(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 104135715 B
(45)授权公告日 2015.10.07

(21)申请号 201410256324.1
(22)申请日 2014.06.10
(73)专利权人 腾讯科技(深圳)有限公司
地址 518000 广东省深圳市福田区振兴路赛格科技园2栋东403室
(72)发明人 杨岳军 姚晓光 吴凡凡 樊亮 沙鹰 牟露
(74)专利代理机构 深圳市深佳知识产权代理事务所(普通合伙) 44285
代理人 王仲凯
(51)Int.Cl.
H04W 4/02(2009.01)
H04W 64/00(2009.01)
G01S 19/14(2010.01)

审查员 郝悦

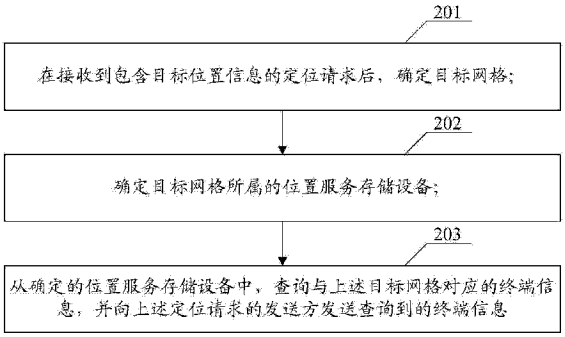
权利要求书4页 说明书21页 附图10页

(54)发明名称

一种位置服务的实现方法、装置,及系统

(57)摘要

本发明实施例公开了一种位置服务的实现方法、装置,及系统,其中方法的实现包括:在接收到包含目标位置信息的定位请求后,确定目标网格;所述目标网格包括所述目标位置信息对应的网格,以及与所述目标位置信息对应的网格距离在设定范围内的网格;所述网格是由地图划分正方形得到;确定目标网格所属的位置服务存储设备;在位置服务存储设备中以网格编号为关键字以终端信息为值进行数据存储,且在每个位置服务存储设备中存储的最小粒度为所述网格对应的正方形的边长的M倍,M大于等于2;从确定的位置服务存储设备中,查询与所述目标网格对应的终端信息,并向所述定位请求的发送方发送查询到的终端信息。可以提高服务器的响应速度,降低计算压力。



CN 104135715 B

1. 一种位置服务的实现方法，其特征在于，包括：

在接收到包含目标位置信息的定位请求后，确定目标网格；所述目标网格包括所述目标位置信息对应的网格，以及与所述目标位置信息对应的网格距离在设定范围内的网格；所述网格是由地图划分正方形得到；

确定目标网格所属的位置服务存储设备；在位置服务存储设备中以网格编号为关键字以终端信息为值进行数据存储，且在每个位置服务存储设备中存储的最小粒度为所述网格对应的正方形的边长的 M 倍，M 大于等于 2；

从确定的位置服务存储设备中，查询与所述目标网格对应的终端信息，并向所述定位请求的发送方发送查询到的终端信息。

2. 根据权利要求 1 所述方法，其特征在于，还包括：

接收终端上报的位置信息以及终端信息；确定所述位置信息所属的网格，以及确定的网格所属的位置服务存储设备；

将所述终端上报的位置信息以及终端信息，发送给确定的位置服务存储设备进行存储。

3. 根据权利要求 2 所述方法，其特征在于，在位置服务存储设备中还以终端信息为关键字以网格编号为值进行数据存储，所述方法还包括：

向确定的位置服务存储设备以外的其他位置服务存储设备发送位置信息删除指令，删除存储的所述终端信息对应的位置信息。

4. 根据权利要求 1 至 3 任意一项所述方法，其特征在于，所述网格由地图划分正方形得到，所有网格的网格编号按照地理位置顺序编号；所有在位置服务存储设备中最小粒度的网格与位置服务存储设备的所属关系符合哈希算法的规则；

所述确定目标网格所属的位置服务存储设备包括：使用目标网格进行哈希查找确定标网格所属的位置服务存储设备。

5. 根据权利要求 4 所述方法，其特征在于，若目标网格分布于两个或两个以上的位置服务存储设备；所述方法还包括：

将从位置服务存储设备查询到的终端信息进行组包。

6. 根据权利要求 1 至 3 任意一项所述方法，其特征在于，若再次接收到来自所述终端的定位请求，还包括：

确定与前次定位请求的时间间隔是否超过预定阈值，若为超过预定阈值，则拒绝执行定位。

7. 根据权利要求 1 所述方法，其特征在于，所述确定目标网格包括：

首先在测试网格中查询包含终端信息的测试网格，所述测试网格是与所述目标位置信息对应的测试网格，测试网格由地图划分正方形得到，且边长大于所述网格；

确定查询到的测试网格中包含的网格为目标网格。

8. 一种位置服务的实现方法，其特征在于，包括：

位置服务存储设备接收位置服务器的查询请求，所述查询请求包含有待查询的目标网格；所述目标网格包括目标位置信息对应的网格，以及与所述目标位置信息对应的网格距离在设定范围内的网格；

在所述位置服务存储设备中以网格编号为关键字以终端信息为值进行数据存储，且在

每个位置服务存储设备中存储的最小粒度为所述网格对应的正方形的边长的 M 倍，M 大于等于 2 ；所述网格是由地图划分正方形得到 ；

使用所述目标网格的编号查询并获得与所述待查询的目标网格对应的终端信息 ；

将获得的终端信息发送给所述位置服务器。

9. 根据权利要求 8 所述方法，其特征在于，

在所述位置服务存储设备存储的网格编号按照哈希算法的规则进行存储，并采用哈希桶排序。

10. 根据权利要求 8 或 9 所述方法，其特征在于，还包括 ：

接收来自所述位置服务器发送的终端信息以及位置信息，查找是否已经存在所述终端信息对应的位置信息，若存在，则删除 ；

若不存在，或者，删除完毕后，将所述终端信息存放到与所述位置信息对应的网格中，所述位置信息采用哈希表存储。

11. 根据权利要求 10 所述方法，其特征在于，所述哈希表的值采用链表的数据结构 ；所述将所述终端信息存放到与所述位置信息对应的网格中包括 ：

将所述位置信息采用头插法将位置信息插入与所述位置信息对应的网格的链表中。

12. 根据权利要求 11 所述方法，其特征在于，还包括 ：

读取网格的回收时间，若回收时间距当前时间超过预定阈值，则从所述链表的后端往前依次删除超时的位置信息。

13. 根据权利要求 12 所述方法，其特征在于，所述读取网格的回收时间包括 ：

按照预定的顺序在所有网格中选取预定数量网格读取回收时间。

14. 根据权利要求 12 所述方法，其特征在于，所述方法还包括 ：

将位置信息的插入操作和删除操作记录到本端业务支撑系统日志中 ；在有本端业务支撑系统日志超时后，删除超时的本端业务支撑系统日志 ；

在清内存恢复过程中，按照本端业务支撑系统日志的记录时间从先到后依次读入并执行业务支撑系统日志记录的操作。

15. 根据权利要求 14 所述方法，其特征在于，还包括 ：

在位置服务存储设备扩容过程中，获取所有位置服务存储设备的业务支撑系统日志 ；读取获取到的业务支撑系统日志的记录，若当前读取到的记录属于本端位置服务存储设备，则执行当前读取到的记录对应的操作。

16. 一种位置服务器，其特征在于，包括 ：

网格确定单元，用于在接收到包含目标位置信息的定位请求后，确定目标网格 ；所述目标网格包括所述目标位置信息对应的网格，以及与所述目标位置信息对应的网格距离在设定范围内的网格 ；所述网格是由地图划分正方形得到 ；

设备确定单元，用于确定目标网格所属的位置服务存储设备 ；在位置服务存储设备中以网格编号为关键字以终端信息为值进行数据存储，且在每个位置服务存储设备中存储的最小粒度为所述网格对应的正方形的边长的 M 倍，M 大于等于 2 ；

查询单元，用于从确定的位置服务存储设备中，查询与所述目标网格对应的终端信息 ；

信息发送单元，用于向所述定位请求的发送方发送查询到的终端信息。

17. 根据权利要求 16 所述位置服务器，其特征在于，还包括：

信息接收单元，用于接收终端上报的位置信息以及终端信息；

所述网格确定单元，还用于确定所述位置信息所属的网格；

所述设备确定单元，还用于确定所述网格确定单元确定的网格所属的位置服务存储设备；

所述信息发送单元，还用于将所述终端上报的位置信息以及终端信息，发送给确定的位置服务存储设备进行存储。

18. 根据权利要求 17 所述位置服务器，其特征在于，在位置服务存储设备中还以终端信息为关键字以网格编号为值进行数据存储，所述位置服务器还包括：

所述信息发送单元，还用于向确定的位置服务存储设备以外的其他位置服务存储设备发送位置信息删除指令，删除存储的所述终端信息对应的位置信息。

19. 根据权利要求 16 至 18 任意一项所述位置服务器，其特征在于，所述网格由地图划分正方形得到，所有网格的网格编号按照地理位置顺序编号；所有在位置服务存储设备中最小粒度的网格与位置服务存储设备的所属关系符合哈希算法的规则；

所述设备确定单元，用于使用目标网格进行哈希查找确定标网格所属的位置服务存储设备。

20. 根据权利要求 19 所述位置服务器，其特征在于，若目标网格分布于两个或两个以上的位置服务存储设备；所述位置服务器还包括：

组包单元，用于将从位置服务存储设备查询到的终端信息进行组包；

所述信息发送单元，用于向所述定位请求的发送方组包单元的组包结果。

21. 根据权利要求 16 至 18 任意一项所述位置服务器，其特征在于，还包括：

定位控制单元，用于若再次接收到来自所述终端的定位请求，确定与前次定位请求的时间间隔是否超过预定阈值，若为超过预定阈值，则拒绝执行定位。

22. 根据权利要求 16 所述位置服务器，其特征在于，

所述网格确定单元，用于首先在测试网格中查询包含终端信息的测试网格，所述测试网格是与所述目标位置信息对应的测试网格，测试网格由地图划分正方形得到，且边长大于所述网格；然后确定查询到的测试网格中包含的网格为目标网格。

23. 一种位置服务存储设备，其特征在于，包括：

请求接收单元，用于接收位置服务器的查询请求，所述查询请求包含有待查询的目标网格；所述目标网格包括目标位置信息对应的网格，以及与所述目标位置信息对应的网格距离在设定范围内的网格；

数据存储单元，用于以网格编号为关键字以终端信息为值进行数据存储，且在每个位置服务存储设备中存储的最小粒度为所述网格对应的正方形的边长的 M 倍，M 大于等于 2；所述网格是由地图划分正方形得到；

查询单元，用于使用所述目标网格的编号查询并获得与所述待查询的目标网格对应的终端信息；

信息发送单元，用于将获得的终端信息发送给所述位置服务器。

24. 根据权利要求 23 所述位置服务存储设备，其特征在于，

所述数据存储单元，用于存储的网格的网格编号按照哈希算法的规则进行存储，并采

用哈希桶排序。

25. 根据权利要求 23 或 24 所述位置服务存储设备，其特征在于，还包括：

信息接收单元，用于接收来自所述位置服务器发送的终端信息以及位置信息；

所述查询单元，还用于查找是否已经存在所述终端信息对应的位置信息；

数据删除单元，用于若所述查询单元查询结果为存在，则删除查找到的位置信息；

所述数据存储单元，用于若所述查询单元查询结果为不存在，或者，所述数据删除单元删除完毕后，将所述终端信息存放到与所述位置信息对应的网格中，所述位置信息采用哈希表存储。

26. 根据权利要求 25 所述位置服务存储设备，其特征在于，所述哈希表的值采用链表的数据结构；

所述数据存储单元，用于将所述位置信息采用头插法将位置信息插入与所述位置信息对应的网格的链表中。

27. 根据权利要求 26 所述位置服务存储设备，其特征在于，还包括：

超时清理单元，用于读取网格的回收时间，若回收时间距当前时间超过预定阈值，则从所述链表的后端往前依次删除超时的位置信息。

28. 根据权利要求 27 所述位置服务存储设备，其特征在于，

所述超时清理单元，用于按照预定的顺序在所有网格中选取预定数量网格读取回收时间。

29. 根据权利要求 27 所述位置服务存储设备，其特征在于，所述位置服务存储设备还包括：

日志记录单元，用于将位置信息的插入操作和删除操作记录到本端业务支撑系统日志中；

日志清理单元，用于在有本端业务支撑系统日志超时后，删除超时的本端业务支撑系统日志；

恢复控制单元，用于在清内存恢复过程中，按照本端业务支撑系统日志的记录时间从先到后依次读入并执行业务支撑系统日志记录的操作。

30. 根据权利要求 29 所述位置服务存储设备，其特征在于，

所述恢复控制单元，还用于在位置服务存储设备扩容过程中，获取所有位置服务存储设备的业务支撑系统日志；读取获取到的业务支撑系统日志的记录，若当前读取到的记录属于本端位置服务存储设备，则执行当前读取到的记录对应的操作。

31. 一种位置服务系统，包括：位置服务器和位置服务存储设备，位置服务器和位置服务存储设备以可通信方式连接，其特征在于，所述位置服务器为权利要求 16 ～ 22 任意一项的位置服务器，所述位置服务存储设备为权利要求 23 ～ 30 任意一项的位置服务存储设备。

一种位置服务的实现方法、装置，及系统

技术领域

[0001] 本发明涉及通信技术领域，特别涉及一种位置服务的实现方法、装置，及系统。

背景技术

[0002] 位置服务（LBS，Location Based Services）又称定位服务，LBS是由移动通信网络和卫星定位系统结合在一起提供的一种增值业务，通过一组定位技术获得移动终端的位置信息（如经纬度坐标数据），提供给移动用户本人或他人以及通信系统，实现各种与位置相关的业务。实质上是一种概念较为宽泛的与空间位置有关的新型服务业务。

[0003] 位置服务可以被应用与不同的领域，例如：健康、工作、个人生活等。此服务可以用来辨认一个人或物的位置，例如发现最近的取款机或朋友同事当前的位置，也能透过客户目前所在的位置提供直接的手机广告，并包括个人化的天气信息提供，甚至提供本地化的游戏。

[0004] 当前，基于个人消费者需求的智能化，位置信息服务将伴随定位技术和无线上网技术的发展，需求呈大幅度增长趋势。LBS不但可以提升企业运营与服务水平，也能为车载定位技术的用户提供了更多样化的便捷服务。定位技术用户，从地址点导航到兴趣点服务，再到实时路况技术的应用，不仅可引导用户找到附近的产品和服务，并可获得更高的便捷性和安全性。

[0005] 目前位置服务已经应用到车载定位技术产品上。通过对定位技术市场了解，车载导航在深化LBS应用的过程中，已将“互动”的理念融入其中，“照片导航”、“主题地图”等独有功能模块的增加和延展，不仅让用户可以享受全新的个性导航服务，直接通过导航仪查询到全国各地的著名景点、酒店、饭店、加油站等丰富资讯，一键导航，同时用户可以基于网络进行数据下载、上传，与其他用户实现互动交流，这将成为未来的发展方向之一。而通过定位技术，也可以为个人用户或集团用户提供特殊信息报警服务。位置服务器也已经被用在终端用户寻找周围用户，例如：寻找附近的人进行社交活动、寻找附近与自己有关联关系（例如相同爱好、玩相同游戏等）的人。

[0006] 目前基于位置服务的系统，通常采用如图1所示的结构，位置服务器接收来自终端的地理位置信息，将地理位置信息存放到位置服务存储设备1～n；如果位置服务器需要解析地理位置信息，还可以将地理位置信息发送给地图服务器，并从地图服务器获得上述地理位置信息的解析结果。

[0007] 目前，地理位置信息使用经纬度来表示和存储，这样可以方便的定位到终端的位置，具有精确的优点，但是在定位附近终端的过程中，使用经纬度计算距离计算量较大，导致位置服务器响应速度很低。

发明内容

[0008] 本发明实施例提供了一种位置服务的实现方法、装置，及系统，用于减少距离计算量，提高位置服务器的响应速度。

[0009] 本发明实施例提供了一种位置服务的实现方法，包括：

[0010] 在接收到包含目标位置信息的定位请求后，确定目标网格；所述目标网格包括所述目标位置信息对应的网格，以及与所述目标位置信息对应的网格距离在设定范围内的网格；所述网格是由地图划分正方形得到；

[0011] 确定目标网格所属的位置服务存储设备；在位置服务存储设备中以网格编号为关键字以终端信息为值进行数据存储，且在每个位置服务存储设备中存储的最小粒度为所述网格对应的正方形的边长的 M 倍，M 大于等于 2；

[0012] 从确定的位置服务存储设备中，查询与所述目标网格对应的终端信息，并向所述定位请求的发送方发送查询到的终端信息。

[0013] 本发明实施例还提供了另一种位置服务的实现方法，包括：

[0014] 位置服务存储设备接收位置服务器的查询请求，所述查询请求包含有待查询的目标网格；

[0015] 在所述位置服务存储设备中以网格编号为关键字以终端信息为值进行数据存储，且在每个位置服务存储设备中存储的最小粒度为所述网格对应的正方形的边长的 M 倍，M 大于等于 2；所述网格是由地图划分正方形得到；

[0016] 使用所述目标网格的编号查询并获得与所述待查询的目标网格对应的终端信息；

[0017] 将获得的终端信息发送给所述位置服务器。

[0018] 本发明实施例还提供了一种位置服务器，包括：

[0019] 网格确定单元，用于在接收到包含目标位置信息的定位请求后，确定目标网格；所述目标网格包括所述目标位置信息对应的网格，以及与所述目标位置信息对应的网格距离在设定范围内的网格；所述网格是由地图划分正方形得到；

[0020] 设备确定单元，用于确定目标网格所属的位置服务存储设备；在位置服务存储设备中以网格编号为关键字以终端信息为值进行数据存储，且在每个位置服务存储设备中存储的最小粒度为所述网格对应的正方形的边长的 M 倍，M 大于等于 2；

[0021] 查询单元，用于从确定的位置服务存储设备中，查询与所述目标网格对应的终端信息；

[0022] 信息发送单元，用于向所述定位请求的发送方发送查询到的终端信息。

[0023] 本发明实施例还提供了一种位置服务存储设备，包括：

[0024] 请求接收单元，用于接收位置服务器的查询请求，所述查询请求包含有待查询的目标网格；

[0025] 数据存储单元，用于以网格编号为关键字以终端信息为值进行数据存储，且在每个位置服务存储设备中存储的最小粒度为所述网格对应的正方形的边长的 M 倍，M 大于等于 2；所述网格是由地图划分正方形得到；

[0026] 查询单元，用于使用所述目标网格的编号查询并获得与所述待查询的目标网格对应的终端信息；

[0027] 信息发送单元，用于将获得的终端信息发送给所述位置服务器。

[0028] 一种位置服务系统，包括：位置服务器和位置服务存储设备，位置服务器和位置服务存储设备以可通信方式连接，所述位置服务器为本发明实施例提供的任意一项的位置服

务器，所述位置服务存储设备为本发明实施例提供的任意一项的位置服务存储设备。

[0029] 从以上技术方案可以看出，本发明实施例具有以下优点：采用网格的方式将地理位置进行划分，并采用划分后的网格进行位置信息存储；在位置服务过程中不必使用经纬度进行距离计算减少计算量，从而可以提高服务器的响应速度。另外，在为位置服务存储设备分配网格时，使用的最小粒度比网格要大，这样可以使更多的相邻网格分配到同一个位置服务存储设备，使得确定的目标网格集中在少数几个位置服务存储设备内，从而减少对位置服务存储设备的调用，降低位置服务存储设备的计算压力，从而进一步的提升真个位置服务的系统性能。

附图说明

[0030] 为了更清楚地说明本发明实施例中的技术方案，下面将对实施例描述中所需要使用的附图作简要介绍，显而易见地，下面描述中的附图仅仅是本发明的一些实施例，对于本领域的普通技术人员来讲，在不付出创造性劳动性的前提下，还可以根据这些附图获得其他的附图。

[0031] 图 1 为现有技术系统架构示意图；

[0032] 图 2 为本发明实施例方法流程示意图；

[0033] 图 3 为本发明实施例方法流程示意图；

[0034] 图 4 为本发明实施例系统结构示意图；

[0035] 图 5 为本发明实施例空间数据模型示意图；

[0036] 图 6 为本发明实施例格子的划分示意图；

[0037] 图 7 为本发明实施例格子搜索示意图；

[0038] 图 8 为本发明实施例数据的存储示意图；

[0039] 图 9 为本发明实施例上报玩家地理信息的方法流程示意图；

[0040] 图 10 为本发明实施例存放和删除信息的系统结构示意图；

[0041] 图 11 为本发明实施例删除 playerID 3 的数据查找过程示意图；

[0042] 图 12 为本发明实施例 hash 表结构示意图；

[0043] 图 13 为本发明实施例 playerID 到 GridGID 的反向索引结构示意图；

[0044] 图 14 为本发明实施例基于 bucket（桶）的定时清理示意图；

[0045] 图 15 为本发明实施例的位置服务器的结构示意图；

[0046] 图 16 为本发明实施例的位置服务器的结构示意图；

[0047] 图 17 为本发明实施例的位置服务器的结构示意图；

[0048] 图 18 为本发明实施例的位置服务器的结构示意图；

[0049] 图 19 为本发明实施例的位置服务存储设备的结构示意图；

[0050] 图 20 为本发明实施例的位置服务存储设备的结构示意图；

[0051] 图 21 为本发明实施例的位置服务存储设备的结构示意图；

[0052] 图 22 为本发明实施例的位置服务存储设备的结构示意图；

[0053] 图 23 为本发明实施例的位置服务系统结构示意图；

[0054] 图 24 为本发明实施例的服务器的结构示意图。

具体实施方式

[0055] 为了使本发明的目的、技术方案和优点更加清楚，下面将结合附图对本发明作进一步地详细描述，显然，所描述的实施例仅仅是本发明一部份实施例，而不是全部的实施例。基于本发明中的实施例，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其它实施例，都属于本发明保护的范围。

[0056] 本发明实施例提供了一种位置服务的实现方法，如图 2 所示，包括 ：

[0057] 201 ：在接收到包含目标位置信息的定位请求后，确定目标网格 ；上述目标网格包括上述目标位置信息对应的网格，以及与上述目标位置信息对应的网格距离在设定范围内的网格 ；上述网格是由地图划分正方形得到 ；

[0058] 在本发明实施例中，网格由地图划分而来，也即是说网格与实际的地理位置是对应的。网格的边长实际上就是这个网格对应地理位置的边长。网格边长越小，最终定位时的精度将会越高，网格越大则存储的网格数量越少，查找所用的时间则可能越少。具体的长度本发明实施例不予限定，推荐可以使用 100m 作为边长。

[0059] 202 ：确定目标网格所属的位置服务存储设备 ；在位置服务存储设备中以网格编号为关键字以终端信息为值进行数据存储，且在每个位置服务存储设备中存储的最小粒度为上述网格对应的正方形的边长的 M 倍，M 大于等于 2 ；

[0060] 这里 M 的具体取值，可以由技术人员设定。若查找周围最大 2KM 范围内的终端信息，那么可以将这个值设定为 100 或者 50 等值，具体取值本发明实施例不予限定。

[0061] 203 ：从确定的位置服务存储设备中，查询与上述目标网格对应的终端信息，并向上述定位请求的发送方发送查询到的终端信息。

[0062] 本发明实施例，采用网格的方式将地理位置进行划分，并采用划分后的网格进行位置信息存储 ；在位置服务过程中不必使用经纬度进行距离计算减少计算量，从而可以提高服务器的响应速度。另外，在为位置服务存储设备分配网格时，使用的最小粒度比网格要大，这样可以使更多的相邻网格分配到同一个位置服务存储设备，使得确定的目标网格集中在少数几个位置服务存储设备内，从而减少对位置服务存储设备的调用，降低位置服务存储设备的计算压力，从而进一步的提升真个位置服务的系统性能。另外，还可以将网格可以设置得较小，提高定位精度，并减少因单个网格位置信息太多导致的冲突。

[0063] 在本发明实施例中，位置服务存储设备中的位置信息可以来源于其他服务器搜集的信息也可以来源于位置服务器直接收集的信息，本发明实施例提供了如何获得并存储终端的位置信息的具体实现方案，具体如下 ：进一步地，上述方法，还包括 ：

[0064] 接收终端上报的位置信息以及终端信息 ；确定上述位置信息所属的网格，以及确定的网格所属的位置服务存储设备 ；

[0065] 将上述终端上报的位置信息以及终端信息，发送给确定的位置服务存储设备进行存储。

[0066] 本发明实施例还提供了优选的存储方案，并且防止存储的数据发生重叠导致数据冲突的情况 ；具体如下 ：在位置服务存储设备中还以终端信息为关键字以网格编号为值进行数据存储，上述方法还包括 ：

[0067] 向确定的位置服务存储设备以外的其他位置服务存储设备发送位置信息删除指令，删除存储的上述终端信息对应的位置信息。

[0068] 在本发明实施例中，向确定的位置服务存储设备以外的其他位置服务存储设备发送位置信息删除指令，具体的发送方案可以采用广播的方式发送，这样可以保持终端在服务器一侧的位置信息的唯一性。

[0069] 在本发明实施例中网格划分后可以进行编号，然后采用哈希算法对网格在位置服务存储设备间进行分配，从而提升查询时候的查询速度，另外也方便后续位置服务存储设备的扩展，具体如下：上述网格由地图划分正方形得到，所有网格的网格编号按照地理位置顺序编号；所有在位置服务存储设备中最小粒度的网格与位置服务存储设备的所属关系符合哈希算法的规则；

[0070] 上述确定目标网格所属的位置服务存储设备包括：使用目标网格进行哈希查找确定标网格所属的位置服务存储设备。

[0071] 本发明实施例，虽然可以减少位置服务设备的调用，但是并不能保证在一个位置服务存储设备完成位置服务器功能，因此目标网格是可以分布于多个位置服务存储设备的，本发明实施例基于此给出了进一步的实现方案如下：进一步地，若目标网格分布于两个或两个以上的位置服务存储设备；上述方法还包括：将从位置服务存储设备查询到的终端信息进行组包。

[0072] 由于终端用户可能会不停的通过终端发送定位的请求，如果每次定位请求都进行定位处理，将会给服务器造成较大的压力；另外，通常来说间隔时间很短的情况下，定位的结果通常也极少发生变化，因此这类频繁的定位也是没有必要的。本发明实施例为了降低服务器的压力，减少不必要的定位操作，提出了如下解决方案：若再次接收到来自上述终端的定位请求，上述方法还包括：确定与前次定位请求的时间间隔是否超过预定阈值，若为超过预定阈值，则拒绝执行定位。

[0073] 在本发明实施例中，将地图划分为网格，网格的边长越短，则定位的精度越高，但是计算量越大；网格的边长越短，则计算量越小响应速度越快；为了综合这两者的优点，本发明实施例提供了如下解决方案：上述确定目标网格包括：首先在测试网格中查询包含终端信息的测试网格，上述测试网格是与上述目标位置信息对应的测试网格，测试网格由地图划分正方形得到，且边长大于上述网格；确定查询到的测试网格中包含的网格为目标网格。

[0074] 在本发明实施例中，测试网格的边长较大，这样可以首先使用大网格来过滤掉目标网格中没有位置信息的网格，避免不必要的查找，从而减少查找量，提高系统响应速度。

[0075] 本发明实施例还提供了另一种位置服务的实现方法，如图 3 所示，包括：

[0076] 301：位置服务存储设备接收位置服务器的查询请求，上述查询请求包含有待查询的目标网格；在上述位置服务存储设备中以网格编号为关键字以终端信息为值进行数据存储，且在每个位置服务存储设备中存储的最小粒度为上述网格对应的正方形的边长的 M 倍，M 大于等于 2；上述网格是由地图划分正方形得到；

[0077] 在本发明实施例中，网格由地图划分而来，也即是说网格与实际的地理位置是对应的。网格的边长实际上就是这个网格对应地理位置的边长。网格边长越小，最终定位时的精度将会越高，网格越大则存储的网格数量越少，查找所用的时间则可能越少。具体的长度本发明实施例不予限定，推荐可以使用 100m 作为边长。

[0078] 这里 M 的具体取值，可以由技术人员设定。若查找周围最大 2KM 范围内的终端信

息，那么可以将这个值设定为 100 或者 50 等值，具体取值本发明实施例不予限定。

[0079] 可选地，本发明实施例还提供了在位置服务存储设备网格的分布以及网格内数据存储的具体实现方案如下：在上述位置服务存储设备存储的网格编号按照哈希算法的规则进行存储，并采用哈希桶排序。

[0080] 302：使用上述目标网格的编号查询并获得与上述待查询的目标网格对应的终端信息；

[0081] 303：将获得的终端信息发送给上述位置服务器。

[0082] 本发明实施例，采用网格的方式将地理位置进行划分，并采用划分后的网格进行位置信息存储；在位置服务过程中不必使用经纬度进行距离计算减少计算量，从而可以提高服务器的响应速度。另外，在为位置服务存储设备分配网格时，使用的最小粒度比网格要大，这样可以使更多的相邻网格分配到同一个位置服务存储设备，使得确定的目标网格集中在少数几个位置服务存储设备内，从而减少对位置服务存储设备的调用，降低位置服务存储设备的计算压力，从而进一步的提升真个位置服务的系统性能。另外，还可以将网格可以设置得较小，提高定位精度，并减少因单个网格位置信息太多导致的冲突。

[0083] 在本发明实施例中，位置服务存储设备中的位置信息可以来源于其他服务器搜集的信息也可以来源于位置服务器直接收集的信息，本发明实施例提供了如何获得并存储终端的位置信息的具体实现方案，并且提供了保持位置信息唯一性的具体实现方案，具体如下：进一步地，上述方法，还包括：

[0084] 接收来自上述位置服务器发送的终端信息以及位置信息，查找是否已经存在上述终端信息对应的位置信息，若存在，则删除；

[0085] 若不存在，或者，删除完毕后，将上述终端信息存放到与上述位置信息对应的网格中，上述位置信息采用哈希表存储。

[0086] 本发明实施例还提供了位置信息的具体存储方案，在本发明实施例中采用链表用头插法的方式对位置信息进行存储，这样可以使位置信息可以按照时间先后次序进行存储，方便后续对失效数据的删除，具体如下：可选地，上述哈希表的值采用链表的数据结构；上述将上述终端信息存放到与上述位置信息对应的网格中包括：将上述位置信息采用头插法将位置信息插入与上述位置信息对应的网格的链表中。

[0087] 由于终端的位置信息（地理信息）是有时限的，随着终端的地理位置的变化，终端的位置信息可能就会失效。因此对于存储在位置服务存储设备中的位置信息，需要清除已经过期的位置信息；这样不仅可以提高定位的准确性，还可以降低计算量提高系统响应速度。进一步地，上述方法，还包括：

[0088] 读取网格的回收时间，若回收时间距当前时间超过预定阈值，则从上述链表的后端往前依次删除超时的位置信息。

[0089] 由于位置服务存储设备中，可能存放的网格很多，哈希桶也可能很多，如果每次对一个位置服务存储设备全部的网格进行过期数据的清理，这可能需要较多的时间，并且这段时间内位置服务器将会停滞，造成位置服务器不稳定。为了提高位置服务的稳定性，清理过期数据的进程可以逐步进行，具体如下：可选地，上述读取网格的回收时间包括：

[0090] 按照预定的顺序在所有网格中选取预定数量网格读取回收时间。

[0091] 在本发明实施例中，预定数量的网格，可以是任意设定的，也可以是按照系统的忙

闲度来调整的，例如系统越忙则这个数量越少，系统越闲则这个数量越多。采用本发明实施例方案，不会造成位置服务的停滞，可以提高位置服务的稳定性。

[0092] 在本发明实施例中，位置信息等数据可以存放在共享内存中，如果执行了清共享内存等处理，是可能需要进行数据恢复的，因此本发明实施例进一步提供了如何进行数据恢复的方案，如下：进一步地，上述方法还包括：

[0093] 将位置信息的插入操作和删除操作记录到本端业务支撑系统日志中；在有本端业务支撑系统日志超时后，删除超时的本端业务支撑系统日志；

[0094] 在清内存恢复过程中，按照本端业务支撑系统日志的记录时间从先到后依次读入并执行业务支撑系统日志记录的操作。

[0095] 基于本发明实施例的数据存储结构，以及数据恢复的方案，本发明实施例还提供了在位置服务存储设备扩容时的数据恢复，具体方案如下：进一步地，上述方法还包括：

[0096] 在位置服务存储设备扩容过程中，获取所有位置服务存储设备的业务支撑系统日志；读取获取到的业务支撑系统日志的记录，若当前读取到的记录属于本端位置服务存储设备，则执行当前读取到的记录对应的操作。

[0097] 以下实施例将以游戏玩家使用手机查找附近玩家的应用场景为例，对本发明实施例的应用过程以及系统实现过程进行详细的举例说明。

[0098] 一、系统体结构，可以参考图 4 所示。

[0099] Lbs(Location Based Service，位置服务）服务器主要有两个功能：一个是存储玩家空间地理信息（玩家对应的是位于终端的账号，因此实际上也是终端的地理信息），另一个功能是查询某个玩家附近的玩家，比如某个玩家周围 2KM 之内的玩家。

[0100] Lbs 服务器作为一个单独的服务存在，提供接口供外部访问玩家地理信息，存储玩家地理信息。

[0101] Lbs 服务器需要合理有效组织玩家地理信息，用于支持快速查找玩家附近的玩家，设置玩家地理信息等需求。图 4 是 Lbs 服务器的总体结构图。

[0102] 在本发明实施例图 4 所示系统中，游戏服务器 (GameServer) 只和位置服务器 (Lbsservice) 进行通信，GameServer 设置玩家地理信息，请求玩家地理信息都交由 Lbsservice 进行处理。Lbsservice 负责分析 GameServer 的请求，并将请求消息发送给合适的位置服务存储设备。

[0103] 对于 GameServer 提交的需要解析的地理信息，由 Lbsservice 通过超文本传输协议 (HTTP-Hypertext transfer protocol) 协议向搜搜的地图服务器请求解析的结果，并进行相应的处理。

[0104] 在图 4 所示的系统结构中，Lbsservice 实现的是代理的功能，位置服务存储设备 (Lbsstore) 负责数据的存储，Lbsservice 与 Lbsstore 之间通信的通道可以采用 tbus 实现。

[0105] 二、Lbs 空间数据模型：

[0106] 位置信息的存储重要的一点是要支持范围查询，比如以某个点为中心查找附近多远距离的人。在服务器一侧，数据的存储结构应该有效的支持这种查询方式。基于这一点，可以将全球地图二维化之后，然后分割成 N*M 个格子，每个格子的大小都是相同的正方形。玩家的地理信息则根据玩家当前的二维坐标 (x，y) 存放到不同的格子中，如图 5 所示。

[0107] 当查找一个玩家某个点附近的玩家这种应用，可以按照下面的步骤进行：

[0108] 1、首先以玩家的地理位置(x，y)为中心，距离r为半径，得到一个圆

[0109] 2、计算出这个圆的外接正方形；

[0110] 3、计算这个外接正方形包含哪些格子，那么半径为r的玩家信息都在这些格子中

[0111] 4、检索这些格子，找出符合条件的玩家

[0112] 如图5所示，半径为r的格子包含1、2、3、4、5、6、7、8、9号格子，那么只需要找出存储在这个9个格子中的所有玩家，然后计算每个玩家是否在圆所在的正方形内，如果在正方形内就是满足条件的玩家。这样，只需要遍历这个9个格子中的玩家，就能查询到点(x，y)附近距离r之内的玩家了。

[0113] 基于以上空间数据模型的要求，本发明实施例提供了如下详细设计方案：

[0114] 2.1、地球坐标二维化：

[0115] 地球是一个略扁的球形，赤道的长度是40076千米，子午线的长度是40009千米。粗略的将地球以赤道为矩形的长边，将地球变成一个长方形（参考我们挂在墙上的世界地图）。那么长方形的长就是赤道的长度，我们以40000KM作为矩形的长，宽就是20000KM。

[0116] 对于经度上的1度，代表的长度简单认为是40000KM/360，纬度上的一度代表的长度是：20000KM/180。

[0117] 选择东经180度，北纬90度作为原点，经度代表X轴，正方向是从东往西。纬度代表Y轴，正方向就是从北到南。比如东经170度，北纬40度这个经纬度坐标，二维化后，x＝(180－170)*40000KM/360，y＝(90-40)*20000KM/180。

[0118] 2.2、格子的划分：

[0119] 本发明实施例方案中，格子的宽度可以为100米的正方形。也就是将全球二维化后，划分为宽度为100的多个正方形。玩家的信息就根据其坐标(x,y)计算出所在格子，然后将数据存放到该格子中。对于每一个格子都有唯一的编号GridGID，GridGID的编号图6所示进行变化。

[0120] 在图6中所示了400m*400m的范围内格子编号的变化情况。如图6所示的地理区域被划分成4*4的格子，那么编号就是从原点开始，沿着X轴的方向增加的。

[0121] 对应于二维化的地球，我们已经知道二维化后的长方形地图长度是40000KM，宽度是20000KM。并且格子的宽度是100米。对于一个点(x，y)，计算对应GridGID的公式如下：

[0122] GridGID(x,y)＝((40000*1000)/100)*(y/100)+x/100+1；

[0123] 因此，对于一个在(x,y)位置的玩家，我们可以按照上面的GridGID计算公式计算出其GridGID。

[0124] 2.3、格子搜索：

[0125] 格子搜索可以采用螺旋搜索算法，如图7所示，假设A是中心搜索点（玩家所在格子），搜索A附近的人时，依次按照箭头的方向进行搜索。

[0126] 2.4、数据的存储

[0127] 在前述实施例中已经将地球划分为长度为100米的格子，玩家信息就按其位置存放在不同的格子中。具体存储玩家地理信息时，我们可以选择以GridGID作为key（键值，有时也称为关键字），格子中的玩家信息作为值存放到一个hash（哈希）表中，如图8所示。

[0128] 查询附近玩家时，按照 2.2 中所记载的公式计算出格子编号，然后根据格子编号从 hash 表中取出玩家信息，再计算满足条件的玩家即可。

[0129] 三、位置服务器以及位置服务存储设备数据存储以及处理流程的设计：

[0130] Lbs 主要包括 Lbsservice 和 Lbsstore 两个部分。Lbsserive 相当于一个代理，它向外提供访问的接口，外部只和它通信。

[0131] Lbsstore 则负责组织玩家空间地理信息，存储玩家空间地理信息。Lbsstore 存储数据时将数据存放在共享内存，同时将玩家信息记录到 OSS(Operations support system，业务支撑系统）日志中，方便清理共享内存重启时进行数据恢复。

[0132] 下面针对这两个部分进行详细的介绍。

[0133] 3.1Lbsservice 详细设计：

[0134] Lbsservice 作为一个代理，负责协调 GameServer、Lbsstore、搜搜的地图服务器(MAP Server) 之间的通信。由上面描述已知 Lbs 服务器的功能就是存储玩家空间地理信息，查询玩家附近的玩家。

[0135] 对于存储玩家空间信息这个功能，如图 9 所示，是上报玩家地理信息的流程图：

[0136] 首先玩家通过客户端将自身的地理位置信息发送给 GameServer，GameServer 再将收到的地理位置信息转发给 Lbsservice。Lbsservice 在收到地理信息后，将这些地理位置信息发送给搜搜的 MAP Server（地图服务器）请求解析地理信息（可以使用 http 协议，在程序中是使用 CURL(curl 是利用 URL 语法在命令行方式下工作的开源文件传输工具）来发送 http 请求和接收 http 数据）。搜搜的 MAP Server 解析后将解析地理信息的结果返回给 Lbsservice，Lbsservice 对位置信息进行解析，计算出该位置信息应该存放的 Lbsstore 服务器编号，然后将玩家的地理信息发送给计算得到的 Lbsstore。Lbsstore 将玩家地理信息数据存储，并写入 OSS 日志中，方便以后重启时进行数据恢复。Lbsstore 存储完毕后向 Lbsservice 发送存储结果响应，告知存储是否成功，Lbsservice 再将收到的存储结果响应转发给游戏服务器。

[0137] 通过以上实施例可以完成各玩家的位置信息的收集，在此之后可以实现获取玩家附近的玩家这个功能，过程描述如下：

[0138] 首先玩家通过 GameServer 发送消息请求给 Lbsservice，消息内容包括自己的位置信息，位置信息可能是一个二维坐标，也可能是一串需要搜搜的 MAP Server 解析的位置串。同时消息包中还包含附近距离范围 (radious)、超时时间 expireTm（即只需要距离当前时间多少秒内更新的玩家的信息）。

[0139] 如果玩家位置信息是一个需要搜搜 MAP 解析的字符串，那么 Lbsservice 先发 http 消息给搜搜 MAP Server 进行位置解析得到玩家的二维坐标。

[0140] Lbsservice 得到玩家的二维坐标 (x，y) 之后，以坐标 (x，y) 作为原点，radious 为半径计算出一个圆的外接正方形，那么附近玩家的数据都在这个外接正方形中所包含的格子中。计算出这个外接正方形包含了哪些格子，然后计算相应格子的数据存放在哪个 Lbsstore 上。再将请求发送到这些 Lbsstore 上。可以参考图 5 所示。

[0141] 由图 5 可知，附近玩家包含在编号为 1 ～ 9 的格子中。假如计算后得知 1 ～ 9 号格子包含的数据分布在编号为 N1、N2、N3 这 3 个 Lbsstore 上。那么发查询请求给 3 个 Lbsstore，请求这些格子内的玩家信息，从而获得上述玩家附近的玩家信息。

[0142] 在以上实施例中，Lbsstore 会根据半径以及玩家坐标点计算出附近的玩家，然后将结果返回给 Lbsservice。然后 lbsserivce 将收到的来自 Lbsstore 返回的结果组装好后发送给 GameServer，由 GameServer 进行进一步的处理。GameServer 对收到的结果如何处理，本发明实施例不予限定。

[0143] 下面针对 Lbsservice 的功能介绍 Lbsservice 的设计方案。

[0144] 3.1.1、清除重复玩家地理信息：

[0145] 在设置玩家地理信息的时候，需要清除玩家之前上报的地理信息，以防止过期地理信息造成的数据错误，提高定位服务的准确性。由于 Lbsservice 并不知道之前玩家地理信息存放在哪个 Lbsstore 上面，计算出当前地理信息存放的 Lbsstore 后，给其发送存放(set)指示消息实现存放地理信息。然后采用了广播的方式发送一条删除(delete)指示消息给其他所有的 Lbsstore，进行删除操作，删除上述玩家的地理信息。如图 10 所示。

[0146] 如图 11 所示，为删除 playerID 3 的示意图。在 delete 消息包中包含玩家的 playerID(玩家标识)。在 Lbsstore 上，建立了一个以 playerID 为 key，GridGID 为值的 hash 表。delete 时，广播的 delete 消息最多只在一个 Lbsstore 上能根据 playerID 找到 GridGID。然后根据这个 GridGID，从存储地理信息的 hash 表中删除玩家的信息，然后在删除这个 playerID 到 GridGID 的索引关系即可。

[0147] 3.1.2、确定格子所属的 Lbsstore 编号：

[0148] 在本发明实施例中，将地球划分为成了 100*100 的格子，每个格子都有一个 GridGID。存储玩家信息的 Lbsstore 可能有多个，对于格子编号为 GridGID1 的玩家信息，需要决定其具体存放在哪个 Lbsstore 上。

[0149] 假设我们有 N 个 Lbsstore 服务器，对于存放 GridGID1 格子数据的 Lbsstore 的编号 M1，我们可以简单如下计算：

[0150] M1 = hash(GridGID1) % N；

[0151] 但是这样有一个坏处：比如我们要查找某个玩家附近 1KM 内的玩家，可以查找的格子数目大概是 2000/100*2000/100 = 400 个格子，对这 400 个格子 hash 计算其存储的 Lbsstore 编号，可能包含了所有的 Lbsstore。也就是说一次查找附近的人的操作需要向所有的 Lbsstore 请求数据，这样会导致 Lbsstore 数据处理压力会很大。

[0152] 如果能够实现单个查找附近的人的请求，大部分数据集中在少部分机器上，将会大大减轻 Lbsstore 的数据处理压力。本发明实施例可以采用在一个更粗的粒度上将一些范围内的数据集中存放在同一个机器上。也就是说存放在同一个机器上的区域范围的最小粒度应该比 100 米大。由于格子划分是 100*100。在本发明实施例中，按照之前的划分方案将地球再次划分为大小为 10000*10000 的格子(如果 10000 过大可以调整)。这样一个大格子中的数据会分发存储到同一台机器上。

[0153] 大格子的宽度是 10000*10000 时，它包含了 10000 个小格子，也就是这 10000 个小格子的数据都会存放到同一台机器上。对每个大格子也按之前小格子的规则编号，称之为虚拟机标识(VirtualMachineGridGID)，后续称为虚拟机标识 GridGID。

[0154] 对于一个坐标点(x,y)计算虚拟机标识 GridGID 所属的 Lbsstore 编号如下：

[0155] 首先计算 VirtualMachineGridGID 如下：VirtualMachineGridGID = ((40000*1000)/10000)*(y/10000)+x/10000+1；假设有 N 个 Lbsstore，那么存储坐标点(x,

y) 所在的 Lbsstore 编号 M1 计算如下 :M1 = hash(VirtualMachineGridGID) % N

[0156] Lbsservice 在存储玩家地理信息时,根据上述方式计算出该玩家地理信息存放的 Lbsstore 编号 M,然后将玩家地理信息分发到该 Lbsstore 上。Lbsstore 再计算 GridGID,然后将数据存入 hash 表中。

[0157] 首先根据玩家坐标以及半径,计算出一个圆的外接正方形,然后看这个外接正方形包含哪些大格子(计算方式跟之前一样,只是格子边长变大了)。得到了大格子的虚拟机标识 GridGID,就可以按照上面的公式计算出相应的 Lbsstore 编号。

[0158] 很显然,本发明实施例通过在一个更粗的粒度上将一些范围内的数据集中存放在同一个机器上,将查找附近的人的需求(查找附近的人范围不会无限大的,不然无意义,现在程序限制最多 2KM),现在单个查找附近的人的请求会集中分发到 1 到 2 个 Lbsstore 上,最多可能分发到 4 个 Lbsstore 上去。这样可以大大减少这种请求对 Lbsstore 的压力。

[0159] 3.3.3、Lbsservice 对数据的组装 :

[0160] 由于请求玩家附近的玩家时,可能附近玩家信息分布在多个 Lbsstore 上,也就是 Lbsservice 需要接收多个 Lbsstore 返回的数据才能得到完整的查询结果,然后需要将多份数据拼装好返回给 GameServer。

[0161] 在 Lbsservice 上可以定义如下的数据结构 :

[0162]

```
struct tagNearPlayersInfo
{
    int iUpdateTm;      // 发送请求的时间, 用于超时处理
    int iNeedDataParts;     // 应该收到的数据总数
    int iReceivedDataPart;      // 已经接受到的数据总数
    int iPosX;      // 玩家 X 坐标
    int iPosY;      // 玩家 Y 坐标
    int iSeq;      // 序列号
    uint32_t dwBusID;      // 发送消息的 GameSvr 的 bus 地址
    int iPlayerCnt;      // 玩家数目
    tagplayerDetail stPlayerList[MAX_LBS_PLAYER_LIST_NUM];
};
```

[0163] 并且建立了一个以 playerID 为 key,值为 tagNearPlayersInfo 的 hash 表。当 Lbsservice 发请求到多个 Lbsstore 请求附近玩家信息时,会插入一个以 playerID 为 key 的数据到一个 Map(一种文本文件)中,其中记录了插入的时间,已经接收的数据,需要接收的数据,消息序列号等(需要接收的数据就是请求分发的 Lbsstore 的数目)。Lbsservice 每收到从 Lbsstore 响应的数据 iReceivedDataPart 就增加 1,如果 iNeedDataParts = iReceivedDataPart(也就是已经接收到完整的数据),或者已经超时,那么就将已经接收好的数据发送给 GameServer。

[0164] 使用上述 Map 还可以启动控制玩家请求频率的作用，比如玩家第一次请求了获取附近玩家，马上又发送一个请求，那么 Map 中应该有该玩家的信息，第二次的请求可以直接拒绝；在服务器一侧可以定义超时时间是 10s，也就是 10s 之内玩家同时发送 2 个获取附近玩家的请求，仅处理第一次发送的请求。

[0165] 3.3.4、程序使用的 hash 函数：

[0166] 在前述实施例的描述中，会对 virtualMachineGridGID 进行一次 hash 处理，然后计算出玩家信息所在的 Lbsstore 编号。

[0167] 在 Lbsstore 上根据玩家 GridGID 插入数据到 hash 表中时，也会先计算 hash(GridGID)，然后用 hash(GridGID) % hashnum 才得到该数据所在的 hash 桶 (hash 桶排序）编号。原因是 GridGID 的计算规律很强，可能会导致一些格子数据集中到同一个 hash 桶中。所以本发明实施例可以再对 GridGID 进行一次 hash 处理，从而增加数据的随机性。

[0168] 对于整个地球二维化后，总的格子数目有 800 亿个，为了区分不同的格子，可以使用一个 uint64_t(64 位无符号整数）类型数据来存储 GridGID。为了使 GridGID 有更好的分布性能，采用如下算法 Wang/Jenkins hash 算法（王 / 詹金斯哈希算法）对 GridGID 进行 hash 处理。

[0169]

```
uint64_t CLbsPosConvert::hash64shift(uint64_t ullKey)
{
    ullKey = (~ullKey) + (ullKey << 21);
    ullKey = ullKey ^ (ullKey >> 24);
    ullKey = (ullKey + (ullKey << 3)) + (ullKey << 8);
    ullKey = ullKey ^ (ullKey >> 14);
    ullKey = (ullKey + (ullKey << 2)) + (ullKey << 4);
    ullKey = ullKey ^ (ullKey >> 28);
    ullKey = ullKey + (ullKey << 31);
```

[0170]

```
    return ullKey;
}
```

[0171] 3.2、Lbsstore 详细设计：

[0172] Lbsstore 用于存储玩家地理信息，在本发明实施例中可以使用共享内存上的一个 hash 表来存储玩家的地理信息。地球被分为 100*100 的格子，玩家信息存放在这样的格子中。hash 表的 key 是格子的 GridGID，值 (value) 就是格子中的玩家信息。由于某些格子中玩家数量可能很大，而某些格子玩家数量可能很少。hash 需要支持在格子中存放不同数量的玩家。如果 value 使用一个数组，会降低共享内存的利用率，而且由于无法估算一个格子中玩家的最大数量，数组大小是一个问题。

[0173] 因此，在本发明实施例中 value 的值采用链表存储，每个格子中的玩家都用链表

的形式链接起来。链表采用头插法，保证最新上报数据的玩家在链表最前面，同时也在逻辑上保证了链表中的数据是按照上报时间排序的。这个特性在玩家信息GC(Garbage Collection，超时清除玩家信息信息）以及查找附近玩家时只查询某个时间点内上报地理信息的玩家时比较有用。

[0174] 图12是存放玩家信息的hash表结构图。

[0175] 在图12中GridIndexNode就是代表一个格子，3个格子节点链接代表的是hash(key)冲突的情况，采用链表的方式可以解决冲突。每个格子节点，也就是GridIndexNode中可以挂玩家节点信息，也就是PlayerNode，格子中有多少玩家，那么该格子中就有多少个PlayerNode，PlayerNode通过链表的形式链接起来。

[0176] GridIndexNode结构形式如下：

[0177]

```
struct tagCGridIndexNodeInfo
    {
        int iPlayerNum;        // 格子中的玩家数目
        int iPlayerIndex;      // 玩家在玩家共享池中的索引
        int iLastGcTm;         // 上一次进行GC（删除过时玩家的时间）
        void Init()
        {
```

[0178]

```
            iPlayerIndex = NIL_PLAYER_INDEX;
            iPlayerNum = 0;
            iLastGcTm = 0;
        }
    };
```

[0179] 在上面结构中，iPlayerNum用于维护格子中总的玩家数量，iLastGcTm是上一次格子GC的时间，用于加快GC的处理的效率，避免不必要的重复GC。

[0180] PlayerNode结构形式如下：

[0181]

```
// 玩家地理信息
struct tagCPlayerPosInfo
{
    int iUpdateTm;      // 信息更新时间,绝对时间
    int iXpos;      // X 坐标，单位 m
    int iYpos;      // Y 坐标，单位 m
    uint32_t uiPlayerID;      // 玩家 ID
    int  iNextPlayerPos;      // 同一个格子中下一个玩家在共享内存池中的索引
    void Init()
    {   iUpdateTm = 0;
        iXpos = -1;
        iYpos = -1;
        uiPlayerID = 0;
        iNextPlayerPos = NIL_PLAYER_INDEX;
    }
};
```

[0182] 另外，为了维护玩家信息在 Lbsstore 上的唯一性，每个 Lbsstore 还有一个从 playerID 到 GridGID 的反向索引，如图 13 所示。

[0183] 因此，在 Lbsstore 上增加一个玩家的信息时，首先查找反向索引表，如果玩家信息已经存在，那么就先删除掉该条信息，然后再往共享内存中插入玩家新的地理信息。

[0184] 这个反向索引结构中数据就包含一个 gridGID，如下所示：

[0185]

```
// 玩家索引信息
struct tagCPlayerIndexInfo
{
    uint64_t ullGridGid;
    void Init()
    {
        ullGridGid = 0;
    }
};
```

[0186] 3.2.1、Lbsstore 对超时数据的处理，即 GC 处理：

[0187] 因为玩家地理信息是有时限的，随着玩家的地理位置的变化，玩家的地理信息就会失效。因此对于存储在 Lbsstore 中的数据，需要清除过时的玩家地理信息。方案中采用基于 bucket（桶）的定时清理方式清除 Lbsstore 上的过时玩家信息。如图 14 所示。

[0188] 因为一个 hash 表中可能存放有数十万的玩家信息，bucket 数目也高达几十万。如果每次定时清理对 hash 表进行一次全扫描的话。假设这段时间是 t，那么 t 时间内 Lbsstore 将失去对外服务的能力。而且玩家信息超时可能是数小时，没有必要一次处理所有的数据。

[0189] 基于上面的原因，Lbsstore 采用基于 bucket 的定时清理方式，也就是每次只处理有限个 hash 桶中的 bucket，并记录下次 tick 中需要处理的 bucket 编号。如图 14 所示。

[0190] 假设每次处理 m 个 bucket，整个 hash 表有 N 个 bucket，tick 的时间间隔是 T，玩家信息超时时间是 L。那么扫描一次表的时间是：Total = (N/m)*T。只要 Total<=L，那么超时数据可以得到及时清除。可以通过设置合适的 m 和 T 值，可以保证 Lbsstore 及时清除超时数据，又不影响其对外服务的能力。例如：每个 tick 处理的 bucket 数目可以是 500 个。在图 14 中，每个 tick 处理的 bucket，为 3 个 bucket 内的值。

[0191] 3.2.2、Lbsstore 上查找附近人的方式：

[0192] 由前面实施例中记载的数据模型可以知道：要查找附近的人时，先得到一个以附近范围为半径的圆的外接正方形，然后看看哪些格子在这个正方形中，再进一步判断这些格子中哪些数据满足条件。这涉及到距离的计算。

[0193] 由于格子划分较小，具体是 100*100。因此可以通过计算出附近范围的外接正方形，然后看它包含哪些格子，只要包含了格子，就认为格子中的所有玩家信息都是属于这个附近范围内的，误差不会超过 100 米，应该是可以接受的。如图 5 所示。图 5 中，半径为 r 的附近玩家在红色的正方形内，但是我们可以简单的认为格子 1,2,3,4,6,7,9 中的玩家都是这个范围内的，因为格子大小是 100*100，可以允许有一定的误差，即：小于 100 米；如果一个格子的边长，那么这个误差小于这个格子的边长。

[0194] 使用上面的计算方式，就可以忽略掉距离的计算，只是找到正方形包含的格子，然后输出其中的玩家数据即可，计算量将会大大减少。

[0195] 以上的方案是基于一级 hash 的，但是在查找附近 2KM（最大允许 2KM 的范围）的玩家，需要搜索的格子数目是 (4000/100+1)*(4000/100+1) = 1681 个格子，也就是不管这个 1681 个格子中多少个格子有数据，都需要 1600 次循环，1600 次从 hash 表获取值（键值）getvalue(key) 的过程，查找次数将会较多。

[0196] 为了解决查找次数多的问题，本发明实施例提供的一种方案是扩大单个格子的边长，比如单个格子的边长为：200*200，那么搜寻就变成了 400 次。但是格子中数据扩大 4 倍，是否会影响程序 GC 等需要考虑：同一个格子数据可能会比较多，哈希桶内数据会发生较多的冲突。

[0197] 本发明实施例提供的另一种方案是：将格子分成 1KM*1KM，建立一个 2 级 hash，查找的时候，先找一级 hash，也就是 1KM*1KM 的格子，这个大格子中放一个大小是 100 的数组，记录数组记录哪个格子有数据。那么 2KM 的范围，最多需要 (4KM/1KM+1)*(4KM/1KM+1) = 25 个大格子。然后在这 25 个大格子中找出有数据的格子，再从这些格子中找玩家数据。如

果有数据的格子比较少，这种方式可以大大减少循环的次数。当然这种方式的维护会复杂一点，包括对数据的增加删除时对这个索引的维护都需要时间。如果采用一级 hash 效果不好，可以考虑该方案。

[0198] 3.2.3、lbsstore 的清共享内存数据恢复：

[0199] 由于在 Lbsstore 中，数据是存放共享内存的，因此需要考虑清共享内存恢复的问题。本发明实施例的方案可以采用记录 OSS 日志，清共享内存时，依次从 OSS 日志中读取一条条的记录进行恢复。记录 OSS 日志时，Lbsstore 对数据的操作总共就两种：一种是插入，另一种是删除。每一次的插入操作和每一次的删除操作（不包括 GC 时对数据的删除）都记录一个 OSS 日志，插入日志和删除日志都记录了玩家的详细信息。日志可以每个小时记录一份。超过一定时间的日志（以玩家地理信息超时时间为准）就清除掉。

[0200] 在清共享内存重启时，首先将这些 OSS 日志按照日期排序，读取文件的时候从时间最早的 OSS 日志开始读入，如果是插入操作就插入记录，删除操作就删除记录。由于 Lbsstore 上有一个 playerID 到 GridGID 的反向索引，可以保证数据唯一，另外读入数据是从最早文件开始，因此可以保证最后的玩家信息是最新的。另外，由于日志文件有过期时间，几个小时，那么文件记录数应该不会过多。

[0201] 3.2.4、Lbsstore 数据扩容时数据恢复：

[0202] 玩家位置信息的数据增多的时候，Lbsstore 数目可能也需要要增多，根据之前 Lbsservice 将数据分发到 Lbsstore 上面的方案，采用的是取模的方式，因此 Lbsstore 数目增多的时候，共享内存中数据可能失效，因此扩容的时候需要重新载入玩家信息，这个过程实质上就是清共享内存重启的过程。

[0203] 区别在于，此种情况下需要将所有 Lbsstore 的 OSS 日志放在每个 Lbsstore 的读取目录下，供清共享内存重启时读取。读取文件时，依然是从最早的文件开始读入，只是现在要根据玩家的坐标 (x,y)（这个坐标 OSS 中有记录）计算出 virtualMachineGID，然后计算 virualMachineGID% LbsstoreNum，如果编号与当前 Lbsstore 的编号相符，那么数据属于当前 Lbsstore，那么就根据 OSS 的记录增加或者删除该记录，否则丢弃不处理。

[0204] 增加 Lbsstore 进行清共享内存重启的过程，时间长短与整个 Lbsstore 中有效 OSS 记录的总数有关。OSS 记录只有上报地理位置时才会有记录，插入，删除操作，频率不会很高。另外，读取数据时，此时磁盘是顺序读取，在磁盘预取等作用下，效率可以得到保证。

[0205] 目前使用这种方案，后期如果数据量太大，Lbsservice 可以考虑用一致 hash 的方式分配存储数据的 Lbsstore，这样扩容时可以保证增加一台机器只会导致一台机器的内容失效，读取的记录数也只与一台 Lbsstore 上的数据量有关，但是一致 hash 解决的是某台机器负载过重，如果要均分所有的压力需要增加多个 Lbsstore 或者采用虚拟节点的概念，增加了复杂性，重新部署时分配文件复杂度会相对较高。

[0206] 本发明实施例还提供了一种位置服务器，如图 15 所示，包括：

[0207] 网格确定单元 1501，用于在接收到包含目标位置信息的定位请求后，确定目标网格；上述目标网格包括上述目标位置信息对应的网格，以及与上述目标位置信息对应的网格距离在设定范围内的网格；上述网格是由地图划分正方形得到；

[0208] 设备确定单元 1502，用于确定目标网格所属的位置服务存储设备；在位置服务存储设备中以网格编号为关键字以终端信息为值进行数据存储，且在每个位置服务存储设备

中存储的最小粒度为上述网格对应的正方形的边长的 M 倍，M 大于等于 2 ；

[0209] 查询单元 1503，用于从确定的位置服务存储设备中，查询与上述目标网格对应的终端信息 ；

[0210] 信息发送单元 1504，用于向上述定位请求的发送方发送查询到的终端信息。

[0211] 在本发明实施例中，网格由地图划分而来，也即是说网格与实际的地理位置是对应的。网格的边长实际上就是这个网格对应地理位置的边长。网格边长越小，最终定位时的精度将会越高，网格越大则存储的网格数量越少，查找所用的时间则可能越少。具体的长度本发明实施例不予限定，推荐可以使用 100m 作为边长。

[0212] 这里 M 的具体取值，可以由技术人员设定。若查找周围最大 2KM 范围内的终端信息，那么可以将这个值设定为 100 或者 50 等值，具体取值本发明实施例不予限定。

[0213] 本发明实施例，采用网格的方式将地理位置进行划分，并采用划分后的网格进行位置信息存储 ；在位置服务过程中不必使用经纬度进行距离计算减少计算量，从而可以提高服务器的响应速度。另外，在为位置服务存储设备分配网格时，使用的最小粒度比网格要大，这样可以使更多的相邻网格分配到同一个位置服务存储设备，使得确定的目标网格集中在少数几个位置服务存储设备内，从而减少对位置服务存储设备的调用，降低位置服务存储设备的计算压力，从而进一步的提升真个位置服务的系统性能。另外，还可以将网格可以设置得较小，提高定位精度，并减少因单个网格位置信息太多导致的冲突。

[0214] 在本发明实施例中，位置服务存储设备中的位置信息可以来源于其他服务器搜集的信息也可以来源于位置服务器直接收集的信息，本发明实施例提供了如何获得并存储终端的位置信息的具体实现方案，具体如下 ：进一步地，如图 16 所示，上述位置服务器，还包括 ：

[0215] 信息接收单元 1601，用于接收终端上报的位置信息以及终端信息 ；

[0216] 上述网格确定单元 1501，还用于确定上述位置信息所属的网格 ；

[0217] 上述设备确定单元 1502，还用于确定上述网格确定单元 1501 确定的网格所属的位置服务存储设备 ；

[0218] 上述信息发送单元 1504，还用于将上述终端上报的位置信息以及终端信息，发送给确定的位置服务存储设备进行存储。

[0219] 本发明实施例还提供了优选的存储方案，并且防止存储的数据发生重叠导致数据冲突的情况 ；具体如下 ：进一步地，在位置服务存储设备中还以终端信息为关键字以网格编号为值进行数据存储，上述位置服务器还包括 ：

[0220] 上述信息发送单元 1504，还用于向确定的位置服务存储设备以外的其他位置服务存储设备发送位置信息删除指令，删除存储的上述终端信息对应的位置信息。

[0221] 在本发明实施例中，向确定的位置服务存储设备以外的其他位置服务存储设备发送位置信息删除指令，具体的发送方案可以采用广播的方式发送，这样可以保持终端在服务器一侧的位置信息的唯一性。

[0222] 在本发明实施例中网格划分后可以进行编号，然后采用哈希算法对网格在位置服务存储设备间进行分配，从而提升查询时候的查询速度，另外也方便后续位置服务存储设备的扩展，具体如下 ：可选地，上述网格由地图划分正方形得到，所有网格的网格编号按照地理位置顺序编号 ；所有在位置服务存储设备中最小粒度的网格与位置服务存储设备的所

属关系符合哈希算法的规则；

[0223] 上述设备确定单元 1502，用于使用目标网格进行哈希查找确定标网格所属的位置服务存储设备。

[0224] 本发明实施例，虽然可以减少位置服务设备的调用，但是并不能保证在一个位置服务存储设备完成位置服务器功能，因此目标网格是可以分布于多个位置服务存储设备的，本发明实施例基于此给出了进一步的实现方案如下：进一步地，进一步地，若目标网格分布于两个或两个以上的位置服务存储设备；如图 17 所示，上述位置服务器还包括：

[0225] 组包单元 1701，用于将从位置服务存储设备查询到的终端信息进行组包；

[0226] 上述信息发送单元 1504，用于向上述定位请求的发送方组包单元的组包结果。

[0227] 由于终端用户可能会不停的通过终端发送定位的请求，如果每次定位请求都进行定位处理，将会给服务器造成较大的压力；另外，通常来说间隔时间很短的情况下，定位的结果通常也极少发生变化，因此这类频繁的定位也是没有必要的。本发明实施例为了降低服务器的压力，减少不必要的定位操作，提出了如下解决方案：进一步地，如图 18 所示，上述位置服务器，还包括：

[0228] 定位控制单元 1801，用于若再次接收到来自上述终端的定位请求，确定与前次定位请求的时间间隔是否超过预定阈值，若为超过预定阈值，则拒绝执行定位。

[0229] 在本发明实施例中，将地图划分为网格，网格的边长越短，则定位的精度越高，但是计算量越大；网格的边长越短，则计算量越小响应速度越快；为了综合这两者的优点，本发明实施例提供了如下解决方案：可选地，上述网格确定单元 1501，用于首先在测试网格中查询包含终端信息的测试网格，上述测试网格是与上述目标位置信息对应的测试网格，测试网格由地图划分正方形得到，且边长大于上述网格；然后确定查询到的测试网格中包含的网格为目标网格。

[0230] 在本发明实施例中，测试网格的边长较大，这样可以首先使用大网格来过滤掉目标网格中没有位置信息的网格，避免不必要的查找，从而减少查找量，提高系统响应速度。

[0231] 本发明实施例还提供了一种位置服务存储设备，如图 19 所示，包括：

[0232] 请求接收单元 1901，用于接收位置服务器的查询请求，上述查询请求包含有待查询的目标网格；

[0233] 数据存储单元 1902，用于以网格编号为关键字以终端信息为值进行数据存储，且在每个位置服务存储设备中存储的最小粒度为上述网格对应的正方形的边长的 M 倍，M 大于等于 2；上述网格是由地图划分正方形得到；

[0234] 查询单元 1903，用于使用上述目标网格的编号查询并获得与上述待查询的目标网格对应的终端信息；

[0235] 信息发送单元 1904，用于将获得的终端信息发送给上述位置服务器。

[0236] 在本发明实施例中，网格由地图划分而来，也即是说网格与实际的地理位置是对应的。网格的边长实际上就是这个网格对应地理位置的边长。网格边长越小，最终定位时的精度将会越高，网格越大则存储的网格数量越少，查找所用的时间则可能越少。具体的长度本发明实施例不予限定，推荐可以使用 100m 作为边长。

[0237] 这里 M 的具体取值，可以由技术人员设定。若查找周围最大 2KM 范围内的终端信息，那么可以将这个值设定为 100 或者 50 等值，具体取值本发明实施例不予限定。

[0238] 本发明实施例，采用网格的方式将地理位置进行划分，并采用划分后的网格进行位置信息存储；在位置服务过程中不必使用经纬度进行距离计算减少计算量，从而可以提高服务器的响应速度。另外，在为位置服务存储设备分配网格时，使用的最小粒度比网格要大，这样可以使更多的相邻网格分配到同一个位置服务存储设备，使得确定的目标网格集中在少数几个位置服务存储设备内，从而减少对位置服务存储设备的调用，降低位置服务存储设备的计算压力，从而进一步的提升真个位置服务的系统性能。另外，还可以将网格可以设置得较小，提高定位精度，并减少因单个网格位置信息太多导致的冲突。

[0239] 可选地，本发明实施例还提供了在位置服务存储设备网格的分布以及网格内数据存储的具体实现方案如下：上述数据存储单元 1902，用于存储的网格的网格编号按照哈希算法的规则进行存储，并采用哈希桶排序。

[0240] 在本发明实施例中，位置服务存储设备中的位置信息可以来源于其他服务器搜集的信息也可以来源于位置服务器直接收集的信息，本发明实施例提供了如何获得并存储终端的位置信息的具体实现方案，并且提供了保持位置信息唯一性的具体实现方案，具体如下：进一步地，如图 20 所示，上述位置服务存储设备，还包括：

[0241] 信息接收单元 2001，用于接收来自上述位置服务器发送的终端信息以及位置信息；

[0242] 上述查询单元 1903，还用于查找是否已经存在上述终端信息对应的位置信息；

[0243] 数据删除单元 2002，用于若上述查询单元 1903 查询结果为存在，则删除查找到的位置信息；

[0244] 上述数据存储单元 1902，用于若上述查询单元 1903 查询结果为不存在，或者，上述数据删除单元 2002 删除完毕后，将上述终端信息存放到与上述位置信息对应的网格中，上述位置信息采用哈希表存储。

[0245] 本发明实施例还提供了位置信息的具体存储方案，在本发明实施例中采用链表用头插法的方式对位置信息进行存储，这样可以使位置信息可以按照时间先后次序进行存储，方便后续对失效数据的删除，具体如下：可选地，上述哈希表的值采用链表的数据结构；

[0246] 上述数据存储单元 1902，用于将上述位置信息采用头插法将位置信息插入与上述位置信息对应的网格的链表中。

[0247] 由于终端的位置信息（地理信息）是有时限的，随着终端的地理位置的变化，终端的位置信息可能就会失效。因此对于存储在位置服务存储设备中的位置信息，需要清除已经过期的位置信息；这样不仅可以提高定位的准确性，还可以降低计算量提高系统响应速度。进一步地，如图 21 所示，上述位置服务存储设备，还包括：

[0248] 超时清理单元 2101，用于读取网格的回收时间，若回收时间距当前时间超过预定阈值，则从上述链表的后端往前依次删除超时的位置信息。

[0249] 由于位置服务存储设备中，可能存放的网格很多，哈希桶也可能很多，如果每次对一个位置服务存储设备全部的网格进行过期数据的清理，这可能需要较多的时间，并且这段时间内位置服务器将会停滞，造成位置服务器不稳定。为了提高位置服务的稳定性，清理过期数据的进程可以逐步进行，具体如下：可选地，上述超时清理单元 2101，用于按照预定的顺序在所有网格中选取预定数量网格读取回收时间。

[0250] 在本发明实施例中，预定数量的网格，可以是任意设定的，也可以是按照系统的忙闲度来调整的，例如系统越忙则这个数量越少，系统越闲则这个数量越多。采用本发明实施例方案，不会造成位置服务的停滞，可以提高位置服务的稳定性。

[0251] 在本发明实施例中，位置信息等数据可以存放在共享内存中，如果执行了清共享内存等处理，是可能需要进行数据恢复的，因此本发明实施例进一步提供了如何进行数据恢复的方案，如下：进一步地，如图 22 所示，上述位置服务存储设备，还包括：

[0252] 日志记录单元 2201，用于将位置信息的插入操作和删除操作记录到本端业务支撑系统日志中；

[0253] 日志清理单元 2202，用于在有本端业务支撑系统日志超时后，删除超时的本端业务支撑系统日志；

[0254] 恢复控制单元 2203，用于在清内存恢复过程中，按照本端业务支撑系统日志的记录时间从先到后依次读入并执行业务支撑系统日志记录的操作。

[0255] 基于本发明实施例的数据存储结构，以及数据恢复的方案，本发明实施例还提供了在位置服务存储设备扩容时的数据恢复，具体方案如下：进一步地，上述恢复控制单元 2203，还用于在位置服务存储设备扩容过程中，获取所有位置服务存储设备的业务支撑系统日志；读取获取到的业务支撑系统日志的记录，若当前读取到的记录属于本端位置服务存储设备，则执行当前读取到的记录对应的操作。

[0256] 本发明实施例还提供了一种位置服务系统，如图 23 所示，包括：位置服务器 2301 和位置服务存储设备 2302，位置服务器 2301 和位置服务存储设备 2302 以可通信方式连接；上述位置服务器 2301 为本发明实施例提供的任意一项的位置服务器 2301，上述位置服务存储设备 2302 为本发明实施例提供的任意一项的位置服务存储设备 2302。

[0257] 本发明实施例，采用网格的方式将地理位置进行划分，并采用划分后的网格进行位置信息存储；在位置服务过程中不必使用经纬度进行距离计算减少计算量，从而可以提高服务器的响应速度。另外，在为位置服务存储设备分配网格时，使用的最小粒度比网格要大，这样可以使更多的相邻网格分配到同一个位置服务存储设备，使得确定的目标网格集中在少数几个位置服务存储设备内，从而减少对位置服务存储设备的调用，降低位置服务存储设备的计算压力，从而进一步的提升真个位置服务的系统性能。另外，还可以将网格可以设置得较小，提高定位精度，并减少因单个网格位置信息太多导致的冲突。

[0258] 图 24 是本发明实施例提供的一种服务器结构示意图，该服务器 2400 可因配置或性能不同而产生比较大的差异，可以包括一个或一个以上中央处理器 (central processing units, CPU)2422（例如，一个或一个以上处理器）和存储器 2432，一个或一个以上存储应用程序 2442 或数据 2444 的存储介质 2430（例如一个或一个以上海量存储设备）。其中，存储器 2432 和存储介质 2430 可以是短暂存储或持久存储。存储在存储介质 2430 的程序可以包括一个或一个以上模块（图示没标出），每个模块可以包括对服务器中的一系列指令操作。更进一步地，中央处理器 2422 可以设置为与存储介质 2430 通信，在服务器 2400 上执行存储介质 2430 中的一系列指令操作。

[0259] 服务器 2400 还可以包括一个或一个以上电源 2426，一个或一个以上有线或无线网络接口 2450，一个或一个以上输入输出接口 2458，和 / 或，一个或一个以上操作系统 2441，例如 Windows ServerTM, Mac OS XTM, UnixTM, LinuxTM, FreeBSDTM 等等。

[0260] 上述实施例中由位置服务器或者位置服务存储设备所执行的步骤可以基于该图24所示的服务器结构。

[0261] 值得注意的是，上述位置服务器和位置服务存储设备实施例中，所包括的各个单元只是按照功能逻辑进行划分的，但并不局限于上述的划分，只要能够实现相应的功能即可；另外，各功能单元的具体名称也只是为了便于相互区分，并不用于限制本发明的保护范围。

[0262] 另外，本领域普通技术人员可以理解实现上述各方法实施例中的全部或部分步骤是可以通过程序来指令相关的硬件完成，相应的程序可以存储于一种计算机可读存储介质中，上述提到的存储介质可以是只读存储器，磁盘或光盘等。

[0263] 以上仅为本发明较佳的具体实施方式，但本发明的保护范围并不局限于此，任何熟悉本技术领域的技术人员在本发明实施例揭露的技术范围内，可轻易想到的变化或替换，都应涵盖在本发明的保护范围之内。因此，本发明的保护范围应该以权利要求的保护范围为准。

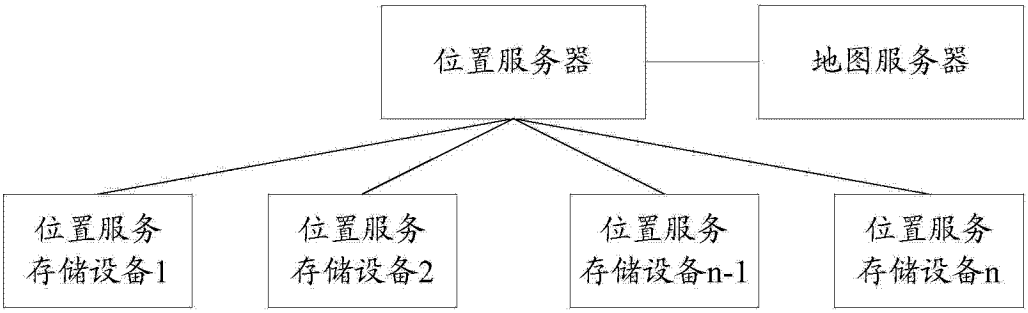

位置服务器
地图服务器
位置服务
存储设备1
位置服务
存储设备2
位置服务
存储设备n-1
位置服务
存储设备n

图 1

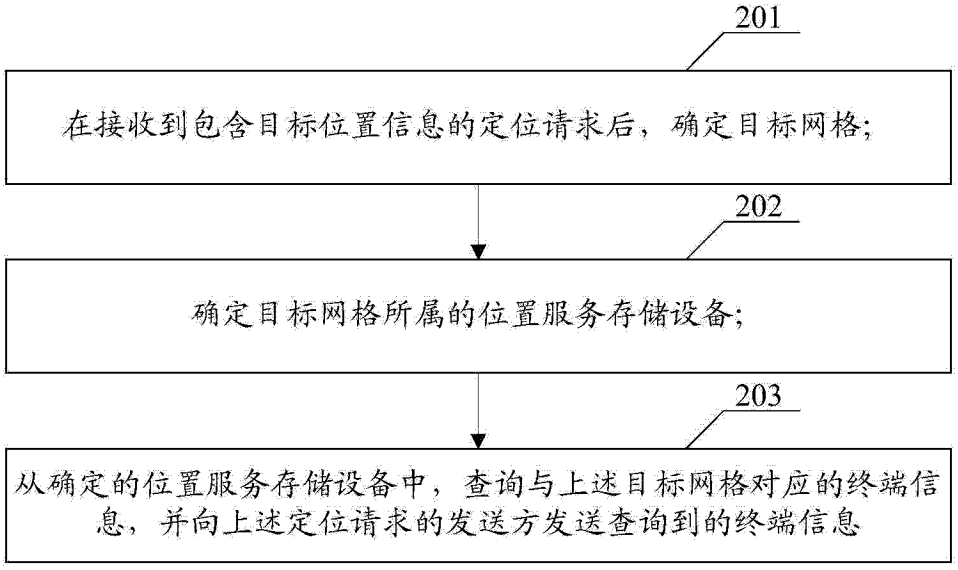

201
在接收到包含目标位置信息的定位请求后，确定目标网格;
202
确定目标网格所属的位置服务存储设备;
203
从确定的位置服务存储设备中，查询与上述目标网格对应的终端信息，并向上述定位请求的发送方发送查询到的终端信息

图 2

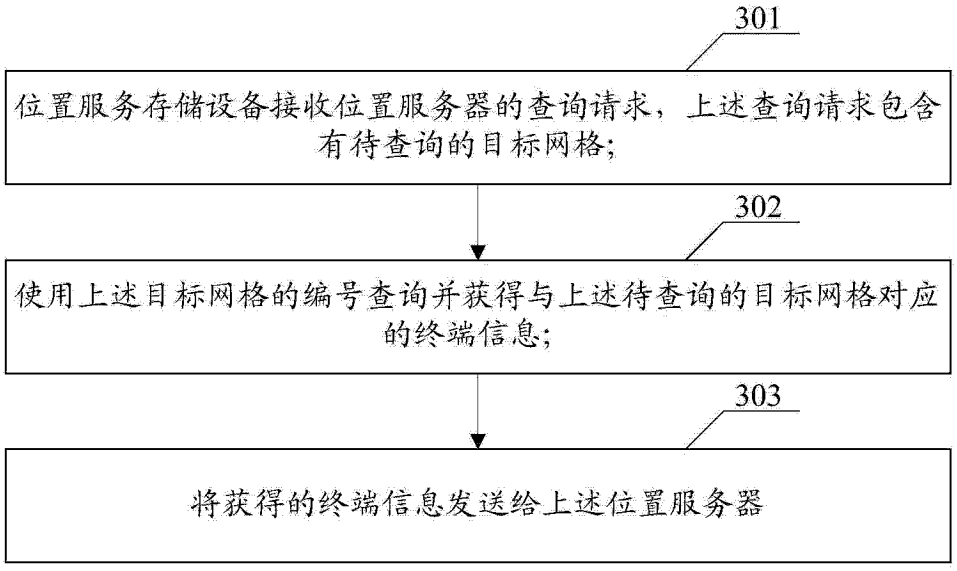

301
位置服务存储设备接收位置服务器的查询请求，上述查询请求包含有待查询的目标网格；
302
使用上述目标网格的编号查询并获得与上述待查询的目标网格对应的终端信息；
303
将获得的终端信息发送给上述位置服务器

图 3

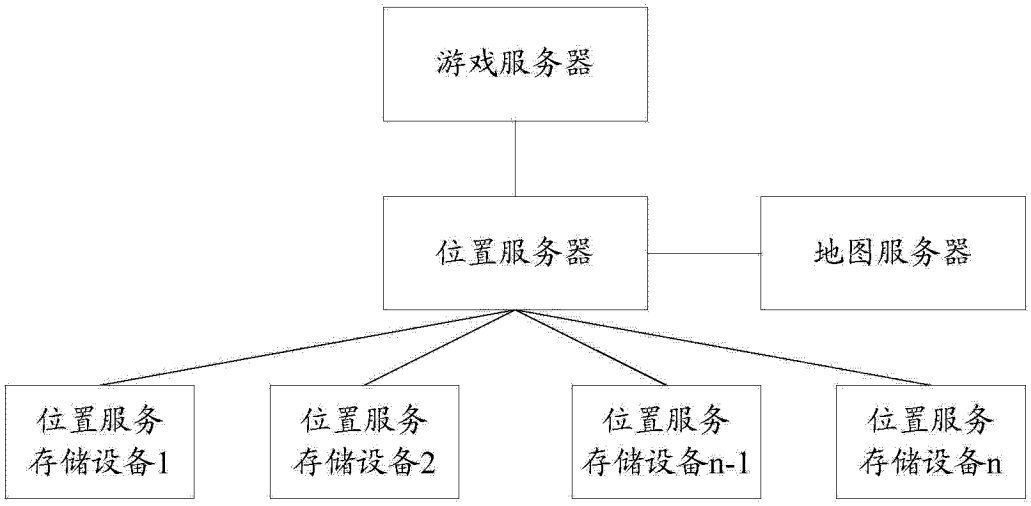

游戏服务器
位置服务器
地图服务器
位置服务存储设备1
位置服务存储设备2
位置服务存储设备n-1
位置服务存储设备n

图 4

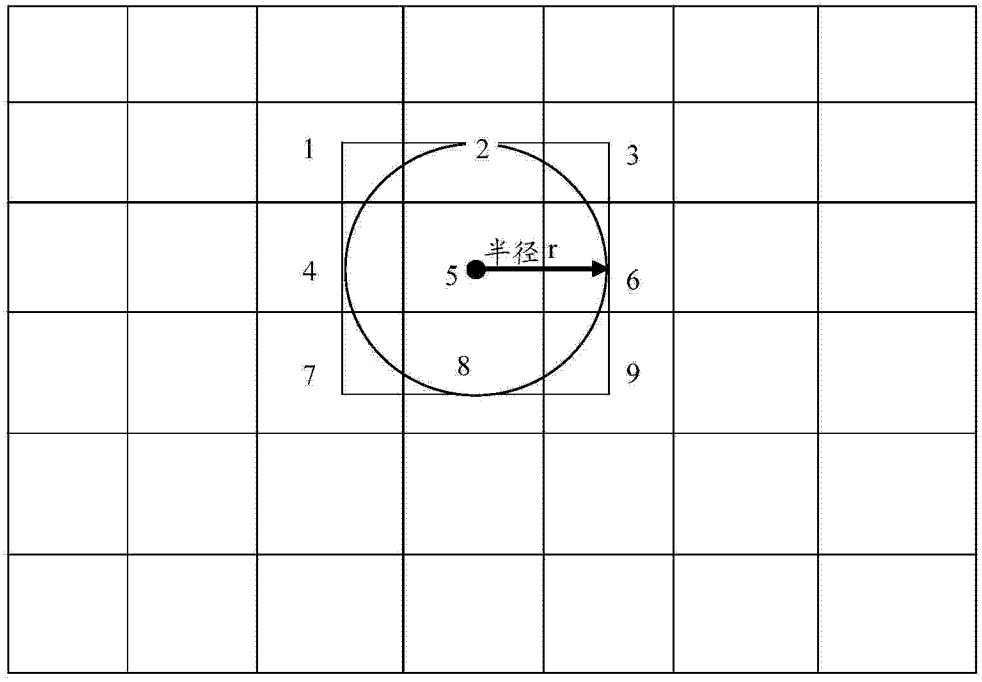


图 5

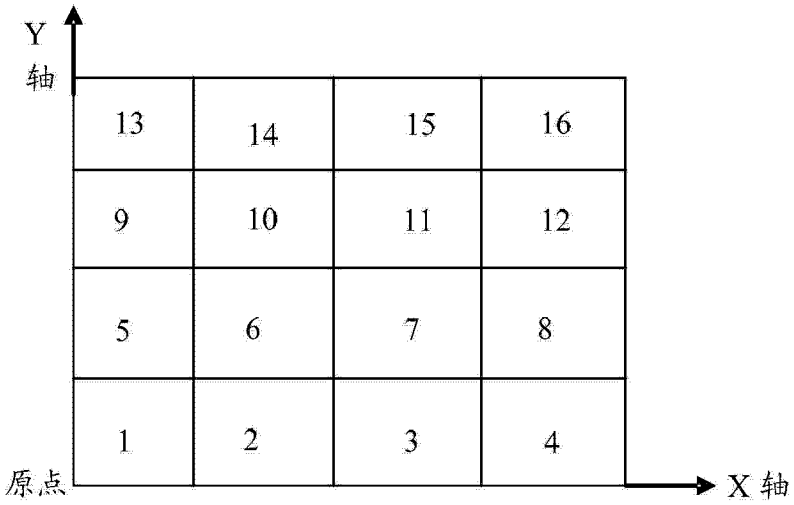


图 6

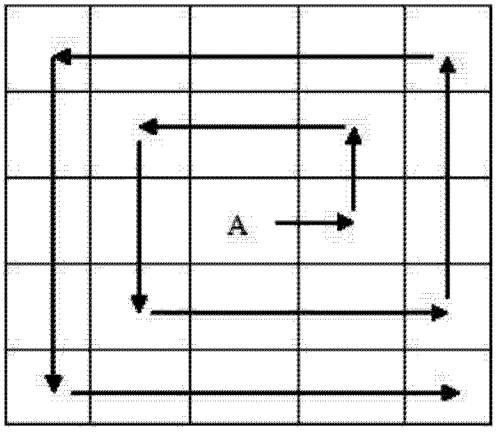

A

图 7

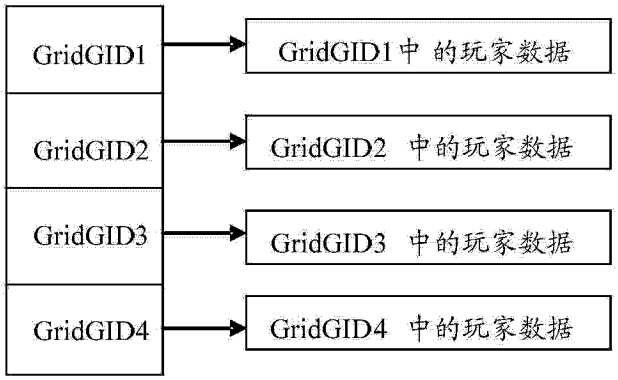

GridGID1
GridGID1中 的玩家数据
GridGID2
GridGID2 中的玩家数据
GridGID3
GridGID3 中的玩家数据
GridGID4
GridGID4 中的玩家数据

图 8

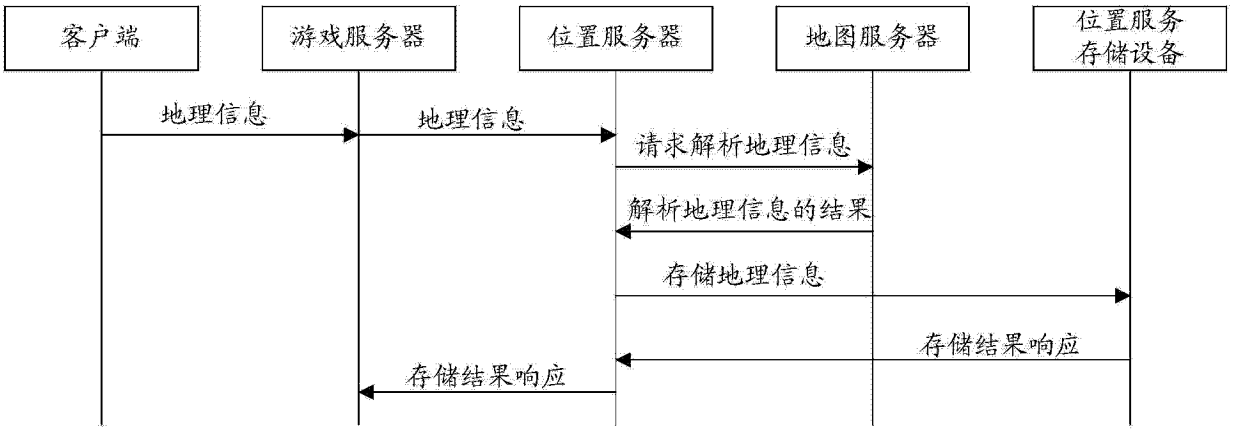

客户端
游戏服务器
位置服务器
地图服务器
位置服务
存储设备
地理信息
地理信息
请求解析地理信息
解析地理信息的结果
存储地理信息
存储结果响应
存储结果响应

图 9

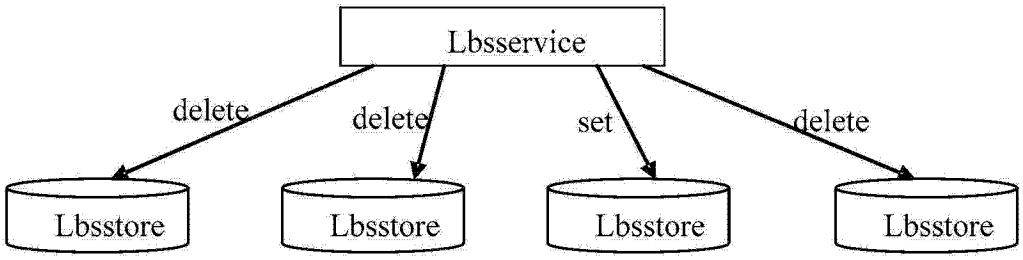

Lbsservice
delete
delete
set
delete
Lbsstore
Lbsstore
Lbsstore
Lbsstore

图 10

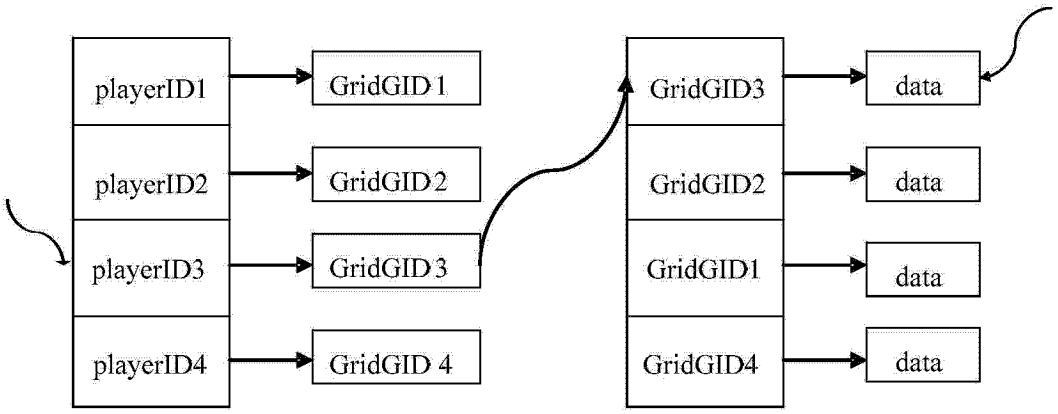

playerID1
playerID2
playerID3
playerID4
GridGID 1
GridGID 2
GridGID 3
GridGID 4
GridGID3
GridGID2
GridGID1
GridGID4
data
data
data
data

图 11

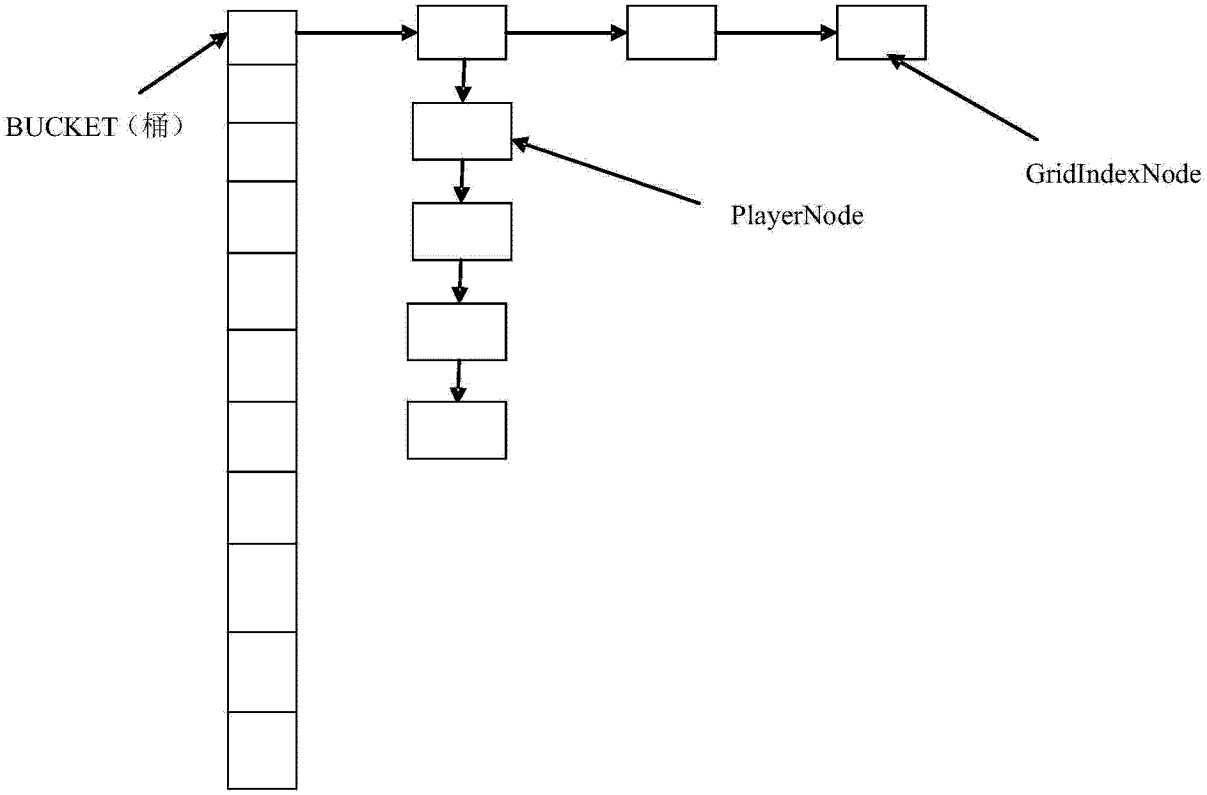

BUCKET（桶）
PlayerNode
GridIndexNode

图 12

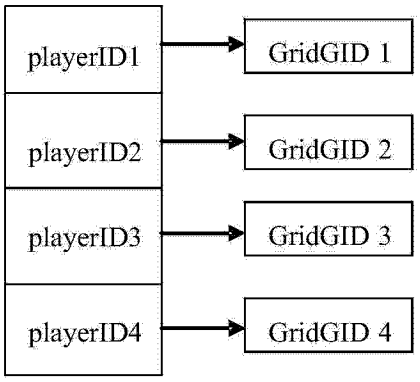


图 13

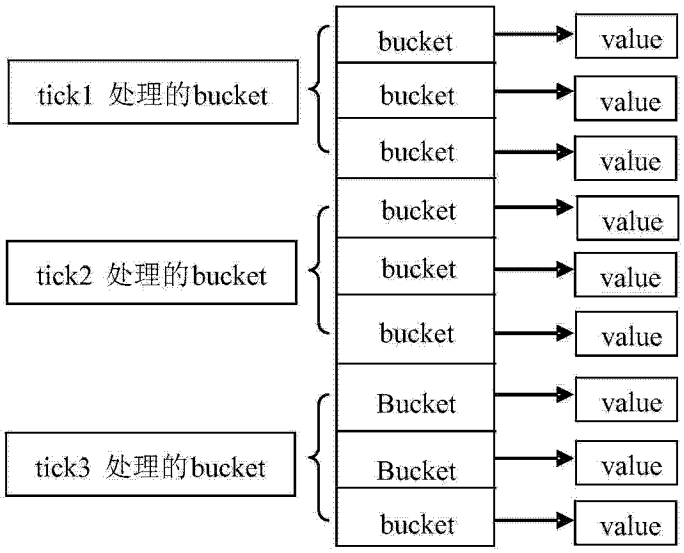


图 14

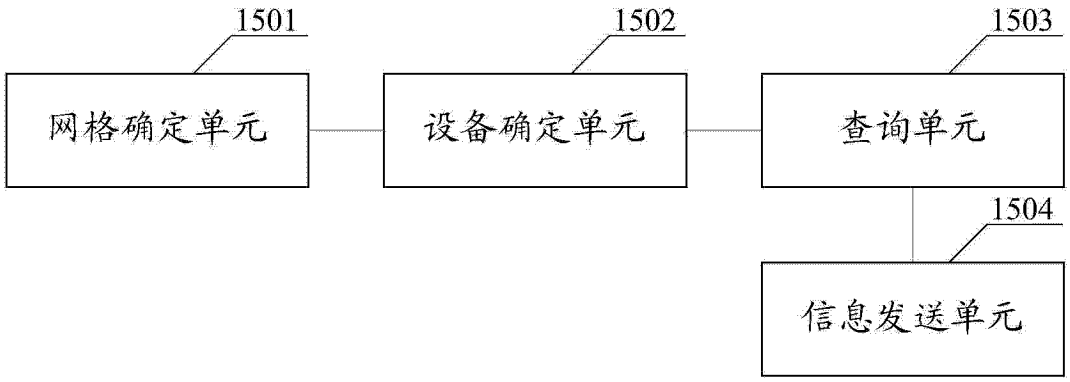


图 15

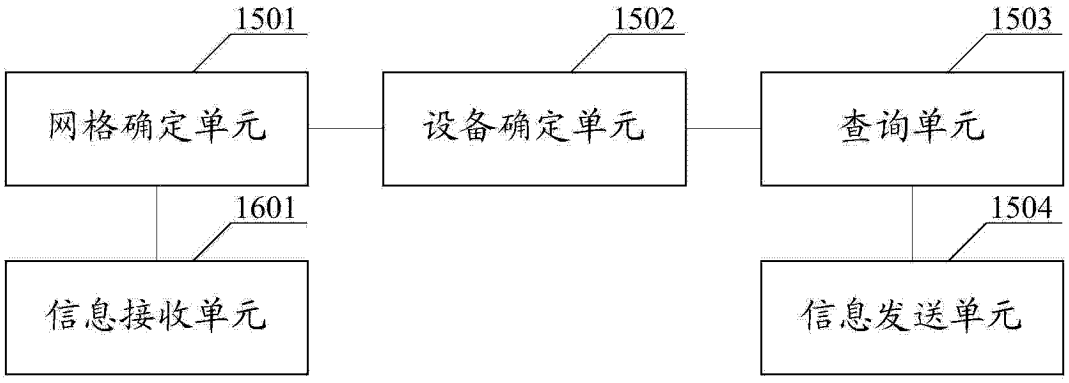


图 16

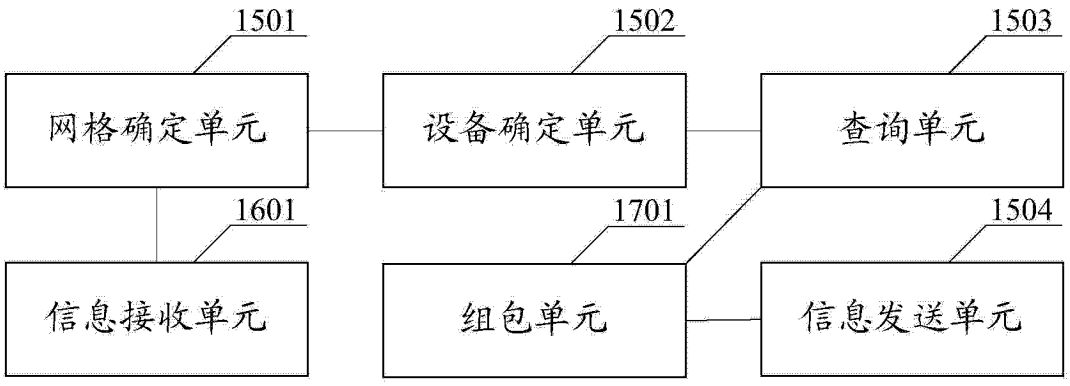

1501
网格确定单元
1502
设备确定单元
1503
查询单元
1601
信息接收单元
1701
组包单元
1504
信息发送单元

图 17

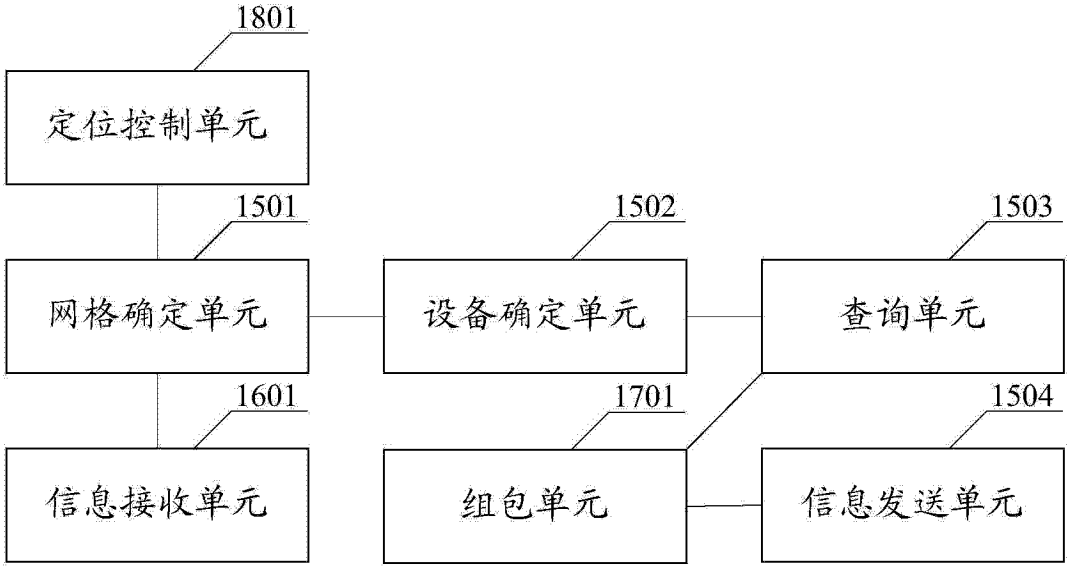

1801
定位控制单元
1501
网格确定单元
1502
设备确定单元
1503
查询单元
1601
信息接收单元
1701
组包单元
1504
信息发送单元

图 18

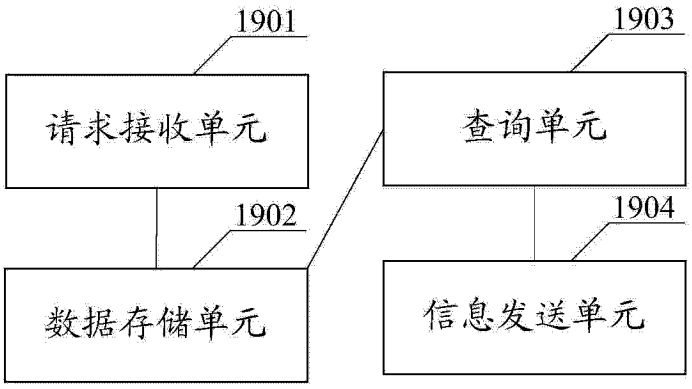

1901
请求接收单元
1903
查询单元
1902
数据存储单元
1904
信息发送单元

图 19

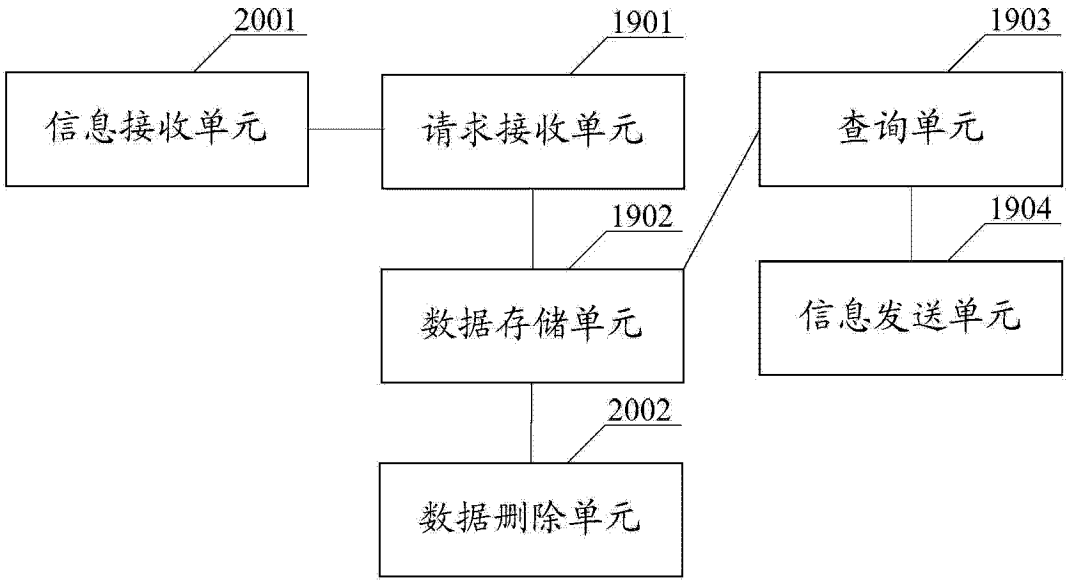

2001
信息接收单元
1901
请求接收单元
1903
查询单元
1902
数据存储单元
1904
信息发送单元
2002
数据删除单元

图 20

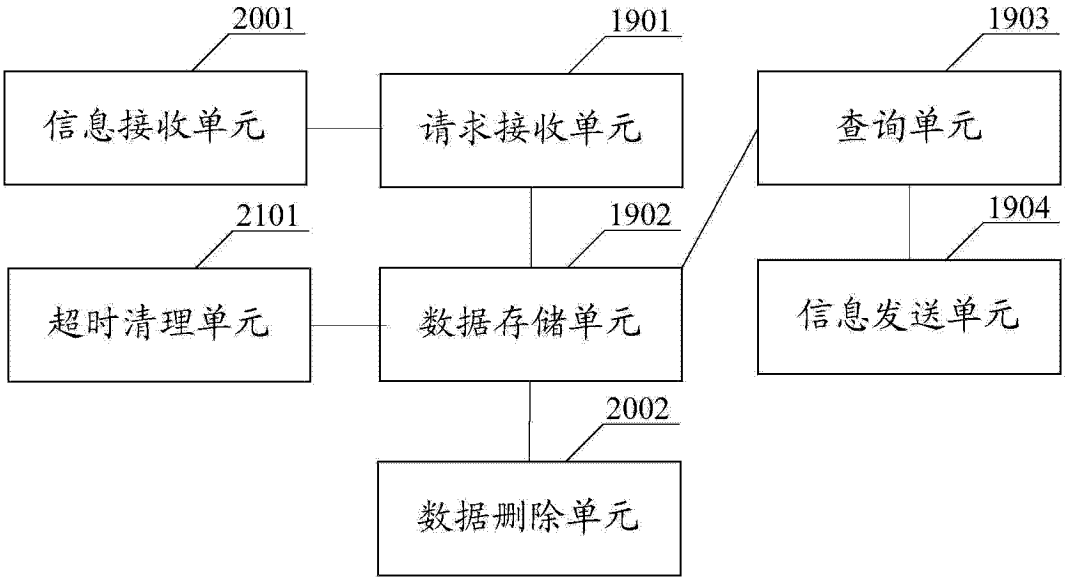

2001
信息接收单元
1901
请求接收单元
1903
查询单元
2101
超时清理单元
1902
数据存储单元
1904
信息发送单元
2002
数据删除单元

图 21

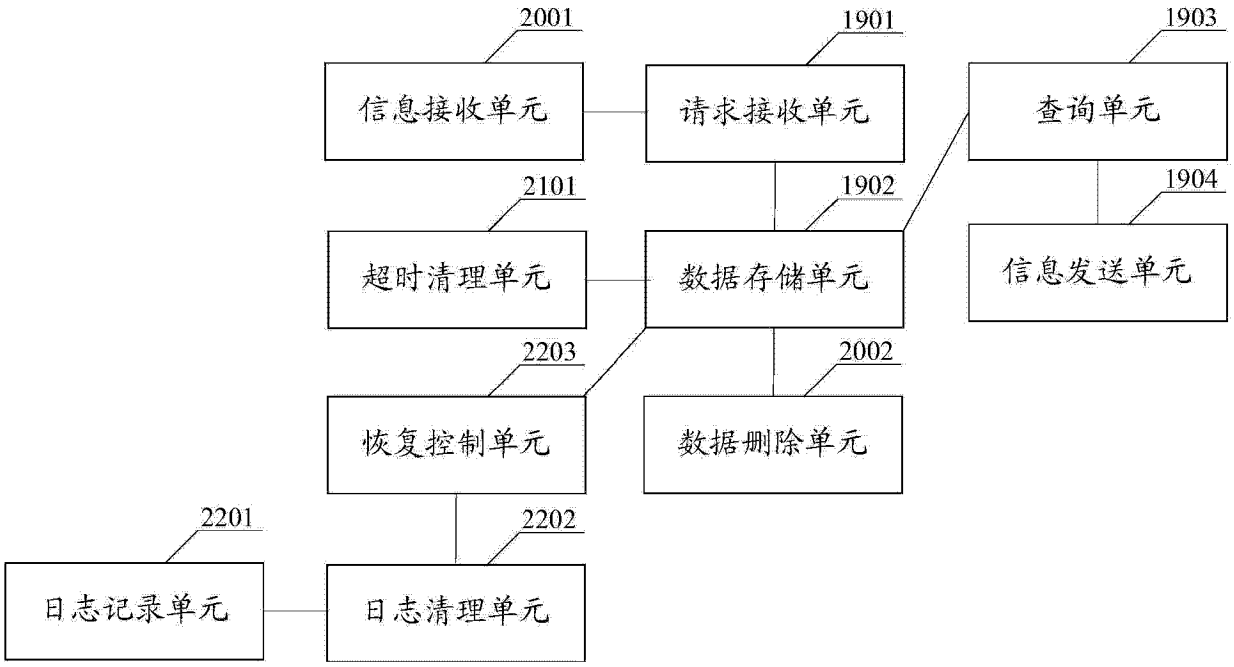

2001
信息接收单元
1901
请求接收单元
1903
查询单元
2101
超时清理单元
1902
数据存储单元
1904
信息发送单元
2203
恢复控制单元
2002
数据删除单元
2201
日志记录单元
2202
日志清理单元

图 22

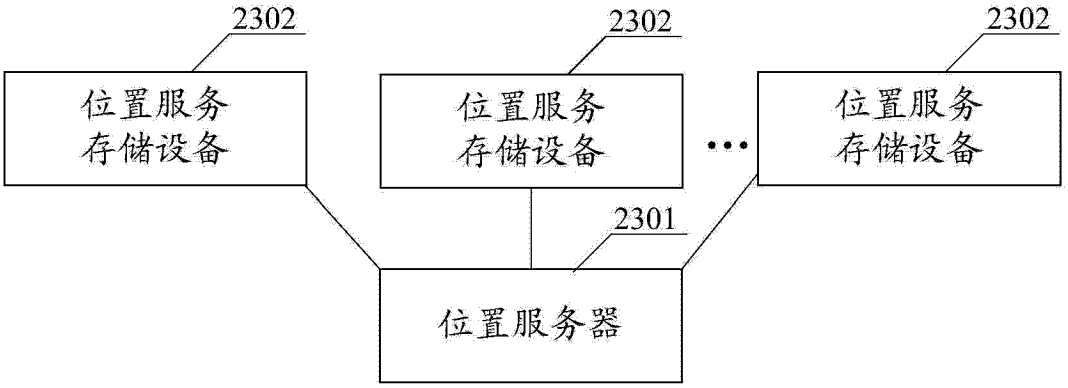

2302
位置服务
存储设备
2302
位置服务
存储设备
2302
位置服务
存储设备
2301
位置服务器

图 23

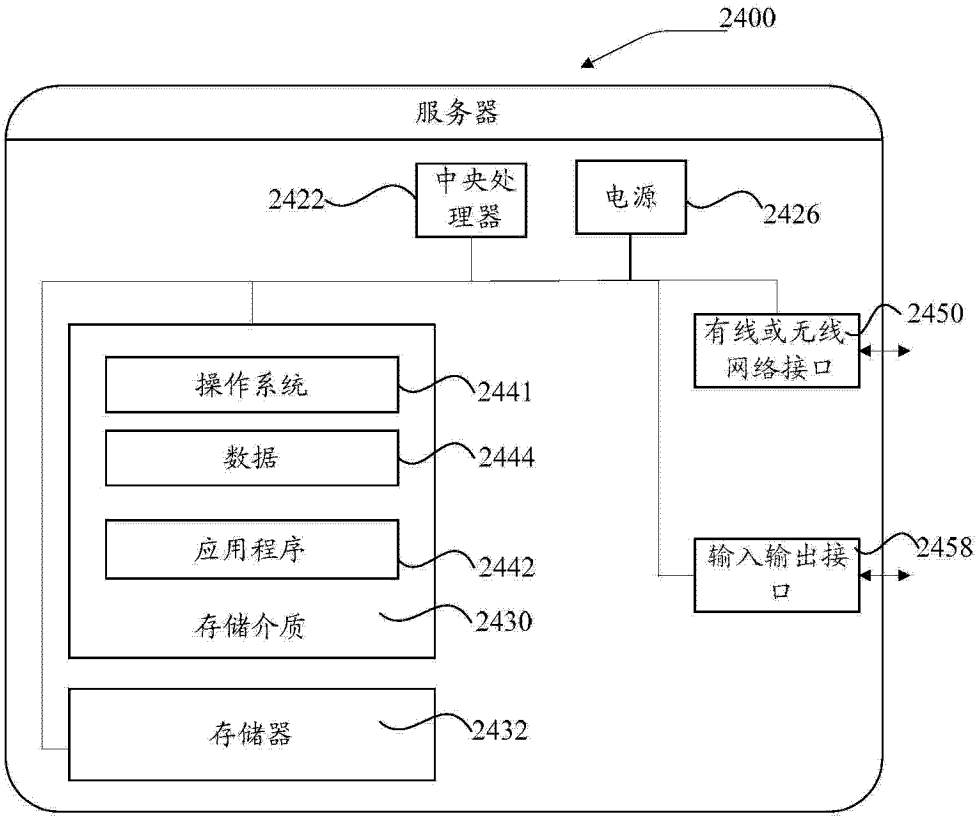

2400
服务器
2422
中央处理器
电源
2426
有线或无线网络接口
2450
操作系统
2441
数据
2444
应用程序
2442
存储介质
2430
输入输出接口
2458
存储器
2432

图 24