Software Engineering and Applications 软件工程与应用, 2019, 8(5), 238-244
Published Online October 2019 in Hans. http://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2019.85029

Hans汉斯

Research on Real-Time Operating System Virtualization Technology for Industrial Internet

Wenbing Chen*, Chao Liang

School of Information, Zhejiang Sci-Tech University, Hangzhou Zhejiang
Email: *1763969403@qq.com, 651835488@qq.com

Received: Oct. 2nd, 2019; accepted: Oct. 15th, 2019; published: Oct. 22nd, 2019

Abstract

Industrial Internet uses cloud computing technology to provide enterprises with high-reliability and low-cost information services, which is an indispensable part of the industrial information revolution. However, the existing operating systems cannot support the smooth migration of the core logic of industrial control to the Internet for processing. The core reason is that the existing industrial Internet does not provide complete real-time support for sensing, transmission, processing and control processes, so it cannot guarantee the real-time performance of all links and systems as a whole. Real-time operation system is one of the important links to guarantee the whole real-time performance of the system. This paper improves the real-time performance of Linux system, including process scheduling, interrupt processing, high-precision clock, real-time lock and so on. It reduces the average delay of Linux system under pressure test to less than 20 μs. It can support virtualization environment, so that industrial control applications can be guaranteed real-time in virtualization environment.

Keywords

Real-Time, Operating System, Virtualization

面向工业互联网的实时操作系统虚拟化技术研究

陈文兵*，梁　超

浙江理工大学信息学院，浙江　杭州
Email: *1763969403@qq.com, 651835488@qq.com

*通讯作者。

文章引用: 陈文兵, 梁超. 面向工业互联网的实时操作系统虚拟化技术研究[J]. 软件工程与应用, 2019, 8(5): 238-244.
DOI: 10.12677/sea.2019.85029

收稿日期：2019年10月2日；录用日期：2019年10月15日；发布日期：2019年10月22日

摘　要

工业互联网利用云计算技术为企业提供高可靠、低成本的信息服务，是工业信息化革命中不可或缺的重要组成部分。然而，现有的操作系统无法支持将工业控制中的核心逻辑顺利迁移至互联网上处理，其主要原因是现有的工业互联网没有为传感、传输、处理和反控等过程提供完备的实时性支持，因此无法保证各个环节与系统整体的实时性。操作系统的实时性是系统整体实时性保障的重要环节之一，本文对Linux系统实时性进了改进，包括进程调度、中断处理、高精度的时钟、锁的实时性等方面，使Linux系统在压力测试下平均延时减小到20 μs以内。并且能够支持虚拟化环境，从而实现工业控制应用能够在虚拟化环境中得到实时性保证。

关键词

实时性，操作系统，虚拟化

1. 引言

工业互联网的概念首先由通用电器公司首先提出，以云计算、工业大数据分析为特征的工业互联网技术呈现出广阔的发展前景[1] [2]。美国、德国等西方发达国家和信息产业强国也对工业互联网的前景十分看好。德国率先提出工业 4.0，并将工业互联网作为工业 4.0 战略的重要支撑。美国于 2012 年提出“先进制造业国家战略计划”，是对工业互联网思路的全面阐释。中国也将工业互联网作为未来产业发展的战略重点，出台一系列的政策支持措施，进而达到抢占工业互联网市场空间和产业发展的制高点的目的。GE，施耐德，西门子等公司凭借其在工业领域的沉淀积累以及应用信息技术改造传统制造业的成功经验，以虚拟化的方式灵活实现跨区域工业信息服务的部署和交付，把数以亿计的终端工业设备连入互联网，通过提供强大的数据传输、存储和处理能力，并为特定的行业提供数字化、网络化、智能化转型的软件应用和服务。亚马逊、微软等公司近年也相继推出了 AWSIoT、Azure IoT 等物联网云平台[3] [4]，由于工业互联网是近几年提出的新概念，企业没能完全理解其在工业应用场景中的作用，平台还在逐渐完善和建设中，还没形成针对特定业务的专门化定制。我国已经意识到和发达国家的差距，并且为此追赶工业互联网的脚步，于 2013 年，提出“6 + 1 专项行动”，探索制造业未来发展方向，实现制造业转型升级。但由于底子薄，发展时间短，我国工业化和信息化水平相对较低，工业企业两化融合尚未达到发达国家的最新标准，仍处在单项覆盖向集成提升过渡阶段。但近几年我国开始重视工业互联网的发展，不断推出新的规划，加大投入，已形成爆发式发展的态势。国内的工业互联网云平台以此为契机，得到了长足的发展，其中生态最完善，技术最先进，应用最为广泛的是航天科工云网、海尔 Cosmoplat 平台，为国民经济发展，实现工业制造转型，先进制造，智能制造做出了巨大贡献[5] [6]。

2. 相关概念与本文贡献

现有的工业互联网的核心问题在于缺少对实时性的支持。工业互联网实时系统的正确性有两个决定

性指标，一是系统计算的逻辑结果，二是系统产生这个结果的时间成本。实时系统最大的目标是适应不同的应用场景，在不同的传输截止时间内将数据安全，无损，同步或异步响应到系统[7] [8]。因此实时系统必须在事先定义的时间范围内处理离散事件的能力，系统能够及时地处理和储存控制系统所需要的大量数据。工业应用对实时性保证的需求包括：时间约束性、可预测性。时间约束性是指实时系统的任务具有一定的时间约束(即截止时间)。根据截止时间，实时系统的实时性分为“硬实时”和“软实时”。可预测性是指系统能够对实时任务的执行时间进行判断，确定是否能够满足任务的时限要求。由于实时系统对时间约束要求的严格性，使可预测性称为实时系统的一项重要性能要求。除了要求硬件延迟的可预测性以外，还要求操作系统的可预测性，包括中断处理程序的响应时间是可预测的，即在有限的时间内完成必须的工作。工业操作系统的可预测性要求实时原语、调度函数等运行开销应是有界的，以保证应用程序执行时间的有界性。

本文改进了 Linux 操作系统的实时性，包括进程调度的实时性、中断处理的实时性、高精度的时钟、锁的实时性等方面。此外，为了提高服务器的利用率和可靠性，需要采用虚拟化技术为不同客户的提供独立的运行环境，并充分利用空闲资源同时为多个客户提供服务。因此，在虚拟化环境中的实时性保证也是工业互联网必须提供的特性。工业互联网实时操作系统的虚拟化技术具有重要的现实意义，通过这项技术，能够将工业控制中对时间敏感的控制过程放至云端进行集中管理，使控制过程之间的交互与合作变得更加便利，同时降低了系统的成本，具有较高的应用价值。

并且本文在 Linux 操作系统内核的实时性改造的基础上使其能够支持虚拟化环境，从而实现工业控制应用能够在虚拟化环境中得到实时性保证。

3. 实时性改进

3.1. 工业互联网操作系统的实时性保证

虽然现有的工业操作系统功能完善且资源利用率高，但是这类系统一开始就被设计成通用操作系统，其目的是尽量缩短系统的平均响应时间，提高吞吐量，注重操作系统的整体功能需求，达到更好的平均意义上的性能。然而工业互联网应用强调工业控制过程的实时性，通用系统其在这方面的就用中存在一定的局限性。本文研究工业互联网云操作系统的实时性保证，通过改造现有的通用操作系统的内核可抢占性、进程调度方式、中断处理机制、时钟粒度等多个方面以支持实时性保证[9]-[14]。

1) 进程调度

实时调度理论是分析任务的可调度性和进行调度的基本理论[15] [16]。任务调度技术主要包括两个方面一是调度策略，二是可调度性分析方法，这两个方面缺一不可，两者结合才能对任务调度做出更好的解决方案。任务调度针对的研究对象涉及任务进行时的系统资源(主要有处理机、内存、I/O、网络等资源)的策略和机制，并且提供了能否事先预测系统性能的方法。例如，任务调度运行的耗时、什么时候开始运行、当系统为多处理机系统或分布式系统时在哪运行的问题等等；以及判断一个添加参数约束的实时任务能不能正确运行和调度。然而一个系统的大部分任务一般都处于阻塞或待命状态，这是因为一个时刻只有一个任务可以得到的 CPU 的执行权。因此，在系统的任务数和调度器类型不同的时候，一般都会有大量项目在待命队列里等待被执行。一般而言，对于简单的时间触发的调度程序，等待任务列表的数据结构应设计为在最坏的情况下最大程度地减少调度程序关键部分的执行时间，以防止其他任务无法及时得到执行机会。因此，抢占式任务要避免在这种类型调度器中应用，有为了保证所有任务及时得到执行甚至将除调度器之外的所有中断都关闭。另一方面，还应根据系统所需的最大任务数，进一步优化等待任务列表的数据结构。如果平时有大量的任务在等待列表里，这样会导致性能下降，这时最好使用双

向链表作为任务列表的数据结构。如果等待任务列表通常包含少量任务，只是偶尔会出现较大量任务等待的情况，那么任务应该根据优先级排序，使系统性能更好发挥。这样，要找到优先级最高的任务，就不必在整个列表中逐个寻找，减少搜索时间。而插入任务需要在列表中逐个寻找，插入到比将要插入的任务的优先级低的任务之前；如果将要插入的任务的优先级是最低的，就插入到任务列表末尾。其中要注意的是在插入任务列表的过程中，要尽量避免抢占的发生。对于运行时间较长的关键部分要拆分成多个小的部分分别执行。若在寻找任务列表以插入低优先级任务时发生了中断，使高优先级任务进入待命状态，那么高优先级任务应该在低优先级任务被插入之前立即执行。Linux 系统提供符合 POSIX 标准的调度策略，包括 FIFO 调度策略(SCHED_FIFO)、带时间片轮转的实时调度策略(SCHED_RR)和静态优先级抢占式调度策略(SCHED_OTHER)。Linux 进程默认采用的调度策略为 SCHED_OTHER，但缺点是，这种调度方式不能保证实时性要求高的任务或者优先级高的任务严格先于其他要求低的任务执行，因为这种调度策略是所有任务公平使用系统资源。显然这不适合有实时性要求的任务来使用。Linux 系统在用户态支持可抢占调度策略，而在内核态却不完全支持抢占式调度策略。因此，将实时进程的调度策略设置为 SCHED_FIFO 或 SCHED_RR，也不能很好地支持实时调度。所以在 Linux 内核态运行的任务(包括系统调用和中断处理)是不能被抢占的，并且会导致优先级逆转问题。为解决这个问题需要内核支持抢占机制，以满足实时性要求。

2) 内核抢占机制

Linux 的系统进程运行分为用户态和内核态两种模式如图 1 所示。这两种模式最大的不同是进程是否可被抢占。运行在用户态的进程，低优先级的进程可能会被较高优先级的进程所抢占。但不同的是当进程运行在内核态时，进程是不能被抢占的，即使优先级高的进程也不能抢占优先级低的进程[17] [18]。这就和实时性的要求产生了矛盾，当进程进入内核态运行时，只有在系统完成前面的任务调用后才有可能让实时性任务得到执行机会。在较新的 Linux 版本中内核是抢占式的。主要有内 3 种抢占模式：PREEMPT_NONE，没有强制性的抢占，整体的平均延时较低，但偶尔也会出现一些较长的延时。它最适合整体吞吐率要求高的应用；PREEMPT_VOLUNTARY，降低延时的第一阶段，它会在内核代码的一些关键位置上放置额外的显示抢占点，以降低延时。但这是以牺牲整体吞吐率为代价的；PREEMPT_DESKTOP，这种模式使内核在除临界区外的任何地方都是可抢占的，对于没有硬实时性要求的任务比较合适。

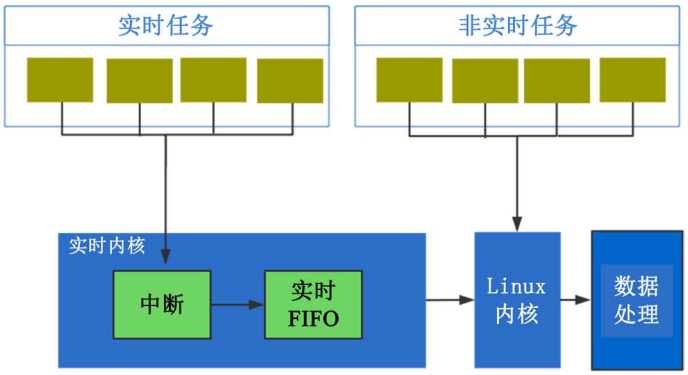

Figure 1. Real-time Linux interrupt mechanism
图 1. 实时 Linux 中断机制

3) 中断屏蔽

默认情况下，Linux 操作系统会在发生中断处理时关闭中断，这是为了在完成任务前不被打断，能更

迅速、更安全地完成自己的任务，但是这时更高优先级的任务中断也无法得到响应，直至当前中断任务处理完毕，才能得系统资源[19]。这对实时性要求高的任务是不可接受的，因为，这种状况下会导致中断延时和调度延时不可确定。通过利用内核线程处理中断就可以解决这种问题，能让优先级高的中断及时上报及时处理。

4) 时钟粒度

时钟粒度是指操作系统能达到的最小时间间隔。在Linux中时钟粒度是指两次时钟中断的时间间隔。时间间隔越小，系统开销就越大。时间间隔越大，虽然开销可以减小，但带来的问题就是进程的响应延迟变大[20]。所以时钟粒度要根据不同的应用场景调整。在 Linux 内核中，时钟中断发生频率范围是50~1200 Hz，周期不小于0.8 ms，遗憾的是这无法达到工业应用中微秒级的实时性要求。但是处理器可以提供更高精度的定时，能够达到纳秒级。因此想要满足对时间精度的较高需求可以利用处理器的时钟。

5) 虚拟内存管理

虚拟内存管理是操作系统的核心之一。操作系统中诸如进程间隔离，文件缓存，存储交换等高级的功能都是依赖于虚拟内存得以实现。这需要频繁的页面换进换出，过程要消耗大量时间。因此应用在实时系统上是不可接受的。一般可以通过内存锁定功能对实时应用的内存空间进行锁定，避免在实时处理中存储页被换出，提高实时性。

6) 共享资源的互斥访问差异

信号量或自旋锁等机制是用来解决同步时数据可能被破坏的问题。然而却带来了另一个弊端，在基于优先级调度的实时系统中，自旋锁可能会造成优先级倒置问题，致使高优先级任务无法保证及时处理。为了使系统更好地应用于工业实时环境，可以将自旋锁改造为支持优先级继承的互斥量以支持不同优先级任务间的抢占。

3.2. 工业互联网虚拟化环境中的任务实时性保证

由于传统的虚拟化环境是一个独立于宿主机的封闭环境，宿主机无法得知它内部进程的优先级，因此无法保证将虚拟化环境中具有高优先级的实时进程先于宿主机进程或其它虚拟化环境中的进程运行。内核提供的锁机制也存在相同的问题，它们对于宿主机也不可见，导致跨虚拟机的优先级调度无法得到保证。

1) 提高虚拟CPU的优先级

解决虚拟化环境中优先级保证的一种方法是提高虚拟CPU的优先级，使虚拟环境中的进程能够先于宿主机进程运行。此时，为了保证外部中断能够传递到虚拟化环境中，宿主机的软中断处理进程的优先级应当高于虚拟CPU的优先级。

2) isolcpus技术

通过isolcpus技术也可以提高任务执行实时性。isolcpus功能用于在SMP均衡调度算法中将一个或多个CPU孤立出来，同时通过亲和性设置将进程置于“孤立CPU”运行，isolcpus后面所跟的CPU参数，可设置孤立的多个CPU。isolcpus带来的好处是有效地提高了孤立CPU上任务运行的实时性。该功能在保证孤立CPU上任务的运行，同时减少了其他任务可以运行的CPU资源，所以需要使用前对CPU资源进行合理的规划。

3) 新的处理器特性

此外，通过利用一些新的处理器特性也有助于实时性的改进。例如，Intel较新的采用了Haswell架构的CPU包含了一项缓存分配(Cache Allocation Technology, CAT)技术，启用该技术后，在任务切换时即使内存和TLB同时都没有未命中，仍然可以保证切换延迟小于50微秒。CAT允许对指定的应用保留部分

缓存，以防止一个工作负载将另一个工作负载驱逐出缓存，这可以使用一个基于控制组的接口进行控制。

4. 对比实验

本实验是在一台非实时的 Linux 内核平台和经过改造的支持实时性的 Linux 内核的平台之间进行对比实验。并且测试使用的是同一台物理机，不存在硬件上的差异。

实验使用的是比较流行 Cyclictest 内核实时性能测试工具进行测试。分别在两个内核上设置了 5 个 Cyclictest 线程，每个 Cyclictest 线程的优先级是 90 并且锁定当前和未来的内存分配。实验是在 loadavg 大于 80 的压力测试情况下进行的，这样更能体现出本文对延时性能改进的效果。如图 2 所示。

```
^C[root@OPCUATEST1 cwb]# cyclictest -t 5 -p 90 -m
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 93.74 62.28 29.63 1/169 3373

T: 0 ( 2959) P:90 I:1000 C: 130521 Min:      1 Act:    1 Avg:    2 Max:      15
T: 1 ( 2960) P:90 I:1500 C:  87011 Min:      1 Act:   11 Avg:    3 Max:      15
T: 2 ( 2961) P:90 I:2000 C:  65256 Min:      1 Act:    1 Avg:    3 Max:      15
T: 3 ( 2962) P:90 I:2500 C:  52203 Min:      1 Act:   11 Avg:    3 Max:      15
T: 4 ( 2963) P:90 I:3000 C:  43501 Min:      1 Act:   11 Avg:    3 Max:      15
^C[root@OPCUATEST1 cwb]#
```

(a)

```
^C[root@OPCUATEST1 cwb]# cyclictest -t 5 -p 90 -m
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 81.06 40.34 17.19 1/137 1352

T: 0 ( 1348) P:90 I:1000 C:  21690 Min:      1 Act:    1 Avg:    3 Max:      39
T: 1 ( 1349) P:90 I:1500 C:  14459 Min:      1 Act:   11 Avg:    4 Max:      26
T: 2 ( 1350) P:90 I:2000 C:  10843 Min:      1 Act:    1 Avg:    3 Max:      27
T: 3 ( 1351) P:90 I:2500 C:   8674 Min:      1 Act:    1 Avg:    3 Max:      34
T: 4 ( 1352) P:90 I:3000 C:   7228 Min:      1 Act:   11 Avg:    4 Max:      23
^C[root@OPCUATEST1 cwb]# cyclictest -t 5 -p 90 -m
```

(b)

Figure 2. Delay contrast test. (a) Real-time kernel test chart; (b) Non-real-time kernel test chart
图 2. 延时对比测试。(a)实时内核测试图；(b)非实时内核测试图

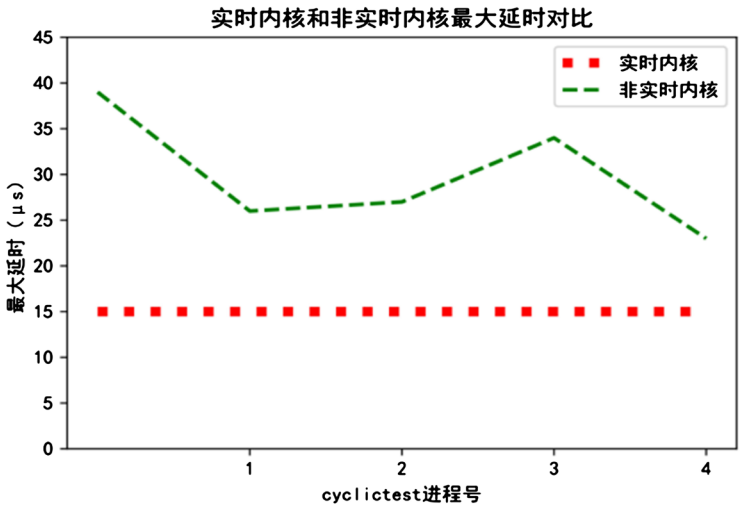

Figure 3. Maximum delay contrast
图 3. 最大延时对比

如图 3 所示，实时性 Linux 内核上的 5 个 Cyclictest 测试线程的最大时延是 15 μs，然而，非实时性

内核的 5 个线程最大延时达到 39 μs。可以看出经过实时内核改造的 Linux 最大延时要比普通 Linux 内核的最大延时有较为显著的改善。

5. 结束语

本文对现有的 Linux 内核进行实时性改造，使其具有高实时性以满足工业环境的严苛要求。可以在支持工业控制应用的优先级实时调度情况下，基本保证调度延迟在 20 μs 以内。并且使 Linux 更好地支持虚拟化技术在工业互联网方面的应用，达到工业级别的延时要求。

参考文献

[1] O'Donovan, P., Leahy, K., Bruton, K., *et al*. (2015) An Industrial Big Data Pipeline for Data-Driven Analytics Maintenance Applications in Large-Scale Smart Manufacturing Facilities. *Journal of Big Data*, **2**, 25. https://doi.org/10.1186/s40537-015-0034-z

[2] Ji, C., Liu, S., Yang, C., *et al*. (2015) IBDP: An Industrial Big Data Ingestion and Analysis Platform and Case Studies. 2015 *International Conference on Identification, Information, and Knowledge in the Internet of Things*, Beijing, 17-18 October 2014, 223-228. https://doi.org/10.1109/IIKI.2015.55

[3] Tärneberg, W., Chandrasekaran, V. and Humphrey, M. (2016) Experiences Creating a Framework for Smart Traffic Control Using AWS IOT. *IEEE/ACM 9th International Conference on Utility and Cloud Computing*, Shanghai, 6-9 December 2016, 63-69. https://doi.org/10.1145/2996890.2996911

[4] Bob, F. and Jeff, B. (2017) Business in Real-Time Using Azure IoT and Cortana Intelligence Suite. Apress, Berkeley.

[5] 朱昱霖. 航天云网国际化战略分析及对中国工业互联网发展的启示[D]: [硕士学位论文]. 沈阳: 辽宁大学, 2018.

[6] 吕文晶, 陈劲, 刘进. 智能制造与全球价值链升级——海尔 COSMOPlat 案例研究[J]. 科研管理, 2019, 40(4): 145-156.

[7] 蒋旭辉. 基于 Xenomai 的嵌入式数控系统实时性实现探究[J]. 电子测试, 2019(16): 68-69.

[8] 马啸. 基于 Zynq 平台的 Linux 实时性研究及在数据采集中的应用[D]: [硕士学位论文]. 兰州: 兰州大学, 2019.

[9] 陈剑飞, 刘胜旺. 嵌入式虚拟化实时系统的研究与应用[J]. 机电信息, 2019(24): 54-55.

[10] 周丹, 陈楚康, 蔡万强, 等. 基于 Linux 内核的用户态网络协议栈的实现[J]. 信息通信, 2019(7): 200-204.

[11] 陈卫屏. 网络数据流高速采集系统设计与实现[D]: [硕士学位论文]. 成都: 电子科技大学, 2009.

[12] 刘剑, 仲宇, 王琦. 嵌入式 Linux 实时性改造技术综述[J]. 航天控制, 2018, 36(2): 93-97.

[13] 杨峰. 基于 Linux 内核的动态内存管理机制的实现[J]. 计算机工程, 2010, 36(9): 85-86.

[14] 刘文, 徐磊, 盛文婷, 等. 基于 ARM 平台的 Linux 内核分析与移植研究[J]. 现代计算机(专业版), 2011(7): 72-74.

[15] 汤元斌. 多线程模拟进程时间片轮转调度算法研究[J]. 四川文理学院学报, 2014, 24(5): 76-79.

[16] 仇阳. Linux 内核进程调度算法发展[J]. 电子世界, 2017(7): 85-87.

[17] 张旭, 顾乃杰, 苏俊杰. Linux 调度器免锁优化方法研究[J]. 小型微型计算机系统, 2017, 38(4): 690-695.

[18] 王帅华, 杨东升, 王允森, 等. 基于 ARM 的 Linux 实时抢占补丁的研究与实现[J]. 组合机床与自动化加工技术, 2015(9): 1-4.

[19] 陈何杰, 郑灵翔. ARM Linux 中断系统移植研究[J]. 厦门大学学报(自然科学版), 2010, 49(3): 339-343.

[20] 王霞, 马忠梅, 何小庆, 等. 提高嵌入式 Linux 时钟精度的方法[J]. 计算机工程, 2006, 32(23): 70-72.