



基于调度器的物联网设备能耗优化策略

赵婉芳¹, 韩勇^{2,3}

(1. 北京电子科技职业学院, 北京 100176;
2. 北京邮电大学, 北京 100087; 3. 联通信息导航有限公司, 北京 100032)

摘要: 针对物联网边缘节点能源消耗需求快速增长问题, 提出一种调度器优化策略, 整体降低边缘节点能耗。在典型的物联网多核终端系统的电源管理架构及能耗特性的基础上, 分析现有 Linux 系统电源管理存在的不足, 提出改进调度器策略来优化系统能耗控制, 并给出了该策略的实现过程及实施要点。该策略在多核模拟系统中进行了测试, 结果表明, 在低负载的运行场景, 即 CPU 不需要进行大规模运算处理的情境下, 如数据采集、设备监控等物联网典型应用场景, 该策略节能效果明显。

关键词: 电源管理; 调度器; Linux

中图分类号: TP311.13

文献标识码: A

doi: 10.11959/j.issn.1000-0801.2019043

Energy optimization strategy on the IoT equipment based on the scheduler

ZHAO Wanfang¹, HAN Yong^{2,3}

1. Beijing Polytechnic College, Beijing 100176, China
2. Beijing University of Posts and Telecommunications, Beijing 100876, China
3. Unicom Information Navigation Co., Ltd., Beijing 100032, China

Abstract: A scheduling strategy to optimize energy consumption was proposed to reduce the energy consumption of the edge system in view of the fast growth of energy consumption demand in the internet of things. Based on the analysis of the power management architecture and the energy consumption composition in the typical multi-core terminal system, in view of the shortcomings of the existing system power management module, the scheduler load balancing mechanism was improved to control system energy consumption and the implementation process and implementation points of the strategy were given. The strategy was tested in the multi-core simulation system. The result shows that in low load scenarios, that is, in scenarios that CPU does not need to do data-intensive computing, the energy-saving effect of this strategy is obvious, such as data acquisition, equipment monitoring and other typical internet of things applications.

Key words: power management, scheduler, Linux

1 引言

随着物联网边缘设备的处理能力、存储器和

网络数据传输量的迅速增加, 设备对更高功耗的需求也随之增加, 受环境因素的影响, 大部分边缘设备能量获取受到较大制约, 因此边缘设备的

能耗问题日益突出。目前存在的动态电源管理策略主要有超时策略、预测性策略和基于随机模型的电源管理策略。超时策略主要是通过时间阈值的设定,进行休眠模式的切换,应用比较广泛,但节能效果不佳。随后,动态电源管理(dynamic power management, DPM)的工作主要集中于预测策略^[1],通过某种预测方法预测 PMC (power management controller) 下一个空闲时间长度值,一旦预测值大于阈值,立即使 PMC 转入低功耗状态,从而避免了超时策略等待超时的功耗开销,但是也存在预测值与实际情况不符的问题。后来参考文献[2]提出了一种基于离散时间马尔可夫决策过程(discrete-time Markov decision process, DTMDP)的电源管理方法,该方法使用数学模型能更加准确地预测系统的下一状态,从而优化能耗。由于离散时间模型需要在周期时间实例上进行策略评估,并且即使系统状态没有发生变化也会消耗大量的功率。为了克服这一缺点,参考文献[3]中提出了一种基于连续时间马尔可夫决策过程(continuous-time Markov decision process, CTMDP)的模型。该模型下的策略改变是异步的,因此更适合作为实时操作系统环境的一部分来实现,只是 CTMDP 假设系统在任何状态的逗留时间都服从指数分布的理想状态,实际情形下不满足该假设。于是参考文献[4]提出了用于系统建模的时间索引半马尔可夫决策过程(time-indexed semi-Markov decision process, TISMDP),避免了指数分布假设所带来的限制,但模型的参数统计比较复杂,需要较大的计算量。

以上算法中,随机模型算法在理论上有较好的节能效果。但是在实际的嵌入式系统中,系统的工作负载类型多种多样,很难找到一种通用的随机算法模型。另外,这类算法的计算开销较大,这些因素使得这类算法难以在嵌入式系统中应用。随着多核芯片在嵌入式系统的应用,电源管

理问题变得更加复杂,如何协调多个 CPU 核的工作状态来优化系统能耗是目前面临的热点问题。本文从进程调度器策略入手,提出一种在多核系统中实现能耗优化的策略,改进现有调度器,将能耗因素加入任务调度决策过程中。该策略基于 Linux 系统,通用性较强,可适用于多种硬件平台,算法计算量小,不会大幅增加系统额外开销,同时改善系统响应时延。

2 ARM 多核系统电源管理问题

2.1 ARMv8 电源管理架构

一个典型的 ARMv8 多核系统的核心主要由 Cortex-A CPU 构成^[5],其中包含两个 cluster (群组),每个 cluster 中有两个 cortex-A CPU,每个 CPU 有单独的时钟开关,并且可以被完全关闭。当时钟被关闭后,CPU 子系统依靠很小的电流就可以保持内部状态,这样当时钟再次被打开时,系统可以很快恢复到之前的状态。当 CPU 的电源被关闭时,软件需要预先保存该 CPU 的状态;当 CPU 被再次供电时,CPU 复位使电路完成初始化,接着软件根据之前保持的状态数据恢复 CPU,使它恢复到原来的工作状态。当进入高级休眠模式时,CPU 的状态被保存在低功耗存储器中,CPU 中大部分的组件可以关闭供电。当退出该模式时,CPU 恢复供电,CPU 的状态可以从低功耗存储器中快速恢复。这些过程都由硬件电路完成,对软件是透明的。

表 1 中显示了 CPU 在各种 idle (空闲) 状态下,时钟和电源的情况。

表 1 CPU idle 状态

	D ₀	D ₁	D ₂	D ₃	D ₄
核时钟	on	off	off	off	off
核电源	on	on	off	off	off
群组时钟	on	on	on	off	off
群组电源	on	on	on	on	off

表 1 中每种 idle 状态都有不同的进入和退出



时延,消耗的功耗也不同。 D_0 是工作状态; D_1 是最浅的空闲状态,时延也是最小的; D_4 是最深的空闲状态,这个状态下CPU核和群组的电源都被关闭,达到最低功耗。进入和退出各个idle状态会导致时延并消耗额外的能量。这里的时延不仅包括硬件的处理时间,也包括软件的处理时间。进入或退出每个状态硬件和软件都需要做很多工作,如等待电压稳定、等待时钟稳定、刷新L2缓存、保存或恢复CPU状态等。

在该CPU子系统中,单个CPU从属于cluster。即如果关闭某个cluster的时钟,那么整个cluster都会停止工作,包括其中所包含的CPU。因此这个cluster内的所有CPU都需要关闭时钟或电源。当要关闭某个cluster的电源时,这个cluster内的所有CPU核都被断开电源。

2.2 能耗分析

在多核的系统中,CPU系统的功耗主要由动态功耗PD和静态功耗PS构成。当工艺固定时,动态功耗PD的量值与供电电压的平方及运行频率成正比。而静态功耗PS则主要由电压和漏电流决定。在实际的物联网设备中,为了简化电路、降低成本,使用固定电压的供电电路,CPU工作在一个固定电压下,频率可以通过内部PLL电路进行调整。

假设芯片中每个CPU都是一致的,而且CPU的供电电压固定,单个CPU可被完全关闭(切断单个CPU核的电源)。在这种情况下,芯片中第*i*个CPU核的静态功耗为 PS_i ,当CPU核供电时为 $PS_i=PS$,当CPU核断电后 $PS_i=0$ 。第*i*个CPU核的动态功耗 PD_i 可写为: $PD_i=a_iPD$,其中 a_i 为CPU的利用率,PD为CPU利用率为100%时的动态功耗,*i*是CPU核的编号。所以,系统中所有CPU核的功耗可表示为:

$$P = PD \sum_i a_i + \sum_i PS_i \quad (1)$$

其中,对CPU性能有贡献的部分是动态功耗部分

$PD \sum_i a_i$,它与CPU利用率相关。静态功耗 $\sum_i PS_i$

对性能没有贡献,它与CPU利用率无关,CPU核只要加电就会产生。在不影响性能的情况下,为了降低总功耗 P ,可以通过减小静态功耗 $\sum_i PS_i$ 来实现。

物联网设备大部分用于数据采集、监控、传感器等应用,设备大部分时间处于休眠模式,CPU利用率较低,静态功耗对总功耗的贡献很大。尤其是在多核系统中,CPU核较多,消耗的静态功耗也比较大。现代微处理器可以关闭单个CPU供电,甚至关闭整个CPU cluster的供电。因此如果CPU利用率较低,将任务集中到少数CPU核上,完全关闭空闲CPU核(使CPU核处于断电状态),则可以降低CPU系统静态功耗,由此降低系统总功耗。

2.3 Linux多核系统电源管理问题

典型的系统工作状态下,CPU大部分时间处于空闲状态,而此时系统如果由空转进入任何低功耗状态都会很大程度上降低能耗,减少热量产生,但同时不可避免地会产生唤醒时延。在Linux系统中,由cpuidle governors模块预测CPU可能在空闲状态中停留的时间,然后将该时间和备选状态中的停留时间字段(target residency)相比较,选取预测停留时间大于该字段的空闲状态,以此来保证避免产生过度的时延。而当系统处于活跃状态时,Linux中的cpufreq模块将对系统负载进行监控,并根据系统负载自动升降工作频率,然而目前大多数cpufreq驱动,甚至CPU频率调节算法只允许CPU频率设置为预定义的固定值,为了提供动态频率调节,cpufreq的核心必须能够告诉这些驱动一个“目标频率”,而该目标频率只是系统根据历史运行情况给出的一个预测值。

在以上电源模块工作过程中,governor子模块根据CPU利用率、唤醒间隔等情况决定何时进入何种低功耗状态,在实际应用中,这种预判准

确率并不高。例如,在网络应用中,数据分组到达设备的时间具有随机性,预测算法无法准确预测数据分组到达时间,如果数据分组到达时,CPU处于低功耗状态,系统将通过中断唤醒CPU来处理数据分组。唤醒过程需要时间,更严重的是当大量数据分组到达时还会导致分组丢失,引入网络时延,降低网络的传输性能。因此,当预期与实际情况不符时,容易引发系统性能大幅下降,影响系统可用性。

另外,从整个系统的角度来看,每个CPU核有各自独立的cpufreq governors和cpuidle governors电源管理模块,它们各自独立运行,被动地根据CPU运行情况预判CPU将进入何种运行状态,缺少全局协调的机制,只实现了单核局部的能耗优化,没有做到多核之间的全局优化。而操作系统中的进程调度器掌握所有任务的信息,并且具有控制能力,如对其进行能耗优化方面的改进可增强系统的整体能耗控制能力,下面将详细说明。

3 基于调度器的能耗优化策略

3.1 能耗优化策略

在操作系统中,进程调度器处于核心位置,全面掌握当前系统的状态,并且能更加准确地预测未来的任务需求,其控制着系统任务切换、任务唤醒、CPU时间分配等。Linux系统采用了完全公平调度器(completely fair scheduler, CFS),它尽可能公平地让每一个任务都得到执行时间,尤其是那些长时间睡眠的任务。CFS的主要目标是实现CPU利用率最大化和最好的交互性能,所以在服务器和桌面环境下表现优秀。而在物联网场景下,设备大部分时间处于休眠模式,且受环境因素影响,大部分边缘设备能量获取受较大制约,功耗和响应更加重要。CFS调度器没有考虑系统功耗因素,即使系统负载很低,调度器也会将任务分配到每个CPU,即每个CPU都处于加电

状态,这种状态下系统静态功耗较高。如果将能耗因素加入调度器的决策过程中,在同等任务下,通过使边缘设备系统尽量关闭不使用的CPU降低静态能耗,不仅改进了CFS调度器策略在能耗方面的不足,而且使系统能耗管理由原来的被动节能模式变为主动地能源控制,从而获得更佳的节能效果和响应能力。具体优化措施如下。

- 关闭没有任务运行的CPU,最大程度降低系统静态功耗。
- 负载平衡机制根据CPU利用率将任务迁移到尽量少的CPU上,让其他CPU尽可能进入较深的关闭状态。将关系较近的CPU分配为同一个调度域,当有新任务需要唤醒休眠的CPU时,优先唤醒关系较近的CPU。
- 设定CPU利用率阈值,一个是最低利用率阈值(UL),一个是最高利用率阈值(UH)。当CPU利用率低于最低利用率阈值时,将该CPU上的任务迁移到CPU利用率低于最高利用率阈值的CPU上,并将该CPU关闭。当CPU利用率阈值高于最高利用率阈值时,将该CPU上部分任务迁移到同一调度域的其他CPU上。

该优化策略增强了调度器在系统功耗上的控制,尤其是在系统负载比较低时。这时系统中的任务会被聚集到尽量少的CPU上,让其他CPU进入长时间的空闲状态,降低能耗。如果硬件支持,可切断长时间空闲CPU的电源,大幅降低CPU的静态功耗。同时,较多的任务聚集在工作状态的CPU上,降低了这些CPU进入深度睡眠状态的概率,当有事件需要处理时,CPU能较快地响应,因此改善了系统响应。与原有的Linux电源管理方法相比,该策略由原来的被动节能模式改为主动节能模式,通过优化调度器策略,主动地在CPU之间调配任务,不仅节省了静态能耗,也改善了系统响应;同时它使得CPU的电源管理



问题不再是 CPU 内部的状态转换问题,而是从系统全局任务管理的角度出发,通过合理调配任务来实现 CPU 能耗控制。此外,该策略针对调度器进行改进,算法计算量小,通用性较好,可适用于多种硬件平台。

3.2 策略实现

Linux 中负载均衡机制是基于调度域来实现的。根据多核系统的 CPU 拓扑结构,系统中的 CPU 被分为多个 CPU group 和 CPU domain,构成一个层状结构。根据调度器策略,负载均衡机制从调度域的叶节点往上遍历,在这些 CPU 组(域)内部或它们之间迁移任务,直到所有调度域中的负载都是平衡的,使得系统达到期望的状态。

改进策略后,首先,当 Linux 系统在创建新任务,新进程或唤醒任务时,调度器可以选择在哪个 CPU 上执行。这样调度器可以调节各个调度域、调度组和 CPU 的负载。该过程是通过调用结构体 struct sched_class 中的函数指针 select_task_rq 实现的。任务将在该函数返回的 CPU 上执行。在 CFS 调度器中,select_task_rq 被赋值为 select_task_rq_fair(),函数 select_task_rq_fair() 执行时,它将从当前 CPU 开始向上遍历整个调度域,查找符合能耗优化策略的 CPU,然后返回该 CPU。具体过程是,使用宏 for_each_domain() 向上遍历调度域,对每个调度域调用函数 select_cpu_for_power(),直到该函数返回有效的 CPU。在函数 select_cpu_for_power() 中,需要先计算当前调度域的负载。遍历该调度域中每一个调度组,将它们的计算能力和利用率分别累加起来。通过计算,判断该调度域是否能容纳当前任务。如果没有空余的计算能力,表示当前调度域不能容纳更多的任务,后续程序将在其他调度域中继续查找。如果有空余的计算能力,将继续在当前调度域中查找适合的 CPU。首先,在当前调度域中找出利用率较高,但还有空余计算能力的调度组;然后,在该调度组中查找利用率较高,但还有空余计算

能力的 CPU。如果有就返回该 CPU;如果没有,就继续在其他调度组中查找。图 1 显示了调度器能耗优化任务分配方式。

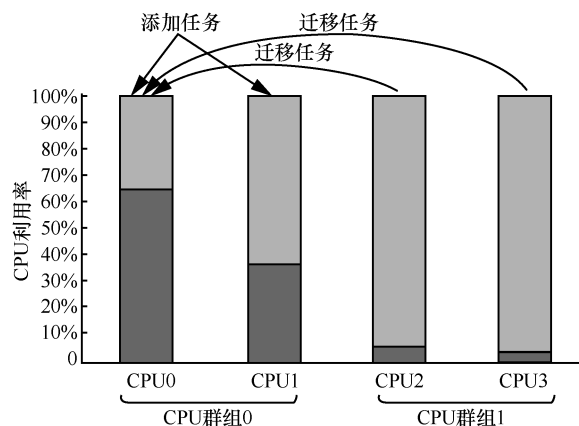


图 1 调度器能耗优化任务分配示意图

其次,当在任务执行过程中, Linux 系统会不断触发软中断 SCHED_SOFTIRQ 来动态监测各个调度域负载平衡。CFS 调度器从当前 CPU 所在的调度域开始,向上遍历,完成调度域内的负载平衡。该任务在函数 rebalance_domains() 中实现,该函数对每个遍历的调度域进行负载平衡。该函数主要完成的工作是:查找当前调度域中不平衡的 CPU(如负载过重或负载过轻),将其上的任务迁移到其他 CPU 上。通过这种方式将任务集中到尽量少的 CPU 上,并且保证各个 CPU 没有过载。主要的实现过程为:

```
void rebalance_domains(this_cpu, idle) {
    for_each_domain(this_cpu, sched_domain) {
        if !(sched_domain->flags & SD_LOAD_
BALANCE)
            continue
        if utilization of this_cpu is more than UH
            break
        group=find_imbalance_group(sched_ do-
main)
        dst_cpu = find_imbalance_queue(group)
        if dst_cpu != this_cpu
```

```

        move_tasks(this_cpu, dst_cpu)
    }
    this_cpu_run_queue->next_balance =
        sched_domain->last_balance + sched_ do-
main-> balance_interval
}

```

另外要注意,为了使 CPU 尽量处于低功耗状态,将空闲 CPU 上的中断绑定到其他 CPU 上,这样就减少了中断唤醒空闲 CPU 的机会。

4 结果测试

测试在 QEMU 仿真器中进行。该仿真器可实现指令级仿真,可采集各种系统运行数据,并能有效控制测试环境。该仿真器支持多种指令架构,包括 x86、PowerPC、ARM 等。它可仿真整个系统,包括 CPU、内存和各种外设,可运行 Linux 系统。在测试中, QEMU 仿真了一个具有 ARMv8 核的多核系统,该系统包含 4 个 CPU 核和 2 GB 内存。该系统中安装了 Ubuntu16.04 系统,运行 Linux 4.4.0 内核。

在测试中,使用计算圆周率的 CPU 压力测试工具。该工具支持多进程,并可设定计算强度。测试程序产生 4 个测试进程,通过调整计算强度,使 CPU 在各种压力下运行。同时,采集仿真器中各个 CPU 的运行状态数据。进行了两组测试,一组使用原来的进程调度器,另一组使用改进后的进程调度器。通过分析测试结果,比较了优化前后系统的能耗变化。为了便于比较两组数据的差异,图 2 中的能耗数据进行了归一化处理。

从图 2 可知,该优化策略在 CPU 利用率低时节能效果明显,但是当 CPU 利用率提高后两种策略没有差别。这个结果与之前的预期相符,当 CPU 利用率低时,未优化的策略会在每个 CPU 上放置 1 个进程,这样 4 个 CPU 核都在工作,无法进入断电模式。而优化后的策略将 4 个进程都放置在

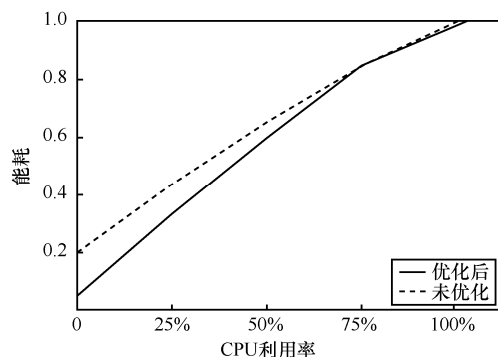


图 2 CPU 能耗比较

一个 CPU 上运行,这样其他 3 个 CPU 就可进入断电状态。所以,优化后节电效果明显。而当计算量提高后,需要比较多的 CPU 参与运算,且利用率都超过了最低利用率阈值, CPU 无法关闭,节电效果不明显。所以,这种进程调度优化策略适用于低负载的运行场景,即 CPU 不需要进行大规模的运算处理,不需要数据采集、设备监控等物联网设备。

5 结束语

本文基于多核系统电源架构及能耗特性,提出优化进程调度器策略,实现能耗优化的目的,改进了现有进程调度策略在能耗方面的不足。该策略算法开销小,仅对系统的内核层进行改进,完全兼容现有的应用程序,通用性较强。实验结果表明,该策略在负载较低的多核系统中效果明显,适用于低功耗的物联网终端应用。下一步的研究中,如果操作系统能更加准确地跟踪每个任务的运行模式以及能耗特性,那么调度器将能更加合理地分配任务到每个 CPU,使能耗进一步优化,这也是今后的工作重点之一。

参考文献:

- [1] SRIVASTAVA M, CHANDRAKASAN A, BRODERSEN R. Predictive system shutdown and other architectural techniques for energy efficient programmable computation[J]. IEEE Transactions on VLSI Systems, 1996(4): 42-55.
- [2] BENINI L, PALEOLOGO G, BOGLIOLO A, et al. Policy optimization for dynamic power management[J]. IEEE Transac-



- tions on Computer-Aided Design, 1999(18): 813-833.
- [3] QIU Q, WU Q, PEDRAM M. Stochastic modeling of a power-managed system-construction and optimization[J]. IEEE Transactions on Computer-Aided Design, 2001(20): 1200-1217.
- [4] SIMUNIC T, BENINI L, GLYNN P, et al. Event-driven power management[J]. IEEE Transactions on Computer-Aided Design, 2001(20): 840-857.
- [5] 韩东升, 郑冰, 陈智雄. 基于能量损耗与能源代价均衡的基站休眠机制[J]. 电信科学, 2017, 33(7): 9-18.
HAN D S, ZHENG B, CHEN Z X. Base station sleep mechanism based on power consumption and energy cost balance[J]. Telecommunications Science, 2017, 33(7): 9-18.
- [6] 孙毅, 王鸿, 宋荣方. 异构网络中两种边缘缓存策略的能耗分析[J]. 电信科学, 2018, 34(1): 17-26.
SUN Y, WANG H, SONG R F. Analysis of energy consumption of two kinds of edge caching strategies in heterogeneous networks[J]. Telecommunications Science, 2018, 34(1): 17-26.
- [7] 卢彩玲, 张静怡, 姚帅. 移动核心机房节能健康度分析与评价[J]. 电信科学, 2016, 32(Z1): 250-254.
LU C L, ZHANG J Y, YAO S. Analysis and evaluation of energy saving health of mobile core computer room[J]. Telecommunications Science, 2016, 32(Z1): 250-254.
- [8] MAITERTH M, WILDE T, LOWENTHAL D, et al. Power aware high performance computing: challenges and opportunities for application and system developers—survey & tutorial[C]//2017 International Conference on High Performance

Computing & Simulation (HPCS), July 17-21, 2017, Genoa, Italy. Piscataway: IEEE Press, 2017: 3-10.

- [9] MAITERTH M, KOENIG G, PEDRETTI K, et al. Energy and power aware job scheduling and resource management: global survey—initial analysis[C]//2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), May 21-25, 2018, Vancouver, British Columbia, Canada. Piscataway: IEEE Press, 2018: 685-693.

[作者简介]



赵婉芳(1975—), 女, 北京电子科技职业学院副教授, 主要研究方向为物联网嵌入式开发及网络软件应用与开发等。



韩勇(1980—), 男, 北京邮电大学博士生, 联通信息导航有限公司高级工程师, 主要研究方向为物联网开发、大数据的应用及开发。