一种基于最小代价的网格资源发现模型*

张忠平[a,b], 贾 倩[a]
(燕山大学 a. 信息科学与工程学院; b. 河北省计算机虚拟技术与系统集成重点实验室, 河北 秦皇岛 066004)

摘 要: 引入域和资源路由节点的概念,设置最小代价函数对网格资源进行分层按域划分,形成一种基于最小代价的分层次网格资源发现模型;并为模型设计了相应的资源路由器节点选择算法、资源注册和查询算法。性能分析和模拟实验结果表明,模型具有良好的可扩展性和容错性,且系统代价低;模型能够在屏蔽网格资源异构性的同时很好地满足其动态性、分布性和扩展性的要求,具有较高的资源搜索性能。

关键词: 网格; 覆盖; 自组织; 代价; 资源发现
中图分类号: TP393 **文献标志码**: A **文章编号**: 1001-3695(2012)12-4683-05
doi:10.3969/j.issn.1001-3695.2012.12.073

Mincost-based grid resource discovery model

ZHANG Zhong-ping[a,b], JIA Qian[a]
(*a. College of Information Science & Engineering, b. Key Laboratory for Computer Virtual Technology & System Integration of Hebei Province, Yanshan University, Qinhuangdao Hebei* 066004, *China*)

Abstract: By introducing the concepts of domain and resource router, and setting the mincost function to group grid resources into domains and layering, this paper proposed a mincost-based hierarchical grid resource discovery model. Then it designed the algorithms for the resource router selection, resource register and resource search. Performance analysis and simulation experiments show that the mechanism has better scalability and fault tolerance, and less cost. The model can meet the dynamism, decentralization, and scalability of grid, simultaneously avoid the heterogeneous among different resources. It also has a good preformance on searching resources.

Key words: grid; overlay; self-organize; cost; resource discovery

0 引言

网格计算将大量地理上分散的异构资源联系在一起形成一个虚拟的网络环境,使得用户能够动态地共享其中的资源,从而有效地提高了资源利用率和系统性能。网格系统中的资源发现机制是网格实现资源共享的关键所在,网格资源发现机制的性能好坏直接决定了网格系统的性能;又因为网格资源数量巨大、网格计算十分复杂,设计一个效率高、代价低、可扩展性好的网格资源发现模型已经变得非常重要。

目前,国内外的很多组织和研究人员对网格资源发现模型从不同角度和不同层次进行了研究,具有代表性的有三种模型:a)以 Globus 为代表的集中式模型[1],该模型采用统一的策略管理网格中的资源,具有良好的全局控制性和较高的资源发现效率,但缺乏在网格环境下的自适应性和可扩展性;b)以 P2P 网络为代表的分布式模型[2~4],这种模型通过不同资源管理系统之间的相互作用对资源进行管理,与集中式模型正好相反,它具有较好的可扩展性却在整体控制上存在不足,且整个模型的通信代价也远远高于集中式模型;c)agent 式模型[5~7],基于多 agent 式模型是通过将 agent 小程序"释放"在所管辖的网络内进行资源的发现和管理,这种方式具有快速、准确、可控性强等特点,但对 agent 本身的要求很高,较为复杂,难以实现。

针对以上几种模型各自的优缺点,结合 P2P 技术和网格自身的特点,设计一种基于最小代价的资源发现模型。该模型是在自组织覆盖网络(self-organize overlay network,SOON)上建立的分层次资源发现模型[8,9],引入最小代价函数,为每一子域选择代价最小的节点作为资源路由节点,从而对域内节点进行管理,在域间通信过程中也设了最小代价机制,极大地提高了整个系统的工作效率。

1 资源发现模型

1.1 SOON 资源发现模型

覆盖网络(overlay network,ON)是一个建立在物理网络上的虚拟网络,它通过虚拟或逻辑链路连接而成,无须大规模改变现有的网格架构就能提供可靠性更强、容错性更好的应用服务。模型采用 SOON 就是为了充分利用覆盖网络的虚拟性优势,采用树型结构与分层技术相结合的方法来组织网格中数量巨大、异构、动态的资源,如图 1 所示。

模型建立在 SOON 上,以域和层为基本逻辑管理单位,通过分类将数量巨大的资源在逻辑上划分成若干区域,这些区域互为兄弟域。每个区域由一个资源路由节点进行管理,资源路由节点负责联系上层资源路由节点,并管理其域内节点(此时,子域也作为域内虚拟节点存在),同时保存其兄弟域的资

收稿日期: 2012-04-11; **修回日期**: 2012-06-05 **基金项目**: 河北省自然科学基金资助项目(F2012203087)

作者简介: 张忠平(1972-),男,吉林松原人,副教授,博士(后),CCF 会员,主要研究方向为网格、XML 数据库和数据挖掘等;贾倩(1988-),女,山西晋城人,硕士,主要研究方向为网格技术和数据挖掘(455237423@qq.com).

源状态信息。资源按类型分别属于不同的域,当一个域中的虚拟节点增大到一定数量时,从这个域中分裂出一个或多个子域,由原来的资源路由节点统一管理各个子域,而各子域再选出一个节点作为其域内资源路由节点,并告知父域的资源路由节点,父域资源路由节点返回其他兄弟域资源路由节点的服务信息,此时资源类型的粒度变小;反之,当一个域中的节点减少到一定数量时,将其合并到其他兄弟域或父域中形成新的合并域,此时资源类型的粒度变大。这样的资源发现模型能够更好地适应网格的动态环境,具有良好的可扩展性。

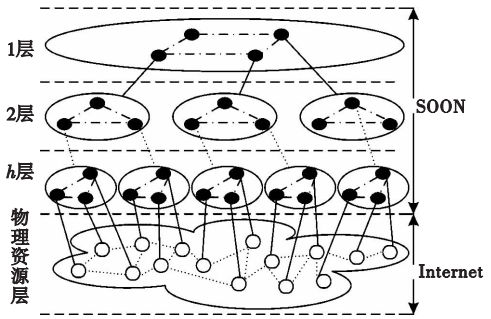

图1 自组织覆盖网络结构

1.2 基本定义

定义 1 资源集合 $R=\langle \mathrm{ID},A,\mathrm{OP}\rangle$。其中,ID 表示全体资源 r 的标志符的集合,A 表示全体资源 r 的全部属性 a 的集合,OP 表示全体资源 r 的全部操作 op 的集合。因此,对于任意的资源 $r_i \in R$ 都可以描述为 $r_i=\langle \mathrm{id}_i, a(r_i), op(r_i)\rangle$。这里,$a(r_i)=\{a_{i1},a_{i2},\cdots,a_{ij}\}$,$\mathrm{op}(r_i)=\{\mathrm{op}_{i1},\mathrm{op}_{i2},\cdots,\mathrm{op}_{ik}\}$。

定义 2 网格 $G=\langle N_G,E,R,T\rangle$。其中,$N_G$ 表示网格中全体节点 n 的集合;E 表示网格中边 e 的集合;R 表示网格中全体资源 r 的集合;T 表示网格中全体资源的类型 t 的集合。若一个资源只属于一种资源类型则记为 $|T(r_i)|=1$,属于多种资源类型则记为 $|T(r_i)|>1$,此时资源可以在多个资源路由器上注册,且其中一个节点为资源注册节点,其他节点为本节点的复制节点。

定义 3 网格节点 $n_v=\langle \mathrm{VID},\mathrm{def},\mathrm{indexlist}\rangle$。其中,VID 表示 ON 中虚拟节点 n_v 的标志符;def 表示节点 n_v 的描述;indexlist 表示节点 n_v 的指针列表。模型为每个资源分配一个全局唯一的 VID 地址,同 IP 地址一样,VID 由域编号(VID_r)和域内节点编号(VID_s)两部分组成,VID_r 和 VID_s 作分段处理,每段代表某种类型的资源。

定义 4 指针列表 $\mathrm{indexlist}=\langle \mathrm{index},\mathrm{data}\rangle$。其中,index 表示节点 n_v 指向某节点的指针;data 表示 index 指向节点的数据描述。Indexlist 包含指向父节点、孩子节点和兄弟节点的指针,即 $\mathrm{Indexlist}=\{\mathrm{index_f},\mathrm{index_s},\mathrm{index_b}\}$。

定义 5 资源发现模型 $G_V=\langle N_V, rf, \mathrm{IndexList}\rangle$。其中,$N_V$ 表示模型中所有节点 n_v 的集合;rf 为资源注册函数,2.2 节详细说明其工作原理;IndexList 表示模型中指针列表集合。

定义 6 定义模型中的非叶子节点为资源路由器(resource router,RR),即对于任意的 $n_{vi}\in \mathrm{RR}$,有 $\mathrm{index_s}(n_{vi})\neq\varnothing$。且 $\forall n_{vi}\in \mathrm{RR}$,其域内节点编号为 0,即 $\mathrm{VID}(n_{vi})=\mathrm{VID}_r(n_{vi})$,$\mathrm{VID}_s(n_{vi})=0$。

模型中的节点 n_v 具有以下性质:a) $\forall n_{vi}\in N_V$,若 $\mathrm{indexlist}(n_{vi})\neq\varnothing$,则 $\exists\, \mathrm{indexlist}_i\in \mathrm{IndexList}$;b) $\forall n_{vi}\in N_V$,若 $\mathrm{index_f}(n_{vi})=\varnothing$,则 n_{vi} 为模型中的顶层资源路由器节点;c) $\forall n_{vi}\in N_V$,若 $\mathrm{index_s}(n_{vi})\neq\varnothing$,则 $n_{vi}\in \mathrm{RR}$;d) $\forall n_{vi}\in N_V$,若 $\mathrm{index_s}(n_{vi})=\varnothing$,则 n_{vi} 为模型中的叶子节点;e) $\forall n_{vi}\in N_V$,有 $\mathrm{VID}_r(n_{vi})=\mathrm{VID}(\mathrm{index_f}(n_{vi}))$,$\mathrm{VID}(n_{vi})=\mathrm{VID}(\mathrm{index_f}(n_{vi}))+k$,其中,$\mathrm{VID}_s(n_{vi})=k$,$k$ 为一整型数。同理可知,$\mathrm{VID}(n_{vi})=\mathrm{VID}_r(\mathrm{index_s}(n_{vi}))$。

模型中的非叶子节点(即资源路由器 RR)只负责对以它为根的资源子树进行管理,并为其上层节点提供该子树的信息。在逻辑上,资源发现模型区分叶子与非叶子节点;在实际应用中,每个资源节点都体现为叶子节点,非叶子节点实质是物理节点为组建模型而提供的服务。

2 模型实现与维护

2.1 模型中 RR 节点的服务部署

资源发现模型的构造是资源管理机制的核心技术,首先必须由网格系统运营商提供一些高性能物理节点在 SOON 上构成最高层资源路由器,并在这些节点上部署网格资源管理服务[10],从而形成模型最基本的结构。

一般情况下,当有新的节点进入网格系统时,只需要根据它的资源属性将其分配给某个域即可。但是当该域内的节点数达到一定程度时,新节点的到来将会导致域的分裂,即生成新的子域,此时系统必须为新形成的域挑选出合适的资源路由节点。

为了保证系统的工作效率和服务质量,在选择资源路由节点 RR 时,应该优先考虑花费代价最低的节点,这是因为在响应和处理任务时,这样的节点付出的计算代价和通信代价很小,相应的系统负担也会很轻。而模型中的资源路由节点与其他普通节点相比参与响应和处理任务的次数要多很多,因此使用一个好的资源路由节点选择标准生成一个合理的 RR 服务部署非常重要。

为保证选择 RR 的合理性,引入最小代价函数 mincost(),定义式(1)作为选择 RR 的标准:

$$\mathrm{mincost}(n_v)=\sum_{j=1}^{n}\delta_j\times k_j w(a_j) \tag{1}$$

其中,"最小代价"的意义在于某一节点作为资源路由节点付出的代价相比较域内其他节点是最小的;$\{a_1,a_2,\cdots,a_n\}$ 是节点 n_v 的属性子集;δ_j 是属性 a_j 对于网格系统工作效率的影响因子,且 $\sum_{j=1}^{n}\delta_j=1$;k_j 为归一转换系数;$|w(a_j)|$ 是属性 a_j 的权值,并且当 $|w(a_j)|$ 越大系统效率越高时 $w(a_j)=-|w(a_j)|$,当 $|w(a_j)|$ 越大系统工作效率越低时 $w(a_j)=|w(a_j)|$;选择的属性 a_j 应该是能够极大地影响到系统工作效率的,如网络带宽、运行速度、生存时间等。

当有新节点 n_{new} 进入网格系统时,系统会将节点信息交给其所属域的 RR,RR 首先计算 $\mathrm{mincost}(n_{\mathrm{new}})$ 的值,然后比较 $\mathrm{mincost}(n_{\mathrm{new}})$ 与 mincost(RR)。如果 $\mathrm{mincost}(n_{\mathrm{new}})<\mathrm{mincost}(\mathrm{RR})$,则表示新节点比原来的资源路由节点更适合做 RR,要将其取代,此时 RR 需要通知其父节点、兄弟节点、孩子节点 RR 已经改变,并将其保存的这些节点信息统统转交给新的 RR;如果 $\mathrm{mincost}(n_{\mathrm{new}})>\mathrm{mincost}(\mathrm{RR})$,则只做普通的节点进入处理。当某个域需要分裂成多个子域时,系统只需在每个域中选择其 mincost()值最小的节点作为其 RR,如果有某几个域合并,新的 RR 必然是所有参与合并的域中 mincost()最小的

那个 RR。

上述资源路由节点选择的过程可以用算法 1 资源路由节点选择算法(resource router selection algorithm, RRSA)进行描述。

算法 1　RR 选择算法

```
RRSA(mincost(RR),NO,NO_sup,n_new)
begin
n_new→RR,a(r_i)→RR;
//新节点的资源属性信息转交给 RR
mincost(n_new) = Σ_{j=1}^{n} ε_j × k_j × w(a_j);
//计算新节点的最小代价函数值
if( + + NO < NO_sup)
//新节点的进入不导致域分裂
{if(mincost(n_new) < mincost(RR))
{//新节点比 RR 更适合做资源路由节点
  index_f(n_new) = index_f(RR),
  index_s(n_new) = index_s(RR),
  index_b(n_new) = index_b(RR);
  //RR 将父亲、兄弟、孩子信息告知 n_new
  RR 将其管理的域内资源信息、兄弟域资源信息告知 n_new;
  RR 告知其父亲、兄弟、孩子将被 n_new 取代;
  index_f(RR) = n_new,分配 VID_s 给原来的 RR;
  新的 RR 的域内编号为 0,RR 更新完毕;
}
else
  //新节点没有 RR 适合作为资源路由节点
  index_f(n_new) = RR,为 n_new 分配域内编号;
}
else //需要进行域分裂
按某种资源属性从域内全部资源节点中划分出一个或多个子域,
并为各个子域挑选出 mincost()值最小的资源节点作为其 RR;
end
```

2.2　资源注册与信息更新

资源注册和信息更新是资源管理模型必须重点考虑的问题。在模型中,域是按资源类型来划分,资源类型依据资源属性来确定,对于每个资源来说,无论是静态还是动态属性,它们都只能活动在某个特定的区域[11]。

1)资源注册函数 $rf()$　假设属性 a_j 的权值 $w(a_j) \in [c, d]$,在区间 $[c,d]$ 中插入 m 个控制点 c_k,令 $c = c_0 < c_1 < \cdots < c_{m-1} = d$,则区间 $[c,d]$ 被划分成 $m-1$ 段,每一段占用一个字符位,得到一个 m 位的二进制串 $b = b_0 b_1 \cdots b_{m-1} = 00\cdots0$,对于任意资源 r_i 的属性 $a_j(r_i)$ 可以用式(2)进行二进制编码:

$$b_k = \begin{cases} 1 & w(a_j(r_i)) = [c_k, c_{k+1}) \\ 0 & \text{else} \end{cases} \tag{2}$$

根据式(2)有 $rf(a_{ji}) = b_1 \cdots b_k \cdots b_{m-1} = 0\cdots1\cdots0$,即资源的属性权值在区间的某一分段上,该分段对应的字符位就用 1 表示,其他字符位用 0 表示。若资源 r_i 具有 $a_1, a_2, \cdots, a_n$ 多个属性,则它的资源注册函数应该为 $rf(a(r_i)) = rf(a_1(r_i))rf(a_2(r_i))\cdots rf(a_n(r_i))$。

2)资源注册　当一个节点 n_i 发布资源时,该节点用统一资源描述语言向距离最近且加入模型的节点 n'_i 发送注册请求,n'_i 向其 RR 转发请求,RR 根据 $rf()$ 产生一个 VID_r,模型根据 VID_r 将资源加入同类资源域,分配 VID_s,生成该资源的逻辑节点 n_{vi},置 n_{vi} 服务状态为有效,并返回资源节点注册信息的副本。模型中资源注册算法(resource register algorithm, RRA)如算法 2 所示。

算法 2　资源注册算法

```
RRA(a(r_i),ttl) //ttl:请求消息的生命周期
begin
a(r_i)→n'_i→RR;
//节点 n_i 向系统发布资源
VID_ri = rf(a(r_i));
//VID_r 是当前域 RR 的标志符
while((VID_ri ∧ VID_r) ≠ VID_r)
//搜索与 VID_ri 类型匹配的资源域
{
ttl - -,若 ttl < 0,则返回 ttl 结束标志;
if(all (index_b(VID) ∧ VID_ri) ≠ index_b(VID))
{//n_i 的资源类型与其他兄弟域不匹配
if(index_f(VID_r) ≠ null)
  //向父节点传送资源注册信息
  VID_ri→index_f(VID_r),
  VID_r = index_f(VID_r).VID_r;
else
  //向兄弟节点传送资源注册信息
  VID_ri→index_b(VID_r),
  VID_r = index_b(VID_r).VID_r;
}
else
//n_i 的资源类型与某一兄弟域匹配
VID_ri→index_b(VID_r),
VID_r = index_b(VID_r).VID_r;
}
while(index_s(index_s(VID_r)) ≠ null)
//资源注册信息向下传递
{
ttl - -,若 ttl < 0,则返回 ttl 结束标志;
VID_ri→index_s(VID_r),
VID_r = index_s(VID_r).VID_r;
}
执行 RR 选择算法为 r_i 分配 VID_si;
r_i 加入模型,op(n_vi) = true;
返回 VID_i 和注册副本;
end
```

3)资源信息更新　资源节点 n_{vi} 在运行期间,发现当前资源信息与副本信息不一致时,向父节点 RR 发送更新信息,更新信息中应包含当前的属性 $a'(r_i)$ 和状态 $op'(n_{vi})$,父节点根据接收的更新信息判断 n_{vi} 的类型是否发生变化。如变化则对 n_{vi} 的类型进行解析处理并返回处理结果,n_{vi} 向其保持的邻居广播结果。

为了节省系统的报文传输代价,本文引入一个临界值 lim,当且仅当域中有 lim 个以上的资源状态改变时,RR 才会通知它的父域与兄弟域其域内信息发生改变。当某一 RR 接收到来自子域或兄弟域的资源更新信息包时,RR 只需要根据信息包的内容修改其维护的子域资源信息表和兄弟域资源信息表。

2.3　资源查找算法

用户提出资源请求后,网格系统能够很快找到与这一请求相匹配的资源域,由资源域中的 RR 负责为其分配最为适合的

资源。而资源域中的资源可以统一用一个 m 维的属性权值坐标 $\{w(a_{m-1}),\cdots,w(a_1),w(a_0)\}$ 来描述,整个资源域也可以映射成为一个 m 维坐标空间,资源请求 $r=\{w(a_{m-1}),\cdots,w(a_1),w(a_0)\}$ 也可以虚拟为该坐标空间当中的一个点。此时为它进行资源匹配就是在坐标空间中查找与它距离最近的 k 个点。

欧式空间中的欧式距离式,可以将 m 维资源属性查询转换为一维距离查询,从而满足网格资源发现机制的多属性查询原则。设 d_i 表示资源 $r_i=(w_i(a_{m-1}),\cdots,w_i(a_1),w_i(a_0))$ 与资源请求点 r 的距离,d_i 可表示为

$$d_i=\sqrt{\sum_{j=0}^{n}\delta_j k_j(w_i(a_j)-w(a_j))^2} \tag{3}$$

其中,δ_j、k_j 和 $w_i(a_j)$ 的描述同式(1)相似,不同之处在于:$w_i(a_j)$ 表示资源 r_i 的属性 a_j 的权值;而 $\delta_j\in[0,1]$,当 $\delta_j=1$ 时,表示资源 r_i 的属性 a_j 对请求执行的任务的影响是绝对的。例如,当属性 a_j 表示操作系统时,用户提出请求"Windows",则查找到的资源不能为其他操作系统,否则用户提交的任务可能无法正常执行,此时属性"操作系统"对任务的执行情况起绝对的影响作用,即 $\delta(\mathrm{OS})=1$;而为了不影响距离公式的可用性,对于影响度 $\delta=1$ 的属性,模型将统一将其归一转换因子 k_j 赋值为固定值 $d_{\lim}$。$d_{\lim}$ 的意义在于:当 $d_i>d_{\lim}$ 时,模型认为资源 r_i 不符合资源请求 r 的要求,不能为其提供服务;如果一个资源请求 r 与某一匹配的资源域内的全部资源的距离都大于 $d_{\lim}$,则表示这一次资源搜索失败。

模型使用的资源查找算法(resource search algorithm, RSA),如算法 3 所示。

算法 3 资源查找算法

```
RSA(W,request(R_i(A)),k,ttl)
begin
用户向模型发送 request(R_i(A));
r_i(a) = request(R_i(A));
VID_i = rf(r_i(a));
while((VID_i ∧ VID) ≠ VID)
//搜索匹配资源域
{ttl - -,若 ttl <0,则返回 ttl 结束标志;
if(all (index_b(VID) ∧ VID_i) ≠ index_b(VID))
{//兄弟域中无匹配的资源域
if(index_f(VID) ≠ null)
RR 向父亲传递 request(R_i(A))和 VID_i;
else
  RR 向兄弟传递 request(R_i(A))和 VID_i,采用最佳邻居策略;}
else //某一兄弟域是匹配的资源域
  RR 向兄弟传递 request(R_i(A))和 VID_i;}
while(index_s(index_s(VID)) ≠ null)
//寻找匹配资源的父节点 RR_j
{ttl - -,若 ttl <0,则返回 ttl 结束标志;
RR 向孩子节点传递请求和 VID_i,请求在域内采用最佳邻居策略传递;}
RR_j 向孩子节点转发请求;
叶子节点 r_j 计算与 r_i(a)距离,并返回 d_j;
返回距离最小的 k 个资源;
end
```

3 模型性能分析

3.1 容错性能分析

在构造分层次结构时,引入最小代价函数 mincost( )作为资源路由节点选择的标准,在每个资源域内选取 mincost 函数值最小的节点作为资源路由节点,这样在当前的资源路由节点发生故障或者退出网络时,资源域只需选取 mincost 函数值次小的节点作为新的资源路由节点就能保证工作的正常进行,克服了传统的分层次网格资源发现模型对资源路由节点依赖性强的弱点,其容错性能更强。

3.2 网络流量分析

在分布式的环境中,随着网络规模的扩大,网络中资源信息更新和查询产生的消息数量也快速增长。网络中消息数量是评价网络性能的重要因素,资源发现模型中消息来源主要包括域内信息转发和高层路由传递两部分。

设总消息个数为 N,域内消息个数为 N_i,高层路由传递产生的消息个数为 N_r,则 $N=N_i+N_r$。模型中顶层路由消息传递时采用最佳邻居的转发策略,所以产生的消息个数的最大值为 $\max(N_r)=\log_{d'}^{n}+2(h-2)\leqslant \mathrm{ttl}$。其中,$h$ 表示模型的高度,d' 表示模型中节点的度,n 表示顶层节点的个数。域内消息个数的最大值为 $\max(N_i)=\sum_{i}^{\mathrm{ttl}}d'_i\leqslant n_i$,其中 $i=\mathrm{ttl}-(\log_d^n+2(h-2))$,$n_i$ 表示最底层域内节点个数。根据以上分析可知,一次信息更新或查询中所产生的最大消息个数为

$$\max(N)=\log_d^n+2(h-2)+\sum_{i=\mathrm{ttl}-(\log_d^n+2(h-2))}^{\mathrm{ttl}}d'_i$$

$$\max(N)\leqslant \mathrm{ttl}+n_i \tag{4}$$

根据式(4)可知,一次信息传递过程产生的最大消息个数与 ttl 值和最底层资源域的节点个数有关,与总的资源节点的个数无关。因此,合理地划分底层的资源域,调整 ttl 值的大小可以控制整个网络中消息的数量。

3.3 模型资源发现成功率

为了更好地描述模型的性能,引入资源密度的概念。

定义 7 资源密度(resource density, RD)是满足查找条件资源数量占全体资源总数的比例,即 $d=|n_r|/|n|$。其中:d 表示与用户请求对应的资源 r 的密度,精度单位为 1/1000;n_r 表示与请求匹配的资源节点集合;n 表示全体资源节点集合。

资源发现成功率 p_s 是指一次资源请求在消息生命周期 ttl 内查找到匹配资源的成功率,可用式(5)来表示 p_s,即

$$p_s=1-(1-p_m)^{d\times n_i\times n(\mathrm{ttl})} \tag{5}$$

其中:p_m 表示某类资源与请求匹配的成功率,其大小受到匹配机制和资源描述方法等诸多因素的影响;d 表示与用户申请对应的资源 r 的密度;n_i 表示最底层域内节点个数;$n(\mathrm{ttl})$ 表示 ttl 跳时节点的平均邻居数。

根据 Faloutsos 的幂率定理,$n(\mathrm{ttl})$ 是关于 ttl 和指数 e 的函数 $n(\mathrm{ttl})=(d'+1)\mathrm{ttl}^e-1$,则式(5)可表示为

$$p_s=1-(1-p_m)^{d\times n_i((d'+1)\mathrm{ttl}^e-1)} \tag{6}$$

其中 d' 为节点平均度数。

对于确定的资源发现模型,p_m 值大小固定,因此,要提高资源发现的成功率,就应适当增加资源的密度 d、节点的度 d'、最底层域内节点个数 n_i 或 ttl。根据 3.2 节分析可知,如果过多地增加 n_i 和 ttl,又会急剧增加网络中消息的数量。增大节点的度 d 可以有效地提高资源发现的成功率,但也会增加网络的复杂性。

3.4　RSA 仿真实现

采用 GridSim[12] 仿真工具包对本文提出的资源发现机制与经典的采用 P2P 形式的基于属性查询的 flooding 资源发现方法[13] 进行仿真分析,仿真程序运行在 Windows XP 平台上,使用 JDK 和 JCreator 作为编辑环境。

3.4.1　实验环境设置

在实验中只模拟网格环境中的计算资源,暂时不考虑其他的网格资源;在仿真过程中,忽略一次资源发现过程中网络拓扑的动态变化,且网络中所有的资源信息保持稳定和有效。一次资源发现的过程与网格资源环境的变化过程相比是很短的。根据系统展现出来的宏观统计特性进行仿真,忽略某一小段时间内系统的各种细节的动态变化,可以在把握系统本质的情况下有效简化问题。

设模型中资源节点的数量为 50 000,即 $|N_G| = 50\ 000$, $|R| = 50\ 000$;资源类型数为 10,即 $|T(R)| = 10$,各种类型的资源均匀分布;同时,模型中的节点 n_{vi} 需满足三个条件:

a) $\forall n_{vi} \in N_V, \exists \text{index_f}(n_{vi}) \in \varnothing$,则 $50 \leqslant |n_{vi}| \leqslant 5000$;

b) $\forall n_{vi} \in N_V, \forall \text{index_s}(n_{vi}) \in \varnothing$,则 $50 \leqslant |n(\text{index_f}(n_{vi}))| \leqslant 500$;

c) $\forall n_{vi} \in N_V$,有 $d'(n_{vi}) \leqslant 4$,其中,$d'(n_{vi})$ 表示模型中节点 n_{vi} 在某层与其兄弟节点的度。

在 flooding 资源发现机制中,设节点间的平均度数 d' 为 4,采用 TTL 的方式控制网络中请求信息的生存周期。

3.4.2　仿真结果分析

在仿真实验过程中,采用不同的用户请求进行多次实验,仿真结果取实验的平均值。

资源发现过程中涉及的节点数、响应时间和查找成功率是评价资源搜索算法的三个重要性能指标。本文主要通过这三个指标及用户对返回资源的满意程度四个方面对 RSA 与 flooding 资源发现算法进行比较。

1)涉及节点数的比较　如图 2 所示,为两种资源发现算法在资源查找时涉及的节点数的比较。从图 2 中可以看出,随着资源密度的增加,RSA 涉及到的节点数远远少于 flooding 算法,尤其当资源密度较小时,RSA 具有更加明显的优势。

2)响应时间的比较　响应时间是指用户从发出资源请求到收到成功应答之间所需要的平均时间。基于最小代价的资源路由选择策略决定了资源路由节点的处理时间与请求在转发路径上的延迟相对较小,又因为网络延迟具有不确定性,所以在实验中,对响应时间的判断主要根据服务请求转发所经过路径的跳数来决定。

如图 3 所示,为两种资源发现算法在查找跳数上的比较。从图 3 中可以看出,本文提出的 RSA 的查找跳数要明显少于 flooding算法的查找跳数;但随着网络中资源密度的增大,系统中满足请求的资源数量增加,两种资源发现算法的跳数趋向平稳、一致。

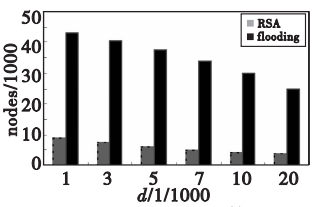

图2　涉及节点数

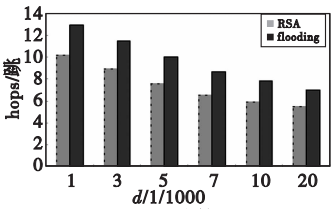

图3　跳数

图 3 中的 RSA 在跳数上并没有太大优势,但综合查找跳数和涉及节点数两个因素来分析,RSA 在响应时间上的优势就明显地增加。Flooding 算法在请求转发时采用广播的方式,资源查找过程中涉及节点数较多,造成网络中的通信量增多,网络延迟增大,响应时间延长;RSA 在资源查找时采用最佳邻居的转发策略,请求传递时涉及节点数较少,网络延迟较小,请求响应时间较快。

3)查找成功率的比较　如图 4 所示,为两种资源发现算法在查找成功率上的比较。从图 4 中可以看出,两种算法随着消息生命周期的增加总可以找到所需要的资源,RSA 在 TTL 值较小时更能有效地发现资源。

4)用户满意度的比较　在资源查找时,flooding 算法采用的是属性对比的资源搜索方法,它往往给用户返回属性偏差过大的结果,即 $|a_{return} - a_{request}| \to \infty$,这个结果一般对用户来说并非是最佳的,而且资源节点和请求节点间的通信效率也得不到保证;RSA 能根据用户的请求一次返回 k 个最合适的资源,即 $|a_{return} - a_{request}| \to 0$,用户可以根据需要挑选或系统推荐返回结果中最佳的且通信效率最高的资源,所以 RSA 具有较好的用户满意度。

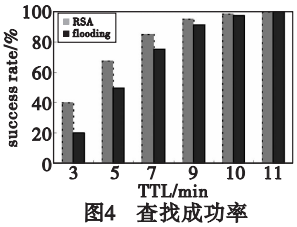

图4　查找成功率

由以上仿真结果可以看出,RSA 可在数量巨大的资源中,以较少的节点访问量和跳数取得较好的查询效率,同时还可以查找到最合适的 k 个资源。

4　结束语

根据网格环境的特点,结合 SOON 网络和分层次网格资源发现机制的特点,提出了一种基于最小代价的网格资源发现模型。该模型根据资源类型对网格资源进行分层按域划分,并设置最小代价函数作为选取资源路由节点的标准。同时讨论了模型的路由节点服务部署、资源注册与更新,以及资源发现等资源管理的相关问题。性能分析和仿真实验表明,基于最小代价的资源发现模型的容错性强、可扩展性好、系统代价低。本模型能很好地满足网格动态性、分布性和扩展性的要求,并能屏蔽资源间的异构性,同时具有较高的资源搜索性能,因此是有效可行的。下一步的工作是对最小代价函数作进一步改进,使其更加灵活,并能够更加合理地对资源路由节点作出选择。

参考文献:

[1] CZAJKOWSKI K, FITZGERALD S, FOSTER I. Grid information services for distributed resource sharing[C]//Proc of the 10th IEEE International Symposium on HPDC. Washington DC: IEEE Computer Society, 2001: 181.

[2] IAMNITCHIX A, FOSTER I, NURMI D. A peer-to-peer approach to resource discovery in grid environments[C]//Proc of the 11th Symposium on High Performance Distributed Computing. Norwell: Kluwer Academic Publishers, 2002: 2-15.

[3] JUNG J J. Trustworthy knowledge diffusion model based on risk discovery on peer-to-peer networks[J]. Expert Systems with Applications, 2009, 36(3): 7123-7128.

(下转第 4690 页)

3.2 参数设定

实验中采用的网络节点数共 500 个。定义节点属性由生存时间 t 和离开概率 ρ 组成,生存时间 t 服从正态分布,均值为 2 300 s,范围在 200 ~ 3600 s,如图 3 所示。节点离开概率 ρ 使用 OverSim 提供的 Pareto 动态流失模型。设 $X = \{0.4t, 0.6\rho\}$,边权值 w 设为时延参数。

3.3 实验结果分析

在不提供失效检测和恢复机制的情况下,对新算法生成的组播树在运行期间由于节点失效影响累积范围进行统计。由新算法生成的组播树由于高层节点具有较强的稳定性,因此高层节点失效概率减小。由图 4 可以看出,新算法组播树累计受影响节点所占总节点比例与 NICE 协议相比减少约 10%。

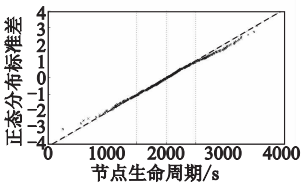

图3 500个网络节点生存时间的分布情况

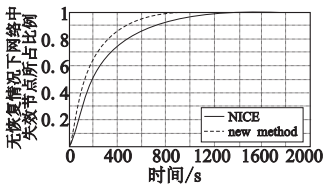

图4 累计节点失效比例对比

由于新算法采用了逐层分簇最小生成树连接方式,新算法生成组播树较 NICE 协议具有更小的加权时延总和。由图 5 所示的时域比较可以得出,在网络运行期间,新算法生成树较 NICE 协议平均时延小。

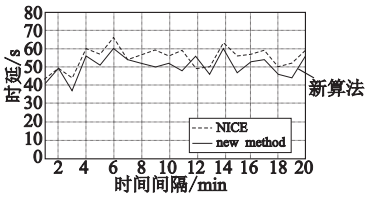

图5 平均时延的比较

4 结束语

实验结果表明:基于多维节点属性通过层次聚类应用层组播生成树算法能优化非稳态组播树结构,提高网络稳定性。仿真实验选取了生存时间和离开概率这两个节点属性,运用多维节点属性通过层次聚类应用层组播生成树算法生成传输路径,网络稳定性提高了约 10%,明显减少了由于节点离开造成的网络时延。在未来的研究中,笔者将完善组播树的维护机制应对生成树对节点动态处理的场景,并进一步优化算法,在实际多媒体通信中进行应用推广。

参考文献:

[1] 苏金树,曹继军,张博锋. 应用层组播稳定性提高技术综述[J]. 计算机学报,2009,32(3):576-590.

[2] RONG Bin, KHALIL I, TARI K. Making application layer multicast reliable is feasible[C]//Proc of the 31st IEEE Conference on Local Computer Networks. 2006: 483-490.

[3] ALKUBEILY M, BETTAHAR H, BOUABDALLAH A. A new application-level multicast technique for stable, robust and efficient overlay tree construction[J]. Computer Networks, 2011, 55(15): 3332-3350.

[4] SRIPANIDKULCHAI K, GANJAM A, MAGGS B, *et al*. The feasibility of supporting large-scale live streaming application with dynamic application end-points[J]. ACM SIGCOMM Computer Communication Review, 2004, 34(4): 107-120.

[5] 韩加炜,KAMBER M. 数据挖掘概念与技术[M]. 范明,孟小峰,译. 2 版. 北京:机械工业出版社,2007: 251-272.

[6] 雷蕾, 张春环, 纪越峰. 分层网络中用于生成树抽象的优化方法[J]. 北京邮电大学学报, 2006,29(2):55-58.

[7] BAUMGART I, HEEP B, KRAUSE S. OverSim: a flexible overlay network simulation framework[C]//Proc of IEEE Global Internet Symposium. [S. l.]: IEEE, 2007: 79-84.

[8] BANERJEE S, BHATTACHARJEE B, KOMMAREDDY C. Scalable application layer multicast[J]. ACM SIGCOMM Computer Communication Review, 2002, 32(4): 205-217.

[9] RZADCA K, YONG J T T, DATTA A. Multi-objective optimization of multicast overlays for collaborative applications[J]. Computer Networks, 2010, 54(12): 1986-2006.

[10] MATHY V, CANONICO R, SIMPSON S, *et al*. Scalable adaptive hierarchical clustering[C]//Lecture Notes in Computer Science, Vol 2345. London: Springer-Verlag, 2006: 1172-1177.

(上接第 4687 页)

[4] 王毅,张毓森,刘鹏. 一种新的基于 P2P 系统的网格资源信息发现方法[J]. 计算机研究与发展,2008,45(4):725-733.

[5] JI Peng, LUO Jun-zhou. A feedback and investigation based resource discovery and management model on computational grid[C]//Proc of the 2nd International Workshop on Grid and Cooperative Computing. Heidelberg: Springer, 2004: 211-214.

[6] 蒋伟进,王璞. 基于 MAS 市场机制的动态计算资源调度模型研究[J]. 计算机研究与发展,2007,44(1):29-36.

[7] LONG N V, WANG Sheng-zu. Resource-grabbing by status-conscious agents[J]. Journal of Development Economics, 2009, 89(1): 39-50.

[8] JAMALI M A J, SANI Y. Adaptive peer-to-peer resource discovery in grid computing based on reinforcement learning[C]//Proc of the 12th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing. Heidelberg: Springer, 2011: 191-196.

[9] CHEN Dong, CHANG Gui-ran, ZHENG Xiu-ying, *et al*. A novel P2P based grid resource discovery model[J]. Journal of Computers, 2011, 6(10): 1390-1397.

[10] 王伟,曾国荪. 构造基于信任机制的自组织资源拓扑[J]. 计算机研究与发展,2007,44(11):1849-1856.

[11] MARZOLLA M, MORDACCHINI M, ORLANDO S. Resource discovery in a dynamic grid environment[C]//Proc of the 16th Internation Workshop on Database and Expert Systems Applications. [S. l.]: IEEE, 2005: 356-360.

[12] BUYYA R, MURSHED M. GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing[J]. Journal of Concurrency and Computation: Practice and Experience, 2002, 23: 13-15.

[13] KUTTEN S, PELEG D, VISHKIN U. Deterministic resource discovery in distributed network[C]//Proc of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures. New York: ACM Press, 2001: 77-83.