基于蚁群算法的对等网模拟器的设计与实现

赵晓怡 杨明福 黄桂敏
(华东理工大学信息科学与工程学院 上海 200237)

摘 要 对等网(Peer to Peer,简称P2P)技术是在Internet上的一种分散式控制的网络技术,它将逐渐取代客户/服务器结构,使Internet的应用从中央服务器模式向网络设备边缘化方向发展。本文针对目前对等网应用范围的多样性和系统设计时性能评估难的特点,结合蚁群算法提出了一种辅助分析对等网应用方案性能的模拟器,同时深入地讨论了模拟器的设计及实现过程。

关键词 对等网 蚁群算法 模拟器 节点

DESIGN AND IMPLEMENTATION OF PEER-TO-PEER NETWORK SIMULATION TOOL BASED ON ANT COLONY SYSTEM

Zhao Xiaoyi Yang Mingfu Huang Guimin
(*Information Engineering College. East China University of Science and Technology, Shanghai 200237*)

Abstract Peer-to-peer systems are characterized by decentralized control. It's going to replace the Internet application development from client-server structure to the Internet edge pattern gradually. Due to the variety of the peer-to-peer application and the difficulty of evaluation, we put forward an implementation for a peer-to-peer application simulation tool designed by ant colony system.

Keywords Peer-to-peer Ant colony system Simulation Peers

1 引 言

对等网技术是Internet上出现的一种分布式计算技术。它和流行的客户/服务器模式不同的是:对等网技术是完全分布式的,所有的对等网节点的地位是平等的,每一个节点既可以是请求服务的客户机,也可以是提供服务的服务器。一个客户的计算机可以通过Internet直接和网络上其他客户的计算机交换文件和信息,不需要通过服务器去浏览、下载。因此,对等网技术使网络应用的核心从中央服务器向网络边缘的终端设备扩散,降低了网络应用的硬件和软件成本,提高了通信效率,避免了服务器瓶颈问题,可以说是Internet的一场全新的技术革命。

从应用的角度看,对等网技术目前主要有四类应用:对等计算、协同工作、搜索引擎、文件交换,此外它也是网格计算的重要基础技术之一。但是,目前一些流行的对等网技术应用都是采用传统的网络技术来实现,具有许多局限性。例如:Napster[1],每个对等网节点之间的信息传输需要中心服务器来协调,虽然服务器不再是网络的中心,但是服务器的瘫痪仍然会导致整个网络系统的瘫痪;Gnutella[2],虽然它的结构是分布式的,但是它采用广播方式来传输信息,使得传输的信息量呈指数增长,其可扩展性受到了严重的挑战。

随着Internet的用户数量的不断增长,利用传统网络技术设计的对等网应用,迫切需要引入新的方法来改进其设计。那么在设计对等网应用方案时,如何评价该方案的可行性及性能呢?这是一个需要解决的问题。本文利用蚁群算法设计了一种对等网应用方案模拟器,通过它可以对对等网应用方案进行定量分析,评估其可行性和性能。

2 蚁群算法

蚁群算法[3]是90年代初意大利学者Dorigo提出的一种模拟进化算法,模拟了自然界蚂蚁总是能找到最佳路径觅食的方式。一般情况下,蚂蚁行动时会在它经过的地方留下一种挥发性的分泌物——信息素(pheromone,即外激素),它会随着时间流失慢慢挥发,降低浓度。开始时蚂蚁随机挑选路径行走,经过一段时间后,如果从源头到目的地有多条路径,那么长路径上留下的信息素浓度肯定低于短路径上留下的信息素浓度。由于蚂蚁选择信息素浓度高的路径行走的概率远大于信息素浓度低的路径,所以随着时间的流逝就形成一个正向反馈,即:最优路径上的信息素浓度越来越大,而其它路径上的信息素浓度逐渐减少直到消失,最终整个蚁群找到最佳路径。

整个蚁群系统是一个分散控制的系统,个体高度自治,可以随意加入或退出蚁群系统。虽然单个蚂蚁的行为极其简单、非智能,但是通过信息素的作用,整个蚁群交换着路径信息,协调工作,最终完成复杂的任务,显现了整体性和智能性的特点。因此,蚁群系统是自然界中一个典型的群体智能系统[4]。

由于蚁群系统对环境变化的自适应、自组织能力很强,它可以应用于复杂问题的优化[5]、多代理系统[6]研究等方面。

3 模拟器设计

首先,本文把对等网系统看作是一个蚁群系统,其主要特点如下:

收稿日期:2003-08-07。赵晓怡,硕士生,主研领域:计算机网络。

• 节点分布在 Internet 的边缘,也就是说:对等网系统是完全分布式的网络系统。

• 节点具有很高的自治性和随意性。

• 节点上的操作十分简单。

• 对等网的控制较复杂。

因此,本文的模拟器的设计是基于蚁群算法并采用 Java 语言实现。采用 java 主要是考虑到:

• Java 的可移植性、跨平台性。

• Java 的远程通信处理功能 RMI。

• Java 的安全性。

整个对等网主要由两个部分组成:Nest 和 Ant。Nest 对应于对等网中的节点,它由唯一的 NestID 来标识,其任务是完成本地的计算处理、存储信息和与其他节点通信。当对等网中某个节点的用户需要服务时,通过 nest 接口向 Nest 发出请求;Nest 接收到请求后,根据请求产生一个或多个对应的 Ant。这些 Ant 可以看作是一个个代理,它们也由唯一的 AntID 来标识。Ant 穿梭在由 Nest 组成的对等网中,和途径的 Nest 交互信息,完成特定的任务,并把结果以应答的方式返给源 Nest。不同的对等网应用(如:搜索,下载文件等),可以根据各自的特点设计不同的蚁群算法来实现,见图 1 示:

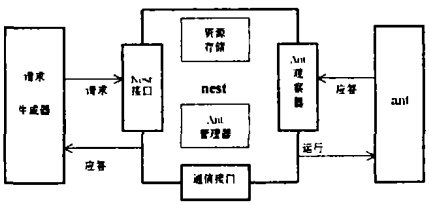

图 1

为了对不同的蚁群算法进行定量分析,评估其性能,本文的模拟器设计为以下五部分:

• Experiment 对蚁群算法的一次模拟,本文称为一次实验,为了准确衡量一种蚁群算法,需要进行多次实验,收集模拟过程中产生的相关数据并求平均值。

• Scenario 每一次实验又需要在不同的资源和请求条件下进行,即需要在不同模拟情况下执行。

• PeerNet 根据设定的 Nest 数量和每个 Nest 上的连接度来模拟对等网的各个节点(Nest)。

• Stats 收集模拟过程中的数据,即对特定的事件进行计数。

• ResourceSet 随机产生要插入的共享资源和请求。

4 系统实现

4.1 对等网模拟部分的实现

(1) 首先请求生成器模拟产生请求,由 Nest 读取后,在 Ant 管理器中产生相应的 Ant,放入 Ant 队列等候处理。然后管理器通过 Nest 和 Ant 之间的接口来执行蚁群算法。

(2) Ant 通过 Ant 观察器(AntView)调用 Nest 的通信接口 Gate 来实现其在 Nest 间的走动。

```
public void move(NestId id)
  throws IOException//通信异常
  {
  nest.getGate().Send(id,desc);//desc 是当前 Ant
  moved = true;
  }
```

(3) 当 Ant 到达某一个 Nest 后,其通信接口 Gate 中的 Ant 监听器负责监听到它的到来,然后通过 ant 观察器将它加入 Ant 管理器中的 Ant 队列中。

(4) 从管理器选择一个 Ant 运行蚁群算法。

```
public void execute(){
  AntView view = (AntView)queue.remove();
  if(view != null){
  Ant ant = view.getAnt();
  ant.run(view);
  }
}
```

(5) Ant 通过 Ant 观察器与 Nest 交互,读取并留下信息素、路由信息(通过资源存储部分实现)和修改 Nest 的邻居信息(通过通信接口 Gate 实现)。

(6) Ant 通过 Ant 观察器,查看是否为初始 Nest,若是,返回应答;否则,继续步骤(2)。

为了安全考虑,在 Ant 管理器中利用 Java 的 sandbox 把 Ant 的操作限制在一定的环境下执行。此外,Ant 也不能直接读取或修改 Nest 中的数据,必须通过 Ant 观察器接口(AntView)才能与 Nest 交互。

4.2 模拟实验部分的实现

(1) 读取以 xml 文件形式存储的配置信息,如 Experiment,Scenario 执行的次数。

```
<?xml version="1.0" encoding="us-ascii"?>
<!DOCTYPE SystemConfiguration SYSTEM "factory.dtd">
<SystemConfiguration version="1.0">
  <Create interface="Experiment" class="ExperimentImpl">
    <Argument name="iterations" type="int" value="100"/>
  </Create>
  <Create interface="Scenario" calss="ScenarioImpl">
    <Argument name="rounds" type="int" value="1000"/>
  </Create>
  <Create interface="Peernet" class="PeernetImpl"
    <Argument name="size" type="int" value="200"/>
    <Argument name="degree" type="int" value="6"/>
  .....
</SystemConfiguration>
```

(2) 执行 Experiment,并内嵌多次循环执行 Scenario。

```
public interface Experiment{  Stats[]evaluate();
  //评价 ant 算法,收集在模拟过程中产生的相关数据并返回
}
public Stats[] evaluate(){
  for (int i=0;i<iterations;i++){
  scenario.clear();//恢复初始状态
  scenario.setStats(stats);//设置数据收集对象
  scenario.evaluate();//评价 ant 算法
}
return new Stats[]{stats};
}
```

(3) Scenario.evaluate 调用 PeerNet 模拟 Nest 网络,获取 Nest 的管理器;调用 ResourceSet 随机产生要插入的共享资源和请求(替代请求生成器),然后执行前一部分所述的对等网的执行过程。

值得一提的是,本文的模拟器采用 xml 来定义配置文件,不

仅可以根据自己意愿使用 DTD 来描述 xml 文档的格式,对于模拟器来说还可以只需编译一次,就能多次修改数据来模拟评价算法,节约了开发时间。

5 模拟实例

以文件搜索为例,设计了一种基于关键字搜索的 ant (SearchAnt)。

5.1 算法概述

每一个文件被共享时,系统会自动提取它文件名或其中的一部分作为该文件的关键词。这个关键词不仅被用来进行匹配比较,还可以被用作路由信息。

系统的每一个 nest 中存储着一些以前搜索过的文件的 URL 资源或是与 ant 交互留下的 URL 资源,这样整个 nest 网络合起来就是一个庞大的分布式索引。

当用户发出查询请求后,nest 就产生一个或多个包含相应关键字的 SearchAnt,并利用安全散列算法将关键字转换为一个散列关键字(hashed keyword),这样就可以将关键字文本转变为 160bit 的数值,有利于比较相似程度。

当 SearchAnt 访问一个 nest 时,首先它将 query 的关键字与该 nest 的 URL 资源进行比较,如果找到完全匹配的,就将 URL 加入到 SearchAnt 的匹配文件 URL 列表中。然后 SearchAnt 用相关的散列关键字在 nest 中检查的其他 ant 留下的路由信息,从最相近的散列关键字的 nest 序列中选一个 nest 作为下一步。每走一步,TTL 数减一(TTL 数即 SearchAnt 定义的最大跳数)。

每一个 SearchAnt 中有一个 path 对象来记录它的行走路径,这样既可以避免重复走过同一个 nest,又便于在 TTL 数为 0 时,SearchAnt 可以沿原路径返回。在返回路上,SearchAnt 把找到的符合要求的 URL 列表和路由信息与沿途的 nest 交互更新,方便以后路过的 SearchAnt。回到源 nest 后,SearchAnt 将 URL 列表应答给相应的请求。

当 SearchAnt 找到第一个匹配文件的 URL 时,该 nest 就产生一个应答 ant(ReplyAnt),返回初始 nest 报告。SearchAnt 则继续寻找匹配文件。

```
Void run(AntView view){
UrlStore = view.getStorage(URL_STORAGE);//读取本地的 URL 资源
Route = view.getStorage(ROUTE_STORAGE);//读取其他 ant 留下的
                                        //路由信息
If(view.getTTL() > 0){
  Path.add(view.getNestId());//将此 nest 记录到 ant 的行走路径中
  Size = path.size();//ant 的行走路径的当前长度
  Urls.add(urlStore.getResoureces(query));//将符合 query 要求的
                                  //URL 加入到 ant 的匹配 URL 列表中
NextList = route.getNextNest(keyhash,path);
                                  //读取下一步可以走的 nest 列表
  Do{
    Moved = view.move(nextList.get(i++));
  }while(!moved);
}else{//当 ant 的 TTL 为 0 时,沿原路径返回
  if(size > 0){
    urlstore.addResources(urls);//将发现的 URL
                                  //列表更新 URL 资源中
  route.addKeyhash(keyhash,path.getLast());//更新路由信息
  view.move(path.get(size - ));//往回走
 }else{
  view.result(rid,urls);//回到原始 nest,将符合要求的 URL 列表回
                        //应给相应的请求
 }
 }
}
```

5.2 模拟结果

通过 xml 文件,我们将整个 nest 网络的节点数设为 2000 个,每个 nest 的连接度设为 6,experiment 和 scenario 的循环次数分别为 5 和 50,每个 ant 的 TTL 为 100 个跳数。模拟结果见表 1。

表 1

搜索请求总数	成功找到的请求数	成功率	第一个响应时,Ant 走过的跳数
200	73	36.5%	23
500	198	39.6%	21
1000	405	40.5%	18
1800	756	42.0%	14
3000	1321	44.1%	12
5000	2343	46.9%	10
7000	3596	51.4	7

由表 1 中数据可知,随着请求总数的不断增加,成功率不断上升,得到第一个响应时 ant 所走过的跳数不断下降。这说明随着系统运行时间的增加,分布式索引和路由信息不断地完善,系统的整体性能不断地优化。

6 结束语

随着网格技术的发展,对等网技术的应用越来越受到人们的关注和青睐。目前国内在对等网的研究还刚刚起步。基于蚁群算法的对等网模拟器的研究还很少。希望本文模拟器的设计和实现会对对等网的研究应用起到一定的积极作用。

参 考 文 献

[1] Napster Home Page.http://www.napster.com.

[2] G.Kan.Gnutella.In a.Oram,editor,Peer to Peer:harnessing the Benefits of a disruptive Technology,Chapter8.O'Reilly & Associates,Mar.2001.

[3] 吕国英等,"基于蚂蚁算法的分布式 QoS 路由选择算法",《通信学报》,Vol.22,No.9,2001.

[4] Eric Bonabeau,Marco Dorigo,Swarm Intelligence:From Natural to Artificial Systems,Oxford University Press,1999.

[5] Gianni Di Caro,Marco Dorigo,AntNet:Distributed Stigmergetic Control for Communication networks.Journal of Artificial Intelligence Research,9:317~365,1998.

[6] Gianni Di Caro,Marco Dorigo,an adaptive multi-agent routing algorithm inspired by ants behavior,Proceedings of 5th Annual Australasian Conference on Parallel and Real-Time Systems,1998.

(上接第 14 页)

[5] P. Coad, E. Yourdon. Object-Oriented Design. New York: Prentice-Hall, 1991.

[6] Ivar Jacobson, Magnus Christerson, Patrik Jonsson, Gunnar Overgaard, "Object-Oriented Software Engineering——A Use Case Driven Approach", Addison-Wesley Publishing Company, 1992.

[7] IBM Object-Oriented Technology Center. Developing Object-Oriented Software: An Experience-Based Approach. Prentice Hall PTR, 1999.

[8] UML Specification. OMG, http://www.OMG.com.