的影响。

1.10 一般控制问题

这里所说的高层操作系统是高度非层次结构的，这就是说，它是单层的并且没有任何内部的主从关系。这一特性再加上部件的自治性，就大大恶化了控制问题。即使多个自治的部件正在协同工作，同时发生冲突的动作的可能性要比分层次系统中的高得多。而且，由于存在显著的时间延迟，对系统中各个控制器的动作进行同步要困难得多。最后，系统内死锁或无限循环的问题与其他系统的情况很不相同。有些建议请求一仲裁人(外界的第三者)来解决这个问题；但是，这种仲裁人必须是瞬时的，因为如果存在一固定的仲裁人的话就表示有了不可接受的分层控制。

从分布处理系统的工作特性来看，可以得出有关系统通讯的一些结论。我们定义的第二个准则要求在进程之间和处理机之间的通讯中，不论是物理的或逻辑的所有传送都要采用报文式协议。其中不得有任何全局变量，在系统部件之间也不得有任何直通"隧道"。所有参数都要通过完善定义而且严格执行的接口。

在单处理机和多处理机环境中通讯方面所做的大量工作在这里也可应用，但是要考虑到分布系统中系统部件的自治性质来加以扩充。

用户必须使用只包含服务名的命令来与系统进行通讯。我们关于系统透明性的准则使得用户指定一系统部件来提供所需的服务是不必要的或者不可能的。然而，这个要求引起了系统故障和用户差错检测方面的新问题，因为没有一个处理机能够确定所请求的服务是否可在系统中某个地方来提供，或者甚至只是确定这个服务是否合法也不可能。

分布处理系统中的资源管理是一多维的任务。因此至今对专用于分布系统的资源管理方面很少进行研究。但是，低层的功能与单处理机中执行的非常相似，例如物理资源的分配，以及在把进程安排在一具体系统部件上以后所需的各种设施的管理。然而，在能这样进行以前可能先要把所需的资源在一个地点汇集起来，或者建立起链接机构以便远程使用这些资源。在此过程中要涉及的问题是对资源进行定位，决定哪些部件是合适的，以及决定把资源移至所选地点的最佳方法。在较高层上是调度问题，即决定何时应启动或

终结一个功能。

采用单层的自治控制的任何系统在系统调度方面都会出现全新的问题。在非层次系统中对服务的请求可能一开始就被所有的物理资源拒绝。在那种情况下，此提出服务请求的实体可能要对此新的请求和当前正执行的任务进行相对优先级的评价。高效地执行这个过程是高层操作系统最重要的功能之一。

当把所有这些问题及解决办法与单处理机系统中遇到的类似问题及解决办法进行比较时，看来使分布系统控制问题复杂化的主要因素是分布处理系统内部的通讯，它相对于功能的详细执行来说是异步的，并且它在通讯处理本身的时间以外还表现出一定的时间延迟。单处理机使用信号灯、标志、加锁门或超时来解决许多问题。但要在相当复杂的分布系统中使用这些方法需要的时间太多，因而大大降低系统的吞吐能力。要记住，在一般网络中发送信号灯的传送时间大约在100毫秒数量级上。除了降低运行性能以外，单处理机中的大多数办法的可靠性和健壮性（robustness）也是有疑问的，因为象TEST AND SET（测试置位）那样的系统操作不能够象单一不可分割的机器指令那样在下一机器周期中立即执行。

时间问题进一步复杂化的原因在于，作为解决传送时间引起的困难的大多数办法（例如选举和软件同步）甚至要求系统中每个部件进行更多的处理。

小结

分布处理系统是一类新的组织和操作，它在所有各维中都表现出高度的分布性，并且在其整个操作和交互作用中具有高度的协作自治性。已经设计了大量的系统来满足分布处理系统定义中的一个或几个准则；但是在我们能够实现一真正的和完善的分布系统以前仍旧还有大量的课题需要加以研究解决。

参考文献

[1.1] P. H. Enslow, Jr, "What is a 'Distributed' Data Processing System?" COMPUTER, Jan. 1978, pp. 13—21.

[1.2] G. A. Champine, "Six Approaches to Distributed Data Bases", DATAMATION, May 1977, pp. 69—72.

[1.3] S. A. Kallis, Jr. "Networks and Distributed Processing", MINIMICRO SYSTEMS, March 1977, pp. 32—40.

[1.4] Arthur Lynch, "Distributed Processing Solves Mainframe Problems", DATA COMMUNICATIONS, Dec. 1976, pp. 17—22.

[1.5] H. Lorin, Aspects of Distributed Computer Systems, Wiley, 1981.

[1.6] G. M. Booth, The Distributed System Environment, McGraw-Hill, 1981.

第二章　分布式数据库概述

近年来，分布式数据库已经成为信息处理中的一个重要领域，并且不难预见，它的重要性还将迅速增加。造成这种趋势有组织上和技术上的两方面原因。分布式数据库消除了集中式数据库的许多缺点，并且很自然地适合于许多单位的分散型的结构。

分布式数据库的典型而有点模糊的定义如下：分布式数据库是一个数据集合，这些数据在逻辑上属于同一个系统，但实际上又分散在一计算机网络的若干站点上。这个定义强调了分布式数据库的两个重要的特点：

1. **分布性**，也就是说，这些数据不是驻留在同一处(处理机)的，所以我们能把分布式数据库与单一的集中式数据库区别开来；
2. **逻辑关联性**，也即这些数据具有某些把它们联系在一起的性质，因此我们能把分布式数据库与驻留在计算机网络不同站点上的一组本地数据库区别开来。

上述定义中存在的问题是，根据分布性和逻辑关联性来区别哪些是分布式数据库、哪些则不是，在很多情况下是太含糊了。为了制定更明确的定义，我们先来看几个例子。

例 2.1　我们考虑一个在不同地方有三个分行的银行情况。在每个分行用一台计算机来控制其出纳终端和帐目数据库（图 2.1）。每个分行的计算机及其本地的帐目数据库组成了此分布式数据库的一个站点，而用通讯网络把这些计算机连接起来。正常工作时，分行终端所要求执行的应用只需要访问该分行的数据库就够了，而这些应用完全由提出要求的那台分行计算机来执行，因此我们称为本地应用。本地应用的一个例子为借贷应用，它只对提出要求执行的同一分行中存储的帐目进行操作。

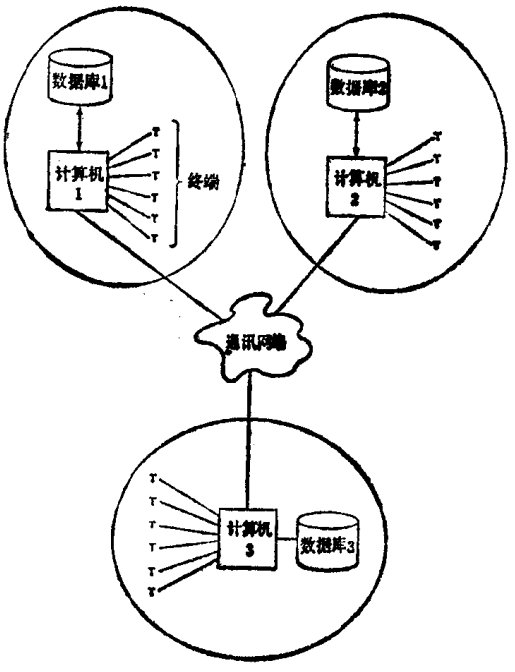


图 2.1　在地域分散的网络上的分布式数据库

如果我们要把分布式数据库的定义应用于上述情况，那么会发现很难说明是否具有逻辑关联性。各分行只涉及自己的帐目信息是否已经足够？上述情况应认为是分布式数据库呢还是仅仅为一组本地数据库？

为了回答这些问题，我们集中研究一下本地数据库集合与分布式数据库有什么不同。从技术观点来看，关键问题是这里有些应用要从一个以上的分行存取数据。这种应用叫做全局应用或分布式应用。可以把它们是否存在作为区别分布式数据库与本地数据库集合的一个重要判据。

全局应用的一个典型例子是把存款从一个分行的帐目转移至另一分行的帐目中，这

要求在两个不同的分行中更新数据库。要注意，这种情况与在两个分行中分别执行本地的更新(例如一个借款,一个贷款)不同,因为在全局应用情况下必须确保两个更新操作或者都执行、或者都不执行。以后我们会看到,要确保这一点是一困难的任务。

在例 2.1 中，计算机分放在地理上相距较远的地方,但是，分布式数据库也可以建立在范围较小的局域网上。

例 2.2 假设前例中的同一银行，具有相同的应用程序，但是其系统配置如图 2.2 所示。把各分行的计算机和数据库移至一公共的大楼中,并且用一高带宽的局域网连接起来,各分行的出纳终端则通过电话线连至相应的计算机上。这样，每台处理机及其数据库组成了该计算机局域网的一个站点。

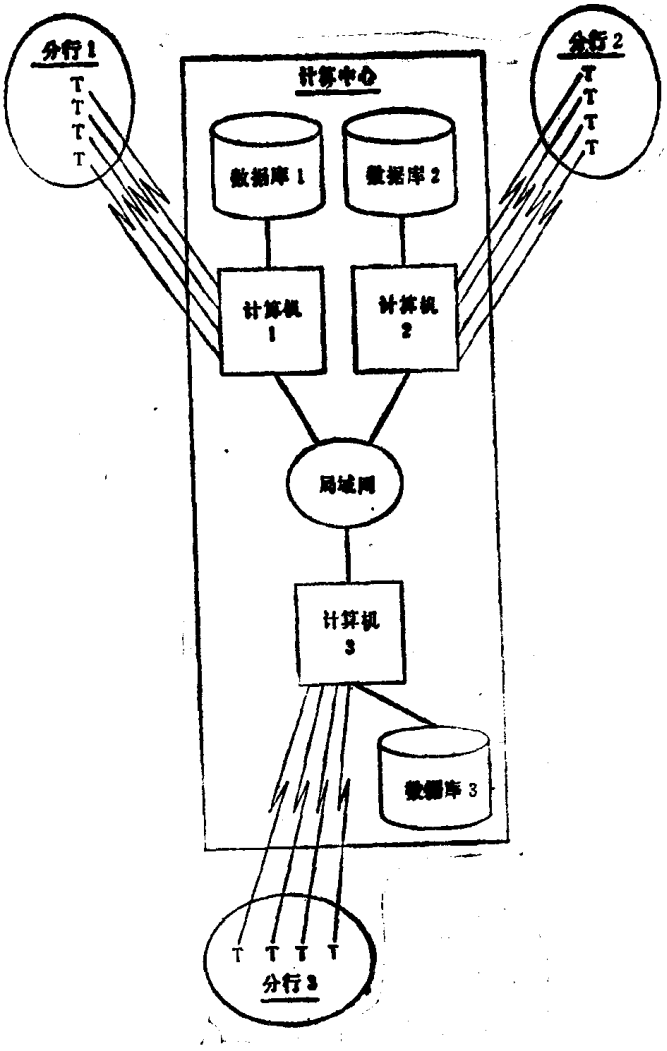


图 2.2 局域网上的分布式数据库

我们可以看出，连接的物理结构已与例 2.1 不同，但是体系结构的特点仍保留不变。特别是,同样的一些计算机执行同样的一些应用,访问同样的一些数据库。如果我们所谓的"本地"是指只涉及到一台计算机及其数据库的话，那么上例中的本地应用在本例中也仍是本地的。

如果存在全局应用，那么可以容易地认为本例是一分布式数据库，因为上例的大部分特征仍旧有效。不过，这个分布式数据库是实现在局域网而不是广域网之上的，由于局域网的吞吐量和可靠性高得多，因此有些情况下某些问题的解决方法会有所不同。

现在我们来考虑一个例子，它在本书中不认为是分布式数据库。

例 2.3 假设上例中同一个银行，但其系统配置如图 2.3 所示。不同分行的数据分散在三台"后端"计算机中，由这些机器来执行数据库管理功能。应用程序则由另一台计算机来执行，当它需要时向后端机请求访问数据库的服务。

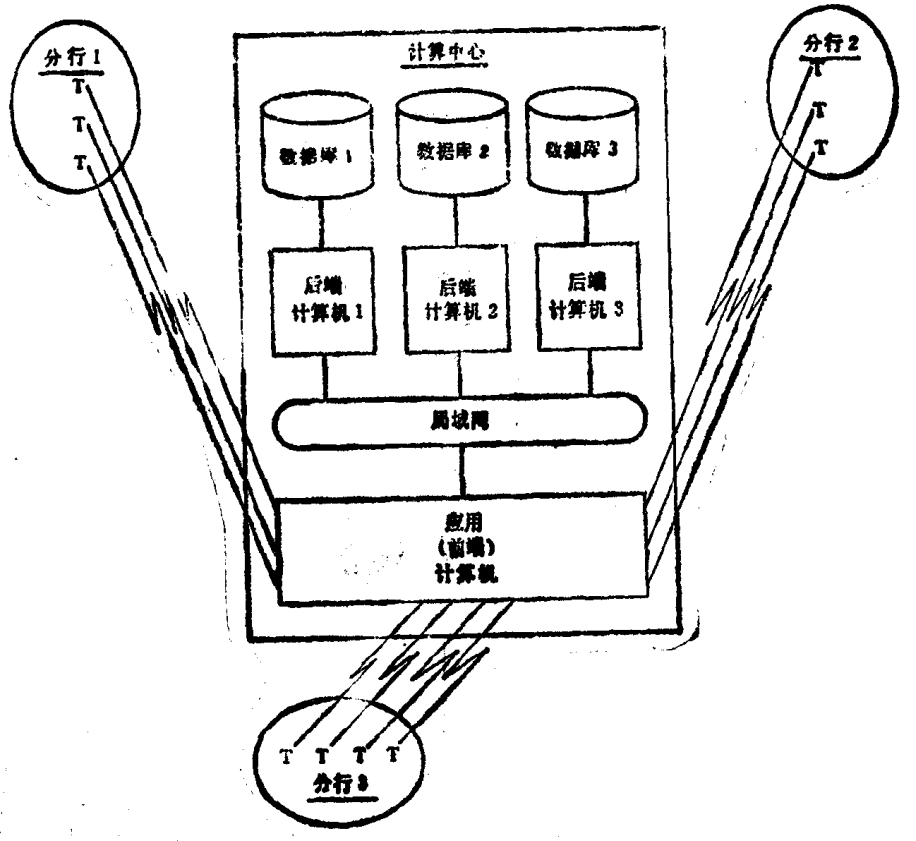


图 2.3 多处理机系统

认为这个系统不是分布式数据库的理由是，虽然数据物理上是分布在不同的处理机中，但是它们的分布并不是从应用的观点来看的。这里不存在本地应用的概念，因为这个系统的几台计算机结合在一起，没有任何一台机器能够只靠自己来执行一个应用。

现在我们把从上面一些例子得到的考虑总结出来，导出了目前使用的分布式数据库的定义如下：

分布式数据库是一个数据集合，这些数据分布在一计算机网络的不同的计算机中。此网络的每个站点具有自治的处理能力，并且能执行本地的应用。每个站点的计算机还至少参与一个全局应用的执行，这种应用要求使用通讯子系统来在几个站点存取数据。分布式数据库中最重要的技术问题就是从"在自治的站点之间协同工作"这一点产生的。上面的定义特别强调了这一特点。

2.1 分布式数据库与集中式数据库的特点

分布式数据库不只是简单地把集中式数据库分布开来，因为分布式数据库系统的设计还可以提供与传统的集中式数据库不同的特点。因此，探讨传统数据库的典型特点，并把它们与分布式数据库相应的特点进行比较是有益的。传统数据库方法的特点是：集中控制，数据独立，减少冗余，复杂的物理结构以提高存取效率，完整性，可恢复性，并发控制，私用性，和安全性。

集中控制 对整个企业或单位的信息资源提供集中的控制是引入数据库的最强有力的推动因素之一，实际上数据库就是从每个应用具有自己私用文件的那种信息系统演进而来的。**数据库管理员**（DBA）的基本职能就是保证数据的安全性；这些数据本身被认为是要求集中管理的企业的一个重要投资。

在分布式数据库中，并不强调集中控制的思想。这也与体系结构有关，从例 2.1 和 2.2 可以看到这一点。或许看起来例 2.2 比例 2.1 更适宜于集中控制。一般来说，分布式数据库具有分层控制的结构，也就是说，由**全局数据库管理员**来担负管理整个数据库的中央职责，而由若干**本地数据库管理员**来分别负责管理各自的本地数据库。但是要强调指出，本地数据库管理员可以具有很高的自治权，甚至可以完全不要全局数据库管理员，而由本地数据库管理员他们自己来进行站点之间的协调。通常把这个特性叫作**站点自治性**。各分布式数据库在站点自治性的程度上可以有很大的不同：从不需要任何集中的数据库管理员的站点完全自治，到几乎完全是集中控制。

数据独立性 数据独立性的要求也是引入数据库方法的主要推动力之一。从本质上来说，数据独立的意思是数据的实际组织对应用程序员是透明的。在编写程序时是根据数据的"概念性"结构，叫做**概念模式**。数据独立的主要优点是当数据的物理组织改变时不会影响到应用程序。

在分布式数据库中，数据独立性和在传统的数据库中同样重要，不过要在通常的数据独立性概念中增加一个新的问题，即**分布透明性**。所谓分布透明性的意思是，可以认为数据库不是分布的那样来编写程序。因此，把数据从一个站点移至另一站点不会影响程序的正确性；但是执行的速度会受到影响。

在传统的数据库中获得数据独立性的方法是采用多层的体系结构，每层具有不同的数据描述，然后在它们之间进行转换。为此发展出概念模式、存贮模式和外部模式等一些概念。与此相似，在分布式数据库中是靠引入新的层次和模式来得到分布透明性的。在第三章中将描述获得分布透明性的一些方法。

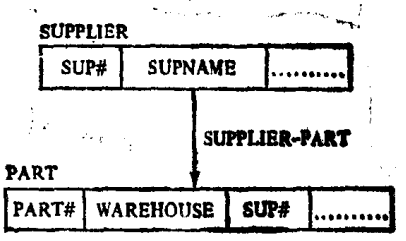
减少冗余 在传统的数据库中，要尽可能地减少冗余度，这有两个原因：第一，同一份逻辑数据的几个拷贝存在不一致性问题，如果只有一份拷贝就自然避免了这一问题；第二，消除了冗余数据也就节省了存贮空间。减少冗余的方法是共享数据，也就是说，允许几个应用访问同一些文件和记录。

但是，在分布式数据库中，有一些理由认为希望数据有冗余。第一，如果在需要那些数据的各站点都有拷贝的话，就可以提高应用的"本地性"；其次，系统的可用性得到提高，因为如果数据有冗余的话，那么一个站点出故障不会妨碍其他站点中应用的执行。一般来说，传统数据库中反对冗余的理由也同样存在，因此，确定最佳的冗余度需要进行相当

复杂的权衡与评价。可以笼统地说，重复一个数据项带来的方便程度随着应用执行的读取访问与更新访问之比的增加而增加。因为一个数据项如果有几份拷贝，读取时只要对一份拷贝进行即可，而更新时则必须对所有拷贝连贯地执行更新才行。在第五章讨论分布式数据库的设计时将更仔细地研究这方面的问题。

复杂的物理结构与存取效率 象二级索引、文件之间的链接等复杂的存取结构是传统数据库的一个主要问题。数据库管理系统（DBMS）的最重要的部分就是支持这些复杂的结构，其目的是为了提高存取数据的效率。

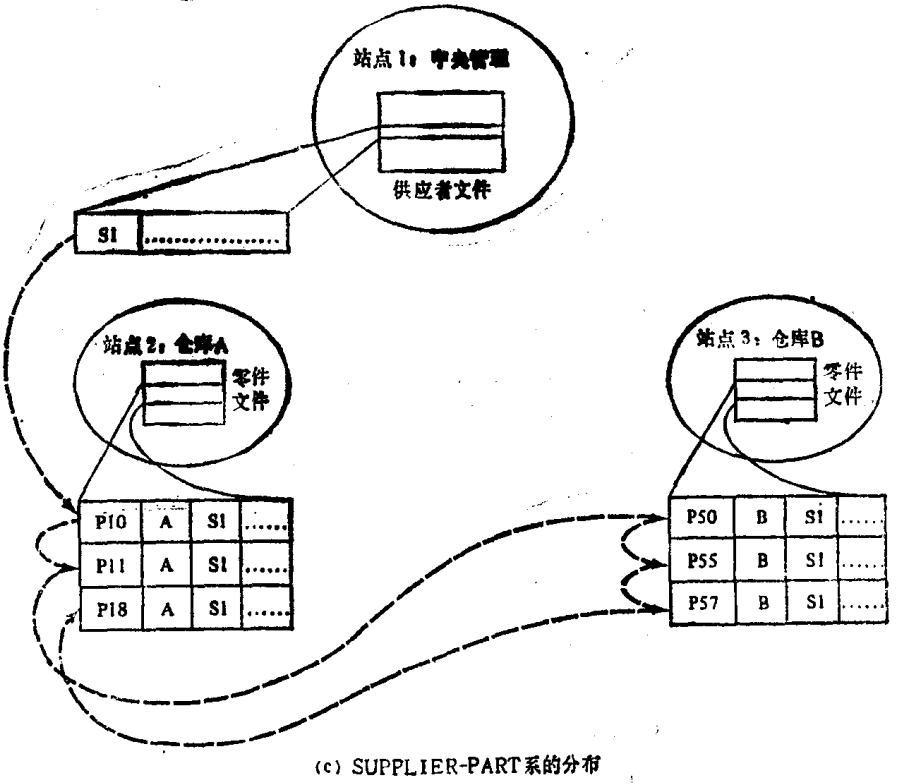
在分布式数据库中，复杂的存取结构并不是提高存取效率的正确方法。因此，当存取效率是分布式数据库的一个主要问题时，物理结构不是一个切中要害的技术问题。采用站点之间的物理结构的方法并不能提高分布式数据库的存取效率，因为这种结构很难建



(a) 一个Codasyl 数据库模式

```
Find SUPPLIER record with SUP# = S1;
Repeat until "no more members in set"
    Find next PART record in SUPPLIER-PART set;
Output PART record;
```

(b) 用于寻找由S1提供的零件的Codasyl-DBMS式程序



(c) SUPPLIER-PART系的分布

图 2.4 一个分布式 Codasyl 式数据库

立及维持，并且分布式数据库中在记录级的“航行”（navigate）是不方便的。我们用一个例子来说明这一点。

例 2.4 考虑图 2.4(a) 所示的 Codasyl 式数据库的模式。其中有两个记录类型 SUPPLIER（供货者）和 PART（零件）及一个系类型 SUPPLIER-PART，它把供货者记录与由其供应的零件记录连接起来。应用用 Codasyl 式 DML 形式写出“找出 SI 供应的全部 PART 记录”的程序如图 2.4(b) 所示。

现在我们假定上述数据库分布在一计算机网络的三个站点上，如图 2.4(c) 所示：供货者文件放在站点 1（中央管理），而零件文件分为两个不同的子文件存放在站点 2 与 3（仓库）。再假定这是 Codasyl 系统的分布式实现，所以我们能够在分布式数据库上运行图 2.4(b) 的同一个（航行的）程序。假定在站点 1 上运行此应用，显然，每一次 “repeat-until” 循环的迭代都要去访问一远程的 PART 记录。因此，每次访问一记录时不仅要传送此记录本身，还必须交换某些报文。

这个应用的更高效的实现方法是尽可能地把所有远程访问分组，如图 2.5 所示。请比较图 2.4(b) 和图 2.5 中的两个程序，前者中的命令 “find next” 要求一个个记录地访问，而后者中的 “find all” 把在同一站点处执行的所有访问归成一组。因此，图 2.5 中所示的过程由两类操作组成：在单个站点处程序的本地执行以及站点之间的文件传输。上面这种过程叫做**分布式访问计划**（distributed access plan）。

1) 在站点 1
把供应者号SN发送给站点 2 和 3
2) 在站点 2 和 3
在接受供应者号时平行执行下述程序：
Find all PARTS records having
SUP # = SN;
Send result to site 1.
3) 在站点 1
从站点 2 和 3 合并结果；
输出结果

图 2.5 访问计划例

分布式访问计划可以由程序员来编写，也可以用优化程序来自动产生。分布式访问计划的编写和集中式数据库中的航行编程在下述意义上有点相似，就是说由程序员指定应如何来访问数据库。但是，应该在记录组的级别上执行站点之间的航行，而通常一次一个记录的航行则可在每个站点的本地处理中执行。因此，航行性语言不如非过程性、面向系的语言更适合于建立访问计划。

对于能自动产生象图 2.5 所示的访问计划的优化程序来说，它的设计中仍有若干问题需要解决。为方便起见可以把这些问题分为两大类：**全局优化**和**本地优化**。全局优化要决定必须在哪些站点访问哪些数据，以及因而在站点之间必须传送哪些数据文件。全局优化的主要优化参数是通讯花费，有些情况还应考虑访问本地数据库的花费。这些因素的相对重要性决定于通讯花费与磁盘访问花费之比，这个比值又与通讯网络的类型有关。

本地优化要决定在每个站点如何来执行本地数据库的访问。本地优化的问题是传统的非分布式数据库的典型情况，所以本书将不作进一步的讨论。

第六章和第七章专门讨论在分布式数据库上执行应用情况下制订全局访问计划的问题。全局优化的研究已经产生了一些结果，即使不是自动产生访问计划这些结果也是很有用的，因为它有助于了解怎样才能提高分布式数据库的存取效率。

完整性、可恢复性和并发控制 在数据库中，完整性、可恢复性和并发控制虽然是各不相同的问题，但是关系却非常密切。在很大程度上，这些问题的解决办法是提出了事务的概念。所谓**事务**（transaction）是一个不可分割的执行单位，叫做**原子单位**（atomic unit），也就是说，它是一个这样的操作序列，要么完整地执行，要么完全不执行。在例 2.1 中所说的"存款转移"程序是一个全局的应用，它必须是一个原子单位，即借款部分和贷款部分要么两者都执行，要么都不执行，不允许只执行两者之一。因此，这个存款转移应用也是一个全局事务。

显然，分布式数据库中事务的不可分割性问题特别有趣：当要求转移存款时，如果"借款"站点正常工作而"贷款"站点不能工作的话，系统应如何行动呢？是否此事务应该中止（即取消在站点出现故障之前已经完成的全部操作），或者是否应巧妙地设计系统使得即使两个站点永不同时正常工作情况下也能正确地执行存款的转移？当然，如果采用后一方案，则故障对用户只有很小的影响。

显而易见，事务的不可分割是获得数据库完整性的一种方法，因为它保证使数据库从一个确定的状态变换至另一状态的全部动作要么都被执行，要么保留数据库的原来确定状态不变。

事务的不可分割性有两个危险的敌人：故障和并发。故障可能会使得系统停止在事务执行的中间，因而破坏了不可分割的要求。不同事务的并发执行可能会使得一个事务观察到另一事务执行期间所产生的不明确的暂时状态。

所谓恢复，在很大程度上是处理故障时保护事务不可分割性的问题的。在分布式数据库中这个问题特别重要，因为如前例所示，事务执行中所涉及到的某些站点可能出故障。分布式数据库的恢复问题将在第十章中讨论。

并发控制的任务是在事务的并发执行时确保事务的不可分割性。这个问题可以看作是典型的同步问题。在分布式数据库中，和在所有分布式系统中一样，同步问题要比集中式系统中的困难。这个问题将在第九章中探讨。

私用性和安全性 在传统的数据库中，具有集中控制权的数据库管理员能够保证只让有权的人来存取数据。但是要注意，集中式数据库本身如果没有专门的控制规则的话，要比基于分开文件的老方法更容易破坏其私用性和安全性。

在分布式数据库中，本地管理员所面临的问题在本质上是和传统数据库的管理员所面临的问题相同的。但是，分布式数据库有两个独特的问题值得说一下：首先，在站点自治程度很高的分布式数据库中，本地数据的拥有者感到比较安全，因为他们能够加强自己的保护措施而不必依赖于中央的数据库管理员；其次，一般来说，安全问题对分布系统来说是固有的，因为通讯网络代表了保护措施方面的一个薄弱点。分布式数据库的私用性和安全性问题将在第十一章中讨论。

2.2 为何要用分布式数据库

为什么要发展分布式数据库有多方面的理由，下面是一些主要的推动力。

组织上和经济上的理由 许多单位在组织上是分散的，因而分布式数据库的方法更自然地适合这种组织结构。分布式组织结构和相应的信息系统是一些著作和论文的主题。随着计算机技术的最新进展，对大型、集中式计算机中心是否经济产生了疑问。这里我们不再进一步讨论这方面的问题，但是，组织上的和经济方面的推动力或许是发展分布式数据库的最重要的原因。

现有数据库的互连 当在一个单位中已经存在有若干数据库，而又需要执行全局应用程序时，分布式数据库是自然的一个解决办法。这种情况下，分布式数据库是从现存的本地数据库开始由底向上地建立的。这个建立过程可能要求本地数据库有一定程度的重新构造，但是，其工作量要比建立一个全新的集中式数据库小得多。

逐步增长 如果一个单位通过增加一些新的相对独立的小单位(新的分行、新的仓库等)而逐步增长的话，那么分布式数据库的方案能够适应于这种逐步增长而对已存在的小单位影响最小。如果采用集中式方案，一个办法是系统的初始规模就考虑到将来的扩充，但这是难以预见而且实现起来花费较大的；否则这种增长将不仅对新的应用程序而且对现存的程序产生很大的影响。

减小通讯开销 在如图 2.1 所示的那种地域上分散的分布式数据库中，事实上许多应用都是本地的，这样比起集中式数据库来可以显著地减小通讯的开销。因此，在分布式数据库的设计中，尽量提高应用的本地性是主要的目标之一。

运行性能的要求 由于存在若干自治的处理机，它们可有高度的平行处理能力，因此提高了运行的性能。这种考虑不仅适用于分布式数据库，也可适用于任何多处理机系统。例如，例 2.3 的结构也允许平行处理，但它不是分布式数据库。但是，分布式数据库的优点在于数据的分解反映了与应用有关的准则，因而提高了应用的本地性；这样，不同处理机之间的互相干扰被减至最小。不同处理机之间得到了负载均分，并且避免了象通讯网络本身或整个系统公共服务那样的瓶颈现象。这是在分布式数据库定义中所说的要求自治处理能力的结果。

可靠性和可用性 分布式数据库的方法，特别是具有冗余数据时，可用来获得较高的可靠性和可用性。但是，达到这个目的不是很简单的，而是要求采用一些至今仍未完全掌握的技术。不同站点的自治处理能力本身并不能保证提高系统的整个可靠性，但是它提供了优越的降级使用能力。换句话说，分布式数据库中由于有大量的元件而要比集中式数据库的故障频繁一些，但是每个故障的影响被限制于使用故障站点数据的那些应用，而整个系统的崩溃是极罕见的。关于建立可靠的分布式数据库的方法将在第十章中介绍。

上面这些对分布式数据库的推动力不是新产生的，那么为什么分布式数据库系统的发展还刚开始呢？这里有两个主要原因：第一，小型计算机的最近进展，能够以较低的价格来提供过去由大型机提供的许多能力，这是发展分布式信息系统所必需的硬件支持。第二，分布式数据库技术是建立在另外两种技术之上的，即：计算机网络与数据库技术，它们在七十年代中才建立起牢固的基础。在计算机网络和各站点的一组本地数据库管理系统之上建立一个分布式数据库是一个很复杂的任务，没有这些基础是不可能实现的。

2.3 分布式数据库管理系统（DDBMS）

分布式数据库管理系统的任务是支持分布式数据库的建立和维护。在分析 DDBMS