RFID数据流过滤算法研究

邓海生[1],李 军[2]
(1.西京学院 教务处,陕西 西安 710123;
2.西安理工大学 计算机与工程学院,陕西 西安 710048)

摘 要:RFID在实时识别、定位、跟踪以及物理对象监控等方面有着广泛的应用前景。为了实现上述应用的目标,RFID数据必须经过采集、过滤等处理过程。采集的原始RFID数据包括噪音数据和冗余数据,只有将这些噪音数据和冗余数据过滤后,才能应用;另外,保证正确的RFID标签数据采集顺序,对于一些应用也非常关键。针对这些问题,提出了基本噪音过滤算法和基于hashtable的有序噪音过滤算法,以及基本冗余过滤算法和基于hashtable的冗余过滤算法,并通过仿真实验验证了该方法的有效性。

关键词:无线射频识别技术;噪音数据;冗余数据;数据过滤
中图分类号:TP139 **文献标识码**:A **文章编号**:1673-629X(2012)06-0026-04

Research of RFID Data Filtering Algorithm

DENG Hai-sheng[1],LI Jun[2]
(1. Dean's Office of Xijing University,Xi'an 710123,China;
2. School of Computer Science & Engineering,Xi'an University of Technology,Xi'an 710048,China)

Abstract:RFID holds a wide ranges of pervasive computing applications of real-time identifying,locating,tracking and monitoring physical objects. To achieve these goals,RFID data contains false readings and duplicates has to be collected,filtered and transferred into semantic application data. Such data cannot be used directly unless they are filtered and cleaned. Meanwhile,the order perservation of RFID tag observations are critical for many applications. In the paper,propose serveral efficient methods to filter RFID data,including both noise removal and duplicate elimination. The performance study demonstrates the efficiency of the methods.

Key words:RFID;false readings;duplicates;data filtering

0 引 言

无线射频识别技术(Radio Frequency Identification,RFID)[1]是利用射频信号自动识别目标对象并获取目标对象相关信息的,鉴于外界环境、读写器、卡片等因素的影响,采集的原始RFID数据往往含有噪音数据和冗余数据[2]。噪音数据是指,除了正常的卡片数据外,读写器产生的额外的、异常的数据,它产生的原因有以下几种:

1)标签的位置超出了读写器正常读写的范围;

2)读写器或环境的不明因素。

冗余数据指的是正常的卡片数据被多次采集,它产生的原因有以下几种:

1)标签长时间(多个读写周期)处于读写器的读写范围,以致于标签被多次读取;

2)标签处于多个读写器读写范围重叠的部分,以致于标签被多个读写器重复读取;

3)为了提高读取的准确性,含有相同EPC的标签被粘贴在同一个物理对象上,使得标识该对象的数据被多次读取。

因此,在将采集的RFID数据转换成可以直接应用的数据之前,首先要对原始的RFID数据进行过滤。另外,过滤后的RFID数据需要保持原来的采集顺序(第一个标签的数据,过滤后要最先输出),例如,在医疗、监控等应用环境中,RFID数据的输出顺序要严格遵循标签的采集顺序。

RFID数据处理既包括事件处理,还包括数据流处理。在文献[3]的事件处理方法中,忽视了处理效率和内存管理的问题,在文献[4]的数据流处理方法中,没有涉及到噪音数据和冗余数据的问题,在文献[5,6]的RFID中间件中虽然提供了RFID数据的过滤功能,但对于大量的实时RFID数据仍需要更有效的过滤算法。文中针对这些问题,提出了几种RFID数据过滤算法,并通过仿真实验验证了方法的有效性。

收稿日期:2011-11-05;**修回日期**:2012-02-11
基金项目:西安科技局信息技术专项(ZX06030)
作者简介:邓海生(1980-),男,讲师,硕士,研究方向为RFID中间件;李 军,教授,博士,研究方向为RFID技术与移动商务。

1　噪音数据过滤

在实际中,为了提高对标签数据的识别率,读写器往往要对标签多次读取,这样可以显著减少噪音数据;同时,噪音数据和正常 RFID 数据相比,产生的机率要低许多。因此,在多个读取周期内,那些多次被重复的数据才被认为是正常的 RFID 数据。当然,这种多次读取的方法也会带来更多的冗余数据[7]。

基于上述分析,通过滑动窗口来解决噪音数据的问题。滑动窗口[8,9]是指随着时间滑动的窗口,假设窗口的大小为 window_size,那么这个滑动窗口可以表示为[t,t +window_size]。经过时间 $t1$ 后,滑动窗口变为[$t+t1$,$t+t1$+window_size]。采集的标签数据进入滑动窗口,并随着时间消亡。噪音数据就是在滑动窗口内,重复的次数小于 threshold 的数据,这里参数 threshold 表示自定义阈值,是噪音数据检测的依据;正常的 RFID 数据就是同一个 EPC 值出现的次数不小于 threshold 的数据。对于噪音数据要采取一定的算法进行过滤;而过滤后的正常的 RFID 数据需要进行冗余处理,即将重复读取的 RFID 数据合并成一条数据。

采集的原始 RFID 数据可以表示成以下的形式:(reader_id,tag_id,timestamp)[10]。其中 reader_id 表示读写器的 EPC,tag_id 表示目标对象的 EPC,timestamp 表示读写器读取数据的时间戳,(reader_id,tag_id)表示 RFID 数据的主键。

1.1　基本噪音过滤算法

对于进入滑动窗口的 RFID 数据 R(用 R 表示任一 RFID 数据),扫描窗口中的每条数据。假如 R 出现的次数大于或等于阈值 threshold,那么 R 就是正常的 RFID 数据,而非噪音数据,因此输出每条 R。并且在输出 R 的时候,设置该 R 的输出状态 state-output 为 true,避免重复输出。方法描述如下:

```
//FIFO 队列用来存放滑动窗口中 RFID 数据
1windowbuffer←empty FIFO queue
2 loop
//变量 INCOMING 临时保存进入滑动窗口的 RFID 数据
3INCOMING←next reading
//RFID 数据进入滑动窗口
4 append INCOMING to WINDOWBUFFER
5 EXPIRETIME←INCOMING. timestamp-window_size
//6,7,8 的作用是去除过期的 RFID 数据
6 while the head of WINDOWBUFFER is older than EXPIRETIME
7remove the head of WINDOWBUFFER
8end while
//变量 COUNT 记录队尾数据在 window_size 这段时间重复的个数
9 COUNT←count of readings in WINDOWBUFFER whose key equals to INCOMING. key
//10~17,对于任何 RFID 数据 R,如果 windowbuffer 中有 threshold 或以上个,认为 R 为有效 RFID 数据,输出每个 R
10f COUNT ≧ threshold then
11for each R in WINDOWBUFFER whose key equals to INCOMING. key
12if R has not been output then
13output R
14state-output = true
15end if
16end for
17end if
18end loop
```

假设在滑动窗口中平均有 n 条数据,k 个主键。上述算法的时间花费取决于四个部分:插入 RFID 数据、去除过期的 RFID 数据、计算变量 COUNT 的大小、输出有效的 RFID 数据和设置 state-output,时间复杂度分别为 $O(1)$,$O(1)$,$\Theta(n)$, $O(n)$。因此总的时间复杂度为:$O(1)+O(1)+\Theta(n)+O(n)=\Theta(n)$;空间复杂度取决于窗口的大小,因此空间复杂度为 $\Theta(n)$[11]。

1.2　基于 hashtable 的有序噪音过滤算法

基本过滤算法比较简单,优点是输出比较及时。但这种方法一经判断出是正常 RFID 数据,就直接输出,可能会造成输出顺序和读取顺序不一致的问题。例如,同个主机上的读写器 A 和 B 分别读取标签 A 和 B,每个标签的读取周期为 100msec,以读取 10 次为例,那么滑动窗口的大小 window_size 为 1000msec。假设自定义阈值为 6,即 1000msec 的时间内,窗口中有 6 个或 6 个以上相同的 RFID 数据,这个 RFID 数据就是正常的数据,否则为噪音数据,还假设卡片 A 的关键字为 1,卡片 B 的关键字为 2,表 1 描述了一种可能的采集情况。其中卡片 A 的"5"和"4",卡片 B 的"5"在这段时间内均为噪音数据,不输出。

表 1　采集情况

Timestamp	Tag A	Tag B
100	1	
200	5	2
300	1	2
400	1	2
500	4	2
600	1	2
700	1	2 *
800	1 *	5
900	1	2
1000	1	2
1100		2

不难发现:在 700msec 的时候,发现标签 B 的"2"

不是噪音数据,于是输出序列:(200,2),(300,2),(400,2),(500,2),(600,2),(700,2),(200,2),然后在800msec的时候,发现标签A的"1"不是噪音数据,于是输出序列:(100,1),(300,1),(400,1),(600,1),(700,1),(800,1)。可见,尽管先采集标签A,但先输出的却是标签B的数据,因此采集顺序和输出顺序不一致的情况出现了。

基于hashtable的有序噪音过滤算法则通过在数据消亡的时刻才输出有效的RFID数据,从而保证采集和输出顺序的一致。采用这种方法会在1100msec时输出标签A的有效数据(100,1),1200msec时输出(200,2),依次类推。方法描述如下:

```
1 WINDOWBUFFER←empty FIFO queue
//hashtable起到计数器的作用
2 TABLE←empty hashtable
3loop
4INCOMING←next reading
5make INCOMING as noise
6 append INCOMING to WINDOWBUFFER
//7~11,通过哈稀表记录窗口中键值为INCOMING.key的个数
7if TABLE[INCOMING.key] does not exist
8TABLE[INCOMING.key].value=1
9else
10TABLE[INCOMING.key].value = TABLE[INCOMING.key].value+1
11end if
12EXPIRETIME←INCOMING.timestamp-window_size
//13~20,在消亡的时刻输出非噪音数据
13 while the head of WINDOWBUFFER is older than EXPIRETIME
14if the head of WINDOWBUFFER is marked as non-noise
15output the head of WINDOWBUFFER
16end if
17remove the head ofWINDOWBUFFER
18TABLE[INCOMING.key].value = TABLE[INCOMING.key].value-1
19if TABLE[INCOIMG.key].value==0 then remove the slot from the hashtable
19end while
20COUNT←TABLE[INCOMING.key].value
//22~28标识非噪音数据
21if COUNT ≧ threshold then
22for each reading R in WINDOWBUFFER with key equals to INCOMING.key
23if R is marked as noise
24mark R as non-noise
25end if
26end for
27end if
28 end loop
```

由于引入了hashtable,因此和基本噪音过滤算法相比,计算COUNT的复杂度由$\Theta(n)$降低为$O(1)$,总的时间复杂度为:$O(1)+O(1)+O(1)+O(n)=O(n)$;空间复杂度增加了$\Theta(k)$($\Theta(k)$为hashtable的空间复杂度)。

2 冗余过滤算法

当噪音数据被过滤后,同一个标签的有效数据被多次输出,产生了冗余,需要过滤掉冗余数据而只保留第一个被输出的标签数据[12,13]。在冗余过滤算法中,引入参数max_distance,如果同一个键值的数据在max_distance时间段内,出现两次或以上,则只保留第一个数据,其余作为冗余过滤掉;如果两个同一键值的数据间隔超过max_distance,则认为是不同的RFID数据。

2.1 基本冗余过滤算法

令max_ distance等于window_size(滑动窗口的大小),基本冗余过滤算法描述如下:

```
1 WINDOWBUFFER←empty FIFO queue
2 loop
3INCOMING←the next reading
4EXPIRETIME←INCOMING.timestamp-max_size
5while the head of WINDOWBUFFER is older than EXPIRETIME
6remove the head of WINDOWBUFFER
7end while
8 search for another reading with the same key as INCOMING
9if nothing is found
10output INCOMING
11 end if
12 append INCOMING to the end of WINDOWBUFFER
13end loop
```

假设在max_ distance内,平均有n条数据,k个主键。上述算法的时间复杂度取决于步骤8,不难发现其复杂度为$\Theta(n)$;空间复杂度取决于窗口的大小,因此空间复杂度为$\Theta(n)$。

2.2 基于hashtable的冗余过滤算法

通过对基本冗余过滤算法的分析,不难发现,为了判断新来的数据是不是冗余数据,设置大小为max_distance的滑动窗口不是必要的。分析如下:可以保存最近的同一个键值数据的时间戳,然后计算新到数据和它的时间间隔,如果时间间隔大于max_ distance,即认为新到数据是不同的数据,否则认为是冗余数据。因此,可以通过hashtable保存不同键值数据的时间戳,方法描述如下:

```
1TABLE←empty hashtable
2loop
```

3INCOMING←the next reading

4if INCOMING. timestamp - TABLE[INCOMING. key] > max_distance

5output INCOMING

6end if

7 update TABLE[INCOMING. key] to be INCOMING. timestamp

8end loop

上述方法和基本冗余数据过滤算法相比,因为引入了 hashtable,时间复杂度由 $\Theta(n)$ 变为 $O(1)$,空间复杂度由 $\Theta(n)$ 变为 $\Theta(k)$。

3　仿真实验

在仿真实验一中,针对基本噪音过滤算法和基于 hashtable 的噪音过滤算法,研究它们在不同的标签到达率时的输出延迟情况。参数如下:每个标签被读 10 次,读取的时间间隔为 200msec,5% 的噪音率,平均标签到达率为 1 个/sec,5 个/sec,50 个/sec 和 500 个/sec(由于每个标签被读 10 次,因此总的数据到达率为 10/sec,50/sec,500/sec,5000/sec)。仿真的结果如图 1 所示:

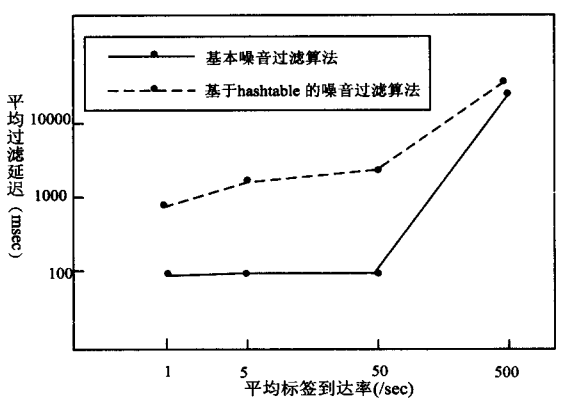

图 1　噪音过滤算法仿真一

通过图 1 可见,当标签平均到达率较低时,由于基本噪音过滤算法忽略了输出顺序,和基于 hashtable 的噪音过滤算法相比,输出比较及时。当标签平均到达率较高时,由于基于 hashtable 的噪音过滤算法引入了哈稀表这种数据结构,缩短了数据扫描时间,和基本噪音算法相比,具有较低的输出延迟。

在仿真实验二中,针对基本噪音过滤算法和基于 hashtable 的噪音过滤算法,研究它们在不同的噪音比率时的输出延迟情况。参数如下:每个标签被读 10 次,读取的时间间隔为 200msec,平均标签到达率为 1 个/sec,噪音比率分别为 1%,5%,20% 和 50%。仿真的结果如图 2 所示。

通过图 2 可见,无论噪音比率低或高,基本噪音过滤算法和基于 hashtable 的噪音过滤算法相比,总能保证较低的输出延迟。并且随着噪音的增加,基本噪音过滤算法能显著降低输出延迟。

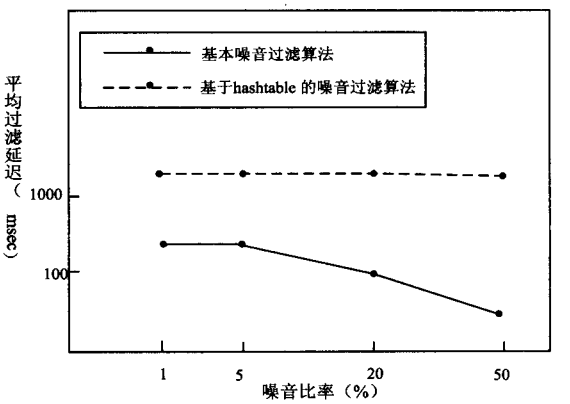

图 2　噪音过滤算法仿真二

在仿真实验三中,针对基本冗余过滤算法和基于 hashtable 的冗余过滤算法,研究它们在不同的标签到达率时的输出延迟情况。参数如下:每个标签被读 10 次,读取的时间间隔为 200msec,0% 的噪音率(由于噪音已经被消除),平均标签到达率为 10 个/sec,50 个/sec,250 个/sec 和 1000 个/sec(由于每个标签被读 10 次,因此总的数据到达率为 100/sec,500/sec,2500/sec,10000/sec)。仿真的结果如图 3 所示:

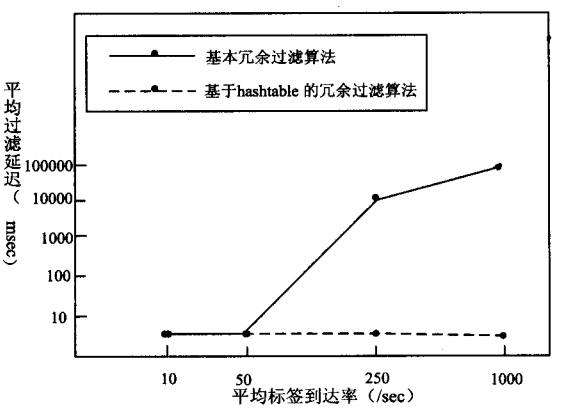

图 3　冗余过滤算法仿真

两种算法都能有效地消除冗余数据。然而,基于 hashtable 的冗余过滤算法,在输出延迟的控制方面,有着十分明显的优势。基于 hashtable 的冗余过滤算法在标签到达率 1000 个/sec 的情况下还能保证较低的延迟,而基本冗余过滤算法在标签到达率 50% 以上时,输出延迟基本成线性增大。

4　结束语

文中分析了 RFID 数据流可能会产生噪音和冗余数据这一问题,并提出了几种有效的解决方法。特别地,针对噪音数据问题,提出了一种有效的、保持输出和观测顺序一致的噪音过滤方法;针对冗余数据问题,

(下转第 34 页)

集成到多核处理器里，使其完成多核互连的功能，将很好地解决现在多核互连在带宽、时延、能耗和可靠性等方面的问题，也必将带动多核处理器进入一个崭新的时代。

4 结束语

多核处理器采用的互连方式，必须根据其实现所需核的数目和互连所带来的延迟、功耗、面积、实现工艺的复杂度等多方面进行考虑。在传统的技术和工艺框架下少于 10 核的多核处理器多采用总线的互连方式，大于 10 核的多核处理器适合采用交叉开关的互连方式，大于 36 核的结构比较适合采用片上网络结构。

随着新材料和新方法的应用及制造技术和工艺的不断进步，一些高效的互连技术已经实现，而随着片上多核处理器核的数目和结构不断的发展，还需要继续探索新型的多处理器核间互连通信技术。光互连技术的提出和各种集成光电子器件及石墨烯材料的应用为多核互连技术的发展提供了广阔的前景，一个高速、低功耗、高性能的多核处理器时代即将来临。

参考文献：

[1] Shen J P, Lipasti M. Modern Processor Design[M]. 北京：清华大学出版社，2007.

[2] Patterson D A, Hennessy J L. 计算机组成与设计[M]. 郑伟民，译. 北京：机械工业出版社，2007.

[3] 黄国睿，张 平，魏广博. 多核处理器的关键技术及其发展趋势[J]. 计算机工程与设计，2009，30(10)：2414-2418.

[4] 王 炜，汤志忠，乔 林. 片上多处理器互连技术综述[J]. 计算机科学，2008，35(9)：7-8.

[5] 卜 凡，赵忠民. 64 位多核 CPU 中交叉开关总线的设计与实现[J]. 计算机与数字工程，2008，36(11)：151-154.

[6] Zhang Y P, Jeong T. A Study of the On-chip Interconnection Network for the IBM Cyclops64 Multi - core Architecture [C]//Proceedings of 20th IEEE International Parallel Distributed Processing Symposium. [s. l.]: [s. n.], 2006: 1-10.

[7] Culler D E, Singh J P, Gupta A. 并行计算机体系结构[M]. 第 2 版. 李晓明，译. 北京：机械工业出版社，2002.

[8] Ravankar A A, Sedukhin S G. "Mesh-of-Tori": A Novel Interconnection Network for Frontal Plane Cellular Processors [C]//Proceedings of IEEE First International Conference on Networking and Computing. [s. l.]: [s. n.], 2010: 281-284.

[9] 胡晨骏，王晓蔚. 基于多核集群系统的并行编程模型的研究[J]. 计算机技术与发展，2008，18(4)：70-73.

[10] 王立炜. 片上网络架构下多核处理器系统的设计[D]. 太原：太原理工大学，2010.

[11] 乔保军，石 峰，计卫星. 多核处理器核间互连的新型互连网络[J]. 北京理工大学学报，2007，27(6)：511-516.

[12] Haroon-Ur-Rashid Khan, Shi Feng, Jia Xinli. Performance of Triplet Based Interconnection Strategy for Multi-core On-chip Processors[C]//IEEE International Conference on High Performance Computing and Communications. [s. l.]: [s. n.], 2009: 163-170.

[13] 李 慧，顾华玺. 多核之间光互连技术的研究[J]. 中国集成电路，2010，19(2)：50-55.

(上接第 29 页)

提出了一种通过 hashtable 保存历史数据的冗余数据过滤方法，大大减少了内存的需要。通过模拟 RFID 数据流进行仿真实验，验证了方法的有效性。文中的思想和方法对于构建 RFID 中间件(数据过滤是 RFID 中间件的重要组成部分)具有一定的参考价值。在噪音数据过滤方法中，如何更有效地提高 RFID 有效数据输出效率是下一步需要研究的问题。

参考文献：

[1] 游战清，李苏剑. 无线射频识别技术(RFID)理论与应用[M]. 北京：电子工业出版社，2004.

[2] Brusey J. Reasoning about Uncertainty in Location Identification with RFID[C]//RUR at IJCAI. [s. l.]: [s. n.], 2003.

[3] 郝忠孝. 主动数据库系统理论基础[M]. 北京：科学出版社，2009.

[4] Madden S. Continuously adaptive continuous queries over stream[C]//SIGMOD. [s. l.]: [s. n.], 2002.

[5] Oracle Sensor Edge Server[EB/OL]. 2008. http://www.oracle.com/technology/products/iaswe/edge_server.

[6] Sybase RFID Solution[EB/OL]. 2005. http://www.sybase.com/rfid.

[7] 张丰贵，程良伦. 基于 KDB 树的 RFID 事件聚合过滤算法[J]. 计算机工程，2009，35(21)：82-84.

[8] 马 岩，张延园，尹方鸣. 基于滑动窗口的 RFID 数据流多标签清洗算法[J]. 科学技术与工程，2009，9(5)：1165-1171.

[9] 阴晓佳，鞠时光. 基于复杂事件处理机制的 RFID 数据流处理方法[J]. 计算机应用，2009，29(10)：2786-2789.

[10] Hahn K. Adaptive workflow management to ensure transactional service composition[J]. Digital Information Management (ICDIM), 2010, 12(2): 373-378.

[11] Cheng Nan, Song Meina, Wang Qian. A web service process transaction framework based on compensation and proxy[J]. Pervasive Computing (JCPC), 2010, 24(8): 369-372.

[12] Vidyasankar K, Gottfried V. Multi - level Modeling of Web Service Compositions with Transactional Properties[J]. Journal of database management, 2011, 22(2): 1-31.

[13] Haddad J E. TQoS: Transactional and QoS-aware selection algorithm for automatic Web service composition [J]. IEEE Transaction on Service Computing, 2010, 14(8): 210-218.