分布式系统中的资源发现机制综述

海 沫
(中央财经大学信息学院 北京 100081)

摘 要 资源发现问题亦即由给定的资源描述找到满足该描述的资源地址。如何快速并且准确地从分布存储的资源中找到所需的资源,是一个挑战性的问题。传统的网格资源发现系统采用注册中心和索引的方式,但这些方式不能满足网格系统规模不断扩大的需求。而对等网络是一种扩展性好的分布式系统,采用已有的P2P技术解决网格环境下的资源发现问题是一种有效的途径。介绍了网格系统中的资源发现、P2P系统的资源发现以及基于P2P的网格资源发现系统,并对它们进行了比较。

关键词 资源发现,网格,注册中心,索引,对等网络
中图法分类号 TP316.4 **文献标识码** A

Survey of Resource Discovery Mechanism in Distributed System

HAI Mo
(School of Informatics, Central University of Finance and Economics, Beijing 100081, China)

Abstract The resource discovery problem is to find the addresses of matching resources by the given description of resources. How to find the needed resources quickly and accurately from the distributed stored resources is a challenge. Traditional grid resource discovery system adopts registry and index, but these methods cannot satisfy the need of increasing scale of grid system. P2P system is a scalable distributed system. It's an efficient method to solve the grid resource discovery problem by adopting current P2P technology. This paper introduced resource discovery in grid system, resource discovery in P2P system, grid resource discovery system based on P2P and gave comparisons.

Keywords Resource discovery, Grid, Registry, Index, P2P

1 引言

分布式系统是将地理上分布的多个机器连接起来,向用户提供单一系统映像。资源发现问题为:根据给定资源描述,找到满足此描述的资源的地址。这些描述有不同的形式,既可以是全局唯一标识符,也可以是属性-值集合。和单机不同的是,分布式系统中的资源分布存储在物理上分布的多台机器上,因而如何快速并且准确地从分布存储的资源中找到所需的资源是一个挑战性的问题。分布式系统既可以采用集中控制的方式,也可以采用分散控制的方式。对于集中控制的分布式系统,资源信息被单一机器所管理,可采用注册中心方式的资源发现机制;对于分散控制的分布式系统,资源信息被分散管理,可采用索引和P2P方式的资源发现机制。

2 网格系统的资源发现

2.1 注册中心

注册中心是被授权的集中管理信息的系统。如Web服务的提供方,在提供一个Web服务前,必须主动地向注册中心提供注册信息。注册中心可以决定谁有权增加和更新注册信息。尽管注册信息的拥有者可以将修改权委托给第三方,但不是任意个体都能修改注册信息。注册者可以决定什么信息发布在注册中心,而其它个体不能独立地扩展这些信息。

统一描述、发现和集成协议(UDDI, Universal Description, Discovery, and Integration)是注册中心的实现实例。UDDI是一套基于Web的、分布的并且为Web服务提供信息注册中心的实现标准规范,同时它包含一组使企业能将自身提供的Web服务注册以使别的企业能够发现的访问协议的实现标准。UDDI标准定义了Web服务的发布与发现的方法。UDDI提供了一种分布的商业注册中心的方法,维护了一个企业提供的Web服务的全球目录,而且其中的信息描述格式是基于通用的XML格式的。目前的UDDI只能支持按照类别和名字发现服务,不能提供基于属性的服务发现。

2.2 索引

与注册中心不同的是索引方式。索引是为存储在不同位置的信息编辑而成的目录,不是集中控制的,它所提供的信息也未经审定。在这种模式下,发布是被动的,Web服务的提供者把服务和功能描述信息提供在Web中,其它想利用该信息的实体可以收集这些信息。任何人可以建立自己的索引,他们可以利用某种自动机制收集这些信息,并加到自己的索引中。多个组织可以拥有这样的索引。

区别注册中心和索引的方法,不仅仅看系统是否采用了注册服务器,而且要看注册的内容是由谁决定的,是任何人都

到稿日期:2010-01-25 返修日期:2010-04-12 本文受中央财经大学"211"工程三期建设项目,国家自然科学基金项目(60970143)资助。
海 沫(1978—),女,博士,讲师,CCF会员,主要研究方向为分布式系统、P2P计算等,Email:haimozhi@gmail.com。

可以加入自己的描述，还是只有信息的发布者才有权更改。

Globus[1]的 MDS(Monitoring and Discovery Service)采用的就是索引方式。Globus Toolkit 是用来开发网格系统和应用的开源服务和软件库，包括安全、信息架构、资源管理、数据管理、通信和错误检测。MDS 实现了资源发现功能，提供了发布和访问网格资源信息的框架。

在 GT2(Globus Toolkit2)中，MDS-2 使用 LDAP 作为信息的统一接口，以提供系统组件状态的信息。MDS-2 包括了可配置的网格资源信息服务(GRIS, Grid Resource Information Service)和可配置的网格目录信息服务(GIIS, Grid Index Information Service)。GRIS 能够提供查询某个特定网格结点中资源的功能，既能提供如主机标识的静态信息，又能提供如 CPU 空闲情况和可用内存的动态信息。一个 GIIS 包含了被一个特定虚拟组织(VO, Virtual Organization)所管理的所有 GRIS 的信息，提供了单一系统映像。一个 GIIS 可以是几个 GIIS 的集合，因而 GIIS 可以是一个层次结构，顶层的 GIIS 可以回答所有关于位于其下的虚拟组织中资源的查询。Globus MDS-2 的结构如图 1 所示。

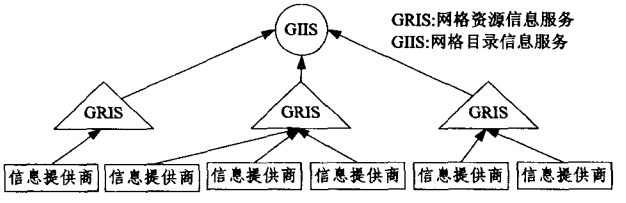

图 1　MDS-2 的结构

GT3 实现了 MDS-3。与 MDS-2 相同的是，MDS-3 也是基于层次结构。不同的是，MDS-3 采用开放的网格服务体系结(OGSA, Open Grid Services Architecture)模型[2]。OGSA 框架中每个资源表示为一个网格服务，因此资源发现问题主要是定位和查询网格服务信息。MDS-3 中，资源信息由索引服务提供。一个索引服务是一个网格服务，它存放所有注册到其上的网格服务的信息。服务信息由服务数据元素组成，描述了单个资源的属性。索引服务的主要功能是提供接口，以查询已注册服务信息的集合视图。

GT4 实现的 MDS-4 符合 WSRF(Web Services Resource Framework)规范[3]。MDS-4 中的触发服务实现了满足特定规则集合的事件产生时的通知机制。

MDS 服务的缺点是：MDS 实现的是基于 LDAP 的树状元数据目录服务，只能管理相对静态的服务信息，不能处理服务信息的频繁更新；MDS 不能处理复杂的查询，LDAP 的树状结构决定了多属性查询和区段查询的效率低；LDAP 风格的查询语言需要用户指定查询的起始节点和查询范围，树结构对用户不透明，结构改变会对用户造成影响。

2.3 Condor

Condor[4]是应用于计算密集作业的资源管理系统。它接收用户提交的作业并且在网络中找到合适的、可用的资源以执行作业。Condor 采用的是集中式的调度模型。中心管理器收集资源的状态信息并且接收用户的请求，将资源描述和资源属主及作业指定的使用策略进行匹配，从而决定如何调度作业。资源和作业都用 ClassAd 规范语言进行描述。ClassAd 的思想与报纸的分类广告一致，能接收任务的空闲资源和寻找可用资源的用户都发布了它们的 aids。

Condor 的资源通过资源属主代理(RA, Resource-owner Agent)表示。每个 RA 周期性地检查资源的状态并且为每个资源创建 ClassAd。每个资源的 ClassAd 包含资源状态和资源属主的使用策略。ClassAd 被发送到中心管理器。这样 Condor 的调度程序能够发现新的资源并能更新旧资源的状态。Condor 的结构如图 2 所示。

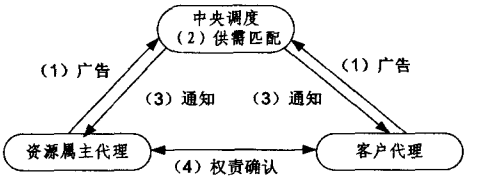

图 2　Condor 的结构

2.4 LCG/EGEE

LCG[5]是支持 E-sciencE 项目的大规模网格应用，其目标是为高能物理组织提供数据存储和分析的基础架构。

LCG/EGEE 框架的信息系统是基于 GT2 提供的 MDS-2。与 GT2 类似，LCG 采用的也是层次化结构，其中位于 GRIS/GIIS 之上的是伯克利数据库信息目录。伯克利数据库信息目录查询给定的虚拟组织中的 GIIS 服务并且用作保持网格状态信息的缓存。用户和其它的网格服务能够查询信息目录，以获得网格状态信息并且搜索需要的资源。每个信息目录以固定频率收集 GIIS 中的信息。通过直接查询 GIIS 和运行在特定资源上的本地 GRIS 能获得及时更新的信息。

3 P2P 系统的资源发现

3.1 非结构化 P2P 系统的资源发现

Napster[6]是历史上第一个获得大规模应用的 P2P 系统。它由一个中央服务器组成，此服务器上存放着所有共享文件的索引。为了定位一个文件，用户向中央服务器发出包含有文件名的查询请求，并接收到存放此文件的对等节点的 IP 地址。在请求节点和存放有此文件的节点之间建立连接，以下载文件。Napster 的资源发现机制是集中式的。集中式的索引服务器不易扩展并且存在单点失效的缺点。

目前的非结构化 P2P 系统中，每个节点都维护着与其邻居的常量数目的连接，因而形成了覆盖网络。Gnutella[7]和 KaZaA[8]是流行的非结构化 P2P 系统，都是通过泛洪的方式进行查询。而泛洪引起了不必要的网络流量，因而扩展性差。为了限制泛洪产生的消息的个数，每个消息中附加了 TTL 字段。TTL 表明了一个查询应该传播的跳数。

为了减少泛洪产生的消息的个数，一些非结构化的 P2P 系统采用了可控的泛洪机制，即动态查询[9]。在 TTL 受限的泛洪机制中，不论是否获得需要的结果，查询请求一直传播到 TTL 值为 0。为了提高查询的准确性，动态查询开始时设置一个小的 TTL 值，并且控制查询的传播，将查询传播到其邻居的一个子集。如果未能得到足够的结果，源节点将增加 TTL 值并继续传播查询请求。这个过程一直进行到接收到令人满意的结果，或者所有节点都被遍历。

还有很多别的技术用来减少泛洪引起的不必要的网络流量[10]。例如，随机漫步算法中，每个节点随机选择一个邻居节点进行转发。随机漫步算法虽然产生非常少的网络流量，每跳只产生一条查询消息，但是减小了网络覆盖率，增加了响

应时间。文献[11]中提出了泛洪和随机漫步算法结合的混和算法。在另一类算法中，查询消息根据某种规则或者统计信息有选择地转发到邻居节点。例如，每个节点选择返回大多数查询响应的头 k 个邻居，或者 k 个容量最大的节点，或者 k 个延迟最小的链接，以转发查询请求[12]。转发索引[13]在每个节点上建立了一个类似路由表的结构，存放了从每个邻居返回的关于每个事先选择的主题的响应。此外，还有查询缓存和结合语义信息的一些技术[14,15]。

实验显示，具有低带宽连接的节点容易被泛洪请求所饱和，因而减慢了非结构化 P2P 系统中的资源发现速度。为了充分利用系统中节点的异构性，低带宽的节点被隔离在查询路由之外。文献[16,17]中构建了两层的层次结构。高带宽的节点，即超级节点，形成了非结构化的覆盖网；而低带宽的节点，即叶子节点，仅与超级节点相连。每个超级节点都存放有所有叶子节点文件的索引信息。任何来自叶子节点的请求都会通过与其相连的超级节点进行转发，而泛洪仅在超级节点形成的覆盖网中进行。这种结构使得系统在保持了非结构化系统简单性的同时，提高了可扩展性。

3.2 结构化 P2P 系统的资源发现

结构化 P2P 系统的典型代表是 CAN[18]，Chord[19]，Pastry[20]和 Tapestry[21]之类的点对点网络。它们的基本思想是将每一个节点的 IP 及端口通过一个哈希函数映射为某个空间(b 位数字空间或者 d 维欧式空间称为“地址空间”)中的一个点，称为该节点的 nodeId。在消息通信时，给定地址空间的一个点作为消息的目标地址，消息在节点之间不断转发，使得收到消息的节点的 nodeId 在地址空间中不断向目标地址逼近，最终到达一个当前所有节点中 nodeId 离目标地址最近的节点，即消息的终点。

分布式哈希表(DHT)是 P2P 系统的一种数据放置策略。对于一份标识为 dataId 的数据，将其 dataId 通过哈希函数映射到地址空间中的某个 hashId，规定该数据必须放置在当前所有节点中 nodeId 离 hashId 最近的节点上。因此结构化 P2P 系统很好地支持了 DHT 的实现。

4 基于 P2P 的网格资源发现系统

4.1 基于非结构化 P2P 的网格资源发现系统

文献[22]中，Iamnitchi 等提出了网格环境下资源发现的一种完全分布的 P2P 结构。在此结构中，虚拟组织中的每一个参与者发布其本地节点资源的信息。这些节点存储并提供对本地资源信息的访问。用户向本地节点发送请求，如果此节点上存放了相匹配的资源，则直接向用户发送响应；否则它将请求转发到其它节点，此请求一直转发到 TTL 值为 0 或者匹配的资源找到。如果某个节点有匹配的资源，则它直接向发出查询请求的源节点发送响应，源节点再向用户发送响应。

文献[23]中提出了一种基于路由转发的网格资源发现模型。每个资源路由器都是对等的节点。资源路由表信息定期更新，解析查询请求时，根据路由表进行转发。

文献[24]中 Talia 等提出了一种符合 OGSA 规范的网格资源发现的 P2P 体系结构。此结构由两层组成，底层由索引服务组成，负责发布每个虚拟组织中的信息；高层是 P2P 层，负责收集和分布信息。P2P 层包括两种符合 OGSA 的 Web 服务：节点服务用来进行资源发现，联系服务将节点服务组织成 P2P 结构。每个虚拟组织中有一个节点服务。每个节点服务与多个节点服务，即它的邻居相连，并以 P2P 方式交换查询/响应消息。节点服务之间通过采用扩展的 Gnutella 协议来交换消息。

文献[25]中 Mastroianni 等采用了超级节点模型设计基于 P2P 的网格信息服务。超级节点模型最开始提出是为了在集中式搜索的高效与分布式搜索的自治、负载均衡和容错性之间达到平衡[26]。超级节点作为普通节点的中央服务器，超级节点之间构成 P2P 覆盖网络。超级节点模型适用于大规模的网格环境。资源发现过程是：某个网格节点发出的查询消息被转发到超级节点。如果超级节点在本地找到所请求的资源，则向请求节点发送响应；如果没找到，则将请求转发到被选择的邻居超级节点，并重复以上过程。

Puppin 等在文献[27]中提出了另一种基于超级节点模型的网格信息服务。网格节点被组织成集群，每个集群包含一个或多个超级节点。系统定义了两种主要的组件：代理和聚合器。代理作为在每个节点上运行的符合 OGSA 规范的网格服务，发布由信息提供者提供的信息。信息提供者周期性地查询资源并将收集的信息存储为服务数据元素。当发布新资源时，它的服务数据名字被广播到集群中的所有聚合器。聚合器作为每个集群的服务器以超级节点的方式工作。聚合器负责收集数据，回答查询，将查询转发到别的聚合器并且存放每个邻居聚合器上的信息的索引。

文献[28]中 Marzolla 等提出了基于路由索引的网格资源发现系统。节点组织成树状结构，每个节点维护着它直接管理的资源信息和以邻居节点为根的子树的资源信息的压缩描述。定位算法利用这些索引路由查询。资源的信息以下列方式进行映射：对于资源的每个属性，属性域被分为 k 个子区域，不同属性的 k 值不同。给定一个资源，属性 A 的索引用 k 位向量表示。某个节点 P 的所有资源的属性 A 的索引通过对所有属性的位向量进行逻辑“或”操作获得。当节点 P 接收到多属性区间查询时，P 将其分解为每个属性的一个子查询集合，每个子查询被映射为与属性的向量位数相同的向量。子查询首先在本地索引中进行匹配，接着在路由索引中进行匹配，并最终路由到索引满足所有子查询的邻居。

表 1 对以上提到的几种基于非结构化 P2P 的网格资源发现系统进行了比较。

表 1 基于非结构化 P2P 的网格资源发现系统的比较

系统	结构	索引	查询解析
Iamnitchi et al.[22]	每个虚拟组织有一个或多个节点	每个节点提供一个或多个资源的信息	随机漫步、基于学习、最佳邻居、基于学习+最佳邻居
Talia et al.[24]	每个虚拟组织有一个符合 OGSA 规范的节点服务	每个虚拟组织中，层次化的索引服务提供了本地资源的信息	消息通过改进的 Gnutella 协议在节点服务之间进行路由
Mastroianni et al.[25]	每个组织中，一个或多个节点作为超级节点	超级节点维护本地组织中所有节点的元数据	查询应转发到哪个超级节点由以前查询的统计信息决定
Puppin et al.[27]	节点组织成集群，每个集群有一个或多个超级节点	每个节点中的代理发布资源信息，信息被广播到节点中的所有超级节点	跳数计数索引用来选择具有最高成功概率的邻居超级节点进行转发
Marzolla et al.[28]	节点组织成树状结构	每个节点维护以邻居节点为根的子树的资源的压缩描述	通过匹配路由索引，找到满足所有子查询的邻居节点

4.2 基于结构化 P2P 的网格资源发现系统

MAAN[29]为资源的每个属性维护一个 DHT。通过保持局部性的哈希函数使得一致分布的属性值空间映射到一致分布的哈希空间。资源的每个属性值对应的节点上都保存着该资源信息的一个副本。每个节点上存放〈属性值,资源信息〉对。多属性查询通过两种方式实现:第一种是迭代的方式,每个查询分解成 M 个子查询,每个子查询分别在对应的属性空间中进行解析,最后在查询的起始节点对所有结果求交集。这是最简单但是最低效的方式。路由跳数为

$$O(\sum_{i=1}^{M}(\log N+N\times S_i))$$

式中,M 为子查询的个数,N 为节点个数,S_i 为子查询 i 的选择因子。第二种方法为单属性控制的路由,即从待查询的属性中选择一个属性,使得查询此属性对应的区间所需的跳数最少。由于资源的每个属性值对应的节点上都存放着〈属性值,资源信息〉对,可以通过资源信息匹配其它的属性。此方法的路由跳数为 $O(\log N+S_{min}\times N)$,$S_{min}$ 是所有属性的 S_i 值中最小的,N 为覆盖网中节点的个数。$S_i=(H(v_h)-H(v_l))/2^m$,其中 $H(v_h)-H(v_l)$ 为属性 i 被查找的范围,m 为哈希值的位数。路由跳数与待查找的属性个数无关。为了构建哈希函数,属性值的分布应该是连续、单调递增而且预先知道的。

文献[30]中,Andrzejak 等提出了一种扩展 CAN 的支持区间查询的网格信息服务。在此结构中,所有的网格资源通过属性集合描述。每个属性根据其类型的不同使用标准的 DHT 或者被扩展的 CAN。属性具有有限值时,由标准的 DHT 处理,而具有连续值时采用扩展的 CAN。为了找到指定属性的资源,每个属性在对应的 DHT 中进行匹配,然后对结果求交集。每个节点存放〈属性值,资源 ID〉对,并负责属性值的一个子区间。

SWORD[31]将属性的值空间通过映射函数映射到 DHT 的 Key 空间。Key 由属性位、属性值位和随机位组成。每个属性被分配到同一 DHT 的一个子区域。设 DHT 的 Key 空间被分成 N 个不重叠的子区域,M 为属性的个数。当 $M=N$ 时,每个子区域对应一个属性;$M>N$ 时,某些子区域对应多个属性;$M<N$ 时,存在一些子区域不对应任何属性。同一资源信息复制到该资源的每个属性值对应的节点上。SWORD 可允许新属性随时加入,资源的属性个数可不固定且无须事先知道。解析多属性查询时,从待查询的属性中选择一个属性,使得查询此属性对应的区间时经过的节点数最少。

XenoSearch[32]在 Pastry 的索引和路由机制之上进行了扩展。它为每个资源属性建立了一个单独的 Pastry 环,以树状的结构存储信息。叶子为 XenoServer 节点,内节点叫做集合点,每个集合点对位于其下的节点的属性值范围进行索引。每个集合点都有一个 Key,集合点的 Key 是其子节点 Key 的前缀。集合点的 Key 映射到 Pastry 上,与此 Key 最接近的 XenoServer 负责维护与集合点相关的信息。解析多属性查询时,将其分解为每个属性的一个子查询。对每个子查询的结果求交集,得到满足条件的资源集合。

INS/Twine[33]中,资源描述和查询请求用属性-值树表示。例如:

```
〈res〉camera
  〈man〉ACompany〈/man〉
  〈model〉AModel〈/model〉
〈/res〉
〈subject〉traffic〈/subject〉
```

其底层的覆盖网用 chord。它将每个资源描述的属性-值对分成多个束。束为资源属性串的前缀子串。假设资源描述的属性-值对共有 a 个,t 为顶层属性的个数,则束的个数为 $s=2a-t$。

例如:如果输入的束为 res-camera-man-ACompany,则

h1=hash(res-camera)

h2=hash(res-camera-man)

h3=hash(res-camera-man-ACompany)

输出的 Key 为 h1,h2 和 h3,将 h1,h2 和 h3 映射到对应的 DHT 节点上。

如果查询请求中有多个束,则对最长的束进行匹配。进行多属性查询时,如果所有节点都正常运行,则查询所需跳数为 $O(\log N)$,N 为节点个数。资源的属性无需事先知道,但不支持区间查询。

Mercury[34]与 SWORD 和 MANN 类似,为每一个属性维护了一个单独的覆盖网。所不同的是,覆盖网不是基于 DHT。Mercury 实现了多属性范围查询,为每个属性创建了一个逻辑的属性中心。每个属性中心由一些节点组成,这些节点构成环状结构。每个节点存储属性值的一段连续的值空间。相邻节点存储的值空间连续。资源信息复制到各个属性中心上。解析多属性查询时,从待查询的属性中选择一个属性,使得对此属性进行区间查询时需要访问的节点数最少。查询请求被送到此属性对应的中心上。路由跳数 $=O(\frac{1}{k}\log^2 N)$。其中,N 为节点个数。

Schmidt 在文献[35]中提出了用单个 DHT 支持多属性查询。多维属性值用空间填充曲线映射到一维空间。每个资源映射到的节点 ID 通过计算属性值获得。查询向与目的 ID 的共同前缀位相比当前节点多一位的节点进行转发。

Ratnasamy 等在文献[36]中提出了利用一致哈希函数在节点间均匀分布存储负载的系统。由于属性值的局部性并未保持,在底层的 DHT 上构建一层覆盖,以有效地进行区间查询。每个属性对应不同的 tier 树,tier 树的根节点分配了该属性的全部值空间。资源向每个属性对应的 tier 树的叶子节点进行注册。初始化时,仅有根节点存在,所有的资源注册到根节点上。资源的个数增多时,根节点创建两个子节点分担负载。tier 树的节点通过一致哈希函数映射到 DHT 节点。DHT 的功能是用来查找负责某一区间的节点,然后此节点向其子树进行广播。

NodeWiz[37]只维护了单个分布式索引,所以更新开销与属性个数无关。NodeWiz 基于 KD 树。初态只有一个节点,将负载由新节点和负载最重的节点进行分担。根据 splitting 算法选出要划分的属性,新节点和负载最重的节点在此属性空间上进行划分。属性空间的划分形成分布式决策树。每个节点的路由表含有层次、属性名、最小值、最大值和 IP 地址这些字段。同一资源的信息只保存到一个节点上,从而减少了存储和更新开销,并能有效地进行负载平衡。

文献[38]中提出了一种基于多层覆盖网结构的网格资源发现机制。将具有同一类型资源的网格信息节点组织在一起,形成域,把节点数量较多的域通过类型粒度组织成层次形结构。利用类型匹配路由的资源搜索技术将资源发现请求转

发到资源所在的域，采用 Top k 技术在相应的域内找出与请求匹配最佳的 k 个资源，从而改善了资源发现的性能，提高了用户的满意度。

文献[39]提出了一种基于索引 P2P 分层的网格资源发现模型。该模型引入了索引 P2P 分层网络，充分利用了网格环境中节点性能的差异，具有较少的查询延迟时间和维护代价，并且对大规模的网格具有较好的适应性。表 2 对几种基于结构化 P2P 的网格资源发现系统进行了比较。

表 2　基于结构化 P2P 的网格资源发现系统的比较

系统	结构	资源注册	查询解析	资源属性
MAAN	每个属性对应一个 DHT	每个属性在对应的 DHT 上注册	对每个子查询分别解析，在查询起始节点求交集	固定且事先知道
SWORD	每个属性分配到同一 DHT 的某个子区域	每个属性注册到 DHT 中对应的子区域	单属性控制的路由	不固定，无须事先知道
XenoSearch	每个属性对应一个 DHT	每个属性在对应的 DHT 上注册	对每个子查询分别解析，在查询起始节点求交集	固定而且事先知道
INS/Twine	资源属性-值对的每个 strand 放到对应 DHT 节点上	对资源的属性-值对的每个 strand 进行哈希，存放到对应的 DHT 节点上	包含好几个 strand 的查询用最长的 strand 解析	不固定，无须事先知道
Mercury	每个属性一个 DHT	每个属性注册在对应的 DHT 上	查找最小区间的属性对应的 DHT	固定且事先知道
NodeWiz	属性空间的划分形成分布式决策树	通过路由表找到合适的节点注册	通过路由表找到合适的节点	固定且事先知道

结束语　本文从网格系统中的资源发现、P2P 系统的资源发现以及基于 P2P 的网格资源发现系统 3 个方面介绍了不同分布式系统中的资源发现机制，并从各方面进行了比较。

参 考 文 献

[1] Foster I, Kesselman C. Globus: A Metacomputing Infrastructure Toolkit[J]. International Journal of High Performance Computing Applications, 1997, 11(2): 115-128

[2] Foster I, Kesselman C, Nick J, et al. Grid Computing: Making the global infrastructure a reality[M]. West Sussex, Wiley, 2003

[3] The WS-Resource Framework Version 1.0 [EB/OL]. http://1www-06.ibm.com/developerworks/library/ws-resource/ws-wsrf.pdf

[4] Litzkow M, Livny M. Experience with the Condor Distributed Batch System[C]// Proceedings of IEEE Workshop on Experimental Distributed Systems. 1990: 122-127

[5] LCG-LHC Computing Grid Project[EB/OL]. http://lcg.web.cern.ch

[6] Napster[EB/OL]. http://free.napster.com/

[7] Gnutella[EB/OL]. http://www.gnutella.com/

[8] KaZaA[EB/OL]. http://www.kazaa.com/

[9] Dynamic Query Protocol[EB/OL]. http://www.the-gdf.org/wiki/index.php? title=Dynamic Query Protocol

[10] Tsoumakos D, Roussopoulos N. A Comparison of Peer-to-Peer Search Methods[C]// Proceedings of the Sixth International Workshop on Web and Databases. San Diego, Florida, USA, 2003: 61-66

[11] Gkantsidis C, Mihail M, Saberi A. Hybrid Search Schemes for Unstructured Peer-to-Peer Networks[C]// Proceedings of IEEE INFOCOM. 2005: 1526-1537

[12] Lv Q, Cao P, Cohen E, et al. Search and Replication in Unstructured Peer-to-Peer Networks[C]// Proceedings of the 16th Annual ACM International Conference on Supercomputing. New York, USA, 2002: 84-95

[13] Crespo A, Garcia M H. Routing Indices for Peer-to-Peer Systems[C]// Proceedings of the 22nd International Conference on Distributed Computing Systems. Vienna, Austria, 2002: 23-34

[14] Sripanidkulchai K, Maggs B, Zhang H. Efficient Content Location Using Interest-based Locality in Peer-to-Peer Systems[C]// Proceedings of IEEE INFOCOM 2003. San Franciso, USA, 2003: 12-23

[15] Zeinalipour Y, Kalogeraki V, Gunopulos D. Exploiting Locality for Scalable Information Retrieval in Peer-to-Peer Systems[J]. Information Systems Journal, 2005, 30(4): 277-298

[16] Chawathe Y, Ratnasamy S, Breslau L, et al. Making Gnutella-like P2P Systems Scalable[C]// Proceedings of ACM SIGCOMM 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication. Karlscruhe, Germany, 2003: 407-418

[17] Yang B, Garcia M H. Designing a Super-Peer Network[C]// Proceedings of International Conference on Data Engineering. Bangalore, India, 2003: 49-60

[18] Ratnasamy S, Francis P, Handley M, et al. A Scalable Content-Addressable Network[C]// Proceedings of ACM SIGCOMM 2001. San Diego, USA, 2001: 161-172

[19] Stoica I, Morris R, Karger D, et al. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications[C]// Proceedings of ACM SIGCOMM 2001. San Diego, USA, 2001: 149-160

[20] Rowstron A, Druschel P. Pastry: Scalable, Decentralized Object Location and Routing for Large-scale Peer-to-Peer Systems[C]// Middleware 2001. Heidelberg, Germany, 2001: 329-350

[21] Zhao Y, Huang L, Stribling J, et al. Tapestry: A Resilient Global-scale Overlay for Service Deployment[J]. IEEE Journal on Selected areas in Communications, 2004, 22(1): 10-25

[22] Iamnitchi A, Foster I, Nurmi D. A Peer-to-Peer Approach to Resource Discovery in Grid Environments[C]// Proceedings of the 11th Symposium on High Performance Distributed Computing. Edinburgh, 2000: 8-15

[23] Li W, Xu Z, Dong F, et al. Grid resource discovery based on a routing-transferring model[C]// Proceedings of the 3rd International Workshop on Grid Computing. Baltimore, USA, 2002: 145-156

[24] Talia D, Trunfio P. Peer-to-Peer Protocols and Grid Services for Resource Discovery on Grids[M]. Grid Computing: The New Frontier of High Performance Computing, Advances in Parallel Computing. Elsevier Science, 2005

[25] Mastroianni C, Talia D, Verta O. A Super-Peer Model for Building Resource Discovery Services in Grids: Design and Simulation Analysis[C]// Proceedings of European Grid Conference. 2005: 132-143

[26] Yang B, Garcia H. Designing a Super-Peer Network[C]// Proceedings of International Conference on Data Engineering. Bangalore, India, 2003: 49-60

[27] Puppin D, Moncelli S, Baraglia R, et al. A Grid Information Service Based on Peer-to-Peer[C]// Proceedings of 11th Euro-Par Conference. Monte de Caparica, Portugal, 2005: 454-464

[28] Marzolla M, Mordacchini M, Orlando S. Resource Discovery in a

Dynamic Grid Environment[C]// Proceedings of DEXA Workshop. 2005:356-360

[29] Cai M, Frank M, Chen J, et al. MAAN: A Multi-Attribute Addressable Network for Grid Information Services[C]// Proceedings of 4th International Workshop on Grid Computing. Phoenix, USA, 2003:184-191

[30] Andrzejak A, Xu Z. Scalable Efficient Range Queries for Grid Information Services[C]// Proceedings of the 2nd International Conference on Peer-to-Peer Computing. Linkoping, Sweden, 2002:33-40

[31] Oppenheimer D, Albrecht J, Patterson D. et al. Distributed Resource Discovery on Planetlab with SWORD[C]// Proceedings of WORLDS 2004. San Francisco, USA, 2004:9-15

[32] Spence D, Harris T. XenoSearch: Distributed Resource Discovery in the XenoServer Open Platform[C]// Proceedings of 12th IEEE International Symposium on High Performance Distributed Computing. Seattle, USA, 2003:216-225

[33] Balazinska M, Balakrishnan H, Karger D. INS/Twine: A Scalable Peer-to-Peer Architecture for Intentional Resource Discovery [C]// Proceedings of the Pervasive 2002. Zurich. Switzerland, 2002:149-153

[34] Bharambe A R, Agrawal M, Seshan S. Mercury: Supporting Scalable Multi-Attribute Range Queries[C]// Proceedings of ACM SIGCOMM 2004. Portland, USA, 2004:353-366

[35] Schmidt C, Parashar M. Flexible Information Discovery in Decentralized Distributed Systems[C]// Proceedings of 12th International Symposium on High-performance Distributed Computing. Seattle, USA, 2003:226-235

[36] Ratnasamy S, Hellerstein J M, Shenker S. Range Queries over DHTs[R]. IRB-TR-03-009. Intel Corporation, 2003

[37] Basu S, Banerjee S, Sharma P, et al. Nodewiz: Peer-to-peer Resource Discovery for Grids[R]. HPL-2005-36. HP Labs, 2005

[38] 张忠平，雷炳银，刘欣媛. 基于多层覆盖网络结构的资源发现机制[J]. 计算机科学，2008，35(3)：103-105

[39] 朱凌，黄德才，郑月锋. 一种基于索引 P2P 分层的网格资源发现模型[J]. 计算机工程与应用，2010，46(2)：96-100

(上接第 7 页)

列数据库在整个数据管理技术领域中的地位如图 15 所示。

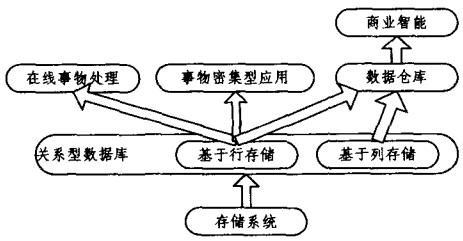

图 15 列数据库在整个数据管理技术领域中的位置

如果 10 年前列数据库只是学术中的一个理论研究的话，如今它已经是一个实实在在的产品，已经在企业决策支持领域使用。不能否认的是，它已经在很多地方展示了自己的特点和优势，已经在竞争激烈的商务智能市场占有一席之地。

列数据库技术还远没有行数据库那么成熟，至今有关它的研究也是数据库领域在学术上一个引人注目的分支，国际上有关的论文频频出现在 SIGMOD, ICDE, VLDB 等重要学术会议上。国内对于列数据库的研究非常欠缺，我们的研究是希望把列存储理念及相关技术引入国内，为我国的数据库领域以及企业决策领域尽一份力量。

列数据库在数据仓库方面有着先天的优势。它即使不会带来数据库领域的革命，也会为商业智能开辟一条新的道路。

参 考 文 献

[1] Copeland G P. A decomposition storage model[C]// SIGMOD '85: Proceedings of the 1985 ACM SIGMOD International Conference on Management of Data. 1985:268-279

[2] 田立中. 列存储在数据挖掘中的应用[J]. 金融电子化，2008(9)

[3] Abadi D J. ColumnStores vs. RowStores: How Different Are They Really? [C]// Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. 2008:967-980

[4] Top Ten TPC-Hby Price / Performance [EB/OL]. http://tpc.org/tpch/results/tpch_price_perf_results.asp

[5] Abadi D J. Integrating Compression and Execution in Column Oriented Database Systems[C]// SIGMOD. Chicago, IL, USA, 2006:671-682

[6] Padmanabhan S, Malkemus T, Agarwal R, et al. Block oriented processing of relational database operations in modern computer architectures[C]// Proceedings of the 17th International Conference on Data Engineering. 2001:567-574

[7] Abadi D J. Materialization Strategies in a Column-oriented DBMS[C]// ICDE. Istanbul, Turkey, 2007:466-475

[8] Abadi D J. Query Execution in Column-oriented Database Systems[C]// SIGMOD. SIGMOD Jim Gray Doctoral Dissertation Award, 2008:145-148

[9] Abadi D J. Column stores for wide and sparse data[C]// CIDR. Asilomar, CA, USA, 292-297

[10] Stonebraker M. C-Store: A Column-oriented DBMS[C]// VLDB. Trondheim Norway, 2005:553-564

[11] Ge Tingjian. Fast, Secure Encryption for Indexing in a Column-oriented DBMS[C]// ICDE. 2007:676-685

[12] Ivanova M. Self-organizing Strategies for a Column-store Database[C]// Proceedings of the 11th International Conference on Extending Database Technology. 2008:157-168

[13] Pranav V. Characterization of TPC-H Queries for a Column-oriented Database on a Dual-core AMD Athlon Processor[C]// Proceeding of the 17th ACM conference on Information and Knowledge Management. 2008:1411-1412

[14] Boncz P A, Kersten M L, Manegold S. Breaking the Memory Wall in MonetDB[J]. Communications of the ACM, 2008, 51(12):77-85

[15] Cornacchia R, Heman S, Zukowski M, et al. Flexible and efficient IR using array databases[J]. VLDB Journal, special issue on IR\&DB integration, 2008, 17(1):151-168

[16] Boncz P A, Grust T, van Keulen M, et al. MonetDB/XQuery: A Fast XQuery Processor Powered by a Relational Engine[C]// Proceedings of the ACM SIGMOD International Conference on Management of Data. Chicago, IL, USA, June 2006

[17] Zukowski M, Boncz P A, Nes N, et al. MonetDB/X100-A DBMS in the CPU Cache[J]. IEEE Data Engineering Bulletin, 2005, 28(2):17-22

[18] 黄鹏，李占山，张永刚，等. 基于列存储数据库的压缩态数据访问算法[J]. 吉林大学学报：理学版，2009(5)

[19] O'Connell S J, Winterbottom N. Performing Joins Without Decompression in a Compressed Database System[J]. ACM SIGMOD Record, 2003, 32(1):6-11