ISSN 1673-9418 CODEN JKYTA8
Journal of Frontiers of Computer Science and Technology
1673-9418/2016/10(11)-1512-12
doi: 10.3778/j.issn.1673-9418.1509013

E-mail: fcst@vip.163.com
http://www.ceaj.org
Tel: +86-10-89056056

面向流式处理系统的对象管理方法*

王家兴[1+],林学练[1],申 阳[1],张 韵[2],张明明[1],马 帅[1]
1. 北京航空航天大学 计算机学院,北京 100191
2. 上海通用识别技术研究所,上海 201100

Data Object Managing Method in Stream Processing System*

WANG Jiaxing[1+], LIN Xuelian[1], SHEN Yang[1], ZHANG Yun[2], ZHANG Mingming[1], MA Shuai[1]
1. School of Computer Science and Engineering, Beihang University, Beijing 100191, China
2. Shanghai General Recognition Technology Research Institute, Shanghai 201100, China
+ Corresponding author: E-mail: wangjx@act.buaa.edu.cn

WANG Jiaxing, LIN Xuelian, SHEN Yang, et al. Data object managing method in stream processing system. Journal of Frontiers of Computer Science and Technology, 2016, 10(11): 1512-1523.

Abstract: In the practice of Internet of vehicles, people run several computing applications in streaming systems to analyze GPS/OBD data collected from vehicles. These applications have some common requirements and features, which are long-cycle running, low processing delay, and the need for keeping states in memory. However, after a long time running, such kind of streaming job needs to keep a lot of computation parameters, status and other data in memory, and large numbers of data objects are not active among them. If let them occupy memory, it will cause a great waste of system resource. This paper proposes a data object managing method for stream tasks, and hopes to optimize the memory use of streaming system to solve the according problem. This paper establishes lifecycle model for streaming data object (SDO), and uses application-driven, data-driven method to achieve appropriate expire-parameters for SDO. Finally, this paper tests the proposed method in applications of Internet of vehicles. Experiments show that the proposed method can effectively reduce inactive SDO number and improve resource utilization while ensuring the processing delay can be accepted by users.

Key words: Internet of vehicles; streaming system; inactive object; lifecycle management; data-driven model

* The National Natural Science Foundation of China under Grant No. 91118008 (国家自然科学基金).
Received 2015-08, Accepted 2015-10.
CNKI网络优先出版: 2015-10-30, http://www.cnki.net/kcms/detail/11.5602.TP.20151030.1620.006.html

摘　要：在车联网的应用实践中，人们将分析车辆数据的任务运行在流式计算系统中。在运行分析中发现，这些任务具有运行周期长，处理延迟低，任务有状态等需求和特点，并且计算过程中需要在内存中保存大量计算参数和中间状态等数据对象，其中大量的数据对象并不活跃，任由其占用内存造成了系统资源的浪费。针对该问题开展研究，给出了流式任务的数据对象管理方法，优化了内存的使用。为流式数据对象建立生命周期模型，采用应用驱动、数据驱动的模型参数确定方法为流式数据对象设置合适的过期参数，设计车联网测试用例，验证该生命周期管理方法的有效性。实验结果表明，该方法在用户可接受的处理延迟范围内，能够有效地减少流式系统中不活跃对象的数目，达到了优化内存，降低资源开销的目的。

关键词：车联网；流式系统；不活跃对象；生命周期管理；数据驱动模型

文献标志码：A　　**中图分类号：**TP391

1　引言

车联网是综合电子信息技术，它将每一辆车作为一个信息源，利用无线通信手段建立以车为节点的信息系统。通过人、车、路间的交互，实现人-车-路的和谐统一发展[1]。

车联网场景下，车载设备上传GPS数据和OBD（on-board diagnostics，发送的数据内容包括车辆位置、发动机参数等）数据；服务器端持续地接收数据，并实时地对数据进行分析，从而为不同类型的用户提供相应的服务。例如，为普通用户提供电子围栏服务、路线规划和推荐服务、驾驶员驾驶行为分析；为租车运营商提供车辆实时位置监控服务，根据驾驶员的驾驶行为分析结果进一步为运营商提供司机绩效考评服务，从而为其提出运营和管理建议；分析和预测一定范围、一段时间内的道路流量情况，以便有效地进行分流、调度和指挥，辅助国家车管部门工作。

本文采用流式处理系统[2]作为基础平台，开发数据计算任务以处理车联网数据。针对持续产生的数据流，流式系统完成实时（准实时）的计算，能够满足人们对线上业务的运行需求。通过对约两万辆车的数据进行处理，发现一些车联网计算任务有以下3个特点：

（1）任务需要长时间运行。车联网应用面向的是车载终端持续产生的GPS数据和OBD数据，因此处理该应用的任务需要长时间运行。一旦任务下线，则持续产生的数据不能得到及时的处理，也就不能为用户提供持续的服务。

（2）处理延迟低，响应速度快。汽车高速行驶带来了实时性要求高的特点，且汽车的安全行驶事关车内人员的生命和财产安全。如电子围栏应用是为了车主能够监控车辆是否处于自己划定的围栏内，目的是监控车辆安全，若实时性不能保证，则会造成早已驶出监控范围很远之后才报警的情况，车辆安全不能得到保证。同时，如路线推荐、驾驶员行为分析等应用，处理-响应时间直接关系到用户体验，因此车联网应用必须保证实时性要求。

（3）任务有状态。任务需要保存一定的内存状态，如电子围栏应用中，需要保存所有车辆的电子围栏数据，从而在接收到GPS数据后马上得到是否越过围栏的反馈；而在疲劳驾驶提醒的应用中，需要保存车辆自开始行驶到现在的持续状态，车辆行程不结束，状态不能从内存中删除。实际开发中，发现这里的状态还有一些计数值和缓存（如为提高数据库写入率，采取批量写入时在内存中缓存的数据）。

由于车联网数据和计算具有上述特点，并且车联网场景下用户基数大但在线用户比例低，流式系统中这些任务运行一段时间后，内存中存在大量不活跃的数据对象。

目前，国内外流式系统层出不穷，主流系统包括Storm[3]、Yahool！S4[4]、Spark Streaming[5]等，目前还没有关于流式系统中的不活跃对象管理的重要研究及成果。当然，对长周期任务中不活跃数据的研究并不是新问题，Corba[6]、J2EE[7]等中间件上运行着大量的长周期服务，为了防止过多的会话、连接浪费内存，它们均为数据对象提供了生命周期服务。

本文分析了车联网流式数据的特点，建立了车联网数据对象的生命周期模型；借鉴中间件技术的对象生命周期管理方法，设计和实现了流式系统的数据对象管理方法，为流式处理系统提供了一套内存优化方法，有效地解决了任务因长期运行而造成的大量不活跃对象无限占用内存的问题。本文研究虽然针对车联网数据，但本文设计的方案有很好的普适性，可以很方便地推广到其他流式数据处理场景中，比如社交网络的用户分析和事件分析。

本文的研究内容和成果包含以下几个方面：

(1)经过对车联网数据的分析，本文定位到流式系统中存在的大量不活跃对象占用内存的问题，将主流中间件系统(Corba/J2EE)采用的对象生命周期服务引入到流式处理系统中。

(2)针对传统生命周期服务中采用经验值设定参数的方式，提出一套应用和数据双驱动的过期参数确定方法，为对象生命周期服务提供更合适的参数，达到内存优化的最优效果。

(3)将数据对象生命周期模块集成到JStorm系统，并且结合车联网场景的真实用例和数据，验证本文方法对流式系统的优化效果。

本文组织结构如下：第2章介绍了车联网的数据处理需求、目前各流式系统基本情况和内存使用情况的相关工作；第3章对车联网数据进行研究，着重分析了数据的活跃程度；第4章介绍了流式数据对象的生命周期模型和数据对象过期策略；第5章将流式数据对象管理作为单独的模块集成到JStorm系统中；第6章以电子围栏应用为例建模、实验并分析结果；第7章是结论和今后的工作介绍。

2 相关工作

2.1 车联网及其数据处理

近年来，车联网作为物联网的重要组成部分，得到了空前的发展。车联网是保证道路安全，优化公共交通资源的关键。通过车联网，可以提供车辆安全、事故管理、车辆监控、流量调度、电子收费、信息娱乐等方面的服务。开发车联网相关产品，借此发展与其相关的汽车类电子产业，推动相关领域的经济变革，是对“互联网+”的一次有效尝试。

目前在车联网方面的研究主要关注在移动无线通信[8-9]、大规模移动对象管理[10-11]、基于车联网的数据挖掘[12]等方面。本文关注基于车联网的大数据处理平台，希望针对车联网数据特点，提供更合适的数据处理系统。

车辆的高速行驶直接关系到车内人员的生命财产安全，相比于其他网络应用，车联网应用具有更高的实时性需求。本文使用流式处理系统来满足车联网数据的流式处理需求，当然车联网场景下，也有批量计算、迭代计算的处理需求，但这并不是本文关注的重点。

车联网下的流式应用主要分为以下几类：

(1)基础统计。基础统计包括总数据量统计、用户活跃性统计、错误数据等各种特征数据统计。

(2)单条数据处理。对单条数据的简单处理包括数据解析、错误数据过滤，稍复杂的有电子围栏应用、车辆所属行政区划监控等。

(3)序列提取，从流式的数据序列中提取到所需的数据序列。例如，行程分段算法将流式数据按照分段规则分成一段一段的行程，行程分段算法是疲劳驾驶提醒服务的基础算法。同理可在数据流中提取到低速行驶段，可结合订单数据为车辆提供实时计价。

2.2 流式处理系统

目前，工业界和产业界推出的较为成型的流式系统实例有：Yahoo!为提高搜索广告有效点击率的问题而开发的S4[4]，安装和使用最多的流式系统Twitter Storm[3]，UC Berkeley建立在Spark[5]上的流式应用框架Spark Streaming，Microsoft StreamInsight[13]针对并行、弹性替换和容错改进之后的升级版本TimeStream[14]，融合了流处理、批处理、图处理的Microsoft Naiad[15]，数据传输通道和流式数据处理系统Data Freeway and Puma[16]，分布式实时统计系统Twitter Rainbird[17]等。研究多关注在系统框架、编程模型、可靠性、容错上，并没有专门的研究关注到流式系统中的不活跃数据问题。

在流式系统上专门针对内存使用方面的研究非

常少，Storm等多数流式系统完全依赖于JVM（Java virtual machine）进行内存管理和回收，而对于不活跃数据对象，并不能也不应该被JVM的GC（garbage collection）机制回收[18]；S4任务重启并恢复状态时，采用PE懒恢复方法，即直到再次用到PE时，才恢复其状态，懒恢复使得S4在重启任务时过滤掉了一部分不活跃的内存数据；Spark Streaming以RDD（resilient distributed datasets）[19]的形式来管理数据，RDD一般保存在内存中，但如果内存不够，可将其持久化到磁盘上。

2.3 生命周期服务

对长周期任务中不活跃数据的研究并不是新问题，Corba、J2EE等中间件系统上运行着大量的长周期服务。Corba、J2EE都为数据对象（EJB/Servant）提供了生命周期服务，对象处于活跃态时将其放在内存中，转为不活跃态时将其持久化到可靠存储中。

J2EE中采用EJB[20-21]容器来限制内存中EJB数量，这让EJB容器能够有效地利用内存、数据库连接、Socket连接等资源，同时增加EJB系统的可伸缩性。EJB容器通过挂起激活Bean实例，有效地完成Bean实例的生命周期管理。

合理的过期参数是生命周期服务是否起作用的基础。然而，这些系统中对过期策略参数由用户通过配置文件自行定制（一般设为经验值）。文献[22-23]指出应通过模型驱动确定参数，但其是从系统角度做出调整以支持动态参数变化，并没有给出一套行之有效的参数确定方法。本文认为过期参数应该视应用类型、数据到达情况而定，因此本文将在第4章给出由数据驱动的过期时间确定方法。

3 车联网数据分析

3.1 数据概述

本文的数据来自国内某租车公司的20 973辆车，月活跃车辆数约18 225辆，日活跃车辆数约为15 200台，每天的数据量约为6 000万条。

本文处理的车联网数据主要包括3种类型：GPS数据、OBD数据和用户请求数据。其中，GPS数据包括该车的经纬度、行驶速度、行驶方向等；OBD数据包括车辆的剩余油量、发动机转速、行驶里程等车辆状态数据；用户请求数据即是对车联网应用发出的定制、取消定制请求，以及对应用的参数设定等。

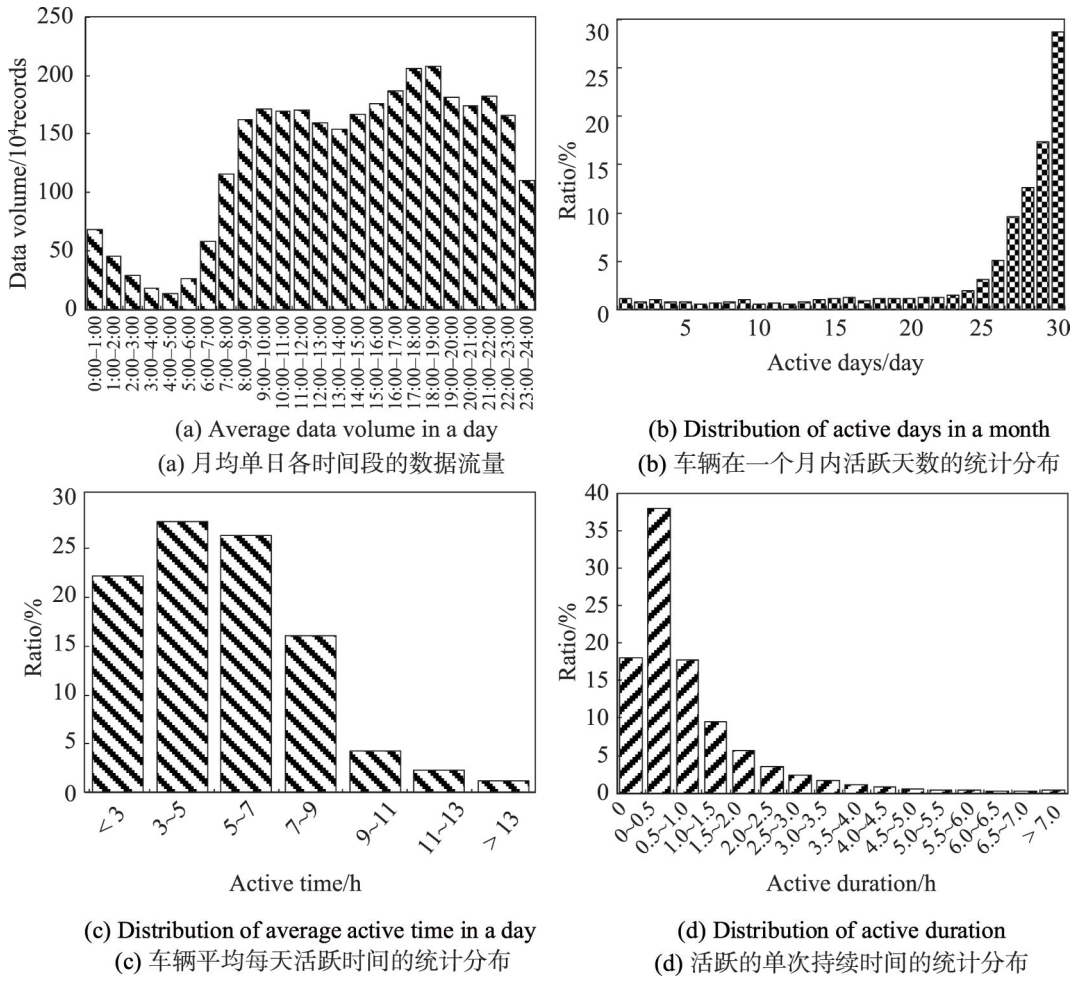
图1（a）为月均单日GPS数据量随时间变化曲线。从图中可以看到明显的早、晚高峰，早高峰数据流量约为170万条/小时，晚高峰约为210万条/小时，凌晨3点到5点数据流量最低（不到20万条/小时）。上述大规模、高密度数据除了能够表示车辆本身状态和行驶状态外，还能够刻画一个城市各个道路的流量、污染排放等情况。

与微博等社交网站相比，车联网数据有如下特征：第一，数据均为结构化数据；第二，数据反映车辆状态，本身不具有情感色彩；第三，数据上传频率更高，车辆在活跃期间会持续上传数据，因此在用户数相同的情况下，车联网数据量要比微博数据量大得多；第四，车联网的用户粘性更大，活跃频率更高，经统计，超过50%车辆的月活跃天数达到28天以上。由统计[24]可知，截止2014年末，我国汽车保有量约1.4亿，其中北京市汽车保有量为537万辆。如果车联网得到进一步推广，按照目前的数据情况推算，车联网的数据规模将明显超过微博等各大社交平台。

3.2 车辆活跃度分析

对于车联网数据，本文尤其关注车辆的活跃性，这里认为车辆有数据上传就是活跃的。人们希望知道每一辆车何时活跃，一次活跃会持续多久，总结出每一辆车的活跃规律，从而为车辆提供个性化的服务，同时也可以在后台计算上采取差异化的处置措施，以达到最佳的处理效果。然而，在数据分析阶段，单独统计一辆车的数据并不具有代表性，因此本文围绕数据活跃性问题，对数据进行了车辆月活跃天数、月平均每天活跃时间、单次活跃持续时间3项统计。

图1（b）为所有车辆一个月内活跃天数的比例分布。由图可知，车辆中约57%的车辆月活跃天数超过27天，其中每天都活跃的车辆约占28%，另外活跃频率为1~22天/月的车辆比例都非常低，共约占18%。图1（c）统计了一辆车平均每天活跃时间的分布情况。由图可知，每天活跃3~5 h的车辆最多，占总车



(a) Average data volume in a day
(a) 月均单日各时间段的数据流量

(b) Distribution of active days in a month
(b) 车辆在一个月内活跃天数的统计分布

(c) Distribution of average active time in a day
(c) 车辆平均每天活跃时间的统计分布

(d) Distribution of active duration
(d) 活跃的单次持续时间的统计分布

Fig.1 Statistics for IoV data

图1 车联网数据统计结果

辆的27%,约50%车辆每天活跃时间低于5 h,活跃时间7 h以下的占75%。图1(b)和图1(c)的统计说明虽然车辆几乎都是每天活跃,但是平均每天活跃的时间仍较低。若在处理时将所有对象都保存在内存中,那么一天中的每个时间点都会有大量对象处于不活跃状态。因此,本文将不活跃对象暂时清出内存,防止不必要的内存浪费。

图1(d)统计了单段活跃持续时间的分布。由图可知,约18%的活跃持续时间为0,即车辆突然上传一条数据,之后一段时间内没有再次产生数据,将这种活跃叫作点状活跃;与点状活跃相对应,将持续一段时间保持活跃的状态叫作段状活跃,37%活跃持续时间在0.5 h以内,90%在2 h以内,约3%的行程超过4 h。由此可知,车辆活跃的段状特征明显,同时有部分数据以零散点状方式上传。当数据表现为明显的段状活跃特征时,说明对象在一次活跃后,短时间内很可能被多次访问,那么可以做出预测:下一次被访问的对象很可能是最近刚被访问过的对象。4.2节将基于这一预测设置数据对象的过期策略。

4 SDO生命周期模型

针对流式系统中存在大量不活跃对象问题,本文给出了流式系统的数据对象管理方法,该方法应具有如下特点:

(1)受管对象可以是流式系统中的任何Java对象,对于自定义对象,用户须为其提供相应的序列化

方法。

（2）能够自动检测不活跃对象，并将其持久化到可靠存储中，防止因不必要的内存占用而浪费资源；当对象恢复活跃时，能够较快地恢复内存状态，正常地为流式系统提供服务。

为方便叙述，将此类受管对象统一命名为流数据对象（streaming data object，SDO）。流式系统使用SDO容器（SDOContainer）组织和管理SDO。SDOContainer功能包括SDO组织和SDO生命周期管理。

4.1 SDO状态及状态转移

本文将每个SDO的生命周期分为3个阶段。

（1）不存在（Does not exist）：SDO还没有被实例化，不在内存中；

（2）活跃态（Ready）：在此状态下，SDO可正常被流式系统使用；

（3）不活跃态（Inactive）：SDO从内存中移除，并持久化到磁盘中，可以恢复内存状态并为流式系统提供服务。

图2是SDO生命周期的状态转换图。

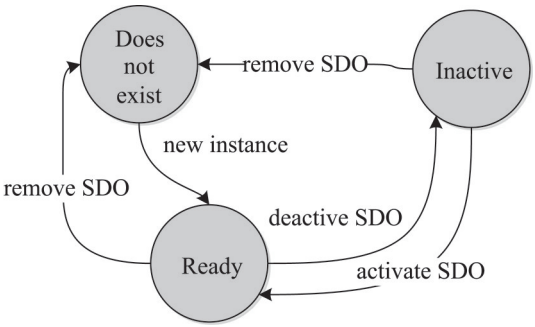


Fig.2 SDO's state machine during its lifecycle

图2 SDO生命周期状态机

（1）Does not exist→Ready：当流式系统用户创建新的SDO实例时，SDO的生命周期开始，完成初始化后，SDO保存在内存里开始为流式系统使用。

（2）Ready→Inactive：SDO被检测为不活跃对象，SDO被去活，去活之前先释放SDO可能持有的所有资源，去活SDO后，SDO容器将SDO的状态持久化到硬盘或其他存储源中。默认情况下，将SDO序列化、持久化到硬盘。

（3）Inactive→Ready：被去活的SDO可能再次被激活，这时需要载入并恢复其被持久化的状态；激活之前SDO须恢复去活时释放的资源，虽然此时的SDO实例已经不是被去活之前的SDO实例，但是这一切对流式应用是透明的。

（4）Ready/Inactive→Does not exist：SDO被显式移除。

4.2 SDO过期策略

基于3.2节对车联网数据活跃规律的分析，对于明显段状活跃的数据，应尽量把刚刚访问过的对象保存在内存中，而将那些很久没有被访问的SDO从内存中移除。基于这种移除方式，SDOContainer提供两种基本的SDO过期策略。

（1）基于空间的过期策略。SDOContainer设定一个SDO数量上限，当到达该上限时，按照最近最少访问顺序来去活对象；此方法是为了防止SDO无限制地占用内存，是对系统资源进行保护。SDO的最大数量根据系统的资源情况和SDO的大小共同估计得到。此方式得到的延迟是在保护系统资源的前提下，为流式任务提供的最低延迟。

（2）基于时间的过期策略。SDOContainer对其管理下的SDO设置一个超时时间，当超过这一时间没有被访问时，就将其从内存中剔除。这种方法用于防止长时间未被使用的数据对象长期占用内存而造成内存浪费。SDO的过期时间需根据应用和数据情况而建模确定，下一节将介绍过期时间的确定方法。此方式能够保证系统延迟在可接受的范围内，将内存使用降到最低。

4.3 SDO过期窗口确定方法

相比于中间件中采用的用户自行设定过期时间的方法（设置经验值），本文采用应用和数据驱动的参数确定方法，为流式任务确定“合适的”SDO过期时间。“合适的”过期时间是指在保证延迟用户可接受的前提下，最大限度地降低内存占用率。可以将内存中SDO看成是最近最少使用SDO链上的滑动窗口范围内的数据（如图3所示），窗口代表着SDO的过期时间。

本文选择流式任务的处理延迟、内存占用作为性能指标，那么需要明确窗口对延迟、内存占用的影响。图4是对内存占用、处理延迟随窗口大小变化的

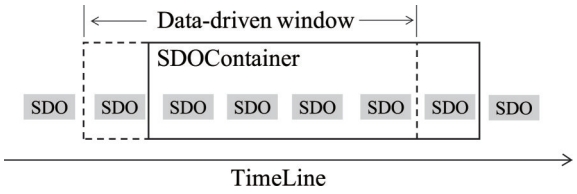


Fig.3 Sliding window of SDOContainer based on expiring time

图3 SDOContainer基于过期时间的滑动窗口

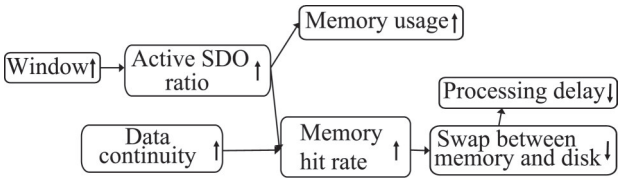


Fig. 4 Influence of window size on processing delay and memory usage

图4 窗口大小对处理延迟和内存占用的影响

经验分析：增大*window*，导致活跃对象的比例增大，内存占用率与活跃对象比例呈正比，因此内存占用率相应增大；而延迟与内存击中率直接相关，内存击中率受活跃对象比例和数据连续性两方面共同影响，对于连续性稳定的数据，整体上内存击中率随活跃对象比例增大而增大。例如：当数据随机到达时，内存击中率与活跃对象比例呈正比关系，数据活跃比例一定时，数据的连续性高，内存击中率大。

根据以上分析，可得目标函数(1)：

$$delay(w,t)=hr(w,t)\times d_{\text{mem}}+(1-hr(w,t))\times d_{\text{load}} \quad (1)$$

其中，w为窗口大小；t为时间；$delay$为采用w窗口在t时刻的处理延迟；hr为击中缓存的概率，它由窗口大小和该时刻的数据情况决定；d_{mem}为直接从内存中取数据并投入计算时应用的处理延时；d_{load}为未击中内存(从可靠存储中取出并恢复SDO状态)时的数据处理延迟。

理论上，用户给定最大可接受的平均延时，在每个时间点会对应一组*window*值，由于内存占用随*window*的增大而增大，取这组值中最小值作为*window*，即可达到在给定最大可接受处理延迟的前提下，将内存占用降到最低。

总结*window*的确定方法为：测试式(1)中参数d_{load}、d_{mem}值，得到处理延迟随内存击中率变化的函数；给定最大可接受的处理延迟d_1，得到相应的缓存击中率hr_1；统计并拟合出$hr=P(w,t)$，取$hr=hr_1$，即可得到随时间变化的窗口大小。此*window*值可以使得在保证用户设定平均延迟的前提下，内存占用降到最低。

5 系统实现

本文将SDO生命周期管理作为单独模块集成进入JStorm系统。JStorm是一个分布式实时计算引擎，它用Java完全重写Storm内核，保持了Storm的编程规范性、健壮性和拓展性，不仅如此，JStorm重新设计了调度、采样、监控、HA，并对Zookeeper[25]和RPC进行大幅改良，从而JStorm比Storm更稳定，功能更强。

JStorm系统由3类节点组成(如图5所示)。

(1)Zookeeper：JStorm集群资源协调者。Supervisor、任务task定期向Zookeeper发送心跳，Nimbus通过检测Zookeeper上的心跳数据来监控集群的资源情况和任务的执行情况。

(2)Nimbus：任务分配器。Client向Nimbus提交任务后，Nimbus为任务进行资源分配并写入Zookeeper，Supervisor从Zookeeper上获取分配给自己的任务并启动相应的进程、线程。

(3)Supervisor：JStorm的工作节点。Supervisor启动Worker执行分配给自己的任务，Supervisor监控本地Worker的状态，以便及时地进行故障恢复。

运行于Supervisor上的Worker进程是任务的实际执行者。图的下半部分是Worker对消息的处理流程，tuple首先进入task的接收队列，BoltExcutor线程根据处理速度从接收队列取出tuple并做处理，向外发送的tuple被放入序列化队列待序列化后发送到下级bolt。

SDOContainer作为一个单独的模块集成到JStorm中，当开发者需要生命周期管理服务时，可申请相应的SDOContainer，将需要被管理的中间结果或任何数据放入其中。同时设置SDOContainer的过期参数、可靠存储的相应参数，SDOContianer即可自动地完成对其内部SDO的生命周期管理。

默认状态下，SDOContainer将Inactive状态的

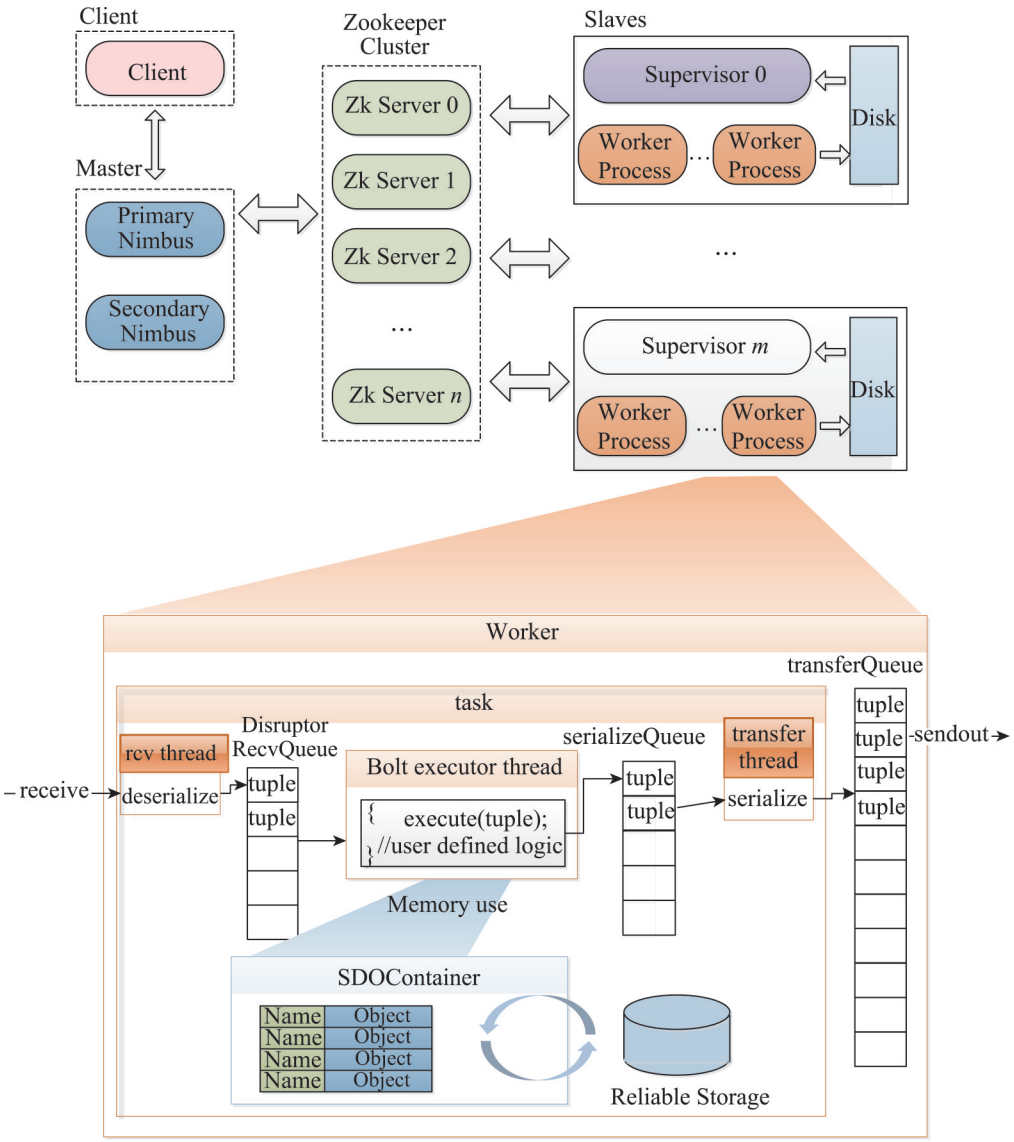


Fig.5 JStorm+SDOContainer system

图5 JStorm+SDOContainer系统

SDO写入到外存中，为用户提供持久化接口，用户通过连接（Connect）、关闭（Close）、写入（Store）、读取（Load）4个函数，即可实现内存数据和任何存储介质的数据交换。

6 实验验证和数据分析

6.1 集群配置和使用

测试资源包括两个物理节点（test1/test2），每个节点的配置是：8核2.80 GHz Intel®Core™ i7 CPU860，16 GB内存和1 TB的SATA硬盘。其中，test1单纯作为JStorm的工作节点（包含Supervisor进程和Worker进程）；test2除作为JStorm的任务分配节点（启动Nimbus进程）外，还安装了MySQL数据库作为可靠存储。

测试中，SDOContainer采用默认持久化策略，将不活跃SDO通过序列化工具Kryo序列化并存储到MySQL数据库中。

6.2 实验设计

6.2.1 用例描述

电子围栏应用是车联网场景下的一个流式应

用。管理员为车辆设置电子围栏范围，车辆持续向服务器端发送GPS位置数据，服务器端实时判断车辆是否越过电子围栏，若超出或重新进入围栏，为车辆管理员发送状态变更提醒。该应用用于满足租车运营商或普通车主的车辆安全需求。

图6为电子围栏用例的任务拓扑图，Storm拓扑中共有4个逻辑节点，其中2个Spout作为数据源和2个Bolt作为数据处理节点。

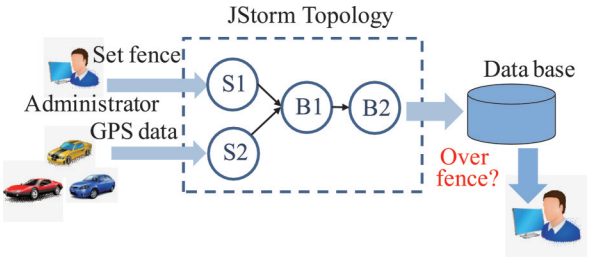


Fig.6 Topology of electronic-fence application

图6 电子围栏任务拓扑

(1)Fence_spout(S1)：接收管理员对电子围栏的设置，发送围栏数据到InFenceMonitor_bolt节点，更改内存中车辆的围栏设置。

(2)Location_spout(S2)：持续接收车辆发送的位置信息，发送位置数据到InFenceMonitor_bolt节点。

(3)InFenceMonitor_bolt(B1)：使用优化射线法[26]，判断车辆位置是否超越电子围栏，若发生状态变更(超出围栏或重回围栏)，则将结果发送到DBvisior_bolt节点。

(4)DBvisitor_bolt(B2)：数据库访问节点，将车辆进出围栏时间写入到数据库中。

6.2.2 数据来源

车辆位置数据来源于国内某租车公司的20 973辆车，车辆未熄火时约5~8 s发送一次GPS数据，熄火时不发送数据。

设置车辆的电子围栏为注册城市区域，城市的GPS区域通过百度地图API获得，平均约为600边的多边形。

6.3 SDO过期参数分析

本节根据4.3节给出的SDO过期参数确定方法，确定SDOContainer最大容量和SDO过期时间。

首先，将任务运行在没有SDOContainer管理的JStorm平台，内存使用量约为10.3 GB，数据处理延迟约为13.11 ms。由此可知，任务未达到系统内存瓶颈，因此不必设置SDOContainer的最大容量，系统可达到的最小处理延迟为13.11 ms。

测试得到hr=0.9时，$delay$=46.79 ms，可知公式中的d_{load}约为349.91 ms/条。因此，目标函数可表示为：

$$delay(w,t)=-336.8\times hr(w,t)+349.91 \quad (2)$$

之后，统计各时间点，内存击中率随窗口大小的变化$hr=P(w,t)$，该统计结果拟合出的应为具有二维变量的三维图形，抽取代表性时间点的数据绘制图表。由图7可知，$window$在0~20 s内，内存击中率随$window$大小急剧增长，几乎所有时间点的击中率都达到90%以上，当$window$进一步增大时，击中率上升缓慢，最终击中率为1。

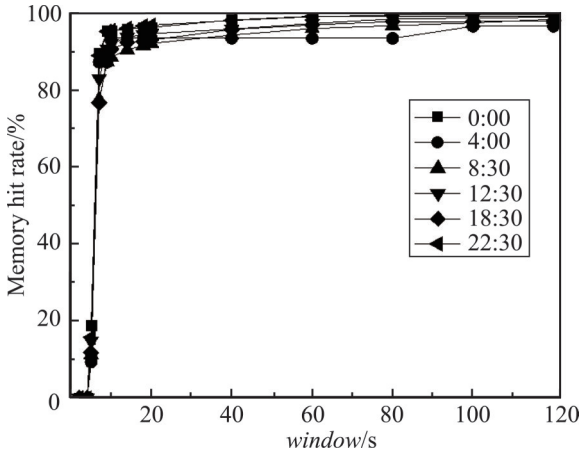


Fig.7 Influence of *window* size on memory hit rate

图7 内存击中率随窗口大小变化

所有时间点的内存击中率随窗口大小的曲线几乎重合，因此可采用不随时间变化的窗口大小。设定最大可接受的平均处理延迟为30 ms，此时需满足hr>94.49%，根据图7的统计结果，对应的窗口大小为30 ms(图7中，30 ms对应的平均内存击中率为95.5%)。

没有加入SDOContainer之前，平均处理延迟为13.11 ms，内存占用为10.3 GB；加入SDOContianer，且$window$=30时，延迟为18.81 ms，内存占用为8.3 GB，内存占用降低了近20%。

6.4 SDO性能分析

本节测试使用SDOContainer后，对该应用处理

延迟、内存占用方面的改变，以及选择*window*=30 s的合理性。

图8为处理延迟、内存占用随窗口大小的变化情况。由图可知，在测试区间（10~120 s）内，处理延迟随窗口的增大而减小，内存使用随窗口增大而增大，要使处理延迟低于30 ms，*window*需大于30 s。虽然由于测试精度的问题，*window*取30 s、40 s时内存占用均为8.3 GB，但从理论分析和总体趋势可以知道，*window*=30是在满足处理延迟低于30 ms的前提下，内存占用率最低的情况。

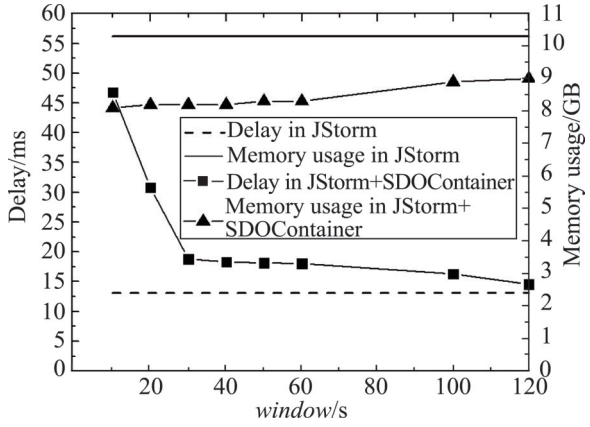


Fig. 8 Processing delay and memory usage in JStorm and JStorm + SDOContainer

图8 加入/未加入SDOcontainer时JStorm处理延迟和内存占用情况

由上述分析可知，在电子围栏应用中，引入SDOContainer模块并设置合适的SDO过期参数之后，可以为流式系统大幅度地节省内存资源。

7 总结和展望

本文针对流式系统中存在大量不活跃对象的问题，对流式数据对象建立生命周期模型，采用数据驱动、应用驱动的参数确定方法，为SDO设置合适的过期参数。实验结果表明，本文方法在用户可接受的处理延迟范围内，能够有效减少流式系统中不活跃对象的数目，节约系统的内存资源，达到内存优化的目的。

对于SDOContainer模型中SDO过期参数明显随时间而变化的用例，本文尚未给出参数确定方法，这也是本文的后续工作。

References:

[1] White paper of Internet of vehicles (IoV)[EB/OL]. [2015-06-26]. http://mddb.apec.org/Documents/2014/TEL/TEL50-PLEN/14_tel50_plen_020.pdf.

[2] Zhu Jianping, Lai Shengqiang. A survey on streaming data mining and future directions in statistical research[J]. Statistical Research, 2007, 24(7): 84-87.

[3] Toshniwal A, Taneja S, Shukla A, et al. Storm@twitter[C]// Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, Snowbird, USA, Jun 22-27, 2014. New York: ACM, 2014: 147-156.

[4] Neumeyer L, Robbins B, Nair A, et al. S4: distributed stream computing platform[C]//Proceedings of the 2010 IEEE International Conference on Data Mining Workshops, Sydney, Australia, Dec 13, 2010. Piscataway, USA: IEEE, 2010: 170-177.

[5] Zaharia M, Das T, Li Haoyuan, et al. Discretized streams: fault-tolerant streaming computation at scale[C]//Proceedings of the 24th ACM Symposium on Operating Systems Principles, Farmington, USA, Nov 3-6, 2013. New York: ACM, 2013: 423-438.

[6] Vinoski S. CORBA: integrating diverse applications within distributed heterogeneous environments[J]. IEEE Communications Magazine, 1997, 35(2): 46-55.

[7] The white paper of JavaEE7[EB/OL]. [2015-06-26]. http://www.oracle.com/technetwork/java/javaee/.

[8] Wu Hao, Tang Hengliang, Dong Lan. A novel routing protocol based on mobile social networks and internet of vehicles [C]//LNCS 8662: Proceedings of the 1st International Conference on Internet of Vehicles - Technologies and Services, Beijing, Sep 1-3, 2014. Basel, Switzerland: Springer International Publishing, 2014: 1-10.

[9] Schönfelder R, Leucker M, Walther S. Efficient profile routing for electric vehicles[C]//LNCS 8662: Proceedings of the 1st International Conference on Internet of Vehicles-Technologies and Services, Beijing, Sep 1-3, 2014. Basel, Switzerland: Springer International Publishing, 2014: 21-30.

[10] Hu Haibo, Xu Jianliang, Lee D L. A generic framework for monitoring continuous spatial queries over moving objects [C]//Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, Baltimore, USA, Jun 14-16, 2005. New York: ACM, 2005: 479-490.

[11] Alamri S, Taniar D, Safar M, et al. A connectivity index for moving objects in an indoor cellular space[J]. Personal and Ubiquitous Computing, 2014, 18(2): 287-301.

[12] Shang Jingbo, Zheng Yu, Tong Wenzhu, et al. Inferring gas consumption and pollution emission of vehicles throughout a city[C]//Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, Aug 24-27, 2014. New York: ACM, 2014: 1027-1036.

[13] Ali M H, Gerea C, Raman B S, et al. Microsoft CEP server and online behavioral targeting[J]. Proceedings of the VLDB Endowment, 2009, 2(2): 1558-1561.

[14] Qian Zhengping, He Yong, Su Chunzhi, et al. Timestream: reliable stream computation in the cloud[C]//Proceedings of the 8th ACM European Conference on Computer Systems, Prague, Czech Republi, Apr 15-17, 2013. New York: ACM, 2013: 1-14.

[15] Murray D G, McSherry F, Isaacs R, et al. Naiad: a timely dataflow system[C]//Proceedings of the 24th ACM Symposium on Operating Systems Principles, Farmington, USA, Nov 3-6, 2013. New York: ACM, 2013: 439-455.

[16] Zheng Shao. Data freeway and puma: realtime data streams and analytics[C]//Proceedings of Hadoop in China 2011, Beijing, Dec 2-3, 2011.

[17] Rainbird: realtime analytics at twitter[EB/OL]. [2015-06-26]. http://www.slideshare.net/kevinweil/rainbird-realtime-analytics-at-twitter-strata-2011.

[18] Liu Yongpo, Jia Xiaoxia, Wu Ji, et al. Analysis of memory usage with low efficiency in Java program[J]. Computer Engineering, 2008, 34(23): 84-85.

[19] Zaharia M, Chowdhury M, Das T, et al. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing[C]//Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, San Jose, USA, Apr 25-27, 2012. Berkeley, USA: USENIX Association, 2012: 2.

[20] Sriganesh R P, Brose G, Silverman M. Mastering enterprise JavaBeans 3.0[M]. Hoboken, USA: John Wiley & Sons Inc, 2005.

[21] Li Huoming. Beginning in EJB3.0[M]//Beijing: Tsinghua University Press, 2008.

[22] Bidan C, Issarny V, Saridakis T, et al. A dynamic reconfiguration service for CORBA[C]//Proceedings of the 4th International Conference on Configurable Distributed Systems, Annapolis, USA, May 4-6, 1998. Piscataway, USA: IEEE, 1998: 35-42.

[23] Coulson G, Blair G S, Clarke M, et al. The design of a configurable and reconfigurable middleware platform[J]. Distributed Computing, 2002, 15(2): 109-126.

[24] National Bureau of Statistics of China. Statistical communiqué of the People's Republic of China on the 2014 national economic and social development[EB/OL]. [2015-06-26]. http://www.stats.gov.cn/tjsj/zxfb/201502/t20150226_685799.html.

[25] Hunt P, Konar M, Junqueira F P, et al. ZooKeeper: wait-free coordination for Internet-scale systems[C]//USENIX Annual Technical Conference, Boston, USA, Jun 23-25, 2010. Berkeley, USA: USENIX Association, 2010.

[26] Wu Jian, Jiang Hong, Wang Xiaochun. A method for the decision of a point whether in or not in self-intersected polygon[J]. Journal of System Simulation, 2003, 15(11): 1592-1594.

附中文参考文献：

[2] 朱建平, 来升强. 流式数据挖掘的现状及统计学的研究趋势[J]. 统计研究, 2007, 24(7): 84-87.

[18] 柳永坡, 贾晓霞, 吴际, 等. Java程序内存低效使用问题的分析[J]. 计算机工程, 2008, 34(23): 84-85.

[21] 黎活明. EJB3.0入门经典[M]. 北京: 清华大学出版社, 2008.

[24] 中华人民共和国国家统计局. 2014年国民经济和社会发展统计公报[EB/OL]. [2015-06-26]. http://www.stats.gov.cn/tjsj/zxfb/201502/t20150226_685799.html.

[26] 吴坚, 姜虹, 王小椿. 快速判断点是否在自交多边形内的方法[J]. 系统仿真学报, 2003, 15(11): 1592-1594.

WANG Jiaxing was born in 1990. She is an M.S. candidate at Beihang University. Her research interests include distributed computing and big data processing.

王家兴(1990—),女,辽宁阜新人,北京航空航天大学硕士研究生,主要研究领域为分布式计算,大数据处理。

LIN Xuelian was born in 1978. He received the Ph.D. degree from Beihang University in 2013. Now he is a lecturer at Beihang University. His research interests include distributed computing and middleware.

林学练(1978—),男,浙江温州人,2013年于北京航空航天大学获得博士学位,现为北航计算机学院讲师,主要研究领域为分布式计算,中间件技术。

SHEN Yang was born in 1991. He is an M.S. candidate at Beihang University. His research interests include distributed computing and big data processing.

申阳(1991—),男,河北邯郸人,北京航空航天大学硕士研究生,主要研究领域为分布式计算,大数据处理。