# 面向车联网应用的数据关联性任务调度算法

丁 男[1),2),3)] 聂率航[1),2),3)] 许 力[3)] 谭国真[1),2)]

[1)](大连理工大学计算机科学与技术学院 辽宁 大连 116023)

[2)](辽宁省物联网与协同感知工程技术研究中心 辽宁 大连 116023)

[3)](软件架构国家重点实验室(东软集团股份有限公司) 沈阳 110179)

**摘 要** 多核系统的广泛应用提高了任务的并发性,同时也带来了任务核间通信这一额外开销.对于具有数据关联性的任务,核间通信会极大地影响任务的调度长度和实时性.结合车联网多任务混合并发的应用特点,针对多核系统中任务响应实时性问题,该文提出了一种面向车联网应用的数据关联性任务调度算法(DTSV).首先,根据车联网协议标准中针对车联网应用相关的三类数据及其特性进行了描述与分析.车联网应用中任务被分为安全关键类、交通效率类和安全无关类,每类任务都包含大量参数.有些参数并不仅存在于一个任务中,而会同时被多个任务所应用.因此,在车联网中,任务之间存在着大量的数据关联性.基于常用关联性任务模型以及车联网中任务特性,定义了一种基于车联网应用的数据关联性模型.其次,根据任务相关性模型给出了任务关联性评价函数,该评价函数的建立主要依据与计算型任务有关的所有通信型任务,生成一个多维的向量,以表示任务与内核中任务之间的强弱关联关系.再次,根据上述关联性模型和评价函数设计了基于此评价函数的关联性任务调度算法,通过将数据关联性较强的任务分配到同一个内核,以减少任务执行过程中核间通信量.算法分为初始化阶段和运行阶段.算法的初始化阶段主要解决了车联网系统启动时大量周期性任务的分配问题,能够明显地减少周期性任务的周期调度长度.算法的运行阶段主要解决了车联网系统运行中随机产生的非周期性任务的分配问题,考虑到非周期性任务的特性,算法能够在一定程度上提高其实时性.同时,在算法的运行阶段,通过对非周期性任务的数据关联的预处理,更进一步提高了非周期性任务的实时性.最后,通过实验将 DTSV 与传统多核任务调度算法做出了比较,结果显示 DTSV 平均能够缩短 10.6%整体任务调度长度,同时非周期性任务的响应时间平均能够减少 33.5%.实验证明,DTSV 相对于传统多核调度算法,针对具有数据关联性的周期性任务以及非周期性任务都能有效地降低其核间通信延时,缩短任务调度长度,提高任务响应实时性.

**关键词** 资源分配;多核系统;数据关联性;车联网;任务调度

**中图法分类号** TP301 **DOI 号** 10.11897/SP.J.1016.2017.01614

# Data Related Task Scheduling for Vehicular Ad Hoc Networks

DING Nan[1),2),3)] NIE Shuai-Hang[1),2),3)] XU Li[3)] TAN Guo-Zhen[1),2)]

[1)](*School of Computer Science and Technology, Dalian University of Technology, Dalian, Liaoning* 116023)

[2)](*Liaoning Engineering Technology Research Center of IoT and Cooperative Sensing, Dalian, Liaoning* 116023)

[3)](*State Key Laboratory of Software Architecture (Neusoft Corporation), Shenyang* 110179)

**Abstract** The extensive application of multi-core system improves the concurrency of tasks, however, brings the extra cost of inter-core communication. Inter-core communication will significant affect the schedule length and real-time of data related tasks. Aiming at the problem of real-time task response in multi-core processor system, this paper proposes a data related task scheduling

收稿日期:2016-11-07;在线出版日期:2017-04-21.本课题得到国家自然科学基金项目(61471084)、国家"八六三"高技术研究发展计划项目(2012AA111902)、软件架构国家重点实验室开放课题基金(SKLSAOP1602)资助.**丁 男**,男,1978 年生,博士,副教授,中国计算机学会(CCF)会员,主要研究方向为实时嵌入式系统设计、物联网及车联网应用研究等.E-mail:dingnan@dlut.edu.cn.**聂率航**,男,1993 年生,硕士研究生,主要研究方向为车联网、嵌入式系统设计等.**许 力**,男,1982 年生,博士,高级工程师,中国计算机学会(CCF)高级会员,主要研究领域为网络虚拟化、云计算及概率图模型理论等.**谭国真**,男,1960 年生,博士,教授,博士生导师,中国计算机学会(CCF)高级会员,主要研究领域为物联网、车联网、智能交通控制等.

algorithm (DTSV) for Vehicular Ad Hoc Networks (VANETs), with considering the multi-task concurrency mechanism in VANETs. Firstly, three types of data in Vehicular Ad Hoc Networks applications and their features are described and analyzed according to the vehicular networks protocol standards. In vehicular networks protocol standards, tasks of vehicular networks applications are divided into three types, Critical Safety, Traffic Efficiency and Non-Safety. Every type of task consists of a lot of parameters. In these parameters, some of them are not included in only one task, but included in several tasks. So it can be inferred that tasks of vehicular networks applications are related by data correlation. Based on the common data related task model and the task characteristics in vehicle networks, a related task model based on data correlation of vehicular networks applications is defined. Secondly, an evaluation function is proposed according to the related task model. The evaluation function is mainly constructed by the communicating tasks of computing task. And then a multidimensional vector is gotten from the evaluation function, to characterize the strong and weak correlation between tasks. Furthermore, a data related task scheduling algorithm based on the related task model and evaluation function is designed to reduce the inter-core communications during task execution by assigning tasks which have deep correlation in data into a same core. There are two phases for the algorithm, initialization phase and running phase. The initialization phase of the algorithm works at the start time of VANET system, to assign periodic tasks which generated at the start time. And the initialization phase can significantly reduce the schedule length of periodic tasks. The running phase of the algorithm works at the running time of VANET system, to assign aperiodic tasks which generated at the running time. It can marginally improve the real-time of aperiodic tasks with considering features of aperiodic tasks. Meanwhile, the data correlation of aperiodic tasks is pre-processed at the running time. After that, the real-time of aperiodic tasks can be further improved. Finally, the experimental and simulation is conducted to compare DTSV with other multi-core task scheduling algorithm. According to the result of the experimental, DTSV can reduce the schedule length by 10.6%. It is a considerable optimization for tasks which are continuously executing. The response time of aperiodic tasks can be reduced by 33.5% averagely. Aperiodic tasks generally require high real-time in vehicular networks. And the decrease of response time can significantly improve the safety of vehicular networks. The results of the experimental show that DTSV can effectively reduce the inter-core communication delay, shorten the task schedule length, reduce response time of tasks and improve the real-time of task response to data related periodic tasks and aperiodic tasks.

**Keywords**　resource allocation; multi-core system; data related; Vehicular Ad Hoc Networks; task scheduling

## 1　引　言

车联网是由运行在交通道路上的拥有感知、计算、存储、和无线通信能力的移动车辆及其周边的基础通信单元而组成的新型自组织网络，是提高交通安全驾驶的有效手段，同时也为出行娱乐等交通增值服务提供了有效平台[1].

随着车联网应用场景的不断复杂化，车联网系统中需要处理的任务数量不断增加，多核技术已成为提升车联网应用性能的有效手段. 为满足车联网多任务并存的应用需求，降低任务响应延时，多核处理器系统是目前普遍采用的硬件系统方案. 例如欧洲 PROTON-PLATA 项目采用 2 核处理器设计车载通信设备[2]，Mobileye 采用 4 核处理器.

根据应用特点，设计有效的任务调度算法是保证多核处理器系统中多任务并发及提高其响应性能的关键[3,4]. 多核处理器可以使多个任务并发执行，增加任务吞吐率，但同时也带来了任务核间通信这一额外开销[5]. 经统计，任务执行过程中核间通信造

成的任务响应延时是核内通信造成任务响应延时的3～10倍[6-7].在汽车行驶过程中,如果安全应急任务响应延时降低100 ms,可以避免90%的交通事故发生[8].车联网系统中任务的执行大多依赖于传感器对信息的捕获以及对信息的后期处理.因此,任务之间大都具有一定的数据关联性,随之产生大量额外的任务核间通信开销,极大影响了车联网任务的实时性.如何调度车联网系统中的任务,使其中具有数据关联性的任务尽量分配到同一个内核中,增加任务的核内通信,减少核间通信开销,提高任务的实时性,是保障车联网安全性的重要研究方向.

本文针对以上目标,依据关联性任务中常用的任务模型,提出了一个判断任务间数据关联性强弱的评价函数,同时提出了依据数据关联性对任务进行调度的算法(DTSV).通过实验验证了DTSV算法可以有效地将具有数据关联性的任务尽量分配到同一个处理器内核中,利用任务执行过程中的核内通信替代核间通信,提高了任务的响应实时性.

## 2 相关工作

在多核系统中,数据关联性任务执行时间主要受任务的计算时间和处理器间的通信开销这两个关键因素所影响.计算处理器间通信开销时,必须在考虑核内通信开销的同时,将核间通信的开销也考虑进去.在多核系统任务运行的实际情况中,核内通信开销与核间通信开销存在着明显差异.其中在核内的通信过程中,在同一个核内的任务中,其相关信息在同一本地的缓存中保存,通信任务大部分只需要通过核内的高速缓存进行,所以任务之间数据交换的时间很短,其通信开销可忽略.而核间通信任务则需要利用核间的缓存进行,由于核间通信中间渠道的使用,难以避免产生延时,故其通信开销不能忽略不计.因此数据依赖关系越大的两个任务若是被分配到两个核内去执行,那么通信开销将会更大,所以在进行任务调度时,考虑任务之间的关联性就显得非常重要.

而在车联网系统中,关联性任务又可以分为周期性任务和非周期性任务.周期性任务是指按一定周期达到需求并请求运行,每次请求称为任务的一个任务实例,任务实例所属任务的起始时刻称为该任务实例的到达时刻,任务实例被置为就绪态的时刻称为该任务实例的释放时刻.而非周期性任务是指随机到达系统的任务[9].因此对于车联网系统来说,任务的调度不仅要考虑到任务的数据关联性,还要考虑到任务的执行特性.

针对于多核处理器的关联性任务调度目前已有一些研究.2007年,Xu等人[10]提出了一种多核系统中提高任务运行效率的方法,主要思想是同时考虑并行处理和流水线的应用,以便对任务进行更加合理的分配.同年,Chen等人[11]提出了一种动态的任务分配和调度方法,以解决多核系统中任务的映射和执行顺序问题.但这两项研究主要是通过增加任务执行的并行性来提高执行效率的,均忽略了核间通信的开销.

Wang等人[12]提出了一种可以很好地消除流媒体应用程序中核间通信的方法.主要思想是将计算型和通信型任务进行重新调度,使得计算型任务和通信型任务重叠执行,这样就可以很好地消除通信型任务带来的开销,也就达到了消除核间通信的目的.该文献算法比较复杂,在将任务进行重新调度时,还进行了任务可调度性的分析,并提出了一个整数线性规划(ILP)方法以生成最优目标任务调度.该算法是基于流媒体应用程序的周期性任务的调度问题,可以达到消除核间通信的目的.2015年,朱怡安等人[13]提出简单树(Simple Tree)模型和越短越优先调度方法,在维护周期性任务的依赖性的同时,提高了系统利用率,降低了死限丢失率.不过这两篇文献只考虑了周期性关联任务的调度问题,并没有将非周期性任务考虑进去.

本文提出一种针对与车联网任务特点的任务调度算法,将任务的内核分配分为两个阶段,有效地解决了车联网系统运行过程中的周期性任务和非周期性任务的内核分配的问题,降低了具有数据关联性的任务间的核间通信,减少了整体任务调度长度,提高了非周期性任务的实时性.

## 3 数据关联性模型的设计

WAVE协议标准定义了三类车联网应用:安全关键类任务(Critical Safety)、交通效率类任务(Traffic Efficiency)、安全无关类任务(Non-Safety).其中具体所涉及到的任务和参数如表1所示[14].

可以看出,同一类任务中,涉及到的参数有多种,并且多个任务涉及到同一个参数的可能性非常大,而涉及到同一个参数的多个任务既可能出现多任务并发执行的情况,也可能出现在具有前驱后继关系的任务中.

**表 1　车联网任务参数类别**

| 分类 | 应用 | 主要涉及参数 |
|---|---|---|
| 安全关键类<br>(Critical Safety) | 并线协助，提前感知碰撞，弯道速度警告，紧急制动警告，碰撞预警，左转协助，车道变换警告，“停车”标志协助等 | 车辆特征，速度，加速度，温度，方向，位置，预警信息，车辆路径预测信息，公路几何机构等 |
| 交通效率类<br>(Traffic Efficiency) | 旅途规划，路径选择，避免堵车，自动巡航，查找停车位，避免交通事故发生点等 | 导航信息，全局公用时钟，交通密度，行程时间，停车位信息，城市实时交通状况，指示灯状况，驾驶员偏好等 |
| 安全无关类<br>(Non-Safety) | 通信交流、服务公告、信息娱乐、支付服务、互联网应用等 | 文字信息，图片信息，语音信息，视频信息，网页文件等 |

在车载系统中，由于许多任务会使用到相同的参数，为了便于同其它车载系统进行通信，使用到了消息调度器来协调运行在车辆上的所有任务的数据交换请求，如图 1 所示[15]. 消息调度器可以将所有任务所包含的参数合并为一个最小集合，减少与其它车载系统间的通信量.

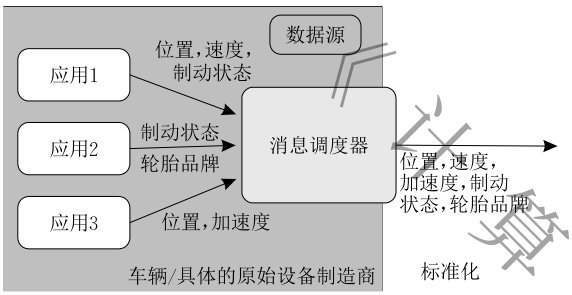


图 1　车联网消息调度器

因此，车联网系统中的任务并不是相互独立的，而是具有一定的数据关联性. 在本节，本文针对任务间这一数据关联性的特性，提出了系统和任务所具有的模型，并以此任务模型为基础，提出了一个判断任务间数据关联性强弱的评价函数.

### 3.1　系统模型

由于目前片上处理系统的所遇到的瓶颈，单核处理器的性能很难得到明显的提升，基于多核处理器的硬件平台已成为趋势. 本文所使用的基于总线的体系结构如图 2 所示，包括 $\{P_0, P_1, \cdots, P_{M-1}\}$ $M$ 个内核，一条共享总线以及总线仲裁. $M$ 个内核通过接口与共享总线连接，总线仲裁允许在某一时刻，只能由一个内核通过总线传送数据，内核之间的任务通信由总线完成.

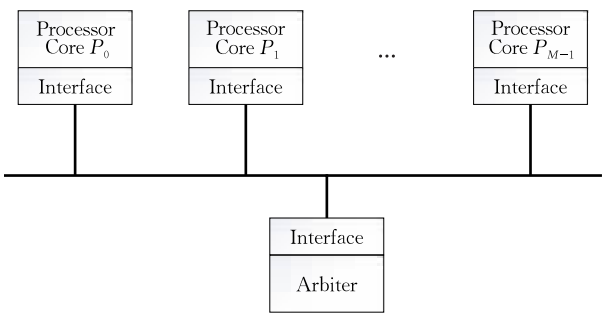


图 2　系统模型

### 3.2　任务模型

本文所描述的任务环境为车联网的应用场景，在车联网系统中，在启动并完成初始化后，其执行的大部分任务为周期性任务，比如图像处理模块以及传感器数据采集和分析模块. 而当车辆遇到突发状况或者接收到相关指令时，会产生非周期性任务. 且这些任务间大多有数据方面的关联性，即一个任务的执行依赖于其前驱任务的相关数据. 只有前驱任务执行完成并传输完相关数据后，这个任务才可以开始执行.

基于上面关于任务场景的描述，本文给出任务模型的以下定义.

**定义 1.**　数据关联性任务图. 本文使用有向无环图(DAG 图)来描述关联性任务，如图 3 所示，整个任务图 $G=\{V,E\}$，包括节点集合 $V$，每个节点表示一个计算型任务，边集合 $E\subset V\times V$，每条边表示一个通信型任务，任务之间有边相连表示两个任务间存在数据关联性.

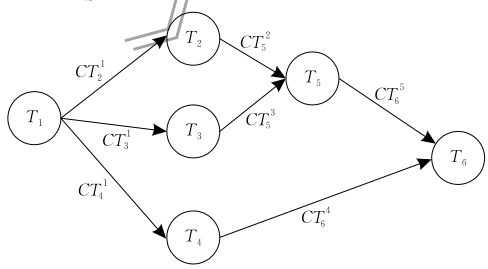


图 3　任务模型

**定义 2.**　计算型任务. 数据关联性任务图 $G$ 中的每个节点代表一个计算型任务，记为 $T_i$，表示第 $i$ 个计算型任务. 每个计算型任务需要分配到处理器内核中执行，其执行时间记为 $e_i$.

**定义 3.**　通信型任务. 数据关联性任务图 $G$ 中的每条边代表一个通信型任务，记为 $CT_j^i$，表示计算型任务 $T_i$ 与 $T_j$ 之间存在数据关联性，需要通信型任务为两者之间传递数据，计算型任务可以开始执行的必要条件为其所有前驱通信型任务执行完毕. 通

信型任务需要一定的总线资源来执行，当具有数据关联性的计算型任务被分配在不同的处理器内核时，这一任务间通信为核间通信，其执行时间为 $e_j^i$；而当具有数据关联性的计算型任务被分配在同一个处理器内核时，任务间通信为核内通信，其执行时间要远远小于核间通信，在本文中假定其执行时间为 0.

**定义 4.** 周期调度长度. 对于一个周期任务簇，其执行一个周期所用的时间长度为周期调度长度，记为 $I$. 如图 4 所示的是图 3 中的任务簇在处理器内核中周期执行的状态.

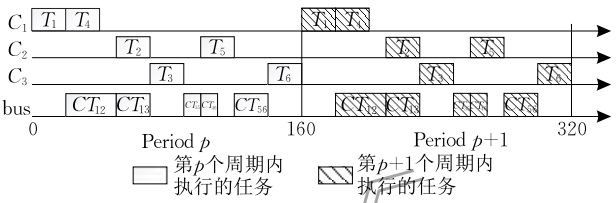


图 4 周期调度长度

### 3.3 基于数据关联性的评价函数

由于任务间的数据关联性，一个计算型任务的执行必须在等待与其相关的前驱通信型任务全部完成时才能开始，而由于任务的核内通信要远远快于核间通信，因此对于任务的内核分配便显得尤为重要，合适的任务内核分配能够有效减少任务的核间通信，降低任务整体的调度长度. 特别对于周期性任务来说，其在完成内核分配后，会在所在内核中周期地执行，周期性任务簇的周期调度长度会极大地影响整体的调度长度. 对于非周期性任务来说，由于可能与其他任务存在数据关联性，内核的分配会影响到核间通信量的大小，又因为计算型任务的执行受到通信型任务执行的限制，非周期性任务的内核分配会很大程度上影响到其任务响应时间.

在此，本文给出一个基于数据关联性的任务内核分配的评价函数，作为数据关联性任务内核分配的主要依据，目的在于最大程度上降低任务间的核间通信.

如图 5 所示，对于待分配任务 $T_p$，其可能与多个任务间存在数据关联性，即存在与之相关的多个通信型任务，评价函数如式(1)所示.

$$R_x(T_p)=\sum_{i=1}^{m}\sigma_{ui}e_p^{ui}+\sum_{j=1}^{n}\sigma_{vj}e_{vj}^{p} \tag{1}$$

式(1)中，对于一个有 $M$ 个内核的处理器来说，$x$ 的取值为区间[0,$M-1$]的整数，评价函数表示的是任务 $T_p$ 相对于内核 $x$ 的数据关联性强度. $\{e_p^{u1},e_p^{u2},\cdots,e_p^{um},e_{v1}^{p},e_{v2}^{p},\cdots,e_{vn}^{p}\}$ 表示的是与任务 $T_p$ 相关的所有通信型任务 $\{CT_p^{u1},CT_p^{u2},\cdots,CT_p^{um},CT_{v1}^{p},CT_{v2}^{p},\cdots,CT_{vn}^{p}\}$ 的执行时间；参数 $\sigma$ 表示的是与任务 $T_p$ 具有数据关联性的计算型任务是否被分配在内核 $x$ 中，$\sigma$ 的取值为 0 或 1，例如 $\sigma_l$ 表示的便是计算型任务 $T_l$ 是否被分配在内核 $x$ 中. 若任务 $T_l$ 已被分配在内核 $x$ 中，则 $\sigma_l$ 取 1；若任务 $T_l$ 被分配在其他内核或尚未分配，则 $\sigma_l$ 取 0.

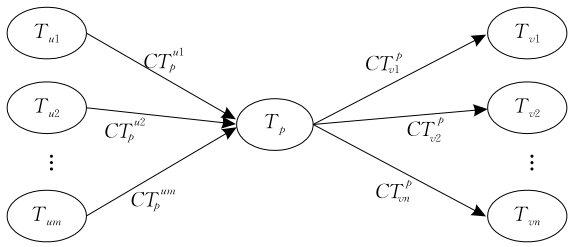


图 5 待分配任务数据关联性

由于本文主要研究的处理器为多核处理器，对一个待分配任务来说，通过式(1)计算出的值具有多个，可得到一个向量，定义如下.

**定义 5.** 关联度向量. 对于待分配任务 $T_p$，若处理器的内核数为 $M$，可以通过式(1)获得一个关于任务 $T_p$ 的 $M$ 维向量 $\boldsymbol{R}(T_p)=(R_0(T_p),R_1(T_p),\cdots,R_{M-1}(T_p))$，称其为关联度向量，表示待分配任务 $T_p$ 相对于每个内核的数据关联性强度.

## 4 基于数据关联性的任务调度算法

### 4.1 基于数据关联性的调度算法

利用上文中提出的数据关联性评价函数，考虑到本文提出一个应用于混合任务场景的算法，完成这一混合场景中任务的内核分配. 由于本文所研究的混合任务场景的特殊性，算法主要分为两个阶段：初始化阶段和运行阶段.

对于车联网来说，关乎车辆安全的任务会在系统启动时便开始执行，这些任务主要为表 1 中安全关键类的任务，比如传感器信息采集、碰撞语境等. 此时系统需要处理的是大量待分配的周期任务簇，这些周期性任务大多数会周期性地执行到系统关闭，为了保证整体调度长度尽可能地短，并且在一定程度上保证内核的负载均衡，算法在初始化阶段的描述如下：

(1) 对于一个任务簇，根据式(1)中的评价函数，得到每个任务关于各内核数据关联性强度的关联度向量.

(2) 对于任务簇中的每个未分配内核的任务，

以拓扑排序的方式根据以下原则完成内核分配：

若待分配任务 $T_p$ 没有前驱任务，则将任务 $T_p$ 以轮询方式分配内核，以 $C$ 表示内核，即：

$$C(T_p) = Cyclic \tag{2}$$

式中的 $Cyclic$ 的取值为[0，$M-1$]的整数，初始为 0，每当有一个任务按照式(2)，即轮询方式分配内核后，$Cyclic$ 的值就会自动增加 1，当 $Cyclic=M$ 时，$Cyclic$ 初始化为 0，这一方式能够在一定程度上实现内核的负载均衡.

若待分配任务 $T_p$ 有多个前驱任务，如图 6 所示，则根据任务的关联度向量，将任务 $T_p$ 分配到数据关联性最强的内核中，即

$$C(T_p) = x(R_x(T_p) = \max\{R(T_p)\},\ x \in [0, M-1]) \tag{3}$$

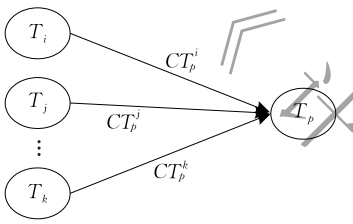


图 6　待分配任务有多个前驱

若待分配任务 $T_p$ 只有一个前驱任务 $T_i$，同时存在其它 $T_i$ 的后继任务与任务 $T_p$ 拥有共同且唯一的前驱，如图 7 所示，此时的情况较为特殊，这些后继任务的关联度向量除了 $R_{C(T_i)}$ 元素不为 0 以外，其它元素均为 0. 在这里，如果待分配任务 $T_p$ 的 $R_{C(T_i)}$ 元素的值为这些后继任务中最大的，则将待分配任务 $T_p$ 按照式(3)方式分配内核，并将其它后继任务按照式(2)方式分配内核；如果待分配任务 $T_p$ 的 $R_{C(T_i)}$ 元素的值不是这些后继任务中最大的，则将待分配任务 $T_p$ 按照式(2)方式分配内核.

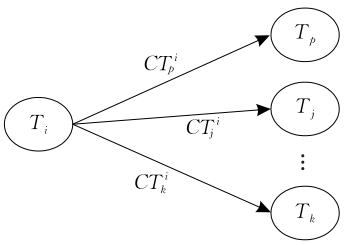


图 7　待分配任务只有一个前驱

以上为算法在初始化阶段的描述. 在初始化阶段，完成对大量周期任务簇的内核分配，并且由于这些任务会周期性地执行到系统关闭，对于这些周期任务的内核分配仅需要在初始化阶段完成一次即可，任务所在内核不再发生改变. 如图 8 所示的为图 3 中的周期任务簇在初始化阶段完成内核分配后的执行过程.

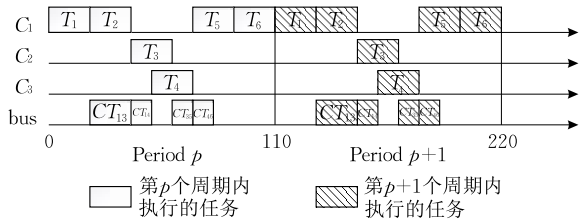


图 8　初始化阶段完成周期任务簇的分配

算法在系统启动时完成初始化阶段，接着在系统开始运行后，算法进入运行阶段. 在这一阶段，不再会出现大量的任务簇等待分配内核，大部分到来的任务为非周期性任务，也有少量的加入某个任务簇的周期任务，且运行阶段到来的任务也会在一定程度上与其它任务具有数据关联性. 若在运行阶段到来的是非周期性任务，则这个任务有极大的可能需要较强的实时性，而由于与这个任务相关的通信型任务会影响到此任务的响应时间，对于非周期性任务来说，内核的分配就成了降低响应时间的关键.

基于以上考虑，算法在运行阶段需要处理的为持续到来的非周期性任务，待分配任务与内核中执行中的任务可能存在数据关联性，因此在运行阶段，算法的描述如下：

(1) 对于在运行阶段中到来的一个待分配任务 $T_p$，根据式(1)中的评价函数，得到任务的关联度向量.

(2) 对于待分配任务 $T_p$，若关联度向量不为 0，以式(3)的方式分配至关联性最强的内核；若关联度向量为 0，以式(2)的方式轮询分配至内核.

以上为算法在运行阶段的描述. 由于算法在运行阶段要持续不断地对到来的任务进行内核分配，因此需要尽量少的计算资源来完成内核分配，以减少对执行中任务的影响，而通过上述算法，能以极小的计算代价使任务分配至关联性最强的内核，并减少最多的核间通信，间接降低了非周期性任务的响应时间. 同时，如果到来的待分配任务与其它任务间没有数据关联性，即关联度向量为 0，则通过式(2)的方式能够在一定程度上考虑到处理器内核的负载均衡. 如图 9 所示，在运行阶段的 $p$ 周期时，到来一个非周期性任务 $T_7$，其具有两个前驱依赖，分别为任务 $T_4$ 和 $T_5$，且 $CT_7^4$ 和 $CT_7^5$ 的执行时间关系为 $e_7^4 > e_7^5$，因此非周期性任务 $T_7$ 被分配到了与任务 $T_4$ 同一个内核中.

由于算法的初始化阶段仅在系统启动时执行一次，系统在大部分时间下执行的是算法的运行阶段，

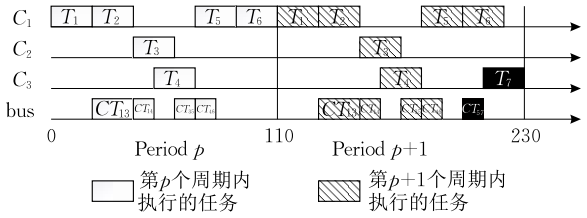


图 9 运行阶段到来非周期性任务

在此,给出算法在运行阶段的性能分析.

**定理 1.** 对于在某个周期到来的非周期性任务 $T_p$,假设与非周期性任务 $T_p$ 有关的所有通信型任务有 $l$ 个,其执行时间为 $\{e_1^p, e_2^p, \cdots, e_l^p\}$,其中最大的为 $e_m^p$,在非周期性任务 $T_p$ 完成内核分配后,与其有关的通信型任务占用总线 $Bus$ 的时间长度将不大于 $\sum_{k=1}^{l} e_k^p - e_m^p (e_m^p = \max\{e_1^p, e_2^p, \cdots, e_m^p\} \mid 1 \le m \le l)$.

证明. 假设计算型任务 $T_m$ 被分配在内核 $r$ 中,非周期性任务 $T_p$ 的关联度向量为 $\boldsymbol{R}(T_p) = (R_0(T_p), R_1(T_p), \cdots, R_{M-1}(T_p))$,则 $\boldsymbol{R}_r(T_p) \ge e_m^p$. 若非周期性任务 $T_p$ 依照算法被分配至内核 $r$ 中,则非周期性任务 $T_p$ 与内核 $r$ 相关的通信型任务的执行时间为 0,因此与非周期性任务 $T_p$ 有关的通信型任务占用总线 Bus 的时间长度为 $\sum_{k=1}^{l} e_k^p - \boldsymbol{R}_r(T_p)$,而 $\boldsymbol{R}_r(T_p) \ge e_m^p$,因此不大于 $\sum_{k=1}^{l} e_k^p - e_m^p (e_m^p = \max\{e_1^p, e_2^p, \cdots, e_m^p\} \mid 1 \le m \le l)$. 若非周期性任务 $T_p$ 依照算法没有被分配至内核 $r$,说明存在 $s (0 \le s \le M-1, s \ne r)$,使得 $\boldsymbol{R}_s(T_p) \ge \boldsymbol{R}_r(T_p)$,非周期性任务 $T_p$ 被分配至内核 $s$,则非周期性任务 $T_p$ 与内核 $s$ 相关的通信型任务的执行时间为 0,因此与非周期性任务 $T_p$ 有关的通信型任务占用总线 Bus 的时间长度为 $\sum_{k=1}^{l} e_k^p - \boldsymbol{R}_s(T_p)$,而 $\boldsymbol{R}_s(T_p) \ge \boldsymbol{R}_r(T_p) \ge e_m^p$,因此不大于 $\sum_{k=1}^{l} e_k^p - e_m^p (e_m^p = \max\{e_1^p, e_2^p, \cdots, e_m^p\} \mid 1 \le m \le l)$. 证得,在非周期性任务 $T_p$ 完成内核分配后,与其有关的通信型任务占用总线 Bus 的时间长度将不大于 $\sum_{k=1}^{l} e_k^p - e_m^p (e_m^p = \max\{e_1^p, e_2^p, \cdots, e_m^p\} \mid 1 \le m \le l)$. 证毕.

上述定理证明了在算法的运行阶段,对于一个在某周期到来的非周期性任务,在完成对其的内核分配后,它对总线资源占用的时间长度上界. 通过此理论,可推出非周期性任务的响应时间,如下.

**引理 1.** 对于在某个周期到来的非周期性任务 $T_p$,假设与非周期性任务 $T_p$ 有关的所有通信型任务有 $l$ 个,其执行时间为 $\{e_1^p, e_2^p, \cdots, e_l^p\}$,其中最大的为 $e_m^p$,在非周期性任务 $T_p$ 完成内核分配后,其响应时间不大于 $\sum_{k=1}^{l} e_k^p - e_m^p + e_p (e_m^p = \max\{e_1^p, e_2^p, \cdots, e_m^p\} \mid 1 \le m \le l)$.

证明. 根据定理 1 可知,非周期性任务 $T_p$ 完成内核分配后,与其有关的通信型任务占用总线 Bus 的时间长度将不大于 $\sum_{k=1}^{l} e_k^p - e_m^p (e_m^p = \max\{e_1^p, e_2^p, \cdots, e_m^p\} \mid 1 \le m \le l)$,而非周期性任务 $T_p$ 开始执行的条件为前驱通信任务执行完毕,因此非周期性任务 $T_p$ 的等待时间将不大于 $\sum_{k=1}^{l} e_k^p - e_m^p (e_m^p = \max\{e_1^p, e_2^p, \cdots, e_m^p\} \mid 1 \le m \le l)$,所以非周期性任务 $T_p$ 的响应时间不大于 $\sum_{k=1}^{l} e_k^p - e_m^p + e_p (e_m^p = \max\{e_1^p, e_2^p, \cdots, e_m^p\} \mid 1 \le m \le l)$. 证毕.

引理 1 证明了在算法的运行阶段,对于每个到来的非周期性任务,其响应时间的上界,保证了非周期性任务一定程度的实时性.

算法的整体过程如下:

**算法 1.** DTSV 算法.

```
输入:
  Data related task clusters: DAG G=(V, E);
  Random task queue: *Tp;
  The processor number: M
输出:
  Assignment result of tasks;
BEGIN:
  Init(G); //初始化阶段,完成对周期任务簇的分配
  WHILE(1) //运行阶段
   IF(Tp)
    Assign(Tp); //对运行阶段的非周期性任务分配内核
   END IF
  END WHILE
END
```

算法在初始化阶段对关联任务簇进行一次分配. 在运行阶段,由于本算法不涉及内核的迁移,对每个到来的任务也仅需要一次内核分配,因此本算法总的时间复杂度为 $O(n)$.

### 4.2 非周期性任务数据关联的预处理

对于本文所提到的混合场景中的非周期性任务来说,影响其响应时间的因素主要有两部分. 首先是与非周期性任务有依赖关系的所有前驱周期性任务的执行结束时间以及非周期性任务所有前驱通信型任务的执行结束时间. 本文算法在初始化阶段完成

对周期任务簇的内核分配后，周期性任务所在的内核便不再会做出调整，即周期性任务的执行情况不会改变，因此可通过对非周期性任务所依赖的前驱通信型任务进行优化，进而减少非周期性任务的响应时间，提高非周期性任务的实时性.

文献[12]中采用类似流水线的方式，能够将周期性任务的所依赖的前驱通信型任务进行预处理，从而减少整个周期任务簇的核间通信开销. 在此，本文针对混合场景中的非周期性任务，采用类似的方式对非周期性任务所依赖的前驱通信型任务进行预处理. 在周期性任务执行过程中，因本文算法在初始化阶段对周期任务簇的内核分配已经避免了大部分的核间通信开销，因此总线在一个周期中的多数时间处于空闲状态. 针对这一条件，本文在非周期性任务执行的上一周期将其所依赖的前驱通信型任务执行完毕，从而缩短非周期性任务所需的等待时间，减少非周期性任务的响应时间.

如图 10 所示，与图 9 中非周期性任务的状况相同，但通过对总线空闲时间的利用，可以将非周期性任务所依赖的数据关联提前一个周期完成，使得非周期性任务可以在前驱计算型任务执行完成后立即开始执行，减少非周期性任务的响应时间，提高实时性.

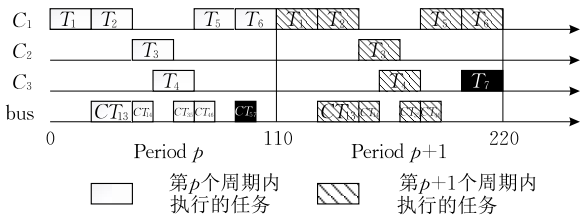


图 10　非周期性任务数据关联的预处理

对于整个任务周期来说，核间通信的完成需要总线资源，因此并不是所有的非周期性任务都可能通过预处理来缩短任务响应时间. 这里提出一个条件，对于某个到来的非周期性任务 $T_p$ 来说，若总线满足这一条件，则可以完成对非周期性任务的预处理.

**定理 2.**　在周期任务执行的 $p$ 周期时，加入了一个非周期性任务 $T_p$，若在整体的周期调度长度 $I$ 中，Bus 的空闲长度不小于 $\sum_{k=1}^{l} e_k^p - e_m^p (e_m^p = \max\{e_1^p, e_2^p, \cdots, e_m^p\} \mid 1 \le m \le l)$，则在下个周期 $p+1$ 周期时，非周期性任务 $T_p$ 的执行不需要额外的等待时间.

证明.　根据定理 1 可知，与非周期性任务 $T_p$ 相关的通信型任务占用总线 Bus 的时间长度不大于 $\sum_{k=1}^{l} e_k^p - e_m^p (e_m^p = \max\{e_1^p, e_2^p, \cdots, e_m^p\} \mid 1 \le m \le l)$. 若整个周期长度 $I$ 中，Bus 的空闲长度不小于 $\sum_{k=1}^{l} e_k^p - e_m^p (e_m^p = \max\{e_1^p, e_2^p, \cdots, e_m^p\} \mid 1 \le m \le l)$，则在下个周期 $p+1$ 周期时，与非周期性任务 $T_p$ 相关的前驱通信型任务已全部在任务 $T_p$ 执行之前执行完毕，因此非周期性任务 $T_p$ 的执行不需要额外等待时间. 证毕.

定理 2 给出了非周期性任务预处理的条件，对于满足此条件的非周期性任务，可推出其响应时间的上界.

**引理 2.**　对于非周期性任务 $T_p$，如果满足定理 2 中的条件，则非周期性任务 $T_p$ 从某个周期 $p$ 加入到执行完成，其响应时间不大于 $I+e_p$，$I$ 为整体的周期调度长度.

证明.　根据定理 2 可知，如果非周期性任务 $T_p$ 满足定理 2 的条件，则非周期性任务 $T_p$ 的执行不需要额外等待时间，又因为非周期性任务 $T_p$ 从到来到开始执行的时间小于等于整体的周期调度长度 $I$，又已知非周期性任务 $T_p$ 的计算量为 $e_p$，则可计算出，非周期性任务 $T_p$ 的响应时间不大于 $I+e_p$.　证毕.

上述理论和性质保证了在算法的运行阶段，对于到来的非周期性任务，能将其响应时间限制在一定范围内，一定程度上保证了非周期性任务的实时性.

而在原本的算法中增加了对非周期性任务数据关联的预处理后，算法的整体过程变为如下：

**算法 2.**　增加预处理后的 DTSV 算法.

```
输入：
 Data related task clusters: DAG G=(V, E);
 Random task queue: *Tp;
 The processor number: M
输出：
 Assignment result of tasks;
BEGIN:
 Init(G); //初始化阶段,完成对周期任务簇的分配
 WHILE(1) //运行阶段
  IF (Tp!=NULL)
  THEN
   Assign(Tp); //对运行阶段的非周期性任务分配内核
   GetBus( );//获取总线周期空闲长度
   IF (Tp.CT<=GetBus())
   THEN
    PreTrans(Tp);
   END IF
  END IF
  END WHILE
END
```

如上所示,在添加了对非周期性任务数据关联的预处理后,算法在车联网系统的初始化阶段保持不变,在运行阶段,每当对一个非周期性任务进行内核分配时,需判断当前总线的空闲情况,以便对非周期性任务的数据关联进行预处理.因此,本算法的时间复杂度依然为 $O(n)$.

## 5 实验与分析

在本节,进行了实验来评估本文方法的有效性.实验中用到的周期任务簇均为现实生活中的流媒体应用程序以及合成任务图,任务图由 TGFF[16] 生成.TGFF 为一种任务图生成器,它可以创建任务调度问题实例的完整描述,包括处理器、通信资源、任务和任务间通信的属性.

本文针对算法性能的整体评估,主要方法是,对于周期任务簇来说,在初始化阶段完成内核分配后,周期性地执行 20 个周期,而对于非周期性任务来说,在这 20 个执行周期中每隔 1 个周期产生 1 个非周期性任务,且非周期性任务与其它任务数据关联性的大小与周期任务簇的规模成正比.对于算法性能的评价,主要观察以下几个指标:任务的整体调度长度、非周期性任务核间通信的减少量、非周期性任务的平均响应时间.本文算法的比较对象分别是以轮询方式简单内核分配的 Cyclic 算法以及以内核负载均衡为主要目的的 LBPSA 算法[17].

### 5.1 算法的整体性能

本文实验采用了四核、五核、六核的配置,测试了对于不同规模的周期任务簇,各个算法的表现情况,任务的整体调度长度如表 2 所示.从表 2 中可以看出,由于所采用的任务簇中任务间具有数据关联性,本文算法可以在一定程度上降低任务的整体调度长度,且与本文算法相比较的两个算法之间的几乎没有明显的差别,这是因为以轮询方式简单内核分配的 Cyclic 算法在一定程度上也考虑到了内核负载均衡的问题,且两个算法均没有考虑到任务之间的数据关联性.

由于相较于计算型任务本身的执行时间来说,通信型任务的执行时间所占的比例较小,且本文算法主要目的是降低核间通信,因此相对来说任务的整体调度长度减少的程度不大.但是考虑到周期任务的执行会从系统启动开始直到系统关闭,在如此长的时间长度内,本文算法对任务整体调度长度的优化会对系统性能有明显的提升.

### 5.2 非周期性任务性能

算法对于非周期性任务的性能,本文首先从非周期性任务的核间通信减少量来进行测试,如表 3 所示.对于非周期性任务来说,影响其响应时间的一大因素就是与之相关的通信型任务的执行,只有当非周期性任务的所有前驱通信型任务的执行完成后,非周期性任务才能够开始执行.本文算法通过比较非周期性任务相对于各个内核的数据关联性强度,能够在最大程度上减少核间通信,即能够尽量多地将与非周期性任务相关的通信型任务变成任务的内核通信,将其执行时间变为 0.

图 11 为表 3 中的数据转换的条形图.从图 11 可以看出,本文算法相较于另外两个算法,非周期性任务核间通信的减少量明显较高,是因为本文算法的内核分配主要依据的是式(1)中的评价函数,总是

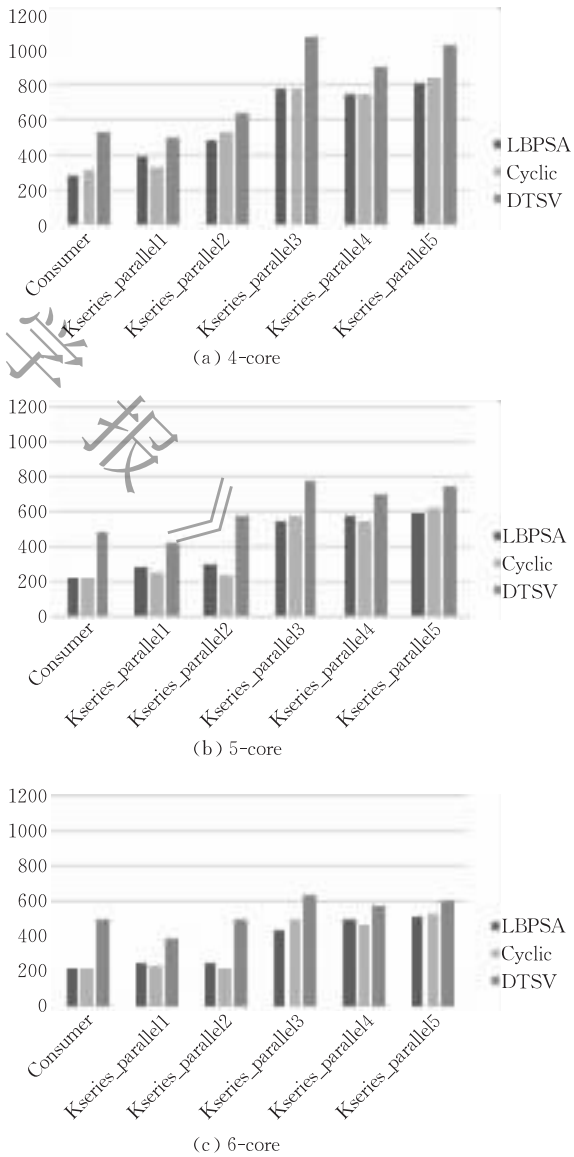


图 11 非周期性任务核间通信减少量

能够将非周期性任务分配至数据关联性最强的内核中，从而能够避免最多的核间通信. 而随着周期任务簇规模的提升，在运行阶段到来的非周期性任务也与越来越多的任务具有数据关联性，因此在整个执行过程所减少的通信量也在增加.

但随着处理器内核数量的提升，非周期性任务的核间通信减少量反而降低. 这是由于随着内核数量的增加，算法出于负载均衡的考虑. 周期任务簇的任务分配不会过于集中，当非周期性任务到来时，非周期性任务的所分配的内核中，与之具有数据关联性的任务量也会减少. 通过表 2 与表 3 的综合比较，处理器内核数量的提升，可以在一定程度上减少任务的整体调度长度，但会增加任务核间通信的开销.

本文对非周期性任务性能的另一测试为非周期性任务的平均响应时间，如表 4 所示. 由 Inference 2 可知，非周期性任务响应时间的上界与任务整体的周期调度长度有关，所以，随着周期任务簇规模的不断增加，任务整体的周期调度长度也不断增加，最终也会导致非周期性任务的平均响应时间不断增加.

而从表 4 可以看出，本文算法相对于另外两个算法，对于非周期性任务的平均响应时间具有很明显的优化效果. 但是随着周期任务簇规模的提升，这一优化效果逐渐减少并趋于稳定，这是由于随着周期任务簇规模的提升，其核间通信的规模也随着提升，总线资源的占有率也相对提升. 而本文算法针对非周期性任务所采用的预处理需要满足 Theorem 2 中所描述的条件，总线资源的占有率越高，Theorem 2 中的条件越难满足，能够满足预处理条件的非周期性任务也随之减少，非周期性任务的平均响应时间达不到最大程度的优化，因此非周期性任务的响应时间必然不会超过 Inference 1 中所描述的上界.

**表 2　任务整体调度长度**

| Benchmarks | Number of task | Number of edge | 4 cores | | | 5 cores | | | 6 cores | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | LBPSA | Cyclic | DTSV | LBPSA | Cyclic | DTSV | LBPSA | Cyclic | DTSV |
| Consumer | 13 | 17 | 15503 | 15115 | 14240 | 13900 | 13445 | 12650 | 13900 | 13445 | 12650 |
| Kseries_parallel1 | 20 | 19 | 10066 | 10770 | 8696 | 10484 | 11547 | 8696 | 10484 | 12124 | 8696 |
| Kseries_parallel2 | 30 | 33 | 19342 | 19010 | 17271 | 18516 | 18470 | 16423 | 17735 | 17673 | 15139 |
| Kseries_parallel3 | 42 | 48 | 28441 | 28240 | 25490 | 27741 | 26944 | 25435 | 22251 | 22124 | 20916 |
| Kseries_parallel4 | 47 | 46 | 26706 | 26563 | 24354 | 19779 | 19420 | 18356 | 19736 | 19326 | 18309 |
| Kseries_parallel5 | 52 | 55 | 31453 | 32547 | 28412 | 28481 | 29345 | 25954 | 27415 | 26849 | 25183 |

**表 3　非周期性任务核间通信减少量**

| Benchmarks | Number of task | Number of edge | 4 cores | | | 5 cores | | | 6 cores | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | LBPSA | Cyclic | DTSV | LBPSA | Cyclic | DTSV | LBPSA | Cyclic | DTSV |
| Consumer | 13 | 17 | 283 | 311 | 529 | 223 | 214 | 487 | 224 | 214 | 487 |
| Kseries_parallel1 | 20 | 19 | 390 | 323 | 507 | 282 | 253 | 424 | 253 | 229 | 393 |
| Kseries_parallel2 | 30 | 33 | 482 | 532 | 642 | 295 | 232 | 573 | 247 | 218 | 494 |
| Kseries_parallel3 | 42 | 48 | 783 | 774 | 1077 | 546 | 582 | 784 | 442 | 493 | 631 |
| Kseries_parallel4 | 47 | 46 | 745 | 744 | 898 | 568 | 551 | 692 | 493 | 468 | 574 |
| Kseries_parallel5 | 52 | 55 | 812 | 843 | 1023 | 595 | 624 | 749 | 511 | 535 | 607 |

**表 4　非周期性任务的平均响应时间**

| Benchmarks | Number of task | Numbetr of edge | 4 cores | | | 5 cores | | | 6 cores | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | LBPSA | Cyclic | DTSV | LBPSA | Cyclic | DTSV | LBPSA | Cyclic | DTSV |
| Consumer | 13 | 17 | 485 | 464 | 148 | 485 | 464 | 148 | 485 | 464 | 148 |
| Kseries_parallel1 | 20 | 19 | 331 | 412 | 198 | 424 | 451 | 198 | 424 | 483 | 198 |
| Kseries_parallel2 | 30 | 33 | 464 | 444 | 329 | 465 | 472 | 325 | 482 | 492 | 268 |
| Kseries_parallel3 | 42 | 48 | 622 | 634 | 546 | 655 | 655 | 587 | 652 | 662 | 564 |
| Kseries_parallel4 | 47 | 46 | 451 | 437 | 368 | 451 | 459 | 379 | 451 | 456 | 373 |
| Kseries_parallel5 | 52 | 55 | 573 | 567 | 484 | 565 | 554 | 469 | 553 | 562 | 461 |

## 6　结束语

针对车联网中不同类型任务调度与响应需求的多样化，结合嵌入式多核系统中任务调度过程，本文提出了一种用于车联网的基于数据关联性的任务调度算法. 该算法依据车联网中任务特性以及关联性任务的常用模型，提出了任务关联性的评价函数，并

依据此评价函数,设计了面向车联网应用的数据关联性任务调度算法.与目前基于负载均衡的多核任务调度算法相比,本文的算法通过将具有强数据关联性的任务分配到同一内核中,避免了过多的核间通信,以降低任务调度长度,提高任务调度与响应性能.最后,通过实验仿真验证了该算法针对周期性任务以及非周期性任务等不同类型任务的调度需求,都具有较好的调度性能.

## 参 考 文 献


[1] Xu C, Zhao F, Guan J, et al. Muntean, QoE-driven user-centric VoD services in urban multihomed P2P-based vehicular networks. IEEE Transactions on Vehicular Technology, 2013, 62(5): 2273-2289

[2] Haziza N, Kassab M, et al. Multi-technology vehicular cooperative system based on software defined radio//Berbineau M, et al, eds. Communication Technologies for Vehicles. Berlin, Germany: Springer, 2013: 84-95

[3] Andersson B. Global static-priority preemptive multiprocessor scheduling with utilization bound 38 percent//Baker T P, Bui A, Tixeuil S eds. Principles of Distributed Systems. Berlin, Germany: Springer, 2008: 73-88

[4] Han Q, Wang T, Quan G. Enhanced fault-tolerant fixed-priority scheduling of hard real-time tasks on multi-core platforms//Proceedings of the International Conference on Embedded and Real-Time Computing Systems and Applications. Hong Kong, China, 2015: 21-30

[5] Wang T, Niu L, Ren S, Quan G. Multi-core fixed-priority scheduling of real-time tasks with statistical deadline guarantee//Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE). Grenoble, France, 2015: 1335-1340

[6] Wang Y, Shao Z, Chan H C B, et al. Memory-aware task scheduling with communication overhead minimization for streaming applications on bus-based multiprocessor system-on-chips. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(7): 1797-1807

[7] Xu X, Center N, Wang L. Task assignments based on shared memory multi-core communication//Proceedings of the International Conference on Systems and Informatics. Shanghai, China, 2014: 324-328

[8] Hartenstein H, Laberteaux K P. A tutorial survey on vehicular Ad Hoc networks. IEEE Communications Magazine, 2008, 46(6): 164-171

[9] Fan M, Quan G. Harmonic-aware multi-core scheduling for fixed-priority real-time systems. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(6): 1476-1488

[10] Xu R, Melhem R, Moss D. Energy-aware scheduling for streaming applications on chip multiprocessors. IEEE Computer Society, 2007, 28(49): 25-38

[11] Chen Y S, Shih C S, Kuo T W. Dynamic task scheduling and processing element alloca-tion for multi-function SoCs. IEEE Real Time & Embedded Technology & Applications Symposium, 2007, 13: 81-90

[12] Wang Y, Liu D, Qin Z, et al. Optimally removing intercore communication overhead for streaming applications on MPSoCs computers. IEEE Transactions on Computers, 2013, 62(2): 336-350

[13] Huang Shu-Juan, Zhu Yi-An, Li Bing-Zhe, et al. Real-time scheduling method for dependency period tasks. Chinese Journal of Computers, 2015, 38(5): 999-1006(in Chinese)
(黄姝娟, 朱怡安, 李兵哲等. 具有依赖关系的周期任务实时调度方法. 计算机学报, 2015, 38(5): 999-1006)

[14] Karagiannis G, et al. Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions. IEEE Communications Surveys & Tutorials, 2011, 13(4): 584-616

[15] Robinson CL. DRAFT SAE J2735-Dedicated short range message set (DSRC) dictionary (rev. 29). Technical Report J2735, USA, 2008

[16] Dick R P, Rhodes D L, Wolf W. TGFF: Task graphs for free//Proceedings of the 6th International Workshop on Hardware/Software Codesign. Seattle, USA, 1998: 97-101

[17] Jain D, Jain D L. Load balancing real-time periodic task scheduling algorithm for multiprocessor environment//Proceedings of the International Conference on Circuits, Power and Computing Technologies. Nagercoil, India, 2015: 1-5



**DING Nan**, born in 1978, Ph. D., associate professor. His research interests include embedded system design, cyber-physical system, and vehicular Ad Hoc Networks.

**NIE Shuai-Hang**, born in 1993, M. S. candidate. His research interests include embedded system design and Vehicular Ad Hoc Networks.

**XU Li**, born in 1982, Ph. D., senior engineer. His research interests include network virtualization, cloud computing and probabilistic graph model.

**TAN Guo-Zhen**, born in 1960, Ph. D., professor. His research interests include cyber-physical system, Vehicular Ad Hoc Networks and ITS.

**Background**

With the development and progress of Intelligent Transportation System (ITS), the data-driven applications running on VANET-supported platform have received more and more attention. Providing support for high Quality of Experience for VANETs applications is a significant challenge. To ensure the multi-task concurrency mechanism in the multi-core processor system and to improve the response performance, the key is to design an efficient task scheduling algorithm. A multi-core processor is capable to handle multiple tasks at the same time and, as a result of that, raises the task throughput, but it also brings about an extra consumption: inter-core communication. The inter-core communication scheduling on multicore architectures has been investigated in the previous work. To solve this problem, DTSV, a data related scheduling algorithm for VANETs applications, is proposed in this paper, with considering the multi-task concurrency mechanism in VANETs. As verified by experiments, this DTSV algorithm can allocate more tasks in which data are related into one core, which thus improves the real-time task response since inter-core communications are replaced by the intra-core ones.

The contribution of this paper is exploration of data related task scheduling for VANETs and inter-core communication overhead minimization. Compared to those multicore scheduling algorithms before, DTSV algorithm allocates the tasks that have related data to the same core, which, partly, avoids the number of inter-core communications and reduces the delay in communications. In the era of big data, our algorithm can significantly improve VANETs real-time performance and, therefore, ensure the safety of the vehicles. Prior to this work, our research group had publish more than 20 papers in high quality journals and international conferences, including IEEE Transactions on Vehicular Technology, IEEE Transactions on Neural Networks and Learning Systems, IEEE Transactions on Cybernetics.

This work was supported in part by the National Science Foundation of China under Grant No. 61471084, the National High Technology Research and Development Program of China (863 Program) under Grant No. 2012AA111902 and the Open Program of State Key Laboratory of Software Architecture under Grant No. SKLSAOP1602.