基于端和云的大规模上下文管理框架的研究与实现

史殿习 吴振东 丁 博
(国防科学技术大学计算机学院 长沙 410073)

摘 要 上下文态势是将大规模、广地域范围内的上下文信息综合在一起形成的一种全局信息。随着各类具备感知能力的移动终端的普及,如何获取这种全局态势并利用态势来为用户提供更好的服务是亟待解决的问题。基于"端+云"相结合的计算模式,提出移动终端的统一抽象模型来实现上下文信息收集,进而提出了在云端对大规模上下文信息进行聚合、基于MapReduce计算模型的态势信息获取算法。通过一个大规模上下文管理框架对研究内容进行验证,并以一个交通态势实例验证了框架的有效性。

关键词 上下文态势,软件抽象模型,聚合,MapReduce,基于规则
中图法分类号 TP311 **文献标识码** A

Research and Implementation of Large-scale Context Management Framework Based on Terminal and Cloud

SHI Dian-xi WU Zhen-dong DING Bo
(School of Computer Science, National University of Defense Technology, Changsha 410073, China)

Abstract The context situation refers the global view extracted from massive context information collected from a wide area. Along with the popularity of various mobile terminals with the ability to sense its context, it is a great problem to get this kind of situation and provide better service based on what we get. On the basis of the "terminal+cloud" computing paradigm, this paper proposed a unified abstract model for mobile terminals to realize context collections. And then, we proposed an algorithm to realize the aggregation of massive context information on the cloud side, which is based on the MapReduce computing pattern. We validated the research content of this paper by a large-scale context management framework as well as a traffic situation application based on this framework.

Keywords Context situation, Software abstract model, Aggregation, MapReduce, Rule-based

1 引言

近年来,随着内嵌传感器的各类终端设备(如智能手机)的迅速普及,上下文感知[1]的成本大幅降低,上下文敏感的各类新型应用层出不穷。用户行为识别[2]就是典型的案例,其应用场景之一是根据手机的加速度传感器信息来判断用户是在步行、跑步或者静止状态,并结合GPS信息判断用户是在骑车还是在坐车,据此为用户提供相应的计算服务。其它典型示例包括基于物联网技术的智能医疗应用、位置敏感的各类服务等。

上下文采集的广度和深度正在不断扩展,与此同时,网络接入手段的发展使得环境信息可以被实时共享。在这一背景下,收集和汇聚大量智能终端收集的环境信息,在更大尺度上得到真实世界状态、获取真实世界的变化态势,这一构想已经成为了可能[3]。当智能终端的数量变得非常大时,将它们收集到的上下文信息汇聚在一起,可以使我们知道社会的各个角落是什么样的情况、正在发生什么事情之类的信息,可以帮助我们从全局的角度去观察真实世界的发展态势。例如,在应急救灾系统中,通过在救灾人员所携带的终端内嵌入特定的传感器来收集和汇聚大量终端的上下文信息,有可能让我们实时、准确、全面地了解灾难的发展态势。

然而,大规模上下文信息的管理面临许多挑战,包括海量上下文如何存储、上下文数据如何大规模实时汇聚等。在传统的普适计算、上下文敏感计算等领域,受限于应用背景,相关研究和实践大多局限于单个结点(如智能手机)或者小规模系统(如智能教室)。要实现大规模上下文信息管理,亟待新的架构、框架和技术突破。

计算模式的发展为应对挑战带来了机遇。近年来,在各类终端设备迅速普及的同时,云计算也取得了一系列的成功商业实践。"云"与"端"具有明显的互补性,二者结合在一起,相互弥补缺陷、扬长避短,催生了"云+端"这一新型计算模式——由终端设备对外提供服务,由云设施在后台提供资源和计算能力支持。本文即以此为背景,针对如何利用"端"来大规模地收集上下文信息、如何利用"云"来对上下文进行有效管理等一系列问题展开研究,提出了基于"云+端"模式的通用上下文态势信息获取机制,以及基于Map/Reduce[4]的

到稿日期:2012-08-09 返修日期:2013-01-08 本文受国家核高基重大专项课题(2011ZX03002-004-01)资助。
史殿习(1966-),男,博士,研究员,主要研究方向为分布计算、普适计算、移动云计算,E-mail:dxshi@nudt.edu.cn;吴振东(1986-),男,博士生,主要研究方向为移动云计算、分布计算;丁 博(1978-),男,博士,助理研究员,主要研究方向为分布计算、软件适应。

大规模上下文汇聚算法，在此基础上，设计并实现了可重用的大规模上下文管理框架。

本文第 2 节将对相关工作进行分析，并指出现有工作的不足；第 3 节将给出面向大规模异构上下文信息获取的统一抽象模型；第 4 节聚焦于如何基于所获取的上下文信息，通过云端上下文聚合得到全局上下文态势；第 5 节描述大规模上下文管理框架的整体实现；最后给出基于该框架的交通态势感知应用及相关实验结果。

2 相关工作

上下文态势是指大量终端的上下文信息综合在一起形成的一种全局、高层次的上下文信息。在上下文态势感知的现有研究中，目前已经有一些实验性质的项目，例如 Dartmouth 大学的 CenceMe[5]，加州大学洛杉矶分校研究的 PEIR[6] (Personal Environmental Impact Report)，麻省理工大学的 VTrack[7]，Nokia、NAVTEQ、加州大学伯克利分校联合研究的 Mobile Millennium[8] 等。

Dartmouth 大学的 CenceMe 项目旨在通过用户手机上的位置信息来提高社会网络的服务质量，使得用户随时能与好友分享自己的动向，在 Twitter、Facebook 和 MySpace 上得到广泛应用，但其中需要处理的位置信息数据规模巨大，同时需要强大的计算能力支持，如 Twitter 中采用 Storm 集群[9] 来对大规模的实时位置信息进行处理分析，Storm 是一个类似于 Hadoop 的即时数据处理工具，主要用于实时处理新数据、连续查询并将结果及时反馈给客户端、并行处理大规模的数据等。

加州大学洛杉矶分校研究的 PEIR 项目旨在通过用户手机上的 GPS 传感器获取到用户所处的位置信息，然后结合城市中各个地方的空气质量情况，来告诉用户周围的环境质量。同时还可以通过用户一天中所处的不同位置，来判断用户一天的行动情况，进一步推算用户这一天的行动对环境产生了什么样的影响，另外还可以帮助相关部门决策哪些地方应该加强环境保护措施。该项目中所处理的上下文信息从整个城市收集得到，最多能支持 20 亿台手机，其数据规模庞大，在后台采用云计算的技术来为用户提供服务。

MIT 研究的 VTrack 项目和 Nokia、NAVTEQ、加州大学伯克利分校联合研究的 Mobile Millennium 项目都是针对实时交通路况信息的研究，其通过司机手机上的 GPS、听筒、加速度等传感器可以获得城市中各条道路上车辆的行驶情况，从而帮助司机了解其周围各个路段的交通路况，以指导司机如何快速地到达目的地。在这两个项目中所处理的上下文信息来自整个城市中各条道路上的所有司机的手机，有些上下文信息可能由于 GPS 定位在建筑物的影响下会存在误差需要修正，处理如此大规模的数据，而且要实时地向用户反馈当前的路况信息，需要非常强大的计算处理能力，因此后台都采用了云计算来为用户提供服务。

虽然上述项目在技术方面取得了一系列突破，但目前研究都没有提出一个有效的中间件来完成大规模上下文信息的感知与聚合过程，只是停留在具体的应用上面，不同的应用之间几乎没有可重用的部分。本文针对上下文态势感知与聚合过程中存在的移动终端异构性显著、数据规模庞大等问题展开研究，提出了面向大规模上下文管理的软件抽象模型，以及具体基于 MapReduce 的上下文数据处理方法，设计并实现了一个高可重用、基于“云+端”架构的大规模上下文管理框架。

3 大规模上下文信息收集抽象模型

基于“云+端”架构的大规模上下文管理主要包括两个环节：信息收集，即如何从大量终端上获取上下文信息；态势聚合，即如何从所获取的大量上下文信息中汇聚得到所关心的真实世界状态。本节主要关注前者，后者的相关算法和实现机制将在下一节阐述。

大规模上下文信息收集的主要挑战来自于异构性：拥有上下文信息的移动终端各式各样，不同终端设备的感知数据具有不同格式、不同精度、不同访问协议、不同采样频率等，要实现对之的有效处理首先需要进行归一化处理。本文提出了软件抽象模型来实现这一目标。该模型引入软件抽象层，将各种上下文信息收集与发送的过程与具体的硬件隔离开，使得用户在使用该模型时不需要关注与传感器交互的细节，直接通过一些简单的配置即可完成上下文信息收集与发送的需求。软件抽象模型的架构如图 1 所示。

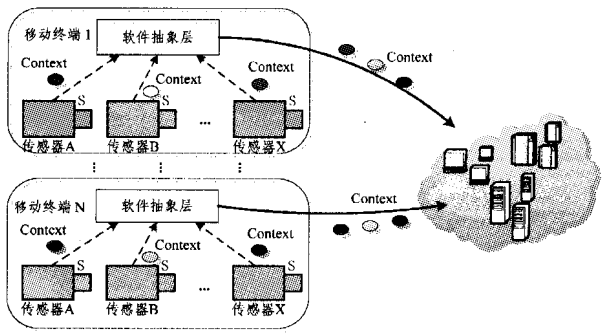

图 1 上下文态势感知的软件抽象模型示意图

以下以 Android 系统为例来说明应用上述软件抽象模型所带来的收益。Android 系统中，上下文信息主要包括位置信息、传感器信息、移动终端自身信息这 3 部分。位置信息通过移动终端内置的 GPS 模块实现；光强、磁场、温度等传感器的信息都可以通过 Android 系统的不同 API 来获取；当前移动终端 CPU、内存、硬盘、网络等使用情况的动态信息通过执行 Linux 命令，并通过解析 Linux 命令的输出结果来获取(例如执行命令/system/bin/cat /proc/stat 就可以收集到系统的 CPU 运行信息)。不同的方式导致开发人员的学习成本显著增加，也不利于上下文的统一处理。软件抽象模型的引入能够使得 Android 上下文信息的收集以一种统一的接口方式提供给用户，用户可以在不了解 Android 系统 API 的情况下只通过简单的配置即可实现相关上下文信息的收集。

除了软件抽象模型之外，移动终端还需要描述、发送上下文。图 2 是大规模上下文管理框架在移动终端上如何实现的描述图。上下文信息首先被软件抽象层订阅获得，然后由上下文建模模块根据用户设置的建模方式(如键值对模型)进行建模，最后由传输模块将上下文信息传输到云端。其中与用户相关的只有 3 条虚线，是用户的工作量，即只需要订阅与应用相关的上下文信息，然后设置好上下文的建模方式，最后还需要设置好与上下文信息传输相关的信息，如传输频率、目标

地址等。使用该模型，用户在开发应用时大部分的工作都由模型完成，而且在大多数情况下都不需要写任何代码就能完成应用移动终端部分的开发。

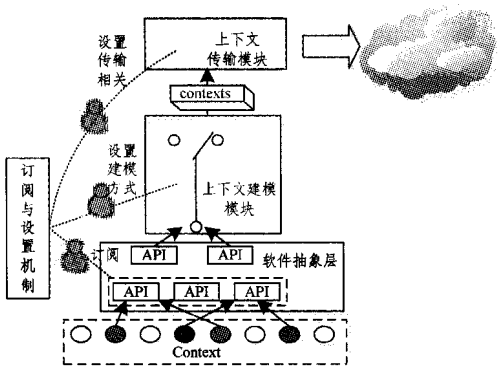

图 2　大规模上下文管理框架移动终端部分实现图

4　云端上下文态势聚合方法

大量移动终端收集的大规模上下文信息都汇集到云端之后，云端就形成了上下文态势信息，从大规模上下文信息中挖掘出对具体应用有用的信息的过程就是上下文态势聚合。文献[10]将传统的上下文聚合算法归结为基于规则和基于用例两类，其中基于规则的方式因简单、可定制、行为可预期性好等优点而使用得比较广泛，如 RCSM[11] (Reconfigurable Context-Sensitive Middleware)中的 CA-IDL 语言可以定义某个操作在若干个上下文的组合情况下被自动触发；SOCAM[12] (Service-Oriented Context-Aware Middleware)中上下文聚合由上下推理引擎完成，其规则可以由用户指定等。基于用例的方式通常采用机器学习方法，包括贝叶斯网络、人工神经网络和基于马尔可夫模型的学习等[13-15]。

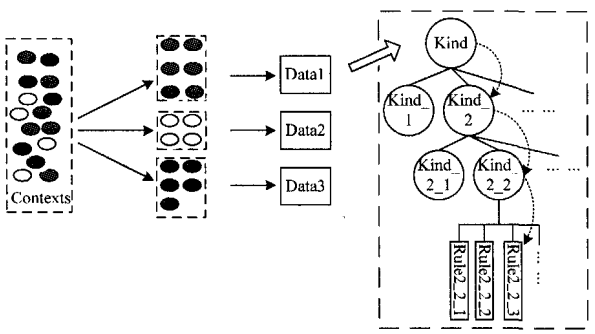

图 3　基于规则的大规模上下文聚合算法示意图

本文中根据 MapReduce 计算模型来实现大规模上下文聚合，在开源的 Hadoop 系统上实现了基于规则的大规模上下文聚合算法。云端在接收到大量的上下文信息之后，首先将大规模的上下文信息分割成几个部分，然后将每个部分都分别分发到云中的每个计算节点上进行计算，如图 3 的左半部分所示，这部分工作在 Hadoop 中已经实现，不需要再做重复的工作。当部分上下文信息被分发到某个计算节点时，该计算节点就对每一个上下文信息进行规则匹配。由于应用的需求，可能在单个计算节点上存在的规则很多，如果每一个上下文都按顺序对每一个规则进行匹配直到满足条件的话，会很消耗资源，因此将规则进行分类，能够大大减少上下文信息按规则匹配的资源消耗。如图 3 的右半部分所示，首先将规则分成 n 个大类，然后将每个大类分成 m 个小类，依次类推，就形成一棵树结构，所有的非叶子节点表示的是规则的类别，所有的叶子节点表示的是具体的规则，一个上下文信息在匹配时先找到某一类别，然后对这个类别中的所有规则按顺序进行匹配。

被分割了的上下文数据块在到达某一个节点时，会在该节点上对原始上下文信息进行 Map 进行，而进行 Map 结果数据的 Reduce 过程则可能在该节点上处理，也可能被调度到其他节点上处理。上下文数据在某节点上的 Map 过程所处理的数据都是具有相同 key 的键值对，用户可以自定义函数来处理这些具有相同 key 的键值对数据，具体的处理方式取决于原始上下文数据的格式和具体的应用需求。当该节点上 Map 过程执行完之后，将计算结果传输到准备进行 Reduce 过程的节点上(如果是本地节点则不需要传输)。Reduce 过程对数据的处理可以由用户来自行定义，也可以调用模型中的算法来获取聚合的结果(模型中提供了数据分类、数据排序等算法)。如果用户需要自定义 Reduce 过程，则模型中的规则由用户根据具体的应用需求自定义，而具体基于规则来进行聚合的算法则由模型实现，用户不需要关心规则匹配的细节，关于单个节点上基于规则上下文聚合的算法描述见表 1。

表 1　单个节点基于规则来进行上下文聚合的算法描述

```
输入：规则集合 Rules={Rule}，数据集合 Datas={Data}
输出：聚合结果集合 Results={results}
步骤 1  //根据输入的规则集合构造一棵规则树结构，其叶节点为各条规则，非叶子节点为规则的种类
        RuleTree ruleTree = constructRuleTree(Rules)
步骤 2  //初始化聚合结果集合
        Results results = {}
步骤 3  //对数据集合中每个数据都用规则树来进行匹配
        For all data in Datas
          Result result = ruleTree. match(data);
          results. add(result);
步骤 4  //返回聚合结果集合，供开发人员进一步处理
Return results
```

其中，规则树与每一个上下文数据进行匹配，得到匹配的那一条规则，最后从该匹配的规则中获取最后的结果(即表 1 中 RuleTree 的 match 方法的过程)，这部分的算法描述见表 2。对每一个上下文数据都进行类似的操作，最后的 Reduce 过程将输出每个上下文的匹配结果。

表 2　规则树匹配一条上下文数据的算法描述

```
输入：上下文数据 Data data，规则树跟节点 Node rootNode
输出：上下文数据匹配结果 Result result
步骤 1  //初始化匹配节点为 null，设定临时节点 tempNode
        Node matchedNode=null，tempNode=rootNode;
步骤 2  //获取临时节点 tempNode 的所有子节点
        Tip：Set<Node> subNodes=tempNode. getSubNodes();
步骤 3  //对 subNodes 中每个节点与上下文数据进行匹配
        For all node in subNodes
          If node. isLeaf()//如果是叶节点，则可直接退出
            If node. match(data)
              matchedNode=node;
              break;
            Else//如果不是叶节点，则将跳转到 Tip
              If node. match(data)
                tempNode=node;
                goto Tip;
步骤 4  //从匹配的条规则中获取该上下文数据匹配得到的结果
        Result result=matchedNode. getResult();
Return result;
```

上下文管理框架中，云端的作用是接收移动终端的上下文信息，并存储和聚合大规模的上下文信息，然后再为应用提供上下文聚合结果服务。图 4 是上下文管理框架在云端部分的描述，即在一部分的机器集群（HttpServer）上部署上下文并发接收器，其负责接收大量移动终端传输过来的上下文信息，最后往上下文服务集群（ContextServer）中存储上下文信息。上下文服务集群中存储着大量的上下文信息，并且通过周期性地运行 MapReduce 作业对新接收到的上下文信息进行聚合，然后，将聚合结果存储到集群中。负责响应上下文聚合结果请求的机器集群在接收到应用请求之后，从 ContextServer 中获取聚合结果然后返回给请求的应用。

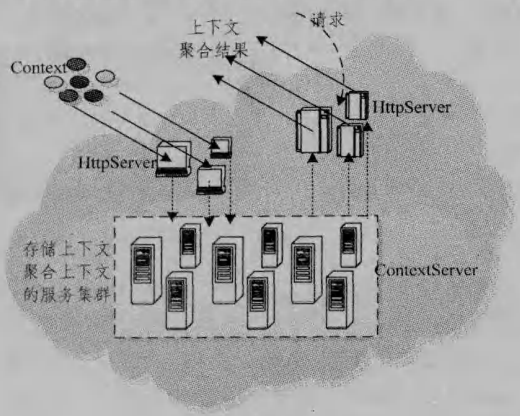

图 4　大规模上下文管理框架云端部分实现图

5　大规模上下文管理框架的实现

根据前文提出的上下文态势感知与聚合技术方案，设计并实现了一个大规模上下文管理框架，其网络结构如图 5 所示。框架工作过程如下（与图 5 中的序号相对应）。

（1）上下文信息收集过程：移动终端根据应用的需求收集上下文信息，然后将这些上下文信息发送至云端中专门负责接收上下文信息的服务器。

（2）上下文信息存储过程：接收了上下文信息的各台服务器分别将上下文信息存储到云平台中，然后对外部提供上下文存储的相关调用接口。

（3）上下文信息聚合过程：在云平台中专门负责聚合上下文信息的那部分机器集群上部署一个 Hadoop 系统，周期性地对新接收到的上下文信息进行聚合，采用基于规则的聚合算法，然后将聚合结果继续存储到云平台中，以供外部应用使用。

（4）上下文聚合结果服务请求过程：应用服务器上的某个应用需要大规模上下文聚合的一些结果信息，因此向云平台发出请求。

（5）聚合结果读取过程：云中专门负责响应上下文聚合结果请求的机器集群在收到请求时，从云平台中读取相关的上下文聚合结果。

（6）聚合结果返回给应用服务器的过程：就像响应普通的 Web 请求一样，将从云平台中读取的上下文聚合结果返回到应用服务器上。

（7）应用服务提供的过程：应用服务器上的应用结合了上下文聚合结果之后，形成一个完整的应用，并为用户提供服务。

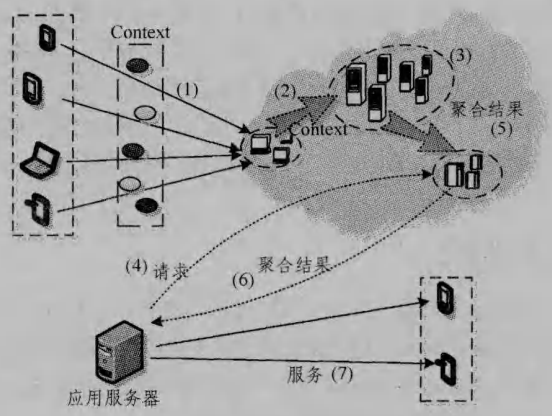

图 5　大规模上下文管理框架的网络结构示意图

从上下文管理框架的网络结构可以看出，框架分成移动终端和云端两个部分，移动终端部分主要负责上下文信息的收集和发送，而云端部分则负责上下文信息的存储和聚合。

图 6 详细地描述了框架的整体软件结构。移动终端部分中的软件抽象层是对移动终端上的所有物理传感器和移动终端自身信息的软件抽象，它向上提供接口用于获取传感器的状态和数据，以获取移动终端的硬件信息和当前的运行情况，其主要通过调用系统 API 和执行命令的方式获取信息，该模块使得具体的上下文应用逻辑与具体的传感器隔离开，以便应用程序在不同硬件平台上的移植。上下文建模模块则负责对收集到的上下文信息进行建模表示，动态地将上下文建模成标记模型或者键值对模型，并将建模好的上下文信息传递给上下文传输模块。上下文传输模块负责将上下文信息以 HTTP 协议传输到云端。模型中提供了上下文信息的订阅机制，以及对移动终端上的上下文信息收集与发送的频率进行设置的能力。同时还提供了一些扩展接口来扩展框架的功能。

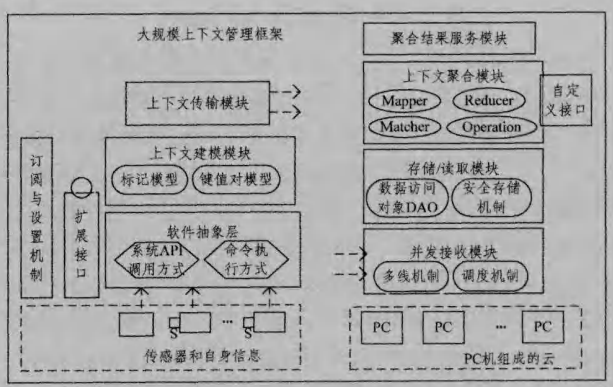

图 6　大规模上下文管理框架软件结构

云端部分中的并发接收模块负责接收所有移动终端发送过来的上下文信息，该模块具有并发接收和解析上下文信息的功能，采用了多线程的机制和线程调度的机制。上下文存储/读取模块具有并发存储/读取上下文信息到云平台中的功能，其通过数据访问对象（DAO，Data Access Object）来完成，并由安全存储机制来保证上下文被安全有效地存储。上下文

聚合模块则对接收到的所有移动终端的上下文信息进行聚合，得到具体应用程序所需要的信息，其主要有 Mapper、Reducer、Matcher、Operation 4 个部分，该模块还提供了一些自定义结构，用于帮助用户设计实现应用领域相关的一些基本需求。聚合结果服务模块将上下文聚合的结果以接口或者服务的方式提供给具体的应用访问，当具体应用请求聚合结束时，该模块就会去云中读取相应的聚合结果，并将其返回给应用。

6 实例验证

为验证大规模上下文管理框架的有效性，本文开发了一个交通态势感知的应用。该应用使得手机可以收集自身的速度和位置信息，并将信息发送至云端，云端对所有上下文信息进行聚合，得到整个城市的交通态势情况。

本文利用上下文管理框架实现了一个仿真实现，实验中采用一台 PC 机做模拟器，其负责模拟出各种上下文数据，包括车辆的 ID(等同于手机 ID)、速度、地理位置信息，然后将这些信息发送到云端。使用一台手机作为交通态势服务的接收方，云端在聚合了所有的速度和位置信息之后，会为手机提供其当前周围交通态势的服务。在云端部分基于 Hadoop 0.19 搭建了一个私有云平台，使用 5 台 DELL Inspiron580s (3.20 GHz Intel i3 4 核处理、2GB 内存)，机器之间使用 100Mbps 的以太网局域网连接；在终端部分采用 Android2.3 版本的 HTC S710。图 7 是实验的整个网络结构。

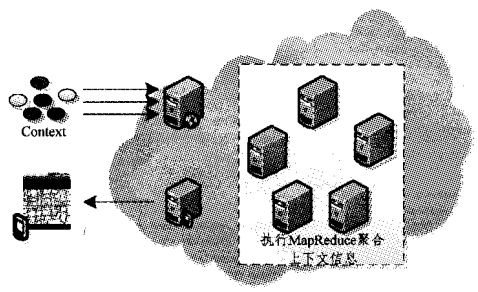

图 7 交通态势感知的实验环境图

在 MapReduce 聚合大规模上下文的过程中，大量的位置和速度信息经过 Map 和 Reduce 过程输出为整个城市各个路段的实时交通路况。位置和速度作为 Map 过程的输入，通过 Hadoop 自动分发数据的功能，将云中的大规模位置和速度数据分发到各个节点，在这些节点上各自对位置和速度数据进行计算，得到城市中各个路段的平均速度。将各个路段的平均速度作为 Reduce 过程的输入，根据开发者自定义的规则，其中各个规则可以根据路段平均速度来判断该路段的交通路况，最后输出各个路段的实时交通路况。Reduce 过程的输出也就是 MapReduce 作业的输出，将该输出结果与具体的交通态势应用(实际的地图数据)相结合，最终在手机上向用户显示出各个路段的实时交通路况。

模拟器在整个实验过程中，每隔一个周期会模拟产生出某一片区域的道路上车辆的位置和车辆的速度等信息，然后把所有车辆的这些信息组装成一个个的数据包发送至云端，如表 3 所列，每一车辆的上下文信息分别由该车辆的 ID、速度、经纬度组成。在模拟产生数据时，考虑到模拟器是一台普通的 PC 机，兼顾性能和实时性的平衡，所以只针对最大车容量为 500 辆的某一区域进行模拟。

表 3 实验中每一车辆的上下文信息表示

{"id":"00001","content":[{"name":"Speed","type":"double","value":"95.85"},{"name":"Longitude","type":"double","value":"22.80"},{"name":"Latitude","type":"double","value":"189.81"}]}
{"id":"00002","content":[{"name":"Speed","type":"double","value":"25.15"},{"name":"Longitude","type":"double","value":"23.60"},{"name":"Latitude","type":"double","value":"187.17"}]}

实验结果如图 8 所示，左图是模拟器上模拟出的车辆行驶情况，图 8 中红色点表示某一时刻车辆的位置。右图是手机上展现的该时刻交通态势服务的效果图，其中红色表示该路段拥堵，黄色表示该路段缓行，绿色表示该路段畅通。

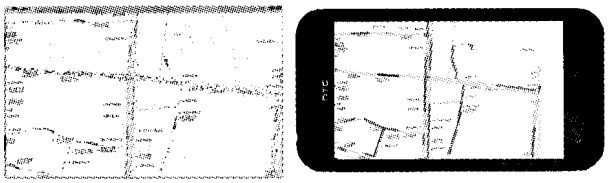

图 8 交通态势应用的实验效果图

从交通态势应用的实验结果中可以看出，该应用能够很好地利用手机终端的上下文信息，即手机的位置和速度信息，为用户提供实时交通路况服务。在该应用中，只要有足够多的手机用户参与，并收集自身的位置和速度信息，就能够充分地反映出整个城市中各个路段的实时交通路况。

实时性对于许多大规模上下文管理应用都十分重要，例如在交通态势应用中，不可能将一小时、一天前的交通路况信息都提供给用户。为了体现本文中云计算在大规模上下文聚合上的优势，我们对单台普通 PC 机和云平台进行大规模上下文聚合所需要的时间进行了比较。在测试的过程中，模拟器以离线的方式预先模拟出大量的上下文信息，然后再将这些上下文信息发送给 PC 机或云端。在测试性能时，分别针对上下文信息的数量规模为 100、500、1000、5000、10000、50000、100000、500000 时进行 10 次实验，然后在去掉最大值和最小值的情况下，取其余 8 次结果的平均值作为该上下文信息数量规模下的测试结果。测试结果如图 9 所示，其中横坐标分别对应 8 种上下文信息数量规模，纵坐标表示云端聚合或者普通 PC 聚合上下文信息的时间。

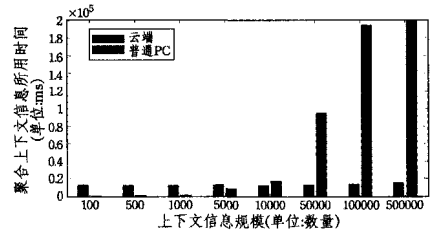

图 9 云端和普通 PC 机聚合性能测试结果比较

从图 9 可以看出，在上下文信息规模比较小的情况下，由于没有 Map 和 Reduce 过程，普通 PC 机的上下文聚合性能比云端更好，但是随着上下文信息规模的增大，当上下文达到 10000 左右时，普通 PC 机的上下文聚合性能与云端的性能几乎相同。当上下文的数量规模超过 10000 时，普通 PC 机聚合上下文的性能优势与云端之间的差距则越来越大。因此，云端在聚合大规模上下文信息上具有明显优势。

结束语 上下文信息的有效利用可以使计算服务更具人性化和智能化。传统的使用上下文信息的应用一般都是针对个体的或者少数群体的上下文信息，本文面向大规模的上下文态势展开了研究，基于端和云相结合的模式，提出了面向大规模上下文信息获取的统一抽象模型及基于 MapReduce 和规则的云端上下文聚合算法，进而设计并实现了一个大规模上下文管理框架。在未来工作中，我们将对云端大规模上下文聚合的性能，以及移动终端收集和发送上下文信息时产生的能耗问题做进一步的研究。

参 考 文 献

[1] Schilit B, Adams N, Want R. Context-Aware Computing Applications[C]// Proceedings of Workshop on Mobile Computing Systems and Applications. 1994

[2] Choudhury T, et al. The Mobile Sensing Platform: An Embedded System for Activity Recognition[J]. IEEE Pervasive Comp, 2008, 7(2): 32-41

[3] Lane N D, Miluzzo E, Lu Hong, et al. A Survey of Mobile Phone Sensing[J]. Communications Magazine, IEEE, 2010, 48(9): 140-150

[4] MapReduce[OL]. http://en. wikipedia. org/wiki/MapReduce, 2011

[5] Dartmouth College. Mobile Sensing Group[OL]. http://sensorlab. cs. dartmouth. edu/

[6] Mun M, et al. Peir, the Personal Environmental Impact Report, as a Platform for Participatory Sensing Systems Research[C]// Proc. 7th ACM MobiSys. 2009: 55-68

[7] Thiagarajan A, et al. VTrack: Accurate, Energy-Aware Traffic Delay Estimation Using Mobile Phones[C]// Proc. 7th ACM SenSys. Berkeley, CA, Nov. 2009

[8] UC Berkeley/Nokia/NAVTEQ. Mobile Millennium [OL]. http://traffic. berkeley. edu/

[9] witterStorm[OL]. http://www. infoq. com/news/2011/09/twitter-storm-real-time-hadoop, 2011

[10] Ding Bo, Wang Huai-min, Shi Dian-xi. Pervasive middleware technology[J]. 计算机科学与探索, 2007(3)

[11] Yau S, Karim F, Wang Y, et al. Reconfigurable context-sensitive middleware for pervasive computing[J]. Pervasive Computing, 2002, 1(3): 33-40

[12] Da T, Zhang Q. A middleware for building context- aware mobile services [C]// Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th, 2004, 5: 2656-2660

[13] Korpipaa P, Koskinen M, Peltola J, et al. Bayesian approach to sensor-based context awareness [J]. Personal and Ubiquitous Computing, 2003, 7(2): 113-124

[14] Schmidt A, Aidoo K A, Takaluoma A, et al. Advanced Interaction in Context [C]// Handheld and Ubiquitous Computing. Springer Berlin Heidelberg, 1999: 89-101

[15] Castro P, Munz R. Managing context data for smart spaces[J]. Personal Communications, 2000, 7(5): 44-46

(上接第 51 页)

[2] Chan H, Perrig A. Security and Privacy in Sensor Networks[J]. IEEE Comp. , 2003, 36(10): 103-105

[3] Shi E, Perrig A. Designing Secure Sensor Networks[J]. Wireless Commun, 2004, 11(6): 38-43

[4] Gura N, et al. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs [C]// CHES'04: Proc. Wksp. Cryptographic Hardware and Embedded Systems. Aug. 2004

[5] Gaubatz G, Kaps J-P, Sunar B. Public Key Cryptogra-phy in Sensor Networks-Revisited[C]// ESAS'04: 1st European Wksp. Security in Ad-Hoc and Sensor Networks. 2004

[6] Rivest R L. RFC 1321. The MD5 message digest algorithm request for comments [S]. Cambridge: MIT and RSA Data Security, 1992

[7] Nits F P. FIPS PUB 180, Secure hash standard [S]. Washington DC: U. S. Department of Commerce, 1993

[8] Wang S Y, Feng J C. Chaotic communication scheme by parameter division multiple access[J]. Acta Electronica Sinica, 2007, 35(7): 35-40

[9] 刘式达，梁福明，刘式适，等. 自然科学中的混沌和分形[M]. 北京：北京大学出版社，2003：1-12，26，31-32，125

[10] 陈帅. 无线微传感器网络混沌加密理论及其关键技术研究[D]. 重庆：重庆大学，2006：55-78

[11] Peng Fei, Qiu Shui-sheng. One-way Hash Functions Based on Iterated Chaotic Systems[C]//Communications, Circuits and Systems International Conference. 2007: 1070-1074

[12] Shannon C E. Communication theory of secrecy systems[J]. Bell System Technology Journal, 1949, 28: 656-715

[13] 郑世慧，张国艳，杨义先，等. 基于混沌的带密钥散列函数安全分析[J]. 通信学报，2011，32(5)：146-152

(上接第 62 页)

[10] Shih H-S, Shyur H-J, Lee E S. An extension of TOPSIS for group decision making[J]. Mathematical and Computer Modeling, 2007, 45(7/8): 801-813

[11] Jahanshahloo G R, Lotfi F H, Davoodi A R. Extension of TOPSIS for decision-making problems with interval data[J]. Interval efficiency, Mathematical and Computer Modeling, 2009, 49(5/6): 1137-1142

[12] Ai Li-feng, Tang Mao-lin, Fidge C. Partitioning composite Web services for decentralized execution using a genetic algorithm [J]. Future Generation Computer Systems, 2011, 27(2): 157-172

[13] Zhao Xin-chao, Song Bo-qian, Huang Pan-yu, et al. An improved discrete immune optimization algorithm based on PSO for QoS-driven webservice composition [J]. Applied Soft Computing, 2012, 12(8): 2208-2216

[14] Fei Tao, Zhao Dong-ming, Hu Ye-fa, et al. Correlation-aware resource service composition and optimal-selection in manufacturing grid[J]. European Journal of Operational Research, 2010, 201(1): 129-143

[15] Ma Yue, Zhang Cheng-wen. Quick convergence of genetic algorithm for QoS-driven web service selection[J]. Computer Networks, 2008, 52(5): 1093-1104