doi:10.3772/j.issn.1002-0470.2021.03.001

基于规则实时性的端云动态分配方法研究①

黄晓辉②*** 崔 莉③** 黄 希**

(*中国科学院大学 北京 100049)

(**中国科学院计算技术研究所 北京 100190)

摘 要 针对物联网控制规则在实际应用中,由于延迟触发执行导致物联网系统引发不同程度的控制安全事故,本文基于规则的实时特性提出了一种面向物联网控制规则的端云动态分配方法(EasiDEP),以提高物联网系统控制规则的实时性。首先,根据规则之间的相关性以及实时特性,提出了一种规则聚类算法,将规则库中的规则分为实时和无实时规则;然后,基于规则聚类结果提出一种端云动态规则分配方法,将实时规则分发至云端和本地规则服务端同时处理,提高实时规则的触发执行率。最后,通过实验验证了EasiDEP能够有效地提高物联网控制规则的实时触发执行率,降低系统发生控制安全事故的风险。

关键词 物联网控制规则;实时特性;控制安全;动态分配;规则聚类

0 引 言

物联网技术为智能家居、医疗、农业、工业监控等不同领域的自动化控制提供了一种便捷的方式[1]。物联网系统由具有不同传感、驱动和计算设备的异构设备网络组成[2]。近年来,规则引擎被引入到物联网系统中,不仅解放了开发人员编写业务规则程序的繁重工作,同时也使用户在没有开发人员支持的情况下,能够自行通过设置控制规则来灵活地配置和定制系统中的各种控制任务[3]。

伴随着物联网规则引擎在各行各业的进一步推广和应用,尤其是在一些实时控制要求较高的领域,比如安全监控系统中,由于安全监控类的规则不能够实时触发执行,将会给用户的生命和财产安全造成无法挽回的灾难性事故。比如,在一个智能家居系统中,规则"当室内烟雾报警器检测到火警时,通知房主采取措施"由云端统一管理,当检测到火警时,云端待触发的火灾应急联动规则由于网络延迟或终端产生规则执行延误,错过了通知外出房主的最佳时间,会给房主造成很大的损失。所以,减少规则触发执行延迟和提高规则实时触发执行率,成为物联网规则引擎在实际应用中亟待解决的一个重要问题。目前,针对物联网规则引擎的规则实时触发执行的相关研究工作,主要是采用云集中和本地触发执行这两种方法。

在云集中方法的工作中,文献[4]扩展了现有的用于用户描述实时约束的IFTTT(if this then that)语法,提出了具有触发器条件感知灵活轮询间隔的实时物联网规则引擎架构RT-IFTTT(real time-if this then that),分析框架中所有小应用程序的当前传感器值、触发条件和执行,并动态计算每个传感器的有效轮询间隔。这项工作从10个物理传感器收集了10 d的真实传感数据,实验表明,使用该调度算法的RT-IFTTT框架与使用固定间隔的框架相比,传感器轮询计数减少了64%,提高了规则触发的实时性。文献[5]使用包含私有和公共云组件的混合云管理

① 国家自然科学基金(61672498)和国家重点研发计划(2016YFC0302300)资助项目。

② 男,1987年生,博士生;研究方向:物联网,无线传感器网络;E-mail:huangxiaohui@ict.ac.cn

③ 通信作者,E-mail:lcui@ict.ac.cn

(收稿日期:2020-04-20)

机制,提出了一种基于规则引擎的分布式混合云管理平台,并结合分布式、可靠的架构,满足批量处理、自动操作的需求,设计了一个简洁的规则模型来定义规则和事件,使得管理来自多个 IaaS 云提供商的云资源更加灵活、高效。文献[6]指出单台计算机的内存和计算能力有限,传统的规则引擎系统无法处理大数据,为了支持大数据推理,提出了基于 Spark 的分布式规则引擎 SparkRE 系统。特别是,SparkRE 使用 DataFrame 表示分布式工作内存,其中包含大量事实,并通过 Spark SQL(structured query language)的查询执行引擎实现规则条件测试机制;还为 SparkRE 设计了一种规则语言和一个高效的推理引擎,提高了系统的性能,减少了规则触发的延时。

本地管理方法中,文献[7]通过添加“关联事实”属性和“议程推理机制”来减小匹配规则集的大小,以及避免匹配过程中不必要的等待时间,提出了一种在物联网网关上实现的轻量化规则引擎(faster light-weight rules engine, FLRE),解决了传统规则引擎无法应用于轻量化物联网网关并且运行时间和响应时间长的问题,并用规则数量不等的规则数据测试,表明 FLRE 能够高效地应用于物联网网关上。文献[8]根据数据的传输会消耗大量电能和网络流量,提出了一种基于低复杂度规则引擎的医疗数据采集和智能传输系统架构,采用 IEEE 802.15.4 标准将数据传输到网关,通过基于事件的传输而不是数据的连续传输,可以减少设备消耗的电能和产生的网络流量。其作者还开发了两种不同的规则引擎,即静态规则引擎和自适应规则引擎,它们根据从数据中提取的重要特征来决定是否传输收集到的数据,从而实现节能、减少网络流量、提高系统实时性。文献[9]关注了物联网监测系统中实时处理的两个问题:(1)如何有效地将大量的原始传感数据转化为有意义的复杂事件;(2)如何适应监测业务逻辑的复杂性和多变性,提出了一种面向物联网实时监测的通用复杂事件处理机制(complex event processing, CEP)和一种包含原始事件、简单事件和复杂事件的形式化层次复杂事件模型(complex of event modeling, COEM),降低了事件建模的复杂性。COEM 模型支持复杂时空语义,通过编程方式定义灵活的复杂事件,基于 COEM 还提出了一个 CEP 系统架构。该系统部署在终端传感设备和云应用之间的网络边缘,将复杂事件定义映射到 CEP 规则逻辑脚本中,及时检测潜在的异常事件。所提出的 CEP 机制是通用的,适用于任何异构传感设备和 CEP 引擎。在工业环境中,文献[10]收集能源使用数据,分析数据,并通过应用控制策略优化能源消耗,提出了一种基于语义规则引擎(semantic rules engine, SRE)的工业网关控制策略,允许网关动态管理规则,而不会导致任何服务中断。此外,它可以处理语义查询,并通过从本体中先前定义的概念推断额外的知识来提供结果。

现有的研究工作从不同的角度分别提出了可用于提高控制规则实时触发执行率的相关技术方法,其中云端集中管理方法能够充分利用云端提供的高性能处理能力,而本地管理方法能够尽可能地减少由于网络传输过程中造成的延迟。所以,上述工作采用了各种有效的方法单方面针对云端或本地的性能进行优化提升,减小规则触发执行过程的延迟,提高了规则的触发执行率。但是,当传输网络出现严重延迟或者网络中断现象,导致规则不能及时触发执行则会影响系统的控制安全。例如,煤矿安全监控系统中监控瓦斯的安全控制规则“CH_4 浓度大于 1.2%,关断电源”,由于传输网络的延迟问题,导致安全控制规则不能及时触发执行,可能引发煤矿发生瓦斯爆炸事故。本地物联网设备由于计算、存储等资源受限,采用本地集中管理方式只能局部减小部分控制规则的延迟,难以提高整个系统的实时性。所以,单方面的性能提升并不能从根本上减少规则触发执行的延迟,无法保障控制规则的实时性,仍存在由于控制规则的延迟触发执行产生的控制安全隐患。

综上所述,现有研究工作主要是采用云端或本地的单边规则处理机制,这类方法受到系统部署环境以及规模的影响比较大,容易导致规则触发执行出现延迟的现象。而本文兼顾了云端和本地管理方法的优势和缺点,采用云端和本地双边规则处理机制,提出了一种新的方法 EasiDEP,进一步减小规则

触发执行的延迟,提高系统的控制安全性能。本文的主要创新点和贡献如下。

(1)为了提取出实时规则,基于规则相关性和欧式距离,提出了一种规则聚类算法 EasiKM,将规则分为有实时性要求和无实时性要求的两类规则。

(2)基于 EasiKM 的聚类结果,结合云端和本地的规则处理能力以及网络延迟因素,提出了一种端云动态规则分配算法(EasiDEP)。

(3)搭建了实验平台,验证了 EasiDEP 能够有效地减小规则触发执行的延迟,降低由于控制规则延迟触发执行引发的控制安全事故的发生率,提高了物联网规则引擎在实际应用中的安全可靠性。

1 规则聚类算法

1.1 基于 EasiDEP 物联网规则引擎概述

本文提出的物联网规则引擎结构如图 1 所示,主要包括规则库、EasiDEP 规则分配模块、推理模块(包含模式匹配和议程)、工作内存(working memory)、执行规则模块部分。规则库中存储着规则引擎中的所有规则,一般都满足 if-then 结构,if 部分描述规则的触发条件,then 部分为规则的执行部分[11]。EasiDEP 规则分配模块主要包括规则聚类、实时规则库和规则分配子模块,规则库中的规则通过规则聚类分为实时规则和无实时规则,并提取出实时规则存入实时规则库中,然后根据规则分配子模块的规则分配结果,分别将规则分到云端和本地处理。工作内存中是等待与规则库中的规则进行匹配的初始数据(又称为事实数据),即 if 触发条件中的传感数据。推理模块中的模式匹配负责将从规则库中取出的规则与工作内存中的数据进行匹配比对,通过匹配并完全触发的规则送入议程模块,由该模块决定这些触发后规则的执行顺序,最后再由执行模块从议程中取出已经触发的规则执行,并反馈给工作内存更新数据[12]。

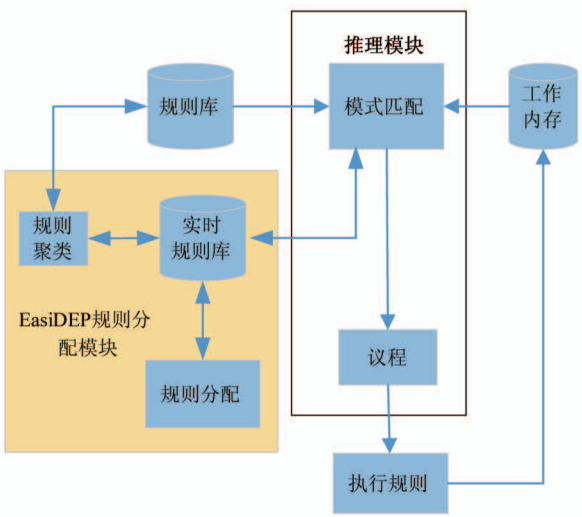


图 1 基于 EasiDEP 规则引擎结构示意图

在规则引擎中,通常会使用某种表述性的语言(而不是编程语言)来描述规则,所以规则描述语言也是规则引擎的一个重要组成部分[13]。目前在规则描述语言方面,并没有一个通用的标准获得规则引擎厂商的广泛支持,大部分规则描述语言都是厂商私有的。但是,不管是哪种规则描述语言,都需要描述一组条件以及在此条件下执行的操作(即规则)。

本文以文献[14]研究工作为基础,制定了规则的统一描述格式,从规则的属性(*Rule _ ID*、*Rule _ Typ*、*Rule _ Pro*)、触发状态 *Rule _ Sta*、触发条件(*Trigger _ Typ*、*Trigger _ ID*、*Trigger _ Value*、*Value _ Au*)以及执行部分(*Actuator _ ID*, *Actuator _ Value*)来定义规则,比如,设定规则 *ID* 为 90,执行优先级为 10,温度传感器 *ID* 为 20,空调 *ID* 为 15,可将规则"当温度大于 30 ℃时,打开空调降温"表示为 *Rule* = {90, 1, 10, 0, 1, 20, 30, 1, 15, 1}。

1.2 EasiKM 规则聚类算法

本文要根据规则的实时特性在本地与云端之间实现动态分配机制,以达到提高实时触发执行规则的目标,首先需要根据规则的实时特性将规则分为实时规则和无实时规则两类,为此,本文基于规则间的相关性和欧式距离提出了一种规则聚类算法 EasiKM,计算过程如下所示。

定义 1 假设规则数据集 $\boldsymbol{D}$ 为 $\{\boldsymbol{D}_1, \boldsymbol{D}_2, \cdots, \boldsymbol{D}_n\}$,每个规则 $\boldsymbol{D}_i$ 具有 d 维特征属性,其中,n 为规则库中的规则总数,$i = 1, 2, \cdots, n$。

定义 2 有实时性要求的规则称为"实时规则",无实时性规则称为"无实时规则"。

设计 EasiKM 的目的，就是要把规则数据集 $\boldsymbol{D}$ 中的所有规则分为实时和无实时规则两类，所以在给定规则分组类别数 $k=2$ 的条件下，可以将规则数据集分为 2 类。

$$\boldsymbol{D}_u=\{\boldsymbol{C}_1,\ \boldsymbol{C}_2\} \tag{1}$$

式中，$\boldsymbol{C}_1$ 为实时规则数据集，$\boldsymbol{C}_2$ 为无实时规则数据集。从规则库中选择优先级最大的规则 $\boldsymbol{D}_u=\{x_{u1},\ x_{u1},\cdots,\ x_{un}\}$ 作为初始实时规则，优先级最小的规则 $\boldsymbol{D}_d=\{x_{d1},\ x_{d1},\cdots,\ x_{dn}\}$ 作为初始无实时规则。由文献[14]可知，规则引擎中存在优先级相同的规则，本文引入皮尔逊相关系数 p(Pearson correlation coefficient)，分别从优先级最高和最低的规则中选择一组规则并计算 p 值，最后取 p 值最小的规则作为 $\boldsymbol{C}_1$ 和 $\boldsymbol{C}_2$ 初始类中心。

对剩余的规则 $\boldsymbol{D}_i(i\neq u,\ d)$，根据每个规则到 $\boldsymbol{D}_u$ 和 $\boldsymbol{D}_d$ 的欧式距离(Euclideam distance，ED) ed 的大小确定归属哪一类，计算公式如下所示：

$$ed=d(x_i,\ x_j)=\sqrt{\sum_{k=1}^{n}|\ x_{ik}-x_{jk}\ |^2} \tag{2}$$

式(2)中的 j 分别取 u 和 d 得到 ed_{iu} 和 ed_{id}，根据这两个值的最小值来判断 $\boldsymbol{D}_i$ 应该归到哪一类。

$$\boldsymbol{D}_i\in\begin{cases}C_1,\ ed_{iu}\geqslant ed_{id}\\ C_2,\ ed_{iu}<ed_{id}\end{cases}\quad i=1,2,\cdots,n\ 且\ i\neq u,\ d \tag{3}$$

剩余的所有规则完成聚类后，重新计算 2 类规则集 $\boldsymbol{C}_1$ 和 $\boldsymbol{C}_2$ 的类中心，然后通过计算已划分类别的每个规则属性的均值得到新的类中心规则，并计算误差平方和函数 E。

$$E=\sum_{i=1}^{n}\sum_{j=1}^{k}r_{ij}\parallel x_i-\mu_j\parallel^2 \tag{4}$$

其中，第 i 个规则属于第 k 类时 $r_{ij}=1$，不属于则 $r_{ij}=0$，μ_j 为第 j 个中心点，$i=1,\ 2,\cdots,\ n,\ k=1$ 或 2。

循环式(2)~式(4)的计算过程，直到函数 E 收敛不再变化，则规则聚类过程结束。EasiKM 规则聚类算法如算法 1 所示。

基于规则相关性和欧式距离的 EasiKM，在选择初始类中心时，并不是采用传统 K-means 算法采用随机选择初始类中心的方法，而是先从优先级最高和最低的规则中选择一组规则并计算两组规则间的 p 值，最后取 p 值最小即相关性最小的规则作为初始类中心，这样不仅降低了聚类过程的计算复杂度，同时也提高了规则聚类结果的准确性。

算法 1　EasiKM 算法

输入：规则数据集 $\boldsymbol{R}$，类别数 $k=2$；
输出：实时规则集 $\boldsymbol{C}_1$，无实时规则集 $\boldsymbol{C}_2$；
1. 初始化 $\boldsymbol{D}$ 中优先级最高的规则 $\boldsymbol{D}_u$ 和最低的规则 $\boldsymbol{D}_d$；
2. if $\boldsymbol{D}_u$ 和 $\boldsymbol{D}_d$ 都是唯一的，即设为初始类中心规则；
3. else 计算优先级最高和最低的规则两两分组的 p 值，最小的一组设为初始类中心规则；
4. $ed\leftarrow\sqrt{\sum_{k=1}^{n}|\ x_{ik}-x_{jk}\ |^2}$ /* 得到剩余每个规则到 $\boldsymbol{D}_u$ 和 $\boldsymbol{D}_d$ 的欧式距离 */；
5. if $ed_{iu}\geqslant ed_{id}$，$\boldsymbol{C}_1\leftarrow\boldsymbol{D}_i$；
6. else $\boldsymbol{C}_2\leftarrow\boldsymbol{D}_i$；
7. 分别选择 $\boldsymbol{C}_1$ 和 $\boldsymbol{C}_2$ 中每个类别中的所有规则属性的均值组成新的类中心；
8. 计算 $E\leftarrow\sum_{i=1}^{n}\sum_{j=1}^{k}r_{ij}\parallel x_i-\mu_j\parallel^2$，循环步骤 4~7，直到 E 收敛不再变化；
9. return $\boldsymbol{C}_1$ 和 $\boldsymbol{C}_2$

2　规则动态分配算法设计

在 1.2 节中，采用 EasiKM 规则聚类算法将规则分为实时规则和无实时规则，本节主要讨论如何实现动态分配至云端与本地触发执行，以达到提高规则触发执行的实时性的目标。

定义 3　设规则库中规则数量为 M，本地共有 n 个规则服务端，每个本地规则服务端单位时间处理规则量为 $d_1,d_2,\cdots,\ d_{n-1}$，云端规则服务端单位时间处理规则量为 d_n。

设每个规则服务端分配规则数量为 $x_i,\ i=1,\ 2,\cdots,\ n$。$f_i(x_i)$ 为每个规则服务端执行 x_i 个规则所需的时间开销以及触发数据传输的时间总和，则系统总共的规则处理时间开销为

$$T=f_1(x_1)+f_2(x_2)+\cdots+f_n(x_n) \tag{5}$$

本文提出的规则分配算法，目的是要根据规则分类后得到的实时规则动态分配至每个规则服务端，并使得系统总的规则处理时间开销最小，以达到减少规则触发执行延迟，提高物联网规则引擎实时

性能的目标。根据上述所提出的问题,可转化为规则静态规划:

$$\begin{cases} \min T = f_1(x_1) + f_2(x_2) + \cdots + f_n(x_n) \\ x_1 + x_2 + \cdots + x_3 = n \\ 0 \leqslant x_i \leqslant M \quad i = 1,2,\cdots,n \end{cases} \tag{6}$$

可以将上式看成是一个多阶段的规则分配决策问题,利用动态规划的递推关系来求解最优分配方式。在应用动态规划的方法来处理规则分配问题时,可以把规则先分配给一个或者几个规则服务端的过程作为一个阶段,把分配给每个规则服务端的规则数量 x_i 作为分配决策变量,分配给第 k 个规则服务端至第 n 个规则服务端的规则总数量作为状态变量 s_k,则规则状态转移方程为

$$s_{k+1} = s_k - x_k \quad k = 1, 2, \cdots, n-1 \tag{7}$$

设满足分配规划的最优值函数 $g(s_k)$ 为 s_k 个规则分配给第 k 个至第 n 个规则服务端所得到的最小规则处理时间总开销,可以得到动态规则分配的逆推关系式为

$$\begin{cases} g_k(s_k) = \min\limits_{0 \leqslant x_k \leqslant s_k} \{f_k(x_k) + g_{k+1}(s_k - x_k)\} \\ g_n(s_n) = \min\limits_{x_n = s_n} f_n(x_n) \end{cases} \tag{8}$$

其中 $k = 1,2,\cdots, n-1$。

根据式(7),当 $x_n = s_n$ 时, $g(s_k) = f_n(x_n)$。由式(8)可知,根据 $g(s_k)$ 与 $f(x_k)$ 的逆推关系,从第 n 个规则服务端根据规则处理时间开销最小的原则逆向往前推断,直到每个规则服务端执行 x_i 个规则所需的时间开销 $f_i(x_i)$ 达到最小,最后可得到每个规则服务端分配到的规则子集 Ri, $i = 1, 2, \cdots, n$。

根据上述所得到的规则子集,便确定了每个规则服务端的规则量,同时考虑到本地每增加一个规则,便可减少一次触发数据和执行控制信息往返云端的网络延迟,并且本地处理能力不如云端高效,所以,计算得到的 T 值需要再加上分配到云端的规则因触发和执行所带来的网络延迟时间,最后得到的规则处理时间开销为 T'。再根据与云端集中处理所有规则的时间(包括网络延迟) t 做对比,当 $T' \geqslant t$ 时,根据式(5)重新计算 T 值,使得 $T' < t$,便得到每个规则服务端的最优规则子集。规则动态分配算法流程如算法 2 所示。

本文提出的端云动态规则分配方法,相比于现有方法,结合了实际部署环境的网络传输延迟,充分发挥了云端集中管理方法的高性能计算能力和本地集中管理方法就近处理的高效性优势,最大程度上保障了实时规则能够及时触发执行,提高了物联网规则引擎的实时性与可靠性。

算法 2 规则动态分配算法

输入: D//输入实时规则集,规则数为 M;
输出: Ri//每个规则服务端的规则子集;
1. for 规则服务端
2. 初始化每个规则服务端;
3. 得到 d_i 与 $f_i(x_i)$;
4. end for
5. $T \leftarrow f_1(x_1) + f_2(x_2) + \cdots + f_n(x_n)$;//处理规则时间开销
6. $s_k \leftarrow x_k + x_{k+1} + \cdots + x_n$;
7. for from k to n
8. $G_k = f_k(x_k) + f_{k+1}(x_{k+1}) + \cdots + f_n(x_n)$;
9. $g(s_k) \leftarrow$ 取 G_k 最小值;//$g(s_k)$ $g(s_k)$ 为 s_k 个规则分配给第 k 个至第 n 个规则服务端所得到的最小规则处理时间总开销
10. end for
11. 根据逆推关系式(8),得到 T 的最小值 i;
12. $t \leftarrow$ 云端集中处理规则的时间开销 + 网络延迟;
13. if (i + 云端集中处理网络延迟) $\geqslant t$
14. 返回步骤 7 重新计算 T;
15. else return 取得 T 值时对应的 Ri

3 实验与性能分析

本节对本文提出的基于规则实时特性的端云动态分配方法 EasiDEP 进行实验验证。

3.1 实验数据与平台搭建

为了 EasiDEP 的有效性,本文基于《煤矿安全规程(2016)》与合作煤矿的 KJ90X 煤矿安全监控系统[15]提取了 1000 个在煤矿中常见的安全监控规则数据集,以煤矿中的气体和温湿度传感器作为触发条件,通风设备(风机、风窗)作为规则执行设备,例如规则“CO 含量大于 50 ppm,打开风窗通风”,CO 传感器采集的气体浓度作为规则的 if 触发条件,风窗开、关作为 then 的执行信息。

本文基于个人电脑(personal computer, PC)机、网关设备、数据采集执行设备搭建了实验验证平台,配置信息如表 1 所示。云端规则服务端(以下简称

“云服务端”)部署到 PC 机上,网关设备作为本地与云端的数据转发中心并组建本地局域网络,将数据采集执行集设备上的 CH_4、CO 和 O_2 传感器数据作为规则触发数据,蜂鸣器作为规则执行的控制设备。

表 1　实验平台配置信息表

设备	硬件配置
PC 机	处理器为 Intel Corei3-8100 CPU @3.60 GHz 内存总容量 16 GB 硬盘总容量 1 TB
网关设备	主控芯片为 AR9344
数据采集执行设备	主控芯片为 STM32F429VGT6

如图 2 所示,本文基于 PC 机、网关设备和 2 个数据采集执行设备(以下称为“设备 1”和“设备 2”)搭建了实验硬件平台。数据采集执行设备实物图如图 3 所示。当云端规则服务端集中管理规则时,设备 1 采集到触发数据,利用 WiFi 网络和网关设备经传输网络送至云端服务端处理,规则执行信息再经传输网络、网关设备和 WiFi 网络送至设备 2 按规则信息执行控制指令。当云端和本地同时管理规则时,云端规则服务端的工作方式与上述一致,而设备 1 和设备 2 互相作为彼此的规则触发数据源和执行设备。

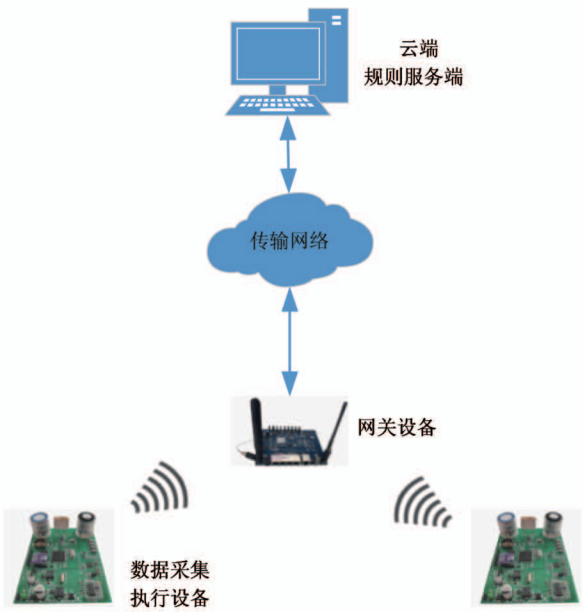


图 2　实验平台结构图

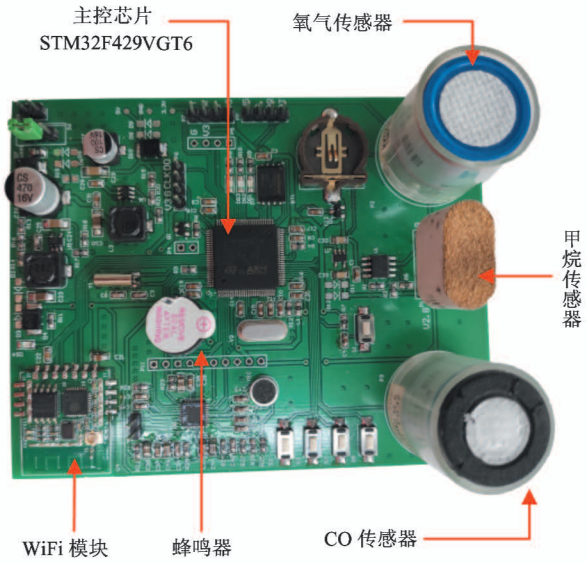


图 3　数据采集执行设备实物图

3.2　规则聚类结果分析

为了验证 EasiKM 算法的正确性,本小节根据上述的规则样本数据集对算法进行了测试。对于实验结果,采用准确率(precision)、ARI(adjusted rand index)系数和 FMI(fowlkes and mallows index)来评估实验结果。

准确率表示正确的聚类规则数与总规则数的比例,计算公式如下:

$$ACC = N_c / N \tag{9}$$

式中,N_c 表示正确聚类的规则数,N 表示规则样本总数。

$$RI = (a+b) / C_n^2 \tag{10}$$

式(10)中,RI 兰德指数(Rand index, RI)取值范围为[0,1],值越大表示规则聚类的准确性越高,规则聚类结果与真实情况越吻合。假定 C 为实际规则类别信息,K 为规则聚类结果,a 表示在 C 与 K 中都是同类规则的元素对数,b 表示 C 与 K 中都是不同类规则的元素对数,C_n^2 表示规则样本组成的总对数。为了实现在规则聚类结果随机产生的情况下指标应该接近 0,提出了调整兰德系数 ARI,以具有更高的区分度,取值范围为[-1,1],同样是值越大意味着规则的聚类结果更接近实际情况。

$$ARI = (RI - \mathrm{E}[RI]) / (\max(RI) - \mathrm{E}[RI]) \tag{11}$$

在知道规则样本数据的真实类别信息的情况

下,可以用 FMI 指标对规则聚类算法进行评估,FMI 定义了准确率和召回率(recall)的几何平均数。

$$FMI = TP/\sqrt{(TP+FP)(TP+FN)} \tag{12}$$

式中,TP(true positive)为对的样本聚类为正样本,FP(false positive)为错的样本聚类为正样本,FN(false negative)为错的样本聚类为负样本。

为了评估本文提出的规则聚类算法的有效性,采用 K-means 算法作为规则聚类的对比方法,该算法是目前被广泛应用的无监督聚类算法之一[16]。规则聚类结果和评估指标如图 4 和图 5 所示。

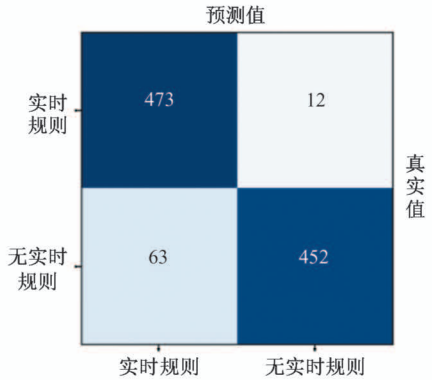
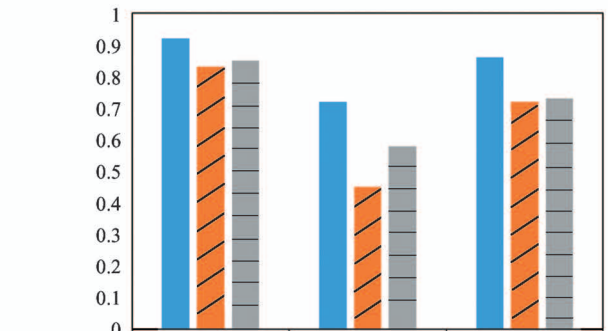


图 4　EasiKM 规则聚类结果



	ACC	ARI	FMI
EasiKM	0.92	0.72	0.86
传统Kmeans	0.83	0.45	0.72
文献 [17]	0.85	0.58	0.73

图 5　评估指标汇总

由图 4 可知,分别有 485 个实时规则和 515 个无实时规则组成的 1000 个规则样本数据通过 EasiKM 聚类后,得到 536 个实时规则和 464 个无实时规则,规则聚类准确率达到了 92%,能够根据规则实时性有效划分规则样本集。文献[17]针对传统 K-means 随机选取聚类中心点的缺点,通过增加新的变量密度方法来确定初始类中心,提高聚类的准确率。为了验证 EasiKM 的有效性,本文在图 5 中对比了传统 K-means 和文献[17]的改进 K-means 算法的结果。从图 5 的数据可知,EasiKM 在 ACC、ARI 和 FMI 指标上都高于对比的方法,其中 ARI 和 FMI 相对于对比方法的结果更接近于 1,表示规则样本集通过 EasiKM 聚类后的结果与实际情况更接近,能够有效地将规则库中的规则根据实时性进行规则聚类。

3.3　规则动态分配性能分析

本节根据 3.1 节中的规则聚类结果实现实时规则的端云动态分配方法。由图 4 可知,根据规则聚类结果得到 536 个实时规则,再采用 EasiDEP 规则分配算法实现端云动态分配机制,减少因为网络延迟和本地计算存储资源受限引起的规则延迟触发执行。

根据 3.1 节中搭建的实验平台,测得规则触发数据在传输网络中的延迟为 5 ms,设备 1 和设备 2 分别距离网关设备 2 m 和 10 m,并且中间都有遮挡,测得无线数据传输的网络延迟分别为 3 ms 和 10 ms。通过调整设备 1 和设备 2 主控芯片的运行频率得到平均规则触发执行时间分别为 1 ms 和 5 ms,云服务端规则触发执行时间小于 1 μs,实时规则样本数只有 536 个,故可以不计云服务端的规则触发执行时间。同时,为了模拟本地设备资源受限的情况,设置设备 1 和 2 最多只能管理 200 个规则,云服务端没有限制。为了进一步验证 EasiDEP 的有效性,采用文献[7]的网关管理方式分析了规则触发执行时间,本文选用的网关设备规则触发执行平均时间为 2 ms。根据上述参数,通过 EasiDEP 计算后得到云端和本地分配到的实时规则数量与总的规则触发执行时间,如表2所示,云服务端、设备1和设

表 2　规则分配与执行结果

指标 / 方法	规则总数	云服务端	设备 1	设备 2	执行时间
EasiDEP	536	268	200	68	8.6 s
集中管理方法	536	536	—	—	9.6 s
文献[7]	536	—	—	—	8.6 s

备 2 分别分配到了 268、200 和 68 个规则，总执行时间为 8.6 s，并且集中式方法的总执行时间为 9.6 s，本文提出的方法比传统方法减少了 1 s 的延迟，与文献[7]的执行时间相同。

在上述实验的基础上，调整设备 2 的规则处理时间为 2 ms，通过 EasiDEP 重新分配规则后得到，如表 3 所示，云服务端、设备 1 和设备 2 分别分配了 136、200 和 200 个规则，总执行时间为 8 s，比传统云集中方法减少了 1.6 s，并且相比文献[7]的方法减少了 0.6 s。实验表明，EasiDEP 能够根据网络延迟时间和本地设备性能变化动态分配规则，规则执行时间小于云端和本地集中管理方法。

表 3 规则分配与执行结果

指标 / 方法	规则总数	云服务端	设备 1	设备 2	执行时间
EasiDEP	536	136	200	200	8.0 s
集中管理方法	536	536	—	—	9.6 s
文献[7]	536	—	—	—	8.6 s

本文实验数据来源于煤矿，减小安全控制规则触发执行的延迟时间，对整个煤矿的安全生产能够起到关键性作用。根据文献[18]可知，瓦斯（主要成分为 CH_4）爆炸是需要一定的引火延迟时间才能形成的一种极其复杂的化学反应过程，这段时间称为瓦斯爆炸感应期，火焰与特定浓度瓦斯的接触时间必须大于爆炸感应期才能产生瓦斯爆炸。不同浓度的瓦斯和火焰温度，爆炸感应期也不相同，例如当瓦斯浓度达到 7%，火焰温度为 1175 ℃时，爆炸感应期为 0.01 s。煤矿数据传输网络会因为复杂的矿井环境产生不同程度的延迟，影响到瓦斯安全控制规则的实时触发执行。例如规则“CH_4 浓度大于 1.2%，关断电源”由于延迟触发并执行，就很有可能因为在瓦斯爆炸感应期内没有及时采取控制措施，导致煤矿发生瓦斯爆炸事故，造成无法挽回的人员伤亡和财产损失。根据上述实验结果，在煤矿安全监控系统中采用本文提出的 EasiDEP 动态规则分配方法，能够有效地减小煤矿中涉及到安全控制规则触发执行的延迟时间，给煤矿的安全生产过程多份保障。

综上所述，本文基于规则实时性提出的 EasiDEP 规则动态分配方法，能够有效地减少规则触发执行过程中的延迟，提高系统的实时性，降低因为规则延迟触发执行产生的控制安全事故的发生率，保障系统的安全性。

4 结 论

针对物联网控制规则在实际应用中存在的延迟触发执行问题，本文基于规则实时特性提出了一种端云动态分配方法 EasiDEP。该方法通过规则聚类算法 EasiKM 将规则划分为实时规则和无实时规则，再结合传输网络延迟时间、云端和本地规则服务端的规则处理能力和性能，实现规则服务端之间的规则动态分配机制，充分发挥云端和本地规则服务端的性能。实验表明，EasiDEP 能够有效地减小物联网控制规则触发执行的延迟时间，降低系统因为规则延迟触发执行引发的控制安全风险。未来的研究将会进一步补充不同场景的实验数据和设备，验证并改进 EasiDEP 的有效性和性能，以适用多种不同的应用场景。

参考文献

[1] Puranik V, Sharmila, Ranjan A, et al. Automation in agriculture and IoT[C]//International Conference on Internet of Things: Smart Innovation and Usages, Ghaziabad, India, 2019: 1-6

[2] Shah T, Venkatesan S, Ngo T, et al. Conflict detection in rule based IoT systems[C]//IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference, Vancouver, Canada, 2019: 276-284

[3] Kargin A, Petrenko T. Internet of Things smart rules engine[C]//IEEE International Scientific-Practical Conference Problems of Infocommunications, Kharkiv, Ukraine, 2018: 639-644

[4] Heo S, Song S, Kim J, et al. RT-IFTTT: real-time IoT framework with trigger condition-aware flexible polling intervals[C]//IEEE Real-Time Systems Symposium, Paris, France, 2017: 266-276

[5] Xu P, Su J W, Zhang Z B. Distributed hybrid cloud management platform based on rule engine[C]//IEEE

International Conference on Cloud Computing, San Francisco, USA, 2018: 836-839

[6] Zhang J D, Yang J X, Li J. When rule engine meets big data: design and implementation of a distributed rule engine using spark[C]//IEEE 3rd International Conference on Big Data Computing Service and Applications, San Francisco, USA, 2017: 41-49

[7] 田瑞琴, 吴尽昭, 唐鼎. 物联网网关中轻量化规则引擎的设计与实现[J]. 计算机应用, 2015, 35(4): 1035-1039

[8] Kiran S, Rajalakshmi P, Krishna B, et al. Adaptive rule engine based IoT enabled remotehealth care data acquisition and smart transmission system[C]//IEEE World Forum on Internet of Things, Seoul, Korea, 2014: 253-258

[9] Lan L N, ShiR S, Wang B, et al. A universal complex event processing mechanism based on edge computing for Internet of Things real-time monitoring[J]. *IEEE Access*, 2019, 7: 101865-101878

[10] Kaed C E, Khan I, Berg A V D, et al. SRE: semantic rules engine for the industrial Internet of Things gateways [J]. *Transactions on Industrial Informatics*, 2018, 14 (2): 715-724

[11] 池深启. 轻量级规则引擎 Drools 在物联网平台中的应用研究[D]. 杭州:浙江工业大学电子与通信工程学院, 2017: 5-15

[12] Sun Y, Wu T Y, Li X M, et al. A rule verification system for smart buildings[J]. *IEEE Transactions on Emerging Topics in Computing*, 2017, 5(3): 367-379

[13] Kargin A, Petrenko T. Knowledge representation in smart rules engine[C]//International Conference on Advanced Information and Communications Technologies, Lviv, Ukraine, 2019: 231-236

[14] 黄晓辉, 李栋, 石海龙, 等. EasiRCC:一种针对智能家居的规则匹配和冲突消除方法[J]. 计算机研究与发展, 2017, 54 (12): 2711-2720

[15] 中华人民共和国应急管理部. AQ6201-2019 煤矿安全监控系统通用技术要求[S]. 北京:应急管理出版社, 2019

[16] 李松, 谢新新, 刘东林, 等. 基于 Dirichlet 过程的无线视频码率变化识别算法[J]. 高技术通讯,2016, 26 (10): 833-840

[17] 刘建花. K-means 聚类算法的改进与应用[J]. 太原师范学院学报, 2020, 19(1): 81-83

[18] 李笑堃. 煤矿井下主动抑爆隔爆系统技术研究[D]. 太原:中北大学研究生院, 2016: 43-46

Research on dynamic end-cloud allocation method based on real-time rules

Huang Xiaohui[*,**], Cui Li[**], Huang Xi[**]

([*] University of Chinese Academy of Sciences, Beijing 100049)

([**] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

Abstract

Aiming at the control rules of the Internet of Things in the practical application, the Internet of Things system causes different degrees of control security accidents due to the delayed trigger execution. Based on the real-time characteristics of the rules, this paper proposes an end-cloud dynamic allocation method (EasiDEP) for the control rules of the Internet of Things system, so as to improve the real-time performance of the control rules of the Internet of Things system. Firstly, according to the correlation between rules and real-time characteristics, a rule clustering algorithm is proposed, which divides the rules in the rule base into real-time and non-real-time rules. Then, based on the result of rule clustering, a dynamic rule allocation method is proposed to distribute real-time rules to the cloud and local rule server for simultaneous processing, so as to improve the execution rate of real-time rules. Finally, the experiment verifies that EasiDEP can effectively improve the real-time trigger execution rate of Internet of Things control rules and reduce the risk of control security accidents.

Key words: control rules of Internet of Things, real-time characteristic, control security, dynamic allocation, rule clustering