软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(Suppl.(1)):11−19

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向异构多处理器设备的自适应命令解释系统*

刘文卿, 李 栋, 崔 莉

(中国科学院 计算技术研究所,北京 100190)

通讯作者: 李栋, E-mail: lidong@ict.ac.cn

摘 要: 智能化赋予了物联网更深刻的实用价值,但是在计算能力强与功耗低的之间寻求性能最优是目前物联网设备极难解决的问题.异构多处理器结构与单一或者同构的多处理器相比可以结合不同处理器的优势,同时满足高计算能力与低功耗的系统需求,但异构多处理器结构下软件编程难度大的问题以及如何优化顶层应用在多处理器设备上的运行性能都是目前亟待解决的技术难题.针对以上问题,设计并实现了一个面向异构多处理器设备的自适应命令解释系统.首先,该系统允许用户将物联网应用安装到设备上,应用程序以命令脚本形式呈现;其次,系统设计了命令在异构多处理器设备上的自动分发算法,该算法考虑性能和功耗的多维参数,在满足时间上限的条件下最优化应用执行能耗.最后,提出了针对同时满足不同用户应用需求的解决方案,在物联网设备的资源受限的条件下,根据具体用户使用习惯,提出了一种基于用户使用历史的命令解释系统自适应方案,可以根据用户个性化习惯自动完成命令解释系统的自适应部署和运行时优化.

关键词: 异构多处理器设备;命令解释器;用户习惯;智能化;物联网

中文引用格式: 刘文卿,李栋,崔莉.面向异构多处理器设备的自适应命令解释系统.软件学报,2017,28(Suppl.(1)):11−19. http://www.jos.org.cn/1000-9825/17002.htm

英文引用格式: Liu WQ, Li D, Cui L. Self-Adaptive command interpretation system for heterogeneous multiprocessor devices. Ruan Jian Xue Bao/Journal of Software, 2017,28(Suppl.(1)):11−19 (in Chinese). http://www.jos.org.cn/1000-9825/17002.htm

Self-Adaptive Command Interpretation System for Heterogeneous Multiprocessor Devices

LIU Wen-Qing, LI Dong, CUI Li

(Institute of Computing Technology Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Intelligence has given the Internet more practical value, but the dual requirements for both the computing power and low power consumption in the current single-processor networking equipment have not been met. The heterogeneous multiprocessor architecture can combine the advantages of different processors with a single or isomorphic multiprocessor to select the right processor to meet the requirements for both high-performance and low-power intelligent device. At present, the software programming under heterogeneous multiprocessor structure and how to optimize the performance of top-level applications on multiprocessor devices are the technical problems that need to be solved. In view of the problems above, this paper designs and implements an adaptive command interpretation system for heterogeneous multiprocessor devices. First, the system allows users to install the Internet of Things application to the device in the form of URL access, and the application is presented in the form of a command script contained in the URL. Secondly, the system designs an automatic distribution algorithm for commands on heterogeneous multiprocessor devices, which adds the energy consumption parameters on the basis of the DAG graph model, and designs a command distribution scheme with the best energy consumption under the condition of satisfying the time limit. Finally, the system is faced with the problem of satisfying the needs of different users at the same time, but the resources of the equipment in the Internet of Things are limited. So the system needs to be customized according to the specific users. This paper presents a self-adaptive scheme of command interpretation system based on users' habits, which can automatically complete the system adapting based on users' specific habits.

* 基金项目: 国家自然科学基金(61672498, 61502461)
Foundation item: National Natural Science Foundation of China (61672498, 61502461)
收稿时间: 2017-05-15; 采用时间: 2017-09-23

Key words: heterogeneous multiprocessor devices; command interpreter; user habits; intelligence; internet of things

物联网涉及到的研究领域不仅包含了数据采集技术、通信技术、能量优化等技术,还涉及到人工智能技术、海量数据存储技术、海量数据处理技术等新兴技术.人工智能、网络技术的飞速发展,推动了物联网的发展.物联网应用广泛,包括可穿戴设备、智能家居、环境监测、智能医疗、智慧农业等很多领域.随着物联网研究的不断深入,据 Gartner 报告显示,截止到 2016 年,物联网设备的总量将达到 64 亿台.但迄今为止,物联网产业仍然不够成熟,还没有完成形成一套成熟通用的物联网应用开发运行环境.在物联网产业中存在着严重限制了物联网发展的“昆虫纲”悖论问题,即产业规模总量看似巨大,但由于其应用可批量规模小且应用种类繁多,碎片化的应用增加了信息系统的复杂性,使得很难形成一种通用的技术生态系统来满足各种不同的应用需求.碎片化的垂直应用,软硬件的异构程度高,水平互联的难度大,重复开发现象严重,限制了物联网的发展.针对物联网应用开发运行环境,目前已经提出了很多有效的成果,比如操作系统方面的研究以及开源硬件提供商提供的统一接口.

智能化赋予了物联网更深刻实用的价值,随着人工智能的飞速发展,物联网智能化也愈演愈烈.但是目前的单处理器物联网设备无法满足物联网中智能化设备计算能力强与功耗低的双重要求.异构多处理器设备可以结合多处理器优势,满足这种双重需求.但是目前异构多处理器结构下软件编程难度大的问题以及如何优化顶层应用在多处理器设备上的运行性能都是目前亟待解决的技术难题.针对该问题,本文面向异构多处理器设备研究能够方便用户开发的自适应命令解释系统的设计与实现.

本文的主要贡献如下:

1) 提出了一种面向异构多处理器设备的能够基于用户习惯进行自适应的命令解释系统架构,该设计允许用户通过 URL 访问的形式把物联网应用安装到设备上;

2) 以时间和能耗两项指标对命令分发进行了建模,并提出了一种在满足时间上限条件下计算应用执行能耗最优分发方案的命令分发算法;

3) 面向物联网中不同用户需求差异大以及设备资源的受限性,本文设计了一种根据用户使用命令习惯进行系统自适应地方案.

1 相关工作

为了解决在物联网中使用异构多处理器设备进行应用开发难度大以及应用在设备上运行性能的问题,现有研究工作进行了以下探究.

微软的研究者设计了一种支持异构多处理器体系结构的新型物联网操作系统——CoMOS 操作系统.它针对同时具有 ARM 处理器和 MSP430 处理器的异构多处理器设备设计,通过设计新的操作系统,让开发者在操作系统的基础上进行应用的设计与开发,方便了物联网应用的开发工作,但是该操作系统目前不具有通用性,仅提供了对一种异构多处理器设备的支持,在其他的异构多处理器设备上移植成本高[1].

开源硬件提供商 Arduino 针对于异构多处理器设备 Arduino YUN 封装了一个处理器间通信的 Bridge 库,该库提供了处理器间调用的接口,该库在 Arduino 系统软件核心层对异构多处理器结构进行封装并提出了通信机制,方便了开发者的使用,提高了生产效率[2].

文献[3]中提出了一种异构多处理器设备中应用设计的流程,并设计了一个装置有 FPGA 协处理器的设备,同时分析比较了该多处理器设备与单处理器设备在执行任务时的能耗使用情况.

文献[4]中提出了一种在异构多处理器上设计和实现程序的方案,提出了一种通用的分层软硬件接口,该接口能够满足微控制器上的任务与 FPGA 上任务的无缝链接.

文献[5]中指出处理器间通信时使用共享互联和共享存储会导致异构设备时间、功率、和时间域的干扰,阻碍了设计人员充分利用多处理器的优势,它提出了一种用于异构无线嵌入式平台构造的第 1 个超低功耗处理器互联方法.

由 Apache 基金会管理的开源物联网操作系统 Zephyr[6]面向支持异构多处理器体系结构提出了物联网操作系统内微内核与超微内核的基础软件架构,能够支持基于多处理器物联网端设备的开发与应用,能够在系统初始化过程中对异构处理器的角色和内核算法进行配置,以此应对物联网应用场景碎片化的问题.

这些工作对在物联网中应用异构多处理器设备进行了深入的探究,提出了一些简易编程以及多处理器间资源调度的方法,然而,这些方法主要针对于具体异构多处理器设备在物联网中的应用进行了设计与开发,没有形成一种通用性的设计.

2 系统设计

本节主要介绍面向异构多处理器设备的自适应命令解释系统的设计.其中,第 2.1 节给出了系统的主要构成.第 2.2 节给出应用在系统中运行过程的说明.

2.1 系统组成

本文提出的命令解释系统系统支持命令序列形式的应用程序在设备上的运行,应用程序形式是:http://myArduinoYun.local/shell/collect:13,1,1000.URL 主要包含两部分:(1) 设备的 IP 地址;(2) 命令序列,类似于脚本程序的形式.该系统允许开发者通过 URL 访问的形式把物联网应用安装到设备上.下面具体介绍下系统的组成:

如图 1 所示,本文以具有一个资源受限处理器和一个资源非受限处理器的异构多处理器设备进行说明,本文提出的命令解释系统主要包含 3 部分:异构多处理器设备上的命令解释器部分,基于用户习惯的命令解释系统自适应部分以及云端服务器部分.异构多处理器设备上的命令解释器包含两个模块:(1) 命令统一解析并自动分发的部分,具体自动分发方式见本文第 3 节;(2) 各处理器上的命令解释器.命令解释器的用户自适应部分,包含了 3 个模块:(1) 用户历史使用命令的收集,在本系统中以键值对的方式保存;(2) 根据收集到的用户历史使用命令的频度对命令集进行筛选的模块,在本文第 4 节对筛选算法具体进行说明;(3) 根据命令集筛选结果进行命令解释器重组的模块.云端服务器部分主要包含 3 个模块:(1) 命令库管理接口,主要负责命令添加、更新等;(2) 数据库部分,设备采集的感知数据库以及存储命令对应程序的程序库;(3) 重新组合生成新的命令解释器的程序,程序根据上文中命令集定制算法结果(命令序号的集合)进行新的命令解释器的自动组合生成.

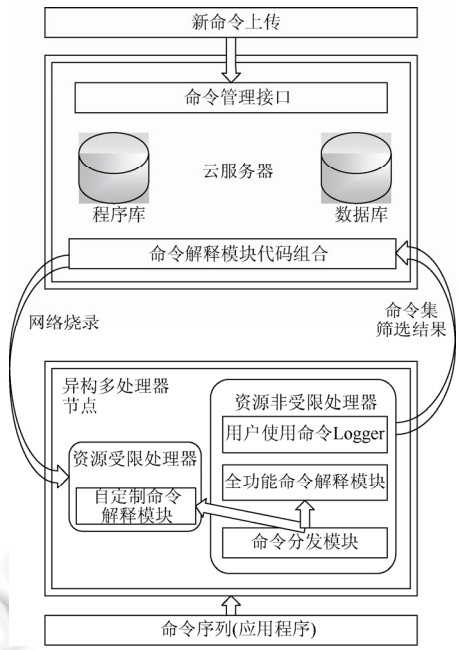


Fig.1 Command interpretation system framework

图 1 命令解释系统架构图

2.2 应用程序在系统中的执行过程

本节对应用在系统中是如何执行的进行具体说明,应用运行的步骤如下:

(1) 系统应用程序在浏览器中以访问地址的形式执行,应用程序形式为一条 URL,http://192.168.191.2/arduino/collect:$1,10,2,10000;compress:*1,//tmp//,data 为一个应用程序实例,该应用完成从数据引脚 10 读取 10 000 次数据,并对数据进行压缩,压缩后的数据保存在/tmp/目录中,以 data 为压缩的文件名,应用中 192.168.191.2 为设备的 IP 地址;

(2) 设备接收到程序后,根据第 3 节中的命令分发方案完成命令的自动分发.之后对 URL 进行更新,加入分

配的标记,比如本例中更新为 http://192.168.191.2/arduino/collect:$1,10,2,10000;compress:$2,*1,//tmp//,data,其中$符合用于指示执行命令的处理器.

(3) 在负责命令分发的处理器中保存本次分配结果“[collect:1]”,用该分配结果并更新系统中收集的用户在处理器上使用命令的频次.

(4) 更新后的 URL 传递到负责统一调度的处理器上完成调度执行.

(5) 各命令按照命令的有向无环图依次分发到具体处理器上执行.

(6) 之后负责命令分发的处理器会判断当前执行情形是否需要对处理器上的命令解释器进行自适应工作,若未达到要求,则执行完毕;否则进入步骤(7),开始进行系统的自适应工作.

(7) 负责命令分发的处理器会根据步骤(3)中统计的各命令在处理器上被用户使用的频度进行命令集的筛选,筛选算法节第 4 节,之后进入步骤(8).

(8) 最后异构多处理器设备以命令集筛选结果和设备网络地址为参数向云服务器请求新的定制的命令解释器.云服务器根据定制结果自动完成命令解释器的组合以及编译,最后通过网络编程到处理器上.远程更新过程可通过现有的重编程技术更新进行优化,本工作中未对此展开研究

3 命令分发算法

本节介绍面向异构多处理器设备的命令解释系统的命令分发问题,通过分析比较已有异构多处理器体系结构下的任务分配方案,针对于物联网中异构多处理器设备的特异性,设计了一种在满足应用执行时间约束下的最低能耗的任务分发方法.相比以往的近似型算法,该算法具有确定性结果,且更加简洁明确,既结合了各处理器的优势,又保证了设备低能耗的特性.

多处理器上的资源分配问题被证明为 NP 难问题.任务调度算法对该问题的研究主要分为 3 种,包括基于周期模型的研究、基于 DAG 模型的研究以及其他模型的研究[7−17].

物联网中设备的命令分发与传统的异构计算系统有其特异性.首先,物联网设备相比于分布式的异构计算系统更注重设备的整体能耗问题,而异构计算系统注重在最短时间内完成用户任务,返回用户反馈;其次,在物联网设备中异构处理器的个数 n 是比较小的,由于设备成本以及能耗双方面的考虑,一般异构多处理器设备的处理器数量是小于 3 个的,在现有物联网工业界成果以及研究成果中尚没有出现过针对某特定物联网场景的处理器个数超过 5 个的异构多处理器设备;第三,异构多处理器设备中的命令数量是有限的,在对物联网应用的统计中,一个应用大多能转换成 4~6 条命令,不会出现分布式异构计算中海量任务的情形,因此,本文定义一般情形下组成单个应用的命令数为 m,通常应用 m<10,在实际应用开发中,本系统要求开发者命令序列中的命令数小于 10,即 m 总是不大于 10 的,若应用包含的命令数超过此要求,系统要求用户自定义新命令,并通过云端服务器进行命令添加;最后,异构多处理器设备中的各处理器功能是不对等的,即一些命令在某些处理器上是可执行的,而在另一些处理器中则是不可执行的,存在功能不对称的问题.

本文基于 DAG 图建立算法模型,DAG 模型使用二元组(V,E)对命令进行建模,其中 V 是设备的集合,每个设备表示一个命令组,数组长度为处理器个数,其中每个命令包含如下属性:命令执行时间,命令执行能耗,命令输出的数据包大小;E 是有向边的集合,有向边代表命令间的执行顺序约束和通信开销,命令间的通信开销由两条命令具体选择的处理器决定,具体地每条边包含如下属性:命令间通信的时间,命令通信的能耗.在本模型中,系统目标是寻找一组命令分配方案,使得在满足应用执行时间 t 的条件下,应用的执行能耗最低.模型中命令执行时间与执行能耗在不同处理器上的差别较大,本文根据命令的时间复杂度粗略计算获得;处理器传输的时间与能耗根据传输的数据包大小与传输单位数据包的时间与能耗计算得到,本文中命令分发算法提出基于 DAG 图模型理论,由于物联网中命令分发相比于分布式集群具有其特有的特点,以往的理论难以适应在此种环境中,因此本文针对此特点通过暴力枚举设计了解决方案,此算法为确定性算法,并适用于该系统环境.

由上文中分析的物联网中命令在多处理器上分发的特异性,本文以 DAG 图为模型提出了 Dijkstra 与枚举相结合的在满足时间上限条件下计算使得应用执行能耗最优的分发方案,具体流程如下:

(1) 应用程序的执行时间要求输入,缺省表示不限制时间上限;

(2) 命令在处理器上执行时间计算;

(3) 各处理器上执行各命令的能耗计算(处理器单位时间能耗×命令执行时间);

(4) 根据应用程序中命令的相互调用关系,生成命令的有向无环图,图中节点的权重为命令的能耗,边的权重为命令的执行时间;

(5) 枚举出各命令在各处理器上分发的所有情况;

(6) 计算每种分发情况的能耗(各命令在对应处理器上的能耗以及转换处理器时的通信能耗),使用 Dijkstra 计算各种分发结果的最早完成时间;

(7) 输出满足时间上限要求的能耗最低分发结果.

4 基于用户习惯的命令解释系统自适应裁剪和更新机制

4.1 命令集筛选算法

在物联网环境中,出于异构多处理器设备成本的考虑,设备中存在一些可编程空间不足的处理器,无法装载有完整的命令解释器.这导致使用上文的命令分发算法进行分发时,会出现分发到某处理器上的命令不能够命令,不得不对命令进行重新分配的现象,因此会影响到命令分发的结果,我们将在命令分配到某处理器上时命令不能在此处理器上被执行的事件出现的概率称为命令在处理器上的的不可执行率或者不命中率.华为手机使用用户历史数据实现了手机资源调度的优化[18].本文根据系统中记录的用户使用设备执行应用的行为信息,提出了一种基于用户习惯的命令集筛选算法,该算法可实现让有限空间最大化地满足用户执行命令的可执行率,进而支持最优命令分发结果的执行,优化系统性能.

针对上述问题,本文使用了 01 背包问题的思想对问题进行了建模,本文将处理器中有限的可编程空间看做背包问题中背包的总容量,而空间中装入的命令的选取看做是否将物品装入到有限容量的背包中.而背包问题中衡量物品间的差别使用的参数为物品的价值,以及物品的大小成本;那么在此问题中我们如何对命令的价值与命令的大小成本呢,对此本文使用了用户使用命令的频次作为命令的价值,该指标可刻画出用户执行不同应用时对命令使用的需求,使用实现命令的函数段需要的存储空间来表征命令大小成本.

```
Algorithm 1 PLOUN
Require: n: number of commands; V[n]: value of commands;
   C[n]: cost of commands; C: memory capacity;
Ensure: A set s maximizing total value V under Input
   capacity C
 1: for node i < n do
 2:     F[i][0] = 0
 3: end for
 4: for W w < c do
 5:     F[0][w] = 0
 6: end for
 7: for index i from 1 to n do
 8:     for k from 1 to C do
 9:         F[i][k] = F[i−1][k]
10:         if (thenk ≥ C[i]);
11:             F[i][k] = max(F[i][k],F[i−1][k−C[i]]+V[i])
12:         end if
13:     end for
14: end for
15: i = N
16: j = C
17: k = 0
18: white i>0 && j>0 do
19:     if F[i][j]==F[i−1][j]−C[i]+v[i] then
20:         S[k] = i;
21:         k = k+1;
22:         j = j−C[i]
23:     end if
24:     i = i−1;
25: end white
```

Fig.2 Command set filtering algorithm PLOUN

图 2 命令集筛选算法 PLOUN

图 2 为本文提出的命令集筛选算法(program library based on user needs,简称 PLOUN)的伪代码.首先,算法的输入包括:(1) 系统中记录的用户使用各命令的频次,该项用来表征命令对用户的价值;(2) 各命令占用的可编程空间,该项用来表征命令装入到处理器上时,处理器牺牲的成本;(3) 处理器的总存储容量;(4) 命令的个数.其次是算法的初始化部分,算法需要开辟 $n\times l$ 的二维数组,其中 n 表示命令的个数,l 表示处理器总存储容量,在算法中二维数组中的项 dp[i][j]的第 1 个维度 i 表示命令号,第 2 个维度表示存储容量,该项标识在总存储容量为 j 时,从前 i 个命令中进行选择,应用列表在存储器中命令解释器上执行的命中命令数.再次是算法的中心执行过程,即对各命令依次进行遍历,最终求出 dp[n][l],即总存储容量为 l 时从前 n 个命令中筛选出命令集使得应用列表在存储器中命令解释器上执行的命中命令数最大,dp[n][l]即为最大的命中数.最后是在找到最大的命中数后,我们需要确定出达到该值时,所使用的各个命令,我们需要对二维数组从 dp[n][l]开

始进行回溯,一次找到各个命令,并添加到结果列表中,该列表即为我们筛选出的命令集.

该部分对算法的时空复杂度进行分析,由上文分析可知该算法可转化为 01 背包问题[19],通过对算法伪代码进行分析,可得出该算法的时空复杂度均为 O(n×l),其中 n 表示命令的个数,l 表示处理器总存储容量;在我们使用的 Arduino Yun 设备中,存在一个处理器的可编程空间受限,其可用的编程空间仅有 16KB.因此,时空复杂度均为 13×16 000,设备有能力执行完成.但随着命令集的扩大,n 会不断增大,可能会存在一些设备编程空间几百 KB,而仍然不足以装入完整的命令集,此时空间复杂度就会增大很多,导致设备执行该算法过程中会出现问题.为了应对这种情形,我们提出了一种使用命令占用空间的行数来表征总存储容量的方法,我们首先对现有系统中的函数进行了测试,并统计出了 1KB 大致可以容纳多少行代码,并给出了最大可编程的代码行数,此方法在空间较小时不适用,因为各命令的差异可能比较大,定制出的命令集总空间与处理器可用编程空间的差别大,但是在空间大、存储命令多时,由大数定理可知,定制出的命令集总空间与处理器可用编程空间的偏差小,结果会更稳定.

4.2 命令解释系统运行时自适应方案

该部分的主要工作为解决命令解释器需要手动进行更新的问题,实现了在系统运行时自动完成处理器中命令解释器根据用户习惯进行自适应.该方案的流程如下:

我们为分发到受限处理器上命令的可执行率设定了一个最低允许值.当用户在执行一定数量的应用时,若应用中命令分发的结果,在受限处理器上可执行的概率小于该允许值时,系统即启动命令解释器的自适应工作,系统首先根据收集到的用户历史使用命令情况完成命令集筛选,根据筛选结果生成新的命令解释器,最后将新的命令解释器更新到具体处理器上的过程.

为支持上述中根据筛选结果生成新的命令解释器的过程,我们在云端服务器构建了一个根据 PLOUN 算法的执行结果来进行处理器中命令解释器自动组合生成的模块,该模块主要包含两部分:(1) 各命令对应的函数组合,其中命令的名称与函数名称一致,命令的参数列表即为函数的参数列表;(2) 各命令与各命令对应函数的映射关系添加.该模块基于 JAVA 语言进行实现.

最后,在云端服务器为异构多处理器设备提供了命令解释器更新的接口,该接口以 PLOUN 算法的执行结果为请求参数,当从异构多处理器设备发起请求后,云端服务器接受到 PLOUN 算法的执行结果,并根据该结果完成命令解释器的自动组合生成.新的模块组合生成后,服务器通过网络烧录的方式烧录到处理器上,我们还可以使用物联网重编程方法对此处进行优化,来减少传输的代码量,如 TinyOS 物联网系统的重编程方法,Contiki 物联网系统中的 Deludge 方法等,出于篇幅问题本文不对此问题进行深入探讨.

5 综合验证平台

本节对本系统的进行综合验证.

5.1 系统的具体实施细节以及平台的适用性

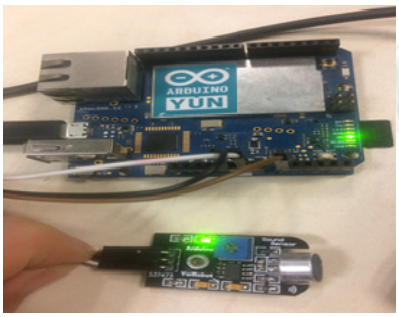


Fig.3 Arduino Yun board
图 3 Arduino Yun 板

在本实施方案系统使用了 Arduino Yun 作为研究的异构多处理器设备.如图 3 所示,Arduino YUN 包含 Atmega32u4 处理器和 Atheros AR9331 处理器,其中 Atmega32U4 为 AVR Arduino 控制器,其 Flash 大小仅为 32kB;Atheros AR9331 处理器为 MIPS 架构的处理器,拥有 16M 的 Flash 存储空间,并且在 Arduino YUN 的处理器间存在一个方便两者通信的 Bridge 桥接库[18].本系统中 Atheros AR9331 处理器上的命令解释器为全功能模块,可以解析执行系统中的所有命令,使用 C++语言实现;而 Atmega32u4 处理器由于 Flash 中可编程空间受限,实际可用空间仅有 16kB,所以该处理器上的命令解释器为非全功能模块,通过 Arduino(C)语言进行实现.云端服务器部分本

系统使用了 Spring 框架开发完成,数据库部分使用了 MySQL 数据库.

5.2 平台的实验测试

我们在此平台上进行了 30 个应用程序的开发工作,并对 30 个应用进行了实验测试.实验展示这些命令可以在系统中正常的运行,图 4 为在系统测试时的部分应用列表.图中的$表示该命令指定在哪个处理器上执行,一些 IO 类命令是在固定处理器上执行的,*表示命令间的引用关系,用于建立邻接表,生成命令的 DAG 图.命令间以分号进行分割,每条命令的冒号后为命令的参数列表,各参数间使用逗号进行分隔.

```
1、http://myArduinoYun.local/shell/ collect: $1,10,1, 10000;
compress:*1; network: $2,data/temp, *2;
2、http://myArduinoYun.local/shell/ collect: $1,10,2, 10000; filter:*1,3;
compress:*2; network: $2,data/temp, *3
3、http://myArduinoYun.local/shell/ collect: $1,10,1, 10000; filter:*1,2;
display: $2, 11,*2
4、http://myArduinoYun.local/shell/ collect: $1,10,2, 10000; filter:*1,2;
data_clean:*2,1; max:*3;display:$2,*4,1;
5、http://myArduinoYun.local/shell/ collect: $1,10,2, 10000; clean:*1,1;
compress:*2; network: $2,data/hum, *2
6、http://myArduinoYun.local/shell/ collect: $1,10,1, 10000; filter:*1,2;
network: $2,data/temp, *2
7、http://myArduinoYun.local/shell/ collect: $1,10,1, 5000; clean:*1,1;
network: $2,data/temp, *2
8、http://myArduinoYun.local/shell/ collect: $1,10,3, 2000;
normalize:*1,1; even:*2; display: $1, 11,*2
9、http://myArduinoYun.local/shell/ collect: $1,10,3, 10000;
normalize:*1,1; sort:*2; network: $2,data/hum, *2
10、http://myArduinoYun.local/shell/ collect: $1,10,3, 10000;
normalize:*1,1; min:*2; display: $1, 11,*2
```

Fig.4 Partial applications used in exprienmt

图 4 实验测试的部分应用列表

本文提出了两个基于贪心算法设计的命令集筛选算法与本文中提出的 PLOUN 进行了对比.其中:

GMV(Greedy-Max value)算法总是贪心地筛选出当前命令集合中价值最大的命令,即用户在处理器上执行频率最高的命令,直到处理器的存储空间不足以装入下一条命令为止,将所有选出的命令组合为命令集.

GMCP(Greedy-Max cost performance)算法总是贪心地筛选出当前命令集合中性价比最高的命令.即用户在处理器上执行频率与占用空间之比最高的命令,直到处理器的存储空间不足以装入下一条命令为止,将所有选出的命令组合为命令集下面对实验对比情况进行说明:

图 5 为 30 个应用在系统中运行时,各命令分发到 Atmega32U4 上的频次,代表了用户在 Atmega32U4 处理器上使用各命令的频次.图 6 为 13 条命令在 Atmega32U4 上实现时,需要占用的代码行数,即为表示将该命令装入到 Atmega32U4 上的命令解释器时所付出的存储代价.在代码书写时,我们规定每行代码的长度不应超过 60 字节.图 7 为应用在由 3 个不同的算法筛选出的命令集上执行时分发到 Atmega32U4 处理器上的命令可执行率的比较,命令可执行率定义请见第 4.1 节的第 1 段.相比于两种贪心算法 GMV 与 GMCP,算法 PLOUN 具有明显的优势.相比于 GMV 算法筛选出的命令集,应用运行在支持 PLOUN 算法筛选的命令集的命令解释器上时命令的可执行率提升了 13%;相比于 GMCP 算法筛选出的命令集,应用运行在支持 PLOUN 算法定制的命令集的命令解释器上时命令的可执行率提升了 4%,即使用此算法后,命令的命中率得以提高,从而命令分发算法所分发出的命令分发结果可以在此算法定制出的命令集上更好地满足命令执行情况.

本文又设计了不同应用数量的两组实验测试了算法的表现.一组实验为从上文中的 30 个应用程序中可重复地随机选取了 50 个应用程序进行实验,另一组实验为从上文中的 30 个应用程序中可重复地随机选取了 100 个应用程序进行实验.图 8 为在 3 组实验中,3 个命令集筛选算法的表现情况.算法 PLOUN 相比于算法 GMV 与算法 GMCP 具有更好的性能.比于 GMV 算法筛选出的命令集,应用运行在支持 PLOUN 算法筛选的命令集的命令解释器上时命令的可执行率平均提升了 9.3%;相比于 GMCP 算法筛选出的命令集,应用运行在支持 PLOUN 算法定制的命令集的命令解释器上时命令的可执行率提升了 3.7%.

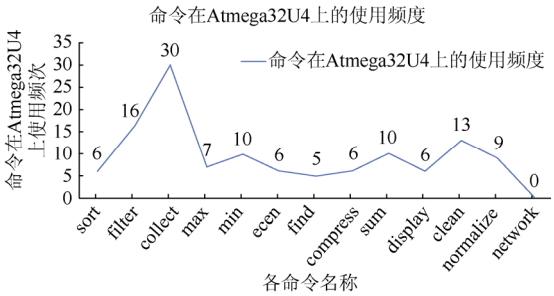


Fig.5 Frequency of command distributed to the Atmega32U4 processor

图 5 命令分发到 Atmega32U4 处理器上的频次

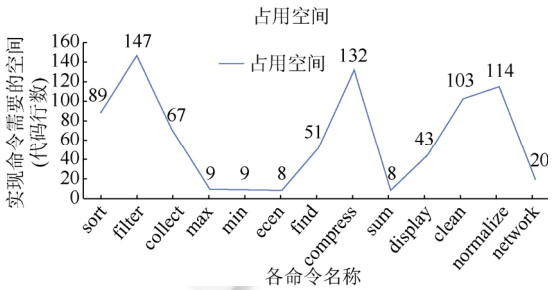


Fig.6 The number of block lines used to interpret each command

图 6 用于解释各命令的程序段代码行数

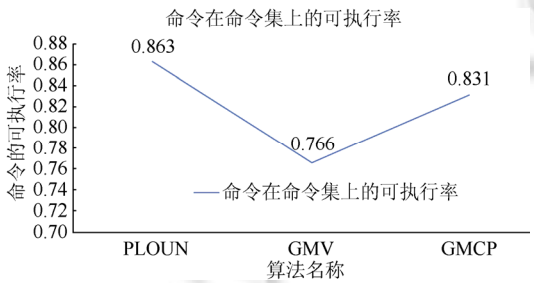


Fig.7 Comparison of executable rates among three algorithmic commands

图 7 3 种算法命令可执行率的对比

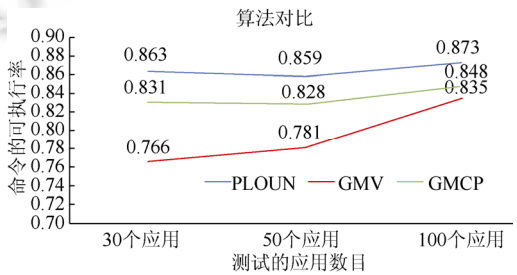


Fig.8 Comparison of executable rates among three algorithmic commands inthree experiments

图 8 3 组实验中 3 种算法命令可执行率的对比

6 总 结

本文设计实现了一种面向异构多处理器设备的命令解释系统.该系统允许用户通过 URL 访问的形式把物联网应用安装到设备上,无需用户考虑资源在多处理器上的调度问题.其次,该系统可根据不同用户习惯进行系统的自适应,系统具有普适性的特点.实验结果表明,通过 URL 访问的形式把物联网应用安装到设备上,同时针对不同用户需求的问题,用户应用在由本文提出的 PLOUN 算法定制出的命令解释器上执行时,命令的可执行率更高.因此,本文的系统是可行且有价值的.

References:

[1] Han CC, Goraczko M, Helander J, Liu J, Priyantha NB, Zhao F. CoMOS: An operating system for heterogeneous multi-processor sensor deviees. https://www.researchgate.net/publication/228467535_CoMOS_An_Operating_System_for_Heterogeneous_ Multi-Processor_Sensor_Devices

[2] ArduinoBoardYun. https://www.arduino.cc/en/Main/ArduinoBoardYun

[3] Zhang JY, Pan ZH, Schaumont P, Yang YL. Application design and performance evaluation for multiprocessor sensor nodes. In: Proc. of the IEEE WCNC'14 Track 4 (Services, Applications, and Business). IEEE, 2014.

[4] Zhang JY, Iyer S, Zheng XW, Schaumont P, Yang YL. Hardware-Software co-design for heterogeneous multiprocessor sensor nodes. In: Proc. of the IEEE Global Commun. Conf. 2014. 20–25.

[5] Sutton F, Zimmerling M, Da Forno R, Lim R, Gsell T, Giannopoulou G, Ferrari F, Beutel J, Thiele L. Bolt: A stateful processor interconnect. In: Proc. of the 13th ACM Conf. on Embedded Networked Sensor Systems. 2015.

[6] Apache a Linux Foundation, Zephyr OS, 2017. https://www.zephyrproject.org/doc/index.html

[7] Baruah S. Feasibility analysis of preemptive real-time systems upon heterogeneous multiprocessor platforms. In: Proc. of the 25th IEEE Int'l Real-Time Systems Symposium (RTSS 2004). Washington DC: IEEE Computer Society, 2004. 37–46. http://dx.doi.org/10.1109/REAL.2004.20.

[8] Baruah S, Fisher Nathan. The partitioned multiprocessor scheduling of deadline-constrained sporadic task systems. IEEE Trans. on Computers, 2006,55(7):918–923.

[9] Valente P, Lipari G. An upper bound to the lateness of soft real-time tasks scheduled by EDF on multiprocessors. In: Proc. of the 26th IEEE Int'l Real-Time Systems Symp. Los Alamitos: IEEE Computer Society Press, 2005. 311–320.

[10] Bertogna M, Cirinei M, Lipari G. Improved schedulability analysis of EDF on multiprocessor platforoms. In: Proc. of the 17th Euromicro Conf on Real-Time Systems. Los Alamitos: IEEE Computer Society Press, 2005. 209–218.

[11] Swiecicka A, Seredynski F, Zomaya AY. Multiprocessor scheduling and rescheduling with use of cellular automata and artificial immune system support. IEEE Trans. on Parallel and Distributed Systems, 2006,17(3):253–262.

[12] Omidi A, Rahmani AM. Multiprocessor independent tasks scheduling using a novel heuristic PSO algorithm. In: Proc. of the IEEE Int'l Conf. on Computer Science and Information Technology. IEEE, 2009. 369–373.

[13] Ferrandi F, Lanzi PL, Pilato C, Sciuto D, Tumeo A. Ant colony heuristic for mapping and scheduling tasks and communications on heterogeneous embedded systems. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 2010,29(6):911–924.

[14] Kaur R, Singh G. Genetic algorithm solution for scheduling jobs in multiprocessor environment. India Conf., 2012,48(11): 968–973

[15] Omara FA, Arafa MM. Genetic algorithms for task scheduling problem. Journal of Parallel & Distributed Computing, 2010,70(1): 13–22.

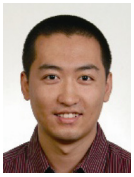
[16] Deng Y, Cheng XH. A heterogeneous multiprocessor task scheduling algorithm based on SFLA. World Automation Congress. 2016. 1–5.

[17] Mok AK, Chen D. A multiframe model for real-time tasks. IEEE Trans. on Software Engineering, 1997,23(10):635–645.

[18] EMUI. http://www.emui.com/plugin/whatsnew/whatsnew

[19] 01 knapsack problem. http://love-oriented.com/pack/P01.html

刘文卿(1992－),男,山东东营人,硕士,主要研究领域为异构多核物联网设备体系结构.



李栋(1979－),男,博士,副研究员,主要研究领域为物联网基础软件和应用系统.

崔莉(1962－),女,博士,研究员,博士生导师,主要研究领域为传感器网络,物联网关键技术.