第 7 章　软件、服务和算法技术

学习任务

本章主要涉及：

1. 环境感知型中间件
2. 嵌入式软件
3. 微型操作系统

学习任务

本章主要涉及：

4 面向服务架构

5 物联网海量数据存储与查询

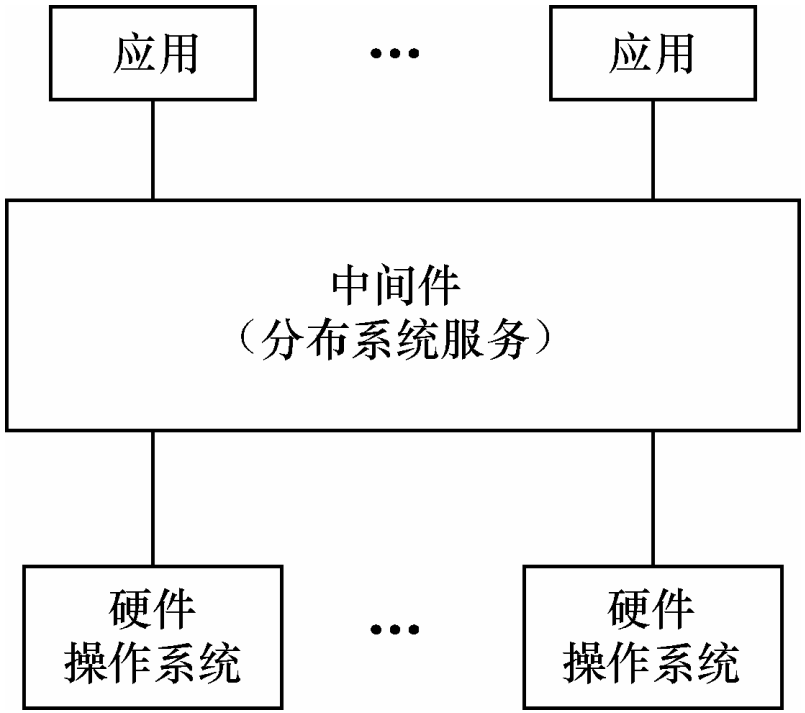
6 物联网数据融合及路由

7.1.1 中间件概述

- 中间件(middleware)是一类连接软件组件和应用的计算机软件，它包括一组服务，以便于运行在一台或多台机器上的多个软件通过网络进行交互。
- 该技术所提供的互操作性，推动了一致分布式体系架构的演进。该架构通常用于支持分布式应用程序并简化其复杂度，它包括web服务器、事务监控器和消息队列软件。

7.1.1 中间件概述

中间件



7.1.1 中间件概述

- 中间件在操作系统、网络和数据库之上，应用软件的下层。
- 简单地讲，中间件是一种独立的系统软件或服务程序，分布式应用软件借助这种软件在不同的技术之间共享资源，中间件位于客户机服务器的操作系统之上，管理计算资源和网络通信。

7.1.1 中间件概述

操作系统、数据库管理系统、中间件的类比

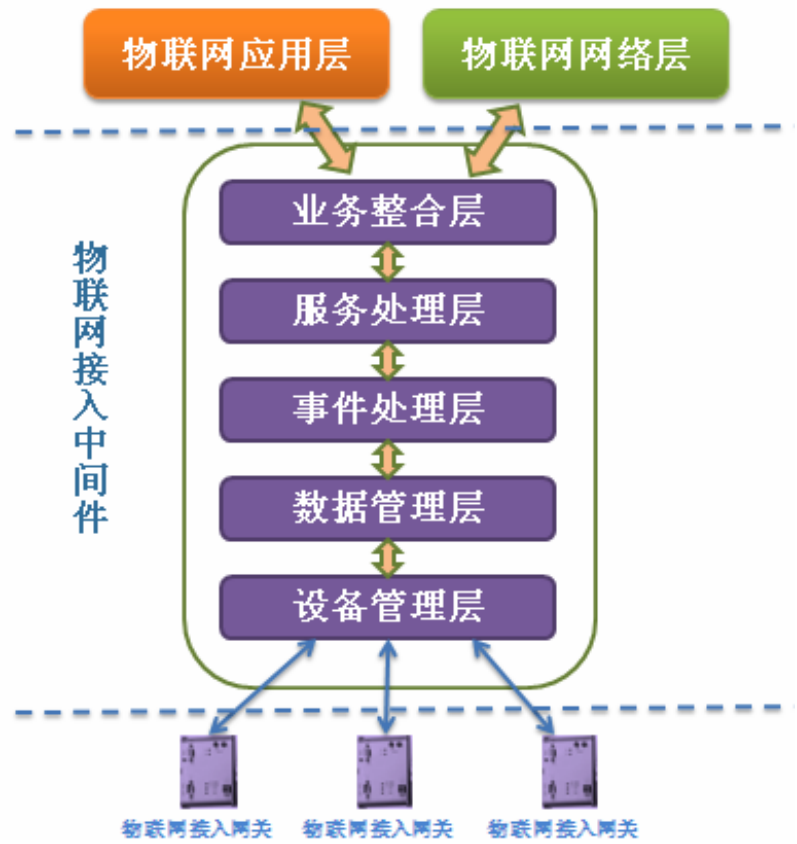
	操作系统	数据库管理系统	中间件
产生动因	硬件过于复杂	数据操作过于复杂	网络环境过于复杂
主要作用	管理各种资源	组织各类数据	支持不同的交互模式
主要理论基础	各种调度算法	各种数据模型	各种协议、接口定义方式
产品形态	不同的操作系统 功能类似	不同的数据库管理系统 功能类似， 但类型比操作系统多	存在大量不同种类中间件 产品， 它们的功能差别较大

7.1.1 中间件概述

- 中间件的核心作用是通过管理计算资源和网络通信，为各类分布式应用软件共享资源提供支撑。
- 广义地看，中间件的总体作用是为处于自己上层的应用软件提供运行与开发的环境，帮助用户灵活、高效地开发和集成复杂的应用软件。

7.1.2 中间件的体系框架与核心模块

- 在物联网中采用中间件技术，以实现多个系统和多种技术之间的资源共享，最终组成一个服务系统。

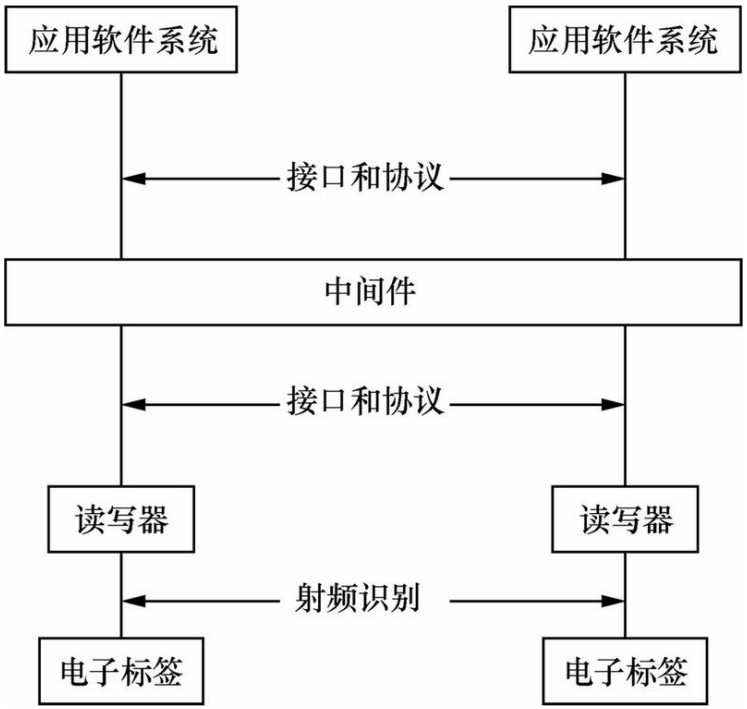


7.1.3 中间件的分类

① 企业服务总线（Enterprise Service Bus）
② 事务处理（Transaction Processing）监控器
③ 分布式计算环境（Distributed Computing Environment）
④ 远程过程调用（Remote Procedure Call）
⑤ 对象请求代理（Object Request Broker）
⑥ 数据库访问中间件（Database Access Middleware）
⑦ 信息传递（Message Passing）
⑧ 基于 XML 的中间件（XML-Based Middleware）

7.1.4 物联网中间件的设计

- 目前，物联网中间件最主要的代表是RFID中间件，其他的还有嵌入式中间件、数字电视中间件、通用中间件、M2M物联网中间件等。



7.1.2 中间件的体系框架与核心模块

- RFID中间件扮演RFID标签和应用程序之间的中介角色，从应用程序端使用中间件所提供一组通用的应用程序接口（API），即能连到RFID读写器，读取RFID标签数据。
- 这样一来，即使存储RFID标签数据的数据库软件或后端应用程序增加或改由其他软件取代，或者读写RFID读写器种类增加等情况发生时，应用端不需修改也能处理，省去多对多连接的维护复杂性问题。

7.2 嵌入式软件

- 嵌入式软件就是嵌入在硬件中的操作系统和开发工具软件，它在产业中的关联关系体现为：芯片设计制造→嵌入式系统软件→嵌入式电子设备开发、制造。

7.2.1嵌入式系统

1. 嵌入式系统的定义

- 嵌入式系统是指用于执行独立功能的专用计算机系统。
- 它由包括微处理器、定时器、 微控制器、存储器、传感器等一系列微电子芯片与器件，和嵌入在存储器中的微型操作系统、控制应用软件组成，
- 共同实现诸如实时控制、监视、管理、移动计算、数据处理等各种自动化处理任务。

7.2.1嵌入式系统

2. 嵌入式操作系统

目前流行的嵌入式操作系统可以分为两类：

- 一类是从运行在个人电脑上的操作系统向下移植到嵌入式系统中，形成的嵌入式操作系统，如微软公司的Windows CE 及其新版本，SUN 公司的Java 操作系统，朗讯科技公司的Inferno，嵌入式Linux 等。

7.2.1嵌入式系统

- 另一类是实时操作系统，如**WindRiver** 公司的**VxWorks**，**ISI** 的**pSOS**，**QNX** 系统软件公司的**QNX**，**ATI** 的**Nucleus**，中国科学院凯思集团的**Hopen** 嵌入式操作系统等，
- 这类产品在操作系统的结构和实现上都针对所面向的应用领域，对实时性高可靠性等进行了精巧的设计，而且提供了独立而完备的系统开发和测试工具，较多地应用在军用产品和工业控制等领域中。

7.2.2 嵌入式软件的应用

1. 概述

- 嵌入式软件与嵌入式系统是密不可分的，嵌入式系统是“控制、监视或者辅助设备、机器和车间运行的装置”，
- 一般由嵌入式微处理器、外围硬件设备、嵌入式操作系统以及用户的应用程序等四个部分组成，
- 用于实现对其他设备的控制、监视或管理等功能。

7.2.2 嵌入式软件的应用

- 而嵌入式软件就是基于嵌入式系统设计的软件，它也是计算机软件的一种，
- 同样由程序及其文档组成，可细分成系统软件、支撑软件、应用软件三类，是嵌入式系统的重要组成部分。

7.2.2 嵌入式软件的应用

2. 应用

- 我们常见的移动电话、掌上电脑、数码相机、机顶盒、MP3等都是用嵌入式软件技术对传统产品进行智能化改造的结果。
- 嵌入式软件在中国的定位应该集中在国防工业和工业控制、消费电子、通信产业。
- 首先一个市场是数字电视市场。
- 第二个市场是移动通信市场。
- 第三个市场是掌上电脑（PDA）。

7.2.3 嵌入式软件的分类

1. 嵌入式操作系统

- 嵌入式操作系统EOS（Embedded Operating System）是一种用途广泛的系统软件，
- 负责嵌入系统的全部软、硬件资源的分配、调度工作，控制、协调并发活动，它必须体现其所在系统的特征，能够通过装卸某些模块来达到系统所要求的功能。

7.2.3 嵌入式软件的分类

- 现在国际上有名的嵌入式操作系统有Windows CE 、Palm OS 、Linux 、VxWorks 、pSOS、QNX、OS-9 、LynxOS等，已进入我国市场的国外产品有WindRiver、Microsoft、QNX和Nuclear等。
- 我国嵌入式操作系统的起步较晚，国内此类产品主要是基于自主版权的Linux操作系统，其中以中软Linux、红旗Linux、东方Linux为代表。

7.2.3 嵌入式软件的分类

2. 嵌入式支撑软件

- 支撑软件是用于帮助和支持软件开发的软件，通常包括数据库和开发工具，其中以数据库最为重要。
- 嵌入式数据库技术已得到广泛的应用，随着移动通信技术的进步，人们对移动数据处理提出了更高的要求，嵌入式数据库技术已经得到了学术、工业、军事、民用部门等各方面的重视。

7.2.3 嵌入式软件的分类

3. 嵌入式应用软件

- 嵌入式应用软件是针对特定应用领域，基于某一固定的硬件平台，用来达到用户预期目标的计算机软件。由于用户任务可能有时间和精度上的要求，因此有些嵌入式应用软件需要特定嵌入式操作系统的支持。
- 嵌入式应用软件和普通应用软件有一定的区别，它不仅要求其准确性、安全性和稳定性等方面能够满足实际应用的需要，而且还要尽可能地进行优化，以减少对系统资源的消耗，降低硬件成本。

7.2.4 嵌入式软件发展趋势

- 嵌入式系统被描述为：“以应用为中心、软件硬件可裁剪的、适应应用系统对功能、可靠性、成本、体积、功耗等严格综合性要求的专用计算机系统”，由嵌入式硬件和嵌入式软件两部分组成。
- 硬件是支撑，软件是灵魂，几乎所有的嵌入式产品中都需要嵌入式软件来提供灵活多样、而且应用特制的功能。

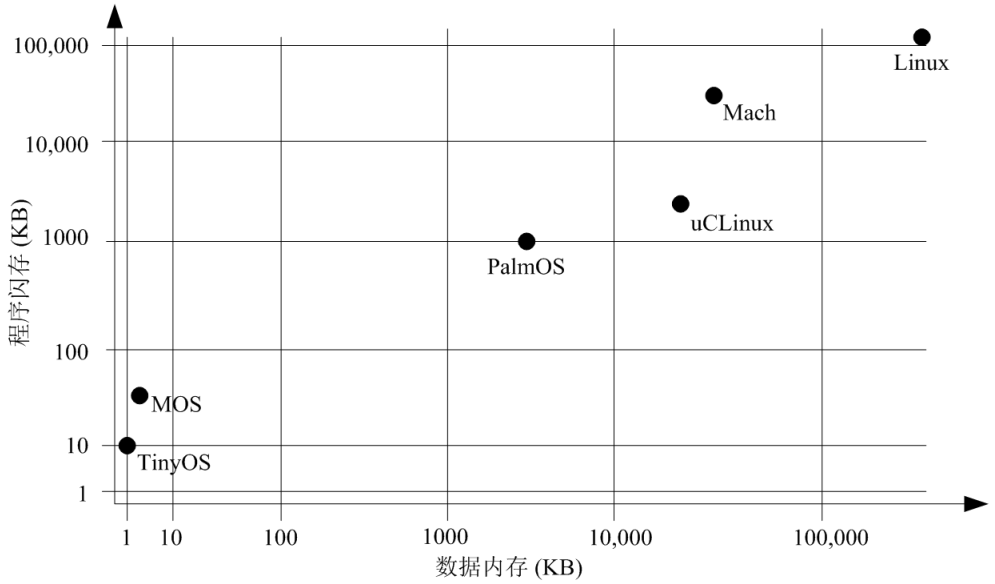
7.2.4 嵌入式软件发展趋势

- 目前的因特网技术只联接了5%左右的计算装置，大量的嵌入式设备急需网络连接来提升其服务能力和应用价值。
- 同时，以人为中心的普适计算技术正推动新一轮的信息技术的革命。计算无所不在，嵌入式设备将以各种形态分布在人类的生存环境中，提供更加人性化、自然化的服务。
- 互联网的"深度"联网和普适计算"纵向"普及所带来的计算挑战，将推动嵌入式软件技术向"纵深"发展，催生了新型嵌入式软件技术。

7.3.1 传感器节点微型操作系统

（1）节点操作系统的发展

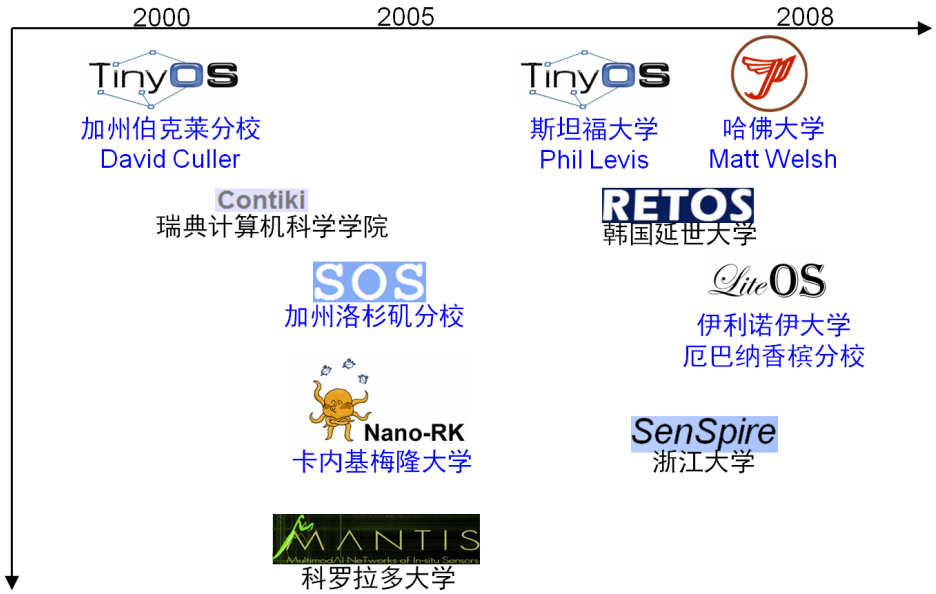
- 操作系统是传感器节点软件系统的核心，为适应传感器网络的特殊环境，节点操作系统与其它使用在计算机或服务器上的操作系统有极大的区别。



节点操作系统VS其他操作系统

7.3.1 传感器节点微型操作系统

- 由上图可清楚地看出，节点操作系统是极其微型化的。节点操作系统的发展近年来取得快速发展，主要发展过程请参看下图。



节点操作系统发展史

7.3.1 传感器节点微型操作系统

（2）TinyOS

- TinyOS由加州伯克莱分校开发,是目前无线传感网络研究领域使用最为广泛的节点操作系统 。TinyOS使用的开发语言是：nesC
- nesC语言是专门为资源极其受限、硬件平台多样化的传感节点设计的开发语言，使用nesC编写的应用程序是基于组件的，组件之间的交互必须通过使用接口。

7.3.1 传感器节点微型操作系统

常用微型节点操作系统对比

	TinyOS	Contiki	SOS	Mantis	Nano-RK	RETOS	LiteOS
发表会议(年份)	ASPLOS (2000)	EmNets (2004)	MobiSys (2005)	MONET (2005)	RTSS (2005)	IPSN (2007)	IPSN (2008)
静态/动态	静态	动态	动态	动态	静态	动态	动态
事件驱动/多线程	事件驱动 & 多线程 TOSThreads	事件驱动 & 多线程	事件驱动	多线程 & 事件驱动 TinyMOS	多线程	多线程	多线程
单核/模块化	单核	模块化	模块化	模块化	单核	模块化	模块化
网络层	主动消息	uIP, uIPv6, Rime	消息	“comm”层	套接字	三层架构	
实时支持	否	否	否	否	是	符合 POSIX 1003.1b	否
语言支持	nesC	C	C	C	C	C	LiteC++

7.3.2 其它常见微型操作系统

1. WinPE

- Windows预先安装环境（Microsoft Windows Preinstallation Environment，简称Windows PE或WinPE）
- 是简化版的Windows XP、Windows Server 2003或Windows Vista。
- WinPE是以光盘或其他可携设备作媒介。

7.3.2 其它常见微型操作系统

2. MenuetOS

- MenuetOS是英国软件工程师Ville Mikael Turjanmaa开发的，完全由x86汇编语言于2000年写成的一款开放源码的32位操作系统。
- 最新的版本可以从其官方网站下载。全部使用汇编语言

7.3.2 其它常见微型操作系统

3. SkyOS

- SkyOS拥有现代操作系统要求的多处理器支持，虚拟内存，多任务多线程等等功能，更令人耳目一新的是它漂亮的GUI系统SkyGI。
- 首个SkyOS系统于1997年底发布。
- SkyOS操作系统并不开放源代码，收费并且用户不可以自由地获取

7.3.2 其它常见微型操作系统

4. ReactOS

- **ReactOS® 完全兼容 Windows® XP 的操作系统。**
- **ReactOS 旨在通过使用类似构架和提供完整公共接口实现与 NT 以及 XP 操作系统二进制下的应用程序和驱动设备的完全兼容。**
- **简单地说，ReactOS 目标就是用您的硬件设备去运行您的应用程序。**
- **任何人都可以免费使用的 FOSS 操作系统！！**

7.3.2 其它常见微型操作系统

5. TriangleOS

- TriangleOS是荷兰人Wim Cools用C和汇编写出来的32位操作系统。

6. Visopsys

- Visopsys由加拿大人Andrew McLaughlin开发，有独特的GUI，开放源码。最新的Visopsys可以从其官方网站下载。

7.3.2 其它常见微型操作系统

7. Storm OS

- Storm OS是由立陶宛的Thunder于2002年开始开发的，有简单的GUI,装在一张软盘上。

8. 实验室中的操作系统

- 这些系统多由高校中的实验室开发，作试验研究之用，如德国的DROPS等，

7.4 面向服务架构

- 面向服务的体系结构（service-oriented architecture，SOA）是一个组件模型，它将应用程序的不同功能单元（称为服务）通过这些服务之间定义良好的接口和契约联系起来。
- 接口是采用中立的方式进行定义的，它应该独立于实现服务的硬件平台、操作系统和编程语言。这使得构建在各种这样的系统中的服务可以以一种统一和通用的方式进行交互。

7.4.1 面向服务架构简介

- 在SOA架构风格中，服务是最核心的抽象手段，业务被划分(组件化)为一系列粗粒度的业务服务和业务流程。业务服务相对独立、自包含、可重用，由一个或者多个分布的系统所实现，而业务流程由服务组装而来。
- 一个"服务"定义了一个与业务功能或业务数据相关的接口，以及约束这个接口的契约，如服务质量要求、业务规则、安全性要求、法律法规的遵循、关键业绩指标(Key Performance Indicator，KPI)等。

7.4.1 面向服务架构简介

- 接口和契约采用中立、基于标准的方式进行定义，它独立于实现服务的硬件平台、操作系统和编程语言。
- 这使得构建在不同系统中的服务可以以一种统一的和通用的方式进行交互、相互理解。
- 除了这种不依赖于特定技术的中立特性，通过服务注册库(Service Registry)加上企业服务总线(Enterprise Service Bus)来支持动态查询、定位、路由和中介(Mediation)的能力，使得服务之间的交互是动态的，位置是透明的。

7.4.1 面向服务架构简介

- SOA架构带来的另一个重要观点是业务驱动IT。以粗粒度的业务服务为基础来对业务建模，会产生更加简洁的业务和系统视图;
- 以服务为基础来实现的IT系统更灵活、更易于重用、更好(也更快)地应对变化;
- 以服务为基础，通过显式地定义、描述、实现和管理业务层次的粗粒度服务(包括业务流程)，提供了业务模型和相关IT实现之间更好的"可追溯性"，使得业务的变化更容易传递到IT。

7.4.1 面向服务架构简介

SOA的主要优点：

- **IT能够更好更快地提供业务价值(Business Centric)、**
- **快速应变能力(Flexibility)、**
- **重用(Reusability)。**

7.4.2 面向服务架构的特征

一般认为SOA具有以下五个特征：

1、可重用

2、松耦合

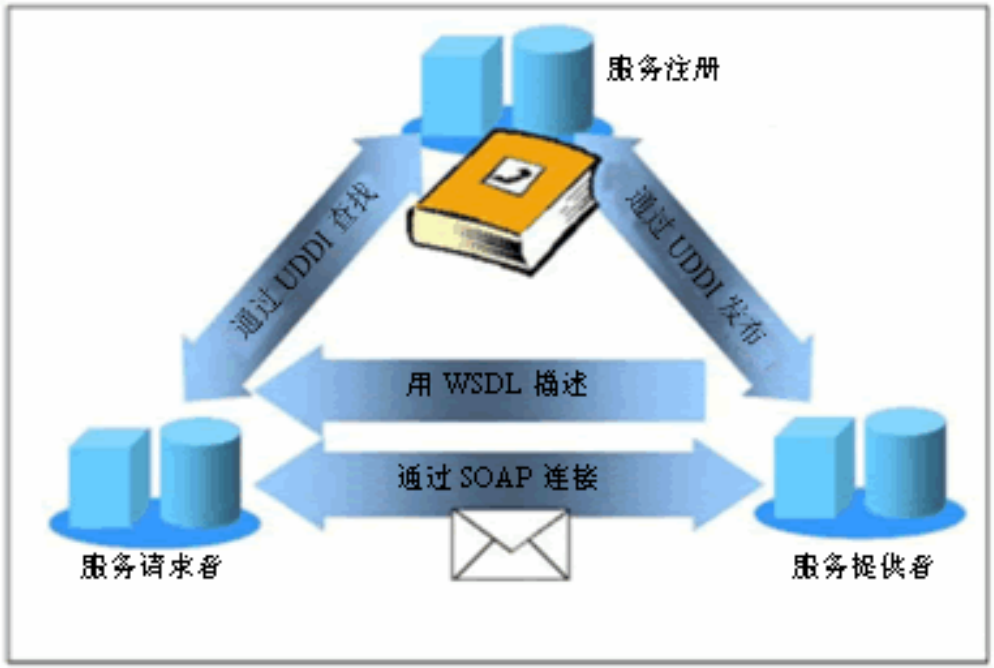
3、明确定义的接口

4、无状态的服务设计

5、基于开放标准：

7.4.3 元素

- 面向服务的体系结构中的角色包括：服务使用者、服务提供者和服务注册中心，如下图所示：



7.4.3 元素

1、服务使用者：

- 服务使用者是一个应用程序、一个软件模块或需要一个服务的另一个服务。
- 它发起对注册中心中的服务的查询，通过传输绑定服务，并且执行服务功能。
- 服务使用者根据接口契约来执行服务。

7.4.3 元素

2、服务提供者：

- 服务提供者是一个可通过网络寻址的实体，它接受和执行来自使用者的请求。
- 它将自己的服务和接口契约发布到服务注册中心，以便服务使用者可以发现和访问该服务。

7.4.3 元素

3、服务注册中心：

- 服务注册中心是服务发现的支持者。
- 它包含一个可用服务的存储库，并允许感兴趣的服务使用者查找服务提供者接口。

7.4.3 元素

- 每个实体都扮演着服务提供者、使用者和注册中心这三种角色中的某一种（或多种）。面向服务的体系结构中的操作包括：
- 发布：为了使服务可访问．需要发布服务描述以使服务使用者可以发现和调用它。
- 发现：服务请求者定位服务．方法是查询服务注册中心来找到满足其标准的服务。
- 绑定和调用：在检索完服务描述之后，服务使用者继续根据服务描述中的信息来调用服务。

7.4.3 元素

面向服务的体系结构中的构件包括：

（1）服务：

- 可以通过已发布接口使用服务，并且允许服务使用者调用服务。

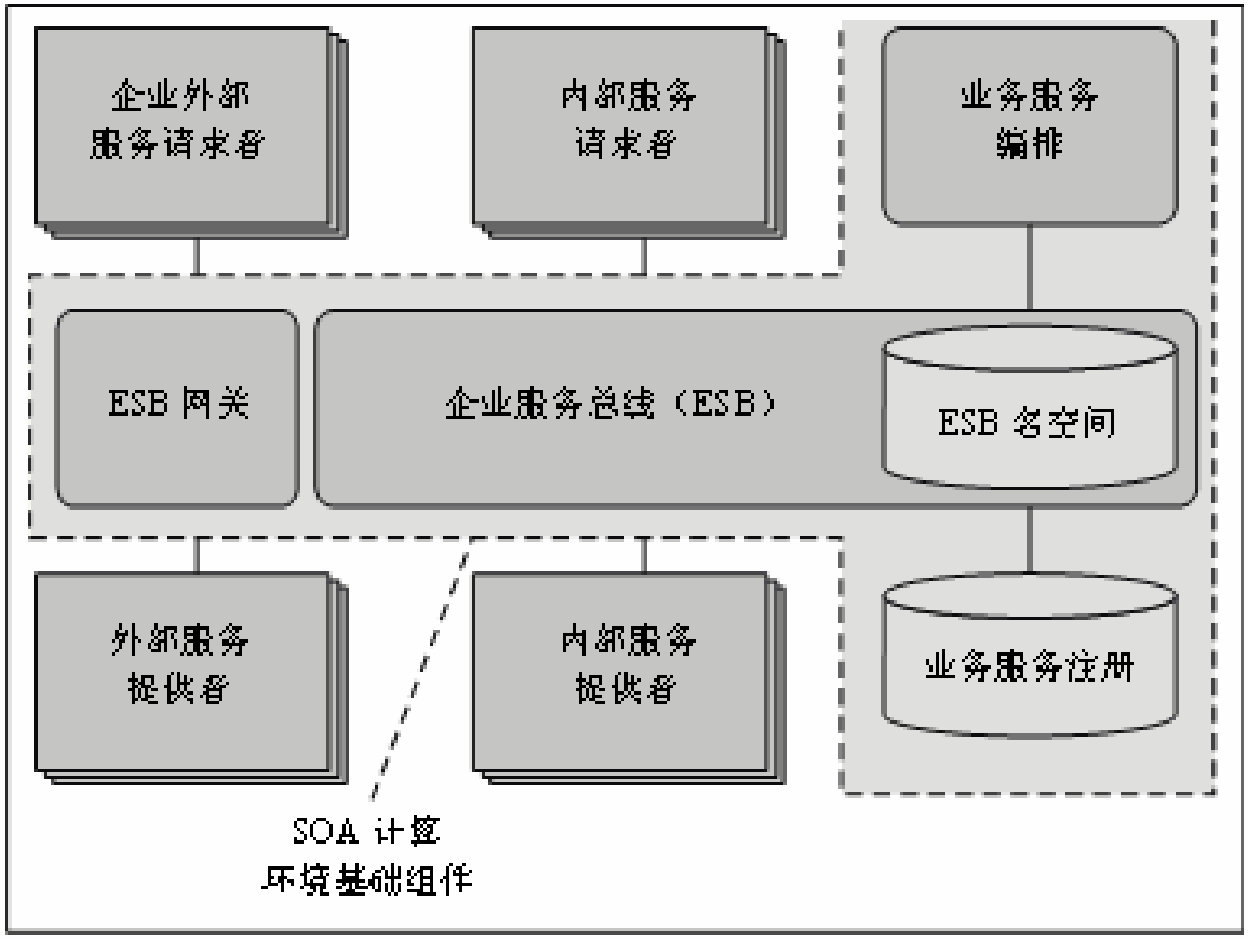
（2）服务描述：

- 服务描述指定服务使用者与服务提供者交互的方式。
- 它指定来自服务的请求和响应的格式。服务描述可以指定一组前提条件、后置条件和/或服务质量（Q0S）级别。

7.4.4 面向服务的计算环境

- 在面向服务的计算环境中，系统可以是高度分布、异构的。它一般包括:
- 服务运行时环境(Service Runtime)、
- 服务总线(Service Integration Infrastructure)、
- 服务网关(Service Gateway)、
- 服务注册库(Service Registry)
- 服务组装引擎(Service Choreography Engine)等.

7.4.4 面向服务的计算环境



SOA计算环境的组成要素

7.4.5 利用价值

- 对 SOA 的需要来源于需要使业务 IT 系统变得更加灵活，以适应业务中的改变。
- 通过允许强定义的关系和依然灵活的特定实现，IT 系统既可以利用现有系统的功能，又可以准备在以后做一些改变来满足它们之间交互的需要。

7.4.5 利用价值

- 对于面向同步和异步应用的，基于请求/响应模式的分布式计算来说，SOA是一场革命。
- 一个应用程序的业务逻辑（business logic）或某些单独的功能被模块化并作为服务呈现给消费者或客户端。
- 这些服务的关键是他们的松耦合特性。

7.4.5 利用价值

- 例如，服务的接口和实现相独立。应用开发人员或者系统集成者可以通过组合一个或多个服务来构建应用，而无须理解服务的底层实现。
- 举例来说，一个服务可以用.NET或J2EE来实现，而使用该服务的应用程序可以在不同的平台之上，使用的语言也可以不同。

7.5 物联网海量数据存储与查询

- 计算机网络的飞速发展导致全球信息总量迅猛增长，据统计2010年全球产生的达到1.2ZB（12 亿TB），世界进入ZB 时代。
- IDC 预测全球数据量从2006 年到2011 年5 年将增长10 倍。
- 而物联网中对象的数量将庞大到以百亿为单位。

7.5 物联网海量数据存储与查询

- 由于物联网中的对象积极参与业务流程的需求、高强度计算需求和数据的持续在线可获取的特性，导致了网络化存储和大型数据中心的诞生。
- 物联网对海量信息存储的需求促进了物联网网络存储技术、海量数据查询技术以及面向物联网的关系型数据库技术的发展。

7.5.1 网络存储体系结构

- 网络存储技术（Network Storage Technologies）是基于数据存储的一种通用网络术语。

  网络存储结构大致分为三种：

- 直连式存储（DAS：Direct Attached Storage）、
- 网络存储设备（NAS：Network Attached Storage）
- 存储网络（SAN：Storage Area Network）。

7.5.1 网络存储体系结构

网络存储技术分类

7.5.1 网络存储体系结构

1. 直连式存储（DAS）

（1） 直连式存储（DAS）简介

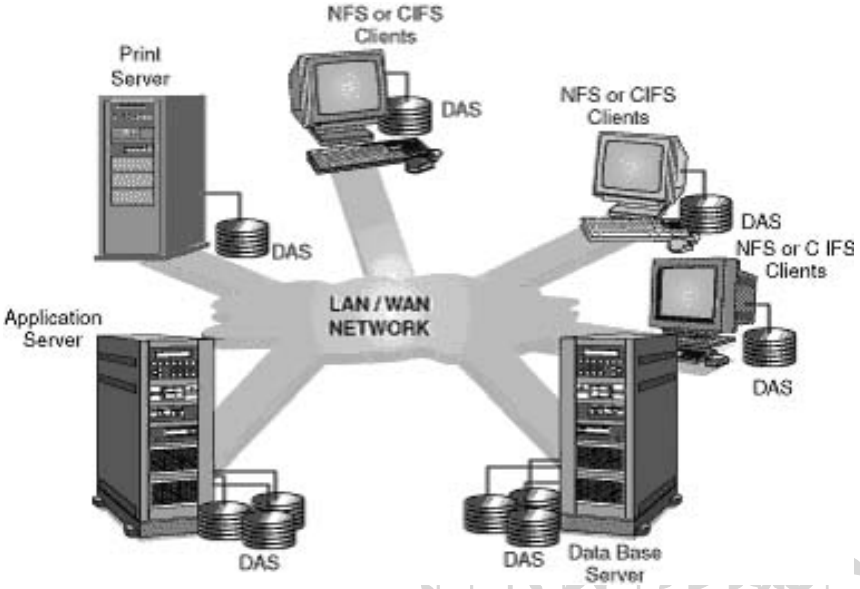
- DAS英文全称是Direct Attached Storage。中文翻译成“直接附加存储”。
- 在这种方式中，存储设备是通过电缆（通常是SCSI接口电缆）直接到服务器的。输入/输入）请求直接发送到存储设备。它依赖于服务器，其本身是硬件的堆叠，不带有任何存储操作系统。
- 这是一种直接与主机系统相连接的存储设备，DAS是计算机系统中最常用的数据存储方法。

7.5.1 网络存储体系结构

DAS的适用环境为：

- ① 存储系统必须被直接连接到应用服务器上时；
- ② 包括许多数据库应用和应用服务器在内的应用，它们需要直接连接到存储器上，群件应用和一些邮件服务也包括在内。

直连式存储（DAS）



7.5.1 网络存储体系结构

（2） 磁盘阵列（RAID）

- 磁盘阵列简称RAID（Redundant Arrays of Inexpensive Disks，RAID），有“价格便宜且多余的磁盘阵列”之意。
- 其原理是利用数组方式来做磁盘组，配合数据分散排列的设计，提升数据的安全性。
- 磁盘阵列主要针对硬盘，在容量及速度上，无法跟上CPU及内存的发展所提出的改善方法。

7.5.1 网络存储体系结构

- 磁盘阵列是由很多便宜、容量较小、稳定性较高、速度较慢磁盘，组合成一个大型的磁盘组，利用个别磁盘提供数据所产生的加成效果来提升整个磁盘系统的效能。
- 同时，在存储数据时，将数据切割成许多区段，分别存放在各个硬盘上。
- 磁盘阵列还能利用同位检查（**Parity Check**）的观念，在数组中任一颗硬盘故障时，仍可读出数据，在数据重构时，将故障硬盘内的数据，经计算后重新置入新硬盘中。

7.5.1 网络存储体系结构

- 磁盘阵列的主流结构是作为独立系统在主机外直连或通过网络与主机相连。
- 磁盘阵列有多个端口可以被不同主机或不同端口连接。主机连接阵列的不同端口可提升传输速度。

磁盘阵列

7.5.1 网络存储体系结构

（3） 磁盘阵列的优点

- RAID通过在多个磁盘上同时存储和读取数据来大幅提高存储系统的数据吞吐量（Throughput）。使用RAID可以达到单个磁盘驱动器几倍、几十倍甚至上百倍的速率。
- 在很多RAID模式中都有较为完备的相互校验/恢复的措施，甚至是直接相互的镜像备份，从而大大提高了RAID系统的容错度，提高了系统的稳定冗余性。

7.5.1 网络存储体系结构

（4）磁盘阵列常用的等级

- 磁盘阵列（Disk Array）是由一个硬盘控制器来控制多个硬盘的相互连接，使多个硬盘的读写同步，减少错误，增加效率和可靠度的技术。
- 常用的等级有1、3、5级等。

7.5.1 网络存储体系结构

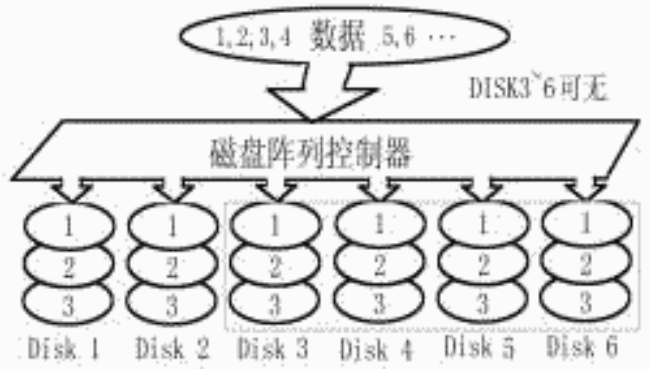
① RAID Level 0

- RAID Level 0是Data Striping（数据分割）技术的实现，它将所有硬盘构成一个磁盘阵列，可以同时对多个硬盘做读写动作，但是不具备备份及容错能力，
- 它价格便宜，硬盘使用效率最佳，但是可靠度是最差的。
- 很少有人冒着数据丢失的危险采用这项技术。

7.5.1 网络存储体系结构

② RAID Level 1

- **RAID Level 1使用的是Disk Mirror（磁盘映射）技术，就是把一个硬盘的内容同步备份复制到另一个硬盘里，所以具备了备份和容错能力，这样使用效率不高，但是可靠性高。**



RAID Level 1磁盘映射技术

7.5.1 网络存储体系结构

③ RAID Level 3

- RAID Level 3采用Byte－interleaving（数据交错存储）技术，硬盘在SCSI控制卡下同时动作，并将用于奇偶校验的数据存储到特定硬盘机中，它具备了容错能力，它的可靠度较佳。

④ RAID Level 5

- RAID Level 5使用的是Disk Striping（硬盘分割）技术，与Level 3的不同之处在于它把奇偶校验数据存放到各个硬盘里，各个硬盘在SCSI控制卡的控制下平行动作，有容错能力，跟Level 3一样，它的使用效率也是安装几个再减掉一个。

7.5.1 网络存储体系结构

（5） 热插拔硬盘

- 热插拔硬盘英文名为Hot－Swappable Disk，
- 在磁盘阵列中，如果使用支持热插拔技术的硬盘，在有一个硬盘坏掉的情况下，服务器可以不用关机，直接抽出坏掉的硬盘，换上新的硬盘。
- 一般的商用磁盘阵列在硬盘坏掉时，会自动鸣叫提示管理员更换硬盘。

7.5.1 网络存储体系结构

2.网络附加存储（NAS）

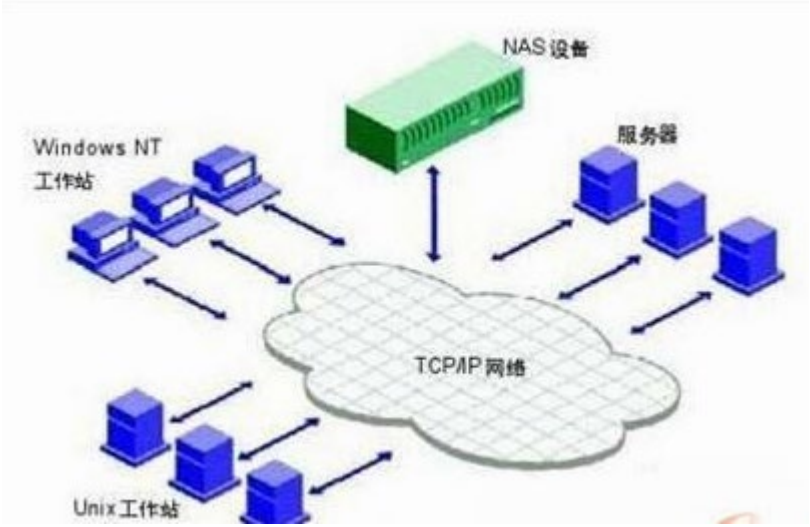
- NAS是英文“Network Attached Storage”的缩写，中文意思是“网络附加存储”。就是连接在网络上，具备资料存储功能的装置，因此也称为“网络存储器”或者“网络磁盘阵列”。
- 在NAS存储结构中，存储系统不再通过I/O总线附属于某个特定的服务器或客户机，而是直接通过网络接口与网络直接相连，由用户通过网络访问。

7.5.1 网络存储体系结构

- NAS是一种专业的网络文件存储及文件备份设备，它是基于LAN（局域网）的，按照TCP/IP协议进行通信，以文件的I/O（输入/输出）方式进行数据传输。
- 在LAN环境下，NAS已经完全可以实现异构平台之间的数据级共享，比如NT、UNIX等平台的共享。

7.5.1 网络存储体系结构

- 一个NAS系统包括处理器，文件服务管理模块和多个硬盘驱动器（用于数据的存储）。
- NAS可以应用在任何的网络环境当中。主服务器和客户端可以非常方便地在NAS上存取任意格式的文件。



网络附加存储
（NAS）

7.5.1 网络存储体系结构

3. 存储区域网络（SAN）

- 英文全称：**Storage Area Network**，即存储区域网络。它是一种通过光纤集线器、光纤路由器、光纤交换机等连接设备将磁盘阵列、磁带等存储设备与相关服务器连接起来的高速专用子网。
- 存储网络（**SAN**）是指存储设备相互连接且与一台服务器或一个服务器群相连的网络。
- 简单的说，**SAN**实际上是一种存储设备池，这一子网上的存储空间可由以太网主网上的每一系统所共享。

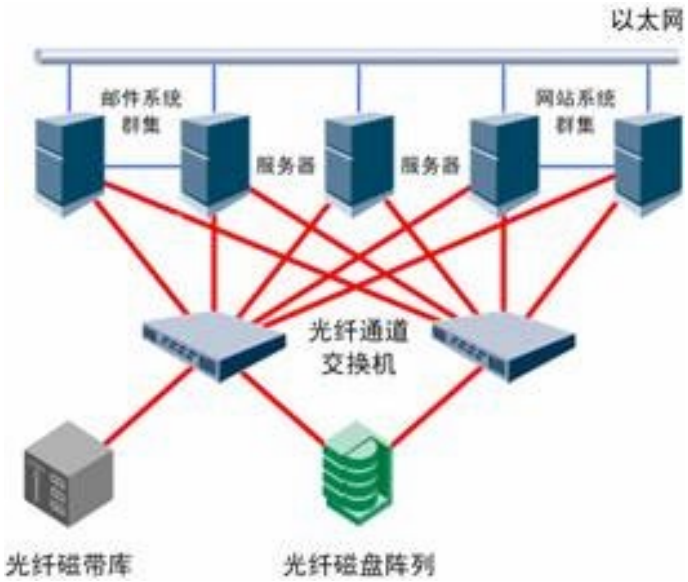
7.5.1 网络存储体系结构

(1) 存储区域网络的基本构成

- SAN由三个基本的组件构成：
- 接口（如SCSI、光纤通道、ESCON等）、
- 连接设备（交换设备、网关、路由器、集线器等）
- 通信控制协议（如IP和SCSI等）。

7.5.1 网络存储体系结构

三个组件再加上附加的存储设备和独立的SAN服务器，就构成一个SAN系统。



存储区域网络
（SAN）

7.5.1 网络存储体系结构

(2) 存储区域网络的适用范围

- ① 对数据安全性要求很高、数据在线性要求高、具有本质上物理集中、逻辑上又彼此独立的数据管理特点的企业，典型行业用户是电信、金融和证券、商业网站。
- ② 对数据存储性能要求高的企业，典型行业用户是电视台、交通部门和测绘部门。

7.5.1 网络存储体系结构

- ③ 在系统级方面具有很强的容量（动态）可扩展性和灵活性的企业，典型行业用户是各中大型企业的ERP系统、CRM系统和决策支持系统。
- ④ 具有超大型海量存储特性的企业，典型行业用户是图书馆、博物馆、税务和石油。

7.5.1 网络存储体系结构

4. 三种网络存储结构的比较

- 从具体功能上讲，三种网络存储结构分别适用于不同的应用环境：

(1) 直接附加存储(Direct-Attached Storage, DAS)

是将存储系统通过缆线直接与服务器或工作站相连,一般包括多个硬盘驱动器，与主机总线适配器通过电缆或光纤,在存储设备和主机总线适配器之间不存在其他网络设备,实现了计算机内存储到存储子系统的跨越。

7.5.1 网络存储体系结构

- （2）网络附加存储(Network Attached Storage, NAS)是文件级的计算机数据存储架构，计算机连接到一个仅为其它设备提供基于文件级数据存储服务的网络。
- （3）存储区域网络 (Storage Area Network, SAN)是通过网络方式连接存储设备和应用服务器的存储架构，由服务器、存储设备和SAN连接设备组成。SAN的特点是存储共享并支持服务器从SAN直接启动。

7.5.1 网络存储体系结构

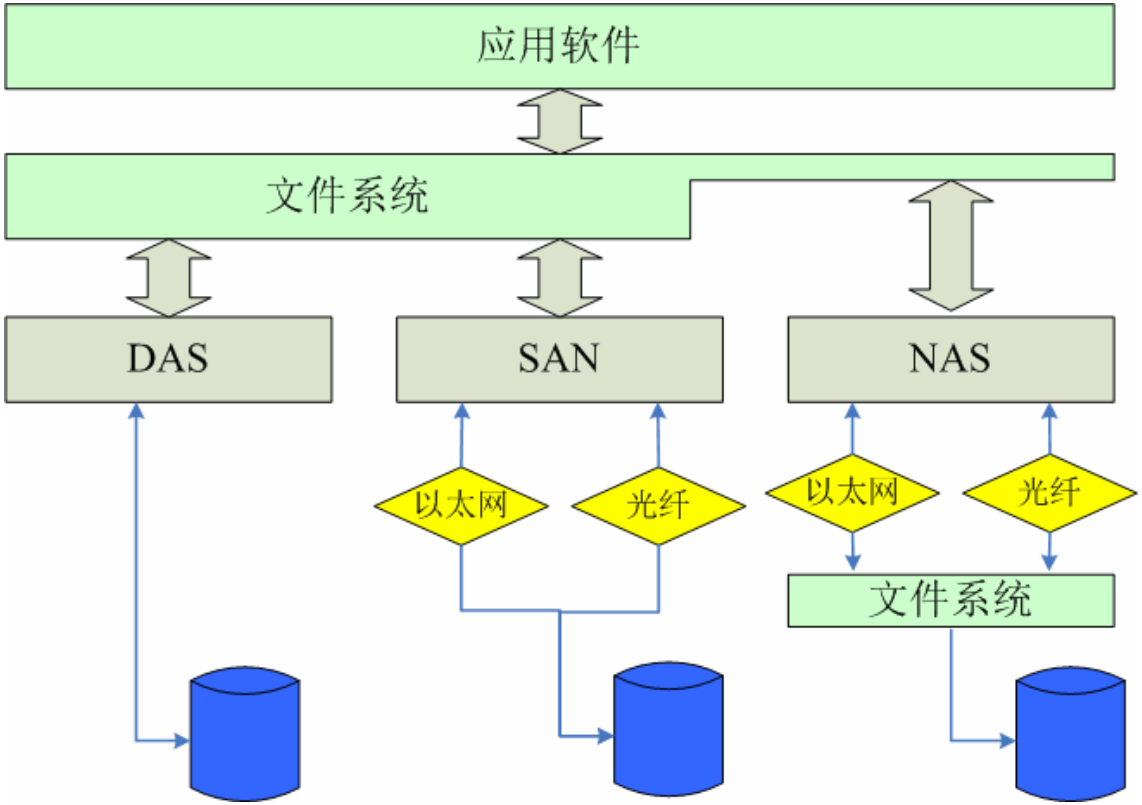
NAS与DAS的区别

- DAS是一种对已有服务器的简单扩展，并没有真正实现网络互联。NAS则是将网络作为存储实体，更容易实现文件级别的共享。NAS性能上比DAS有所增强

SAN与NAS的区别

- NAS结构和SAN最大的区别就在于NAS有文件操作和管理系统，而SAN却没有这样的系统功能，其功能仅仅停留在文件管理的下一层，即数据管理。同时SAN和NAS相比不具有资源共享的特征。

7.5.1 网络存储体系结构



网络存储结构的比较

7.5.1 网络存储体系结构

- SAN和NAS并不是相互冲突的，是可以共存于一个系统网络中的，但NAS通过一个公共的接口实现空间的管理和资源共享，SAN仅仅是为服务器存储数据提供一个专门的快速后方通道，
- 在空间的利用上，SAN是只能独享的数据存储池，NAS是共享与独享兼顾的数据存储池。
- 因此，NAS与SAN的关系也可以表述为：NAS是Network-attached（网络外挂式），而SAN是Channel-attached（通道外挂式）。

7.5.2 海量数据存储及查询

- 现在的网络世界是海量数据的时代，物联网数据存储将使用数据中心的模式。
- 数据中心是一整套复杂的设施。它不仅仅包括计算机系统和其它与之配套的设备（例如通信和存储系统），还包含冗余的数据通信连接、环境控制设备、监控设备以及各种安全装置。
- 在本节案例中以一个典型数据中心（Google 数据中心）加以说明。

7.6 物联网数据融合及路由

- 数据融合一词最早出现在20 世纪70 年代，它是人类模仿自身信息处理能力的结果，类似人类和其它动物对复杂问题的综合处理。
- 数据融合技术最早用于军事，目前，工业控制、机器人、空中交通管制、海洋监视和管理等领域也向着多传感器数据融合方向发展。
- 物联网概念的提出后，数据融合技术将成为其数据处理等相关技术开发所要关心的重要问题之一。

7.6.1 数据融合的基本概念

1. 数据融合的定义

- 数据融合技术是指利用计算机对按时序获得的若干观测信息，在一定准则下加以自动分析、综合，以完成所需的决策和评估任务而进行的信息处理技术。

7.6.1 数据融合的基本概念

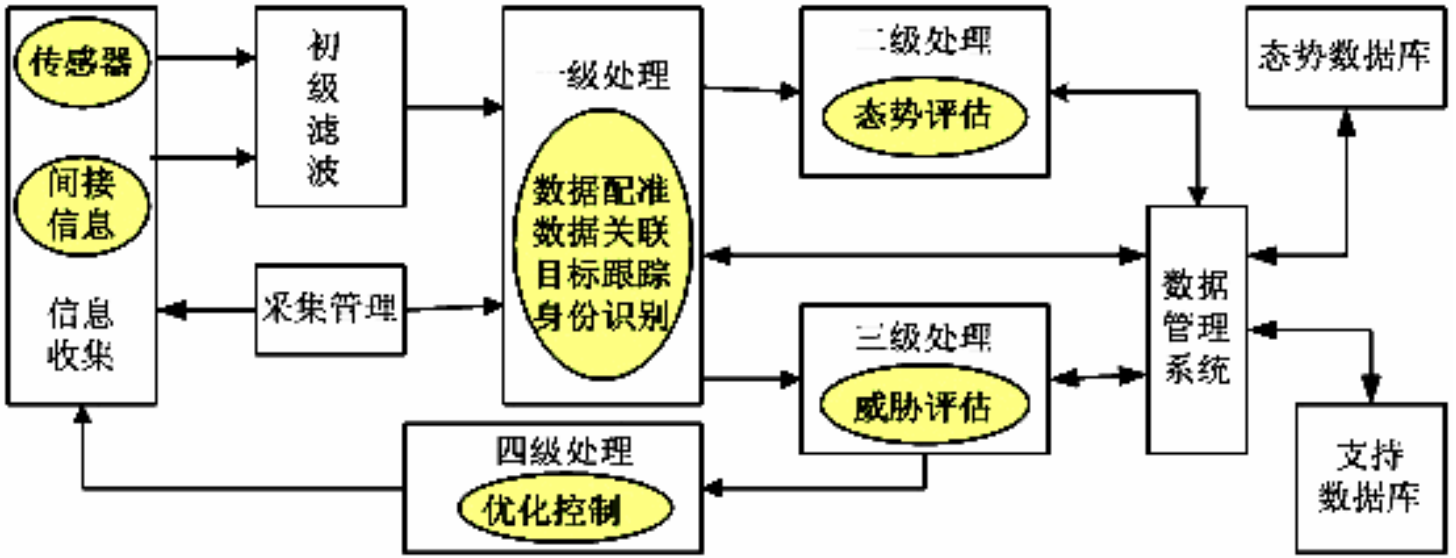
2. 数据融合研究的主要内容

- 1）数据对准;
- 2）数据相关;
- 3）数据识别，即估计目标的类别和类型;
- 4) 感知数据的不确定性;
- 5）不完整、不一致和虚假数据;
- 6) 数据库;
- 7) 性能评估。

7.6.1 数据融合的基本概念

3. 数据融合的体系结构

- 描述数据融合的体系结构的数据融合一般模型如下图。



7.6.2 物联网中数据融合的关键问题

1.物联网数据融合所要解决的关键问题

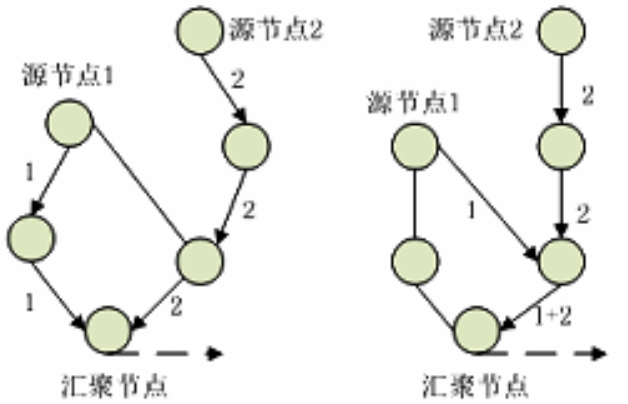
- ① 数据融合节点的选择。
- ② 数据融合时机。
- ③ 数据融合算法。

2.物联网数据融合技术要求

- ① 稳定性;　　② 数据关联;
- ③ 能量约束;　　④协议的可扩展性。

7.6.3 物联网数据融合的基本原理

- 通过对多感知节点信息的协调优化，数据融合技术可以有效地减少整个网络中不必要的通信开销，提高数据的准确度和收集效率。
- 因此，传送已融合的数据要比未经处理的数据节省能量，延长网络的生存周期。



物联网数据融合示意图

7.6.3 物联网数据融合的基本原理

数据融合主要关注一下五点:

- 1）多个不同类型的源节点(如有源或无源的传感器）采集观测目标的数据；
- 2）对源节点的输出数据（离散的或连续的时间函数数据、输出矢量、成像数据或一个直接的属性说明）进行特征提取，提取代表观测数据的特征矢量；

7.6.3 物联网数据融合的基本原理

- 3）对特征矢量进行模式识别处理（例如：汇聚算法、自适应神经网络或其它能将特征矢量变换成目标属性判决的统计模式识别法等）完成各传感器关于目标的说明；
- 4）将各源节点关于目标的说明数据按同一目标进行分组，即关联；
- 5）利用融合算法将每一目标各源节点数据进行合成， 得到该目标的一致性解释与描述。

7.6.4 传感器网络数据融合技术

1. 数据融合技术的产生背景

数据融合技术的产生背景来自于数据融合的几个重要作用:

- （1）节省能量
- （2）获取更准确的信息
- （3）提高数据收集效率

7.6.4 传感器网络数据融合技术

2. 数据融合结合网络的各个协议层来进行

- 在应用层，可通过分布式数据库技术，对采集的数据进行初步筛选，达到融合效果；
- 在网络层，可以结合路由协议，减少数据的传输量；
- 在数据链路层，可以结合MAC，减少MAC层的发送冲突和头部开销，达到节省能量目的的同时，还不失去信息的完整性。

7.6.5数据融合的层次结构

1.传感网节点的部署

- 目前，传感网感知节点的部署方式一般有3种类型，最常用的拓扑结构是并行拓扑。在这种部署方式中， 各种类型的感知节点同时工作。
- 另一种类型是串行拓扑，在这种结构中，感知节点检测数据信息具有暂时性。**SAR(Synthetic Aperture Radar)**图像就属于此结构。
- 还有一种类型是混合拓扑， 即树状拓扑。

7.6.5数据融合的层次结构

2.数据融合的层次划分

- 数据融合大部分是根据具体问题及其特定对象来建立自己的融合层次。
- 根据多传感器数据融合模型定义和传感网的自身特点，通常按照节点处理层次、融合前后的数据量变化、信息抽象的层次，来划分传感网的数据融合的层次结构。

7.6.5数据融合的层次结构

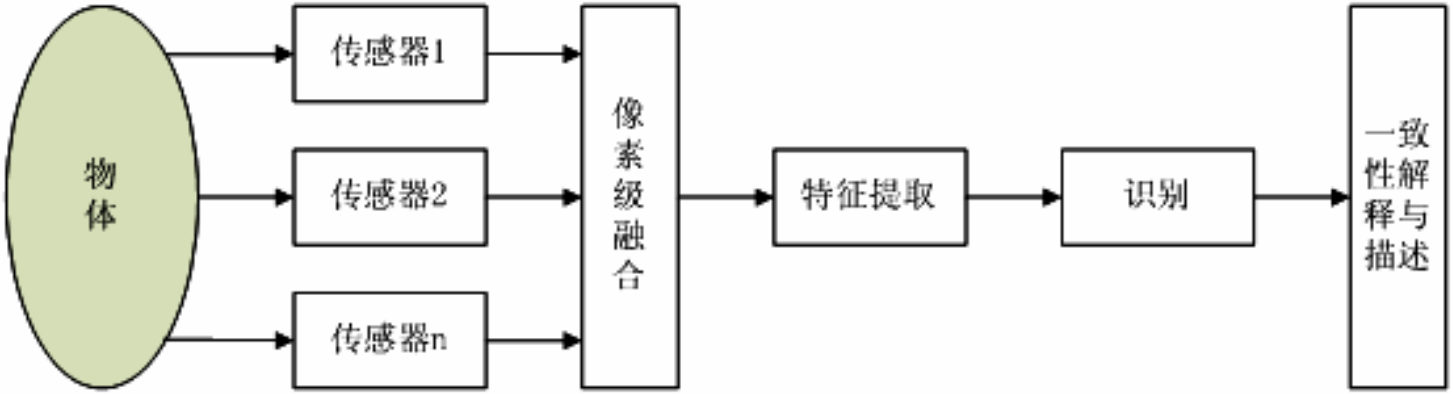
数据融合可分为三类：

① 像素级融合

- 它是直接在采集到的原始数据层上进行的融合，在各种传感器的原始测报未经预处理之前就进行数据的综合与分析。数据层融合一般采用集中式融合体系进行融合处理过程。
- 这是低层次的融合，如成像传感器中通过对包含若一像素的模糊图像进行图像处理来确认目标属性的过程就属于数据层融合。

7.6.5数据融合的层次结构

像素级融合



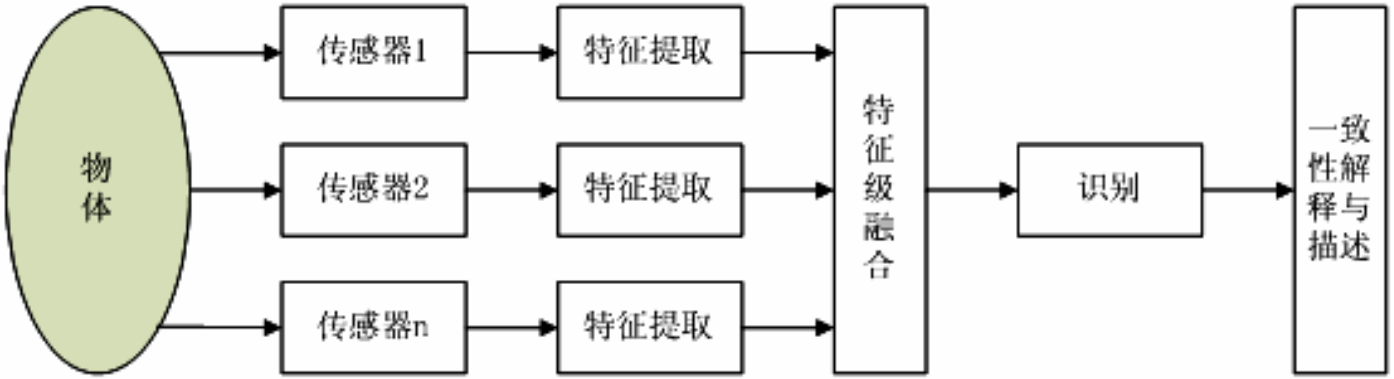
7.6.5数据融合的层次结构

② 特征层融合

- 特征层融合属于中间层次的融合，它先对来自传感器的原始信息进行特征提取(特征可以是目标的边缘、方向、速度等)，然后对特征信息进行综合分析和处理。
- 特征层融合的优点在于实现了可观的信息压缩，有利于实时处理，并且由于所提取的特征直接与决策分析有关，因而融合结果能最大限度的给出决策分析所需要的特征信息。

7.6.5数据融合的层次结构

特征层融合一般采用分布式或集中式的融合体系。特征层融合可分为两大类:一类是目标状态融合;另一类是目标特性融合。

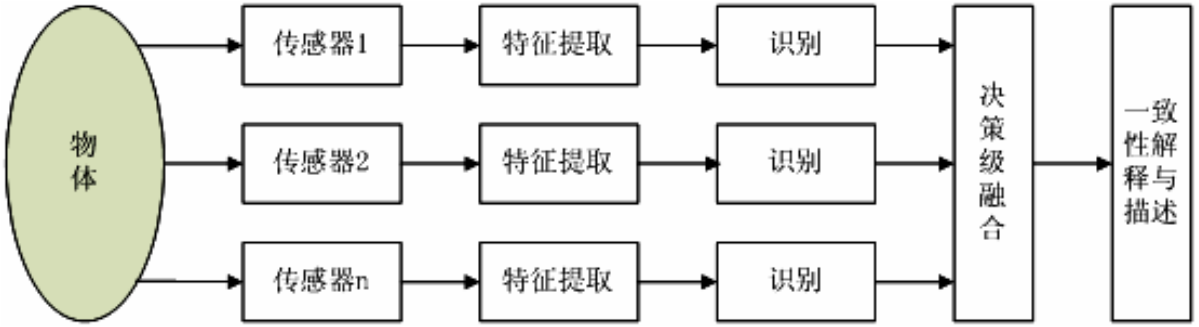


特征层融合

7.6.5数据融合的层次结构

③ 决策层融合

- 决策层融合通过不同类型的传感器观测同一个目标，每个传感器在本地完成基本的处理，其中包括预处理、特征抽取、识别或判决，以建立对所观察目标的初步结论。然后通过关联处理进行决策层融合判决，最终获得联合推断结果。



决策层融合

7.6.6 多传感器数据融合算法

- 在多传感器系统中，由于信息表现形式的多样性，数据量的巨大性，数据关系的复杂性，以及要求数据处理的实时性、准确性和可靠性，都已大大超出了人脑的信息综合处理能力，在这种情况下，多传感器数据融合技术应运而生。
- 多传感器数据融合(Multi-Sensor Data Fusion，MSDF)，简称数据融合；也被称为多传感器信息融合(Multi-Sensor Information Fusion，MSIF)。

7.6.6 多传感器数据融合算法

- 在多传感器系统中，由于信息表现形式的多样性，数据量的巨大性，数据关系的复杂性，以及要求数据处理的实时性、准确性和可靠性，都已大大超出了人脑的信息综合处理能力，在这种情况下，多传感器数据融合技术应运而生。
- 多传感器数据融合(Multi-Sensor Data Fusion，MSDF)，简称数据融合；也被称为多传感器信息融合(Multi-Sensor Information Fusion，MSIF)。

7.6.6 多传感器数据融合算法

目前已有大量的多传感器数据融合算法，基本上可概括为两大类：

- 一是随机类方法，包括加权平均法、卡尔曼滤波法、贝叶斯估计法、**D-S**证据推理等。
- 二是人工智能类方法，包括模糊逻辑、神经网络等。
- 不同的方法适用于不同的应用背景。神经网络和人工智能等新概念、新技术在数据融合中将发挥越来越重要的作用。

7.6.6 多传感器数据融合算法

1. 多传感器数据融合概念

多传感器数据融合比较确切的定义可概括为：

- 充分利用不同时间与空间的多传感器数据资源，采用计算机技术对按时间序列获得的多传感器观测数据，在一定准则下进行分析、综合、支配和使用，获得对被测对象的一致性解释与描述，进而实现相应的决策和估计，使系统获得比它的各组成部分更充分的信息。

7.6.6 多传感器数据融合算法

2.多传感器数据融合原理

- （1）N个不同类型的传感器（有源或无源的）收集观测目标的数据；
- （2）对传感器的输出数据（离散的或连续的时间函数数据、输出矢量、成像数据或一个直接的属性说明）进行特征提取的变换，提取代表观测数据的特征矢量Yi；

7.6.6 多传感器数据融合算法

- （3）对特征矢量Yi进行模式识别处理（如，聚类算法、自适应神经网络或其他能将特征矢量Yi变换成目标属性判决的统计模式识别法等）完成各传感器关于目标的说明；
- （4）将各传感器关于目标的说明数据按同一目标进行分组，即关联；
- （5）利用融合算法将每一目标各传感器数据进行合成，得到该目标的一致性解释与描述。

7.6.6 多传感器数据融合算法

3. 多传感器数据融合方法

- 多传感器数据融合的常用方法基本上可概括为随机和人工智能两大类：
- 随机类方法有加权平均法、卡尔曼滤波法、多贝叶斯估计法、**Dempster-Shafer**（**D-S**）证据推理、产生式规则等；
- 而人工智能类则有模糊逻辑理论、神经网络、粗集理论、专家系统等。

7.6.6 多传感器数据融合算法

4. 应用领域

- （1）军事应用
- （2）复杂工业过程控制
- （3）机器人
- （4）遥感
- （5）交通管理系统
- （6）全局监视

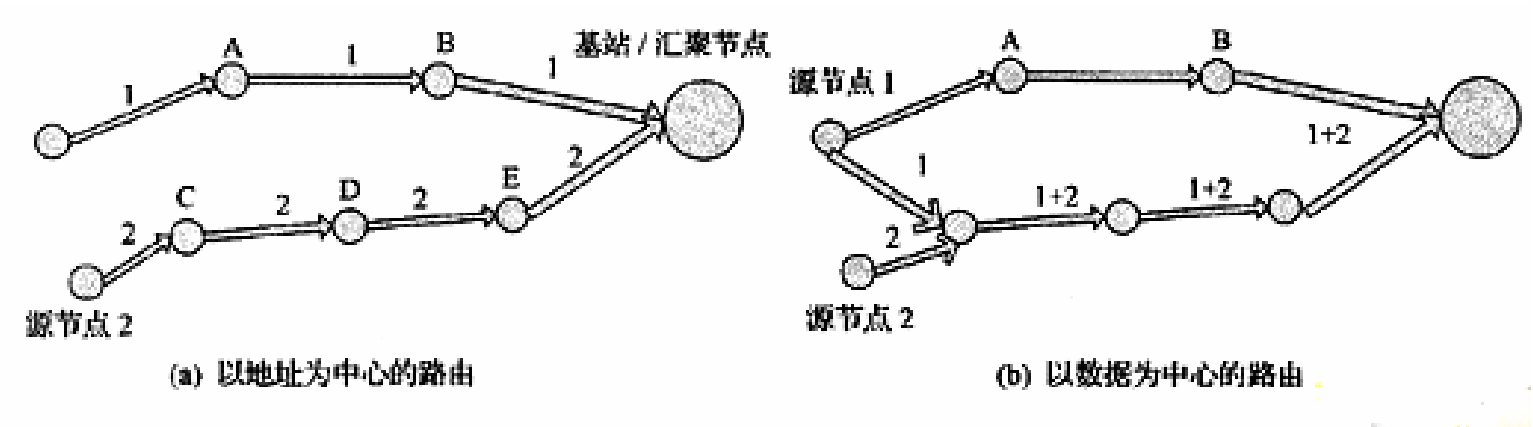
7.6.7 传感网数据融合路由算法

比较典型的数据融合路由协议有：基于数据融合树的路由协议、基于分簇的路由协议，以及基于节点链的路由协议。

- 从网络层来看，数据融合通常和路由的方式有关，例如以地址为中心的路由方式（最短路径转发路由），路由并不需要考虑数据的融合。
- 然而，以数据为中心的路由方式，源节点并不是各自寻找最短路径路由数据，而是需要在中间节点进行数据融合，然后再继续转发数据。

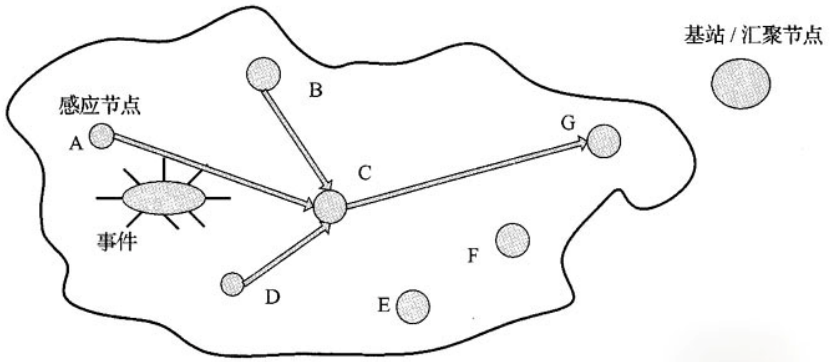
7.6.7 传感网数据融合路由算法

以地址为中心的路由与以数据为中心的路由的区别



7.6.7 传感网数据融合路由算法

- 汇聚节点收集数据时是通过反向组播树的形式从分散的传感器节点将数据逐步汇聚起来的。当各个传感器节点监测到突发事件时，传输数据的路径形成一棵反向组播树，这个树就成为数据融合树。
- 如下图所示，无线传感器网络就是通过融合树来报告监测到的事件的。



7.6.7 传感网数据融合路由算法

关于数据融合树的构造，可以转化为最小Steiner树来求解，它是个NP Com－plete完各难题。以下给出了三种不同的非最优的融合算法。

- ①以最近源节点为中心（center at nearest Source，CNS）
- ②最短路径树（shortest paths tree，SPT）
- ③贪婪增长树（greedy incremental tree，GIT）

7.6.7 传感网数据融合路由算法

- 上面三种算法都比较适合基于事件驱动的无线传感器网络的应用，可以在远程数据传输前进行数据融合处理，从而减少冗余数据的传输量。
- 在数据的可融合程度一定的情况下，上面三种算法的节能效率通常为：**GIT > SPT > CNS**。
- 当基站或汇聚节点与传感器覆盖监测区域距离的远近不同时，可能会造成上面算法节能的一些差异。

7.7 未来的物联网软件、服务和算法技术


服务将在未来的物联网中扮演着关键的角色：

- 服务可以提供了一种很好的方式来封装各种系统功能，比如，可以从各式各样的底层硬件或者具体实现细节中抽象出通用算法来加以实现并且反复应用；
- 服务可以进行自主的协调以创造出新的、更高等级的服务功能；
- 通过这些分布式部署和执行的服务逻辑，可以解决物联网的可扩展性中最为关键的主要问题。

7.8 案例：Google 数据中心

Google的数据中心使用廉价的Linux PC机组成集群，在上面运行各种应用。核心组件是3个：

（1） GFS（Google File System）。

- 一个分布式文件系统，隐藏下层负载均衡，冗余复制等细节，对上层程序提供一个统一的文件系统API接口。
- GFS把文件分成64MB的块，分布在集群的机器上，同时每块文件至少有3份以上的冗余。
- 中心是一个Master节点，根据文件索引，找寻文件块。

7.8 案例：Google 数据中心

（2）MapReduce。

大多数分布式运算可以抽象为MapReduce操作。Map是把输入Input分解成中间的Key/Value对，Reduce把Key/Value合成最终输出Output。

这两个函数由程序员提供给系统，下层设施把Map和Reduce操作分布在集群上运行，并把结果存储在GFS上。

7.8 案例：Google 数据中心

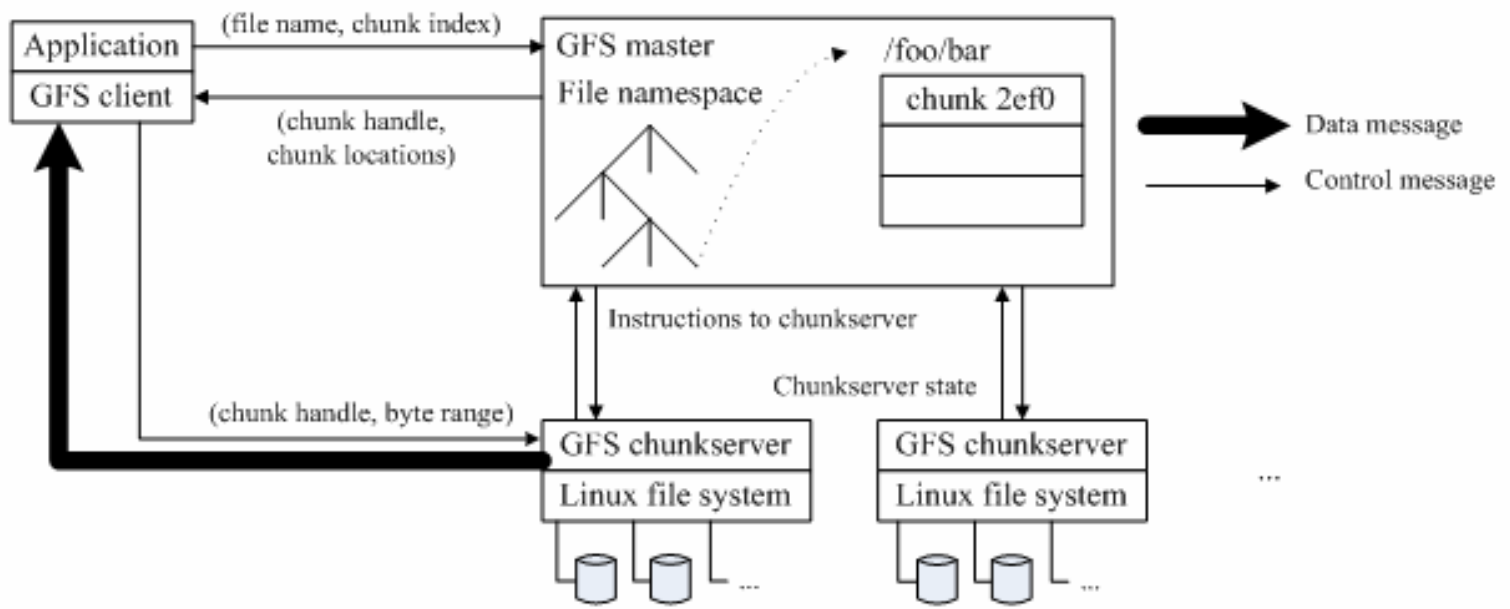
（3） **BigTable**。

一个大型的分布式数据库，这个数据库不是关系式的数据库。

像它的名字一样，就是一个巨大的表格，用来存储结构化的数据。

7.8 案例：Google 数据中心

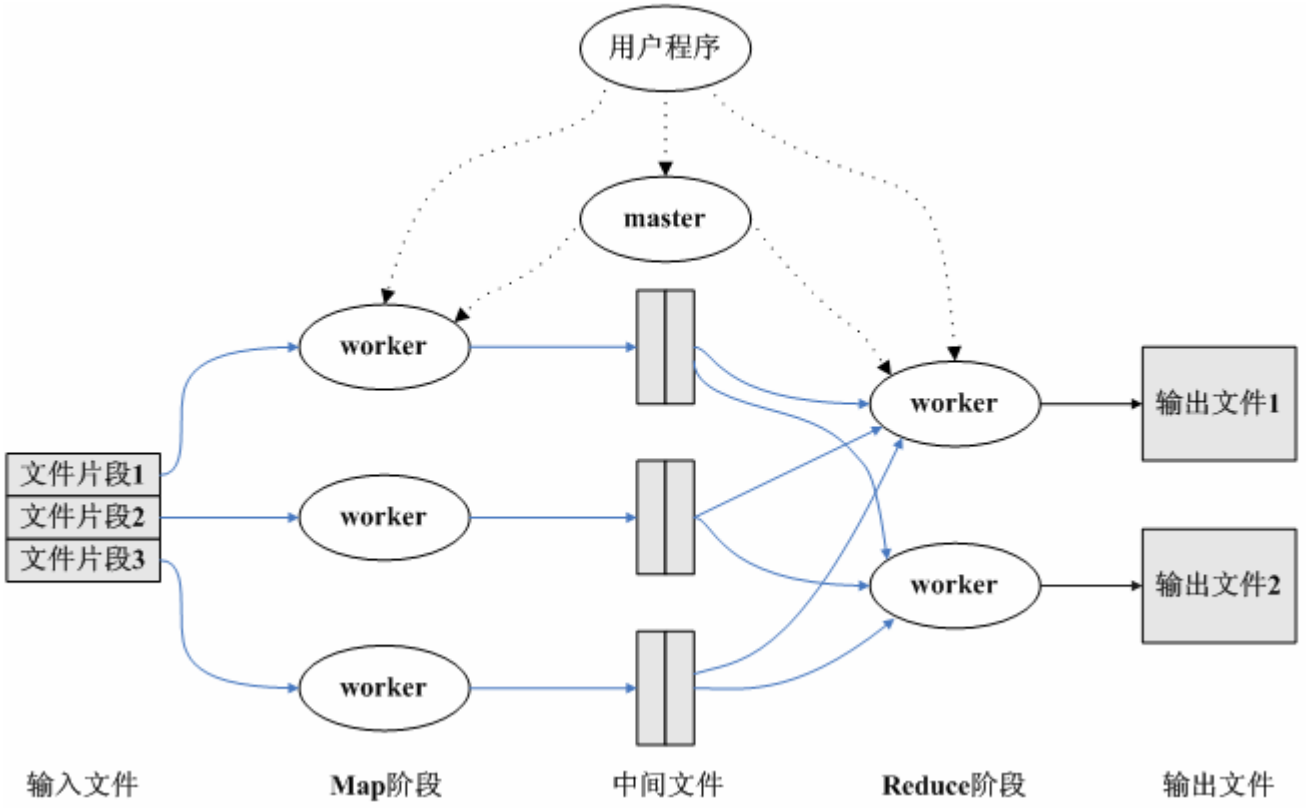
1. 谷歌文件系统（Google File System，GFS ）



GFS的设计架构

7.8 案例：Google 数据中心

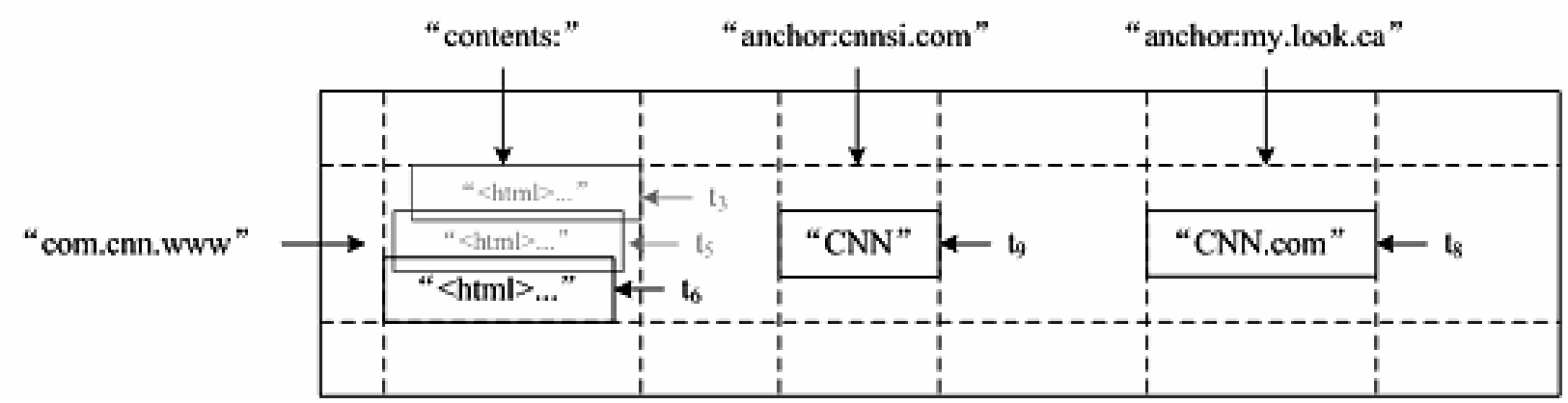
2. MapReduce 编程模型系统



MapReduce程序的执行过程

7.8 案例：Google 数据中心

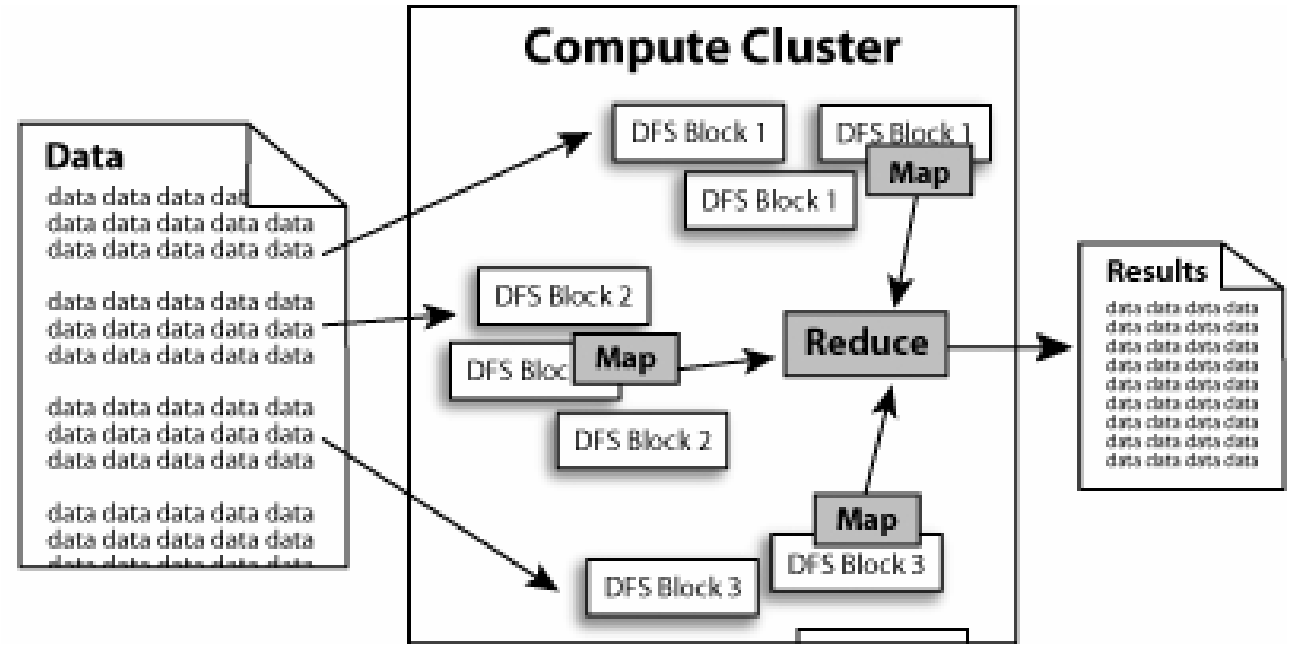
3. BigTable 分布式存储系统



BigTable 分布式存储系统

7.8 案例：Google 数据中心

4. 典型数据中心： Hadoop 分布式文件系统



Hadoop的体系结构

Thank You!