LKL: A way to reuse Linux kernel in userland

华为/OS内核实验室 李雨
2018-6

HUAWEI TECHNOLOGIES CO., LTD.
HUAWEI

自我介绍

- 80后，非科班，持续学习中☺
- 2001年，自学Linux之旅
  - Linux AtoC
- 2004年，正式开始Linux码农生涯
  - 共创开源
  - 系统服务，内核，驱动程序，...
- 2008年，开始熟悉Linux
  - RNI, Alibaba
  - 内核，协议栈，虚拟化，系统工具 ...
- 2017年-，码农仍在路上
  - Huawei
  - 继续学习为Linux为代表的系统软件
  - 探索一下Linux外面的世界，搞点事情

后面会有

- LKL是什么
- LKL能做什么
- LKL是怎么工作的
  - 工作原理
  - 引导过程
- LKL对外接口是什么
- 我们在开源的LKL基础上所做的工作
  - SMP支持：2017下半年的工作
  - LKL作为用户态驱动的一些尝试和思考
- 不包括
  - 大数据，云计算，物联网
  - AI和区块链☺

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

What is it ?

- **第一印象**
  - LKL的设计目标是在用户态复用Linux成熟的功能实现，目前主要协议栈、文件系统，同时不失可移植性和可维护性。
    - 收益与开销
  - 二进制发布形式是一个DSO甚至静态库，供用户态APP使用。
  - 不是为了在bare metal上直接运行设计的，虽然方案上与rumpkernel有几分类似。
    - 即，并非完整的LibOS/Unikernel/Exokernel方案
  - 支持的平台和OS
    - x86_64/AArch64
    - Windows/FreeBSD/Linux
  - 个人观点：
    - LKL对于它的目标而言是一个很好的起点，但还不够好。

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

实现复用

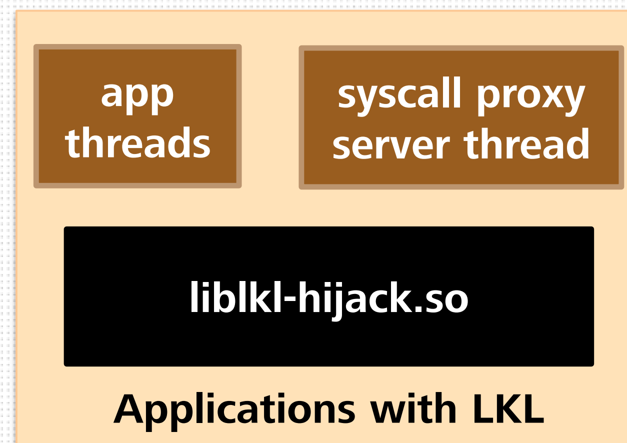
- **文件系统**
  - Ext*, XFS, ... = 几乎完整的存储栈
  - 接口：
    - FUSE：作为hostOS的一个文件系统模块，通过hostOS的syscall接口访问LKL内的文件系统
    - lkl syscall：用syscall式的接口直接访问LKL内的文件系统
- **网络栈**
  - TCP/IPv4/v6, SCTP, ARP, UDP, *tables, tc = 几乎完整的网络栈
  - 接口：
    - lkl syscall
- **Good**
  - 4.11 kernel。尝试过从4.10 rebase到 4.11，不难，0.5 day.
  - 可编译为静态库和DSO，DSO可以用preload的方式override syscall symbols。
- **Bad**
  - Poor performance.
  - UP only, Poor control plane.

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

Internals

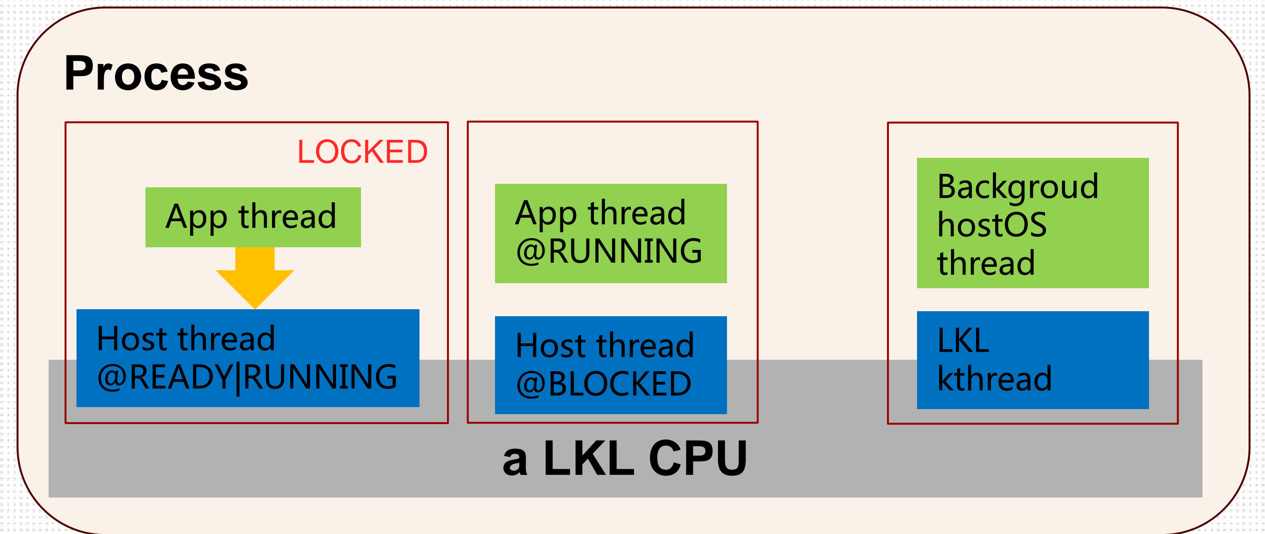
- **如果我们是LKL的作者，会如何实现？**
  - › Genode/lxIP
    - » 暴力肢解 (stubs) 剥离协议栈和驱动，再也回不去了。。。
  - › NUSE
    - » 只关注协议栈，方法与LKL类似，但对kernel本身有不少修改，趋于LKL方向。
  - › 最近还有一些复用Linux其它组件的努力，比如针对驱动。
- **LKL的答案**
  - › 对上（应用程序）：LKL syscalls
    - » 用functions call要模拟Linux syscall
    - » 通过hijack接口实现部分二进制兼容
  - › 对内（要复用的Linux代码）：
    - » 通过一个“POSIX userspace arch”维持kernel核心抽象
  - › 对下（硬件&底层系统）：
    - » The host operations interfaces
    - » 某种程度上类似于现代SASOS变体：RumpKernel，MS Drawbridge



HUAWEI TECHNOLOGIES CO., LTD.
HUAWEI

Internals

- **CPU：并行度**
  - 每个LKL CPU表示一个LKL的入口，有多少LKL CPU就表示同时多少个APP线程可以进入LKL库。
  - LKL CPU的多数字段用于处理内部owner可变的锁逻辑：在LKL内部任务切换时修改lkl cpu的owner。
  - Fast syscall:
    - app threads在LKL syscall时获取cpu锁，然后longjmp到其对应的host thread上下文执行。
    - 如果app threads对应的host thread不存在，就创建一个。
  - Idle thread运行时会唤醒”idle host task”，后者放弃cpu lock，进入休眠。
  - 所有任务都是LKL内核眼中的kernel thread，所以LKL内部基本不需要处理模拟signals的语义。



- **Memory：平坦模型，UMA**
  - 关闭MMU支持，对XFS有个小补丁，2-3行左右。
  - Flat layout意味着，物理内存就只有一个cmdline参数：mem=${HOW-MANY-BYTES}。
  - 和x86_64 port一样，LKL自已截留一部分，其余都放到bootmem allocator（不是memblock），之后这些内存会进入buddy allocator。

HUAWEI TECHNOLOGIES CO., LTD.
HUAWEI

Internals

- **Interrupts**
  - 与PV guest类似，基本就是在enable irq时检查是否有中断发生，另一个点是在释放CPU lock时。
  - 没有特别的irq chip设计，直接复用了builtin simpe irq chip。
- **Timing**
  - oneshot clocksource，对应host timerfd(SIGEV_THREAD) syscall。
  - 读取timestamp，对应host clock_gettime() syscall。
- **I/O：PV like**
  - NICs
    - 直接通过注入MMIO信息注册LKL设备，模拟一块 platform virtio-net NIC, single queue。
    - 后端可以是DPDK、raw sockets、netmap、tap/macvtap。
    - virtio-net设备对应一个"poll thread"，负责在virtio ring和backend之间交换数据。
  - Block devices
    - 类似于NIC。可以将外部文件模拟为一个LKL可见的virtio-blk。Backend就是简单的host pread()/pwrite() syscalls。

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

Internals

- 通向外部世界(HostOS)：host operations interface
  - threading: create, destroy, join, exit
  - timing: alloc, set_oneshot, free
  - mutex/semaphore: alloc, acquire, release, free.
  - memory/TLS: alloc, free, get, set
  - longjump
  - Iomem_access/ioremap
  - Some debug stuff

LKL host operations	Rumpuser interface
sem_down|up mutex_lock|unlock	rumpuser_mutex_enter rumpuser_mutex_exit
Thread_create|destroy Thread_exit|join	rumpuser_thread_create|e xit
Mem_alloc|free	rumpuser_malloc|free
Timer_alloc|free Timer_set_oneslot	Rumpuser_clock*
Ioremap, iomem_access	rumpuser_bio|file*

- 功能组织是有几分类似于Rumpuser interface
  - http://netbsd.gw.com/cgi-bin/man-cgi?rumpuser+3+NetBSD-current

HUAWEI TECHNOLOGIES CO., LTD.
HUAWEI

Internals

- **初始化LKL/UP，其实就是在引导LKL包装的”用户态Linux系统“。**
  - From DSO constructor: hijack_init()
    - 根据各种环境变量构造LKL kernel cmdline
    - 初始化NIC后端：lkl_netdev_{dpdk,raw,netmap,tap,...} _create()
    - lkl_netdev_add() ：通过platform MMIO注册LKL kernel virtio devices
    - lkl_start_kernel(): Bring up LKL kernel
    - LKL引导成功后，设置基本运行环境，例如IP/MAC地址、路由、mount filesystems，设置sysctl vectors.
  - lkl_start_kernel()
    - lkl_cpu_init(): 初始化struct lkl_cpu
    - 创建一个线程执行lkl_run_kernel()，这个线程之后成为LKL CPU0的idle task.
    - 等待LKL init process启动，init process会完成一些重要的初始化。

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

Internals

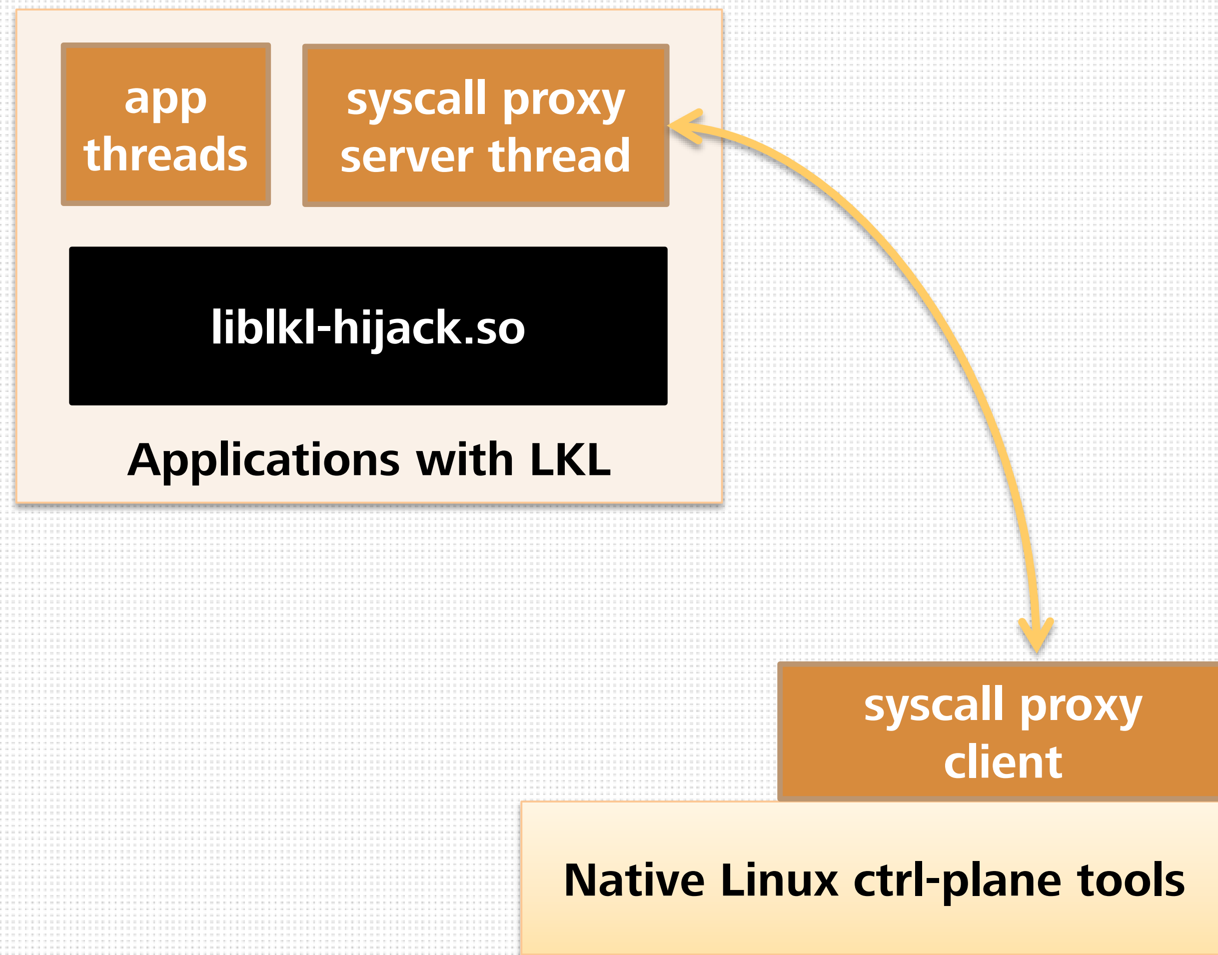
- **初始化LKL/UP**
  - lkl_run_kernel() -> start_kernel()
  - setup_arch()
    - 解析mem=参数，获取内存大小。 这个参数也决定LKL的内存开销。
    - bootmem_init(mem_size)
  - rest_init() -> init calls -> fs_setup()
    - 在rootfs内创建一个空文件 “/init”
    - 为/init注册一个binfmt，其中 “.load_binary=lkl_run_init;”
  - Bring up the init process (lkl_run_init()).
    - syscalls_init()
      - 将当前线程标记为HOST_THREAD，作为syscall enter stub.
      - 创建idle host task。
    - Wakeup user app thread

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

Internals

- **不爽之处**
  - 如果编译成DSO，kallsyms就不工作了
    - 在kallsyms core中一行hack可解。
  - Poor performance: TODO
    - e.g. double memory copy
  - Poor control plane:
    - debug mode：mini shell
    - rumpkernel syscall proxy
      - OK: ip, tc
      - Failure: iptables



HUAWEI TECHNOLOGIES CO., LTD.
HUAWEI

Our work

- **目标&动机**
  - 试图在自研OS里复用networking stack & drivers
  - 可移植性&可维护性
  - 性能
- **内容**
  - LKL networking后端的netmap支持
  - LKL networking后端接口的批处理能力
  - SMP support: x86_64 and AArch64
  - Drivers reuse research

HUAWEI TECHNOLOGIES CO., LTD.
HUAWEI

SMP support

- **Overview**
  - Atomic ops / spin lock / memory barriers are required.
  - 区分多CPU。
    - 多处理器时钟。
    - Bring up secondary LKL processors.
    - Extra Idle tasks.
  - Interrupts affinity & IPI。
  - Tree RCU support.
  - DSO linker script hacks: fix kallsyms and percpu area alignments.
  - Extend LKL host operation interface.

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

SMP support

- **Infrastructure**
  - Atomic ops/cmpxchg/nops:
    - copy from arch/{x86,arm64} with small changes.
  - Spin lock
    - Use Linux kernel built-in queued spin lock in new kernel.
  - Memory barriers
    - copy from arch/{x86,arm64} with small changes.
- **Alternative choice ?**
  - 使用gcc/libc或者其它userland libraries实现以上infrastructure。可移植性更好些，但性能会差些，有些细节可能有语义差异，尤其是memory barriers。
  - A small concern here, see details at LKL github issues page.

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

SMP support

- **CPU and interrupts**
  - struct lkl_cpu 和irq pending bits等核心数据结构都成为数组，对应每个LKL CPU。
  - 为lkl_trigger_irq()增加一个参数，对应IRQ affinity和dispatch的语义。
  - 每个LKL tasks都有一个host TLS variable保存当前CPU。在发生task migration时需要修改这个变量。
- **IPI与其它中断处理略不同：每个LKL CPU对应一个IPI thread.**
  - 如果CPU locked down，就只标记pending bit。
  - 如果CPU unlocked，直接运行所有irq handlers (and IPI)。
  - 如果CPU正在运行idle threads，也会“note RCU context switching”。
- **Timing**
  - 调度器和RCU依赖于多处理器时钟的支持。
  - 利用内置的clock broadcasting framework，CPU0得到timer中断之后，通过IPI广播给其它处理器。

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

SMP support

- **Linux/SMP：留空**
  - Incomplete but important points of vanilla kernel SMP boot:
    - boot_cpu_init(): 为BSP设置各种bitmaps
    - setup_arch(): 为所有BSP/AP设置各种bitmaps
    - setup_per_cpu_areas(): 初始化percpu area
    - smp_prepare_boot_cpu(): 依赖于ARCH
    - rest_init()→smp_prepare_cpus(), smp_init() @ another thread
      - 创建所有idle threads
      - 注册CPU hotplug threads
      - BSP状态机：创建所有HP threads，初始化wq/hrtimer/slab/rcutree，启动其它AP
      - AP状态机：初始化和启动调度器，unpark threads，启用irq affinity/wq/rcutree
    - BSP/APs: rest_init() → cpu_startup_entry(CPUHP_ONLINE)

- **初始化LKL/SMP：填空**
  - Key points of LKL SMP boot:
    - BSP: setup_arch() -> lkl_smp_init(): 设置possible/present bitmaps
    - BSP: smp_prepare_cpus(): 注册IPI中断处理例程，创建IPI线程。
    - PerCPU area: use generic implementation
    - APs: for each AP __cpu_up(): 使cpu online，更新current，唤醒idle threads
    - APs: idle thread start from lkl_start_secondary(), a trick here.
      - 创建对应的idle host thread。
      - 结束SMP/APs引导状态机，BSP得以继续。

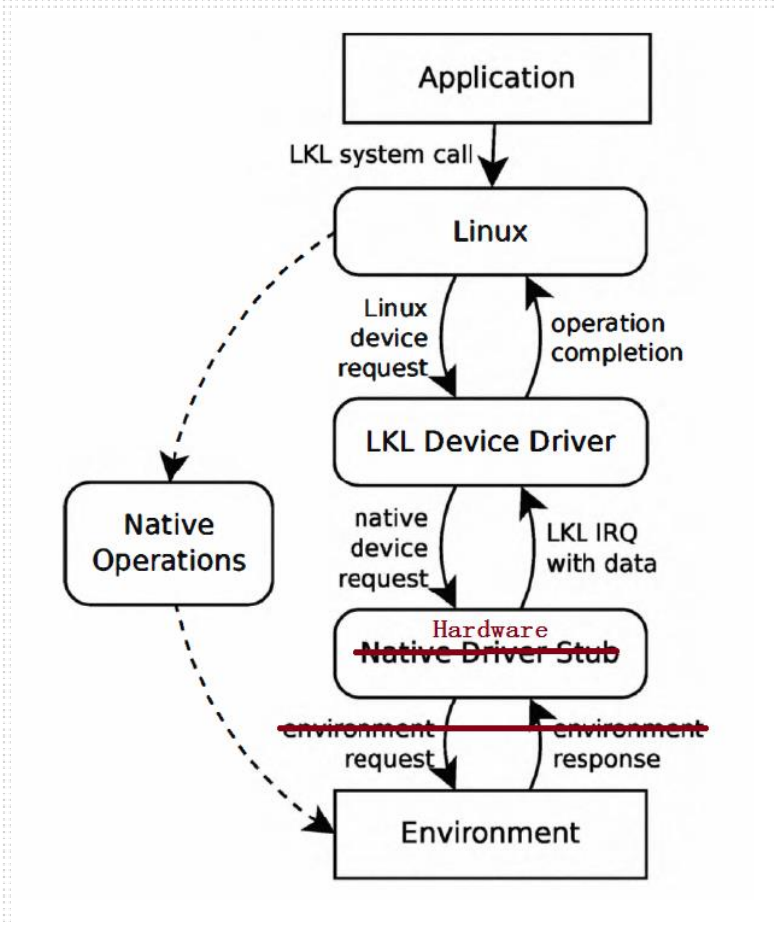
HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

Reuse drivers

- **Linux驱动程序是开源世界中的一份宝藏**
  - Drivers：~400MB
  - Arch/{arm64, x86}：10 -50MB
  - FS: < 50MB
  - Net: < 50MB

- **LKL I/O review**
  - ioremap(addr,size)
  - iomem_access(addr,val,ops,...)
  - Virtio devices over the platform framework.
  - lkl_{get,trigger,put}_irq() API

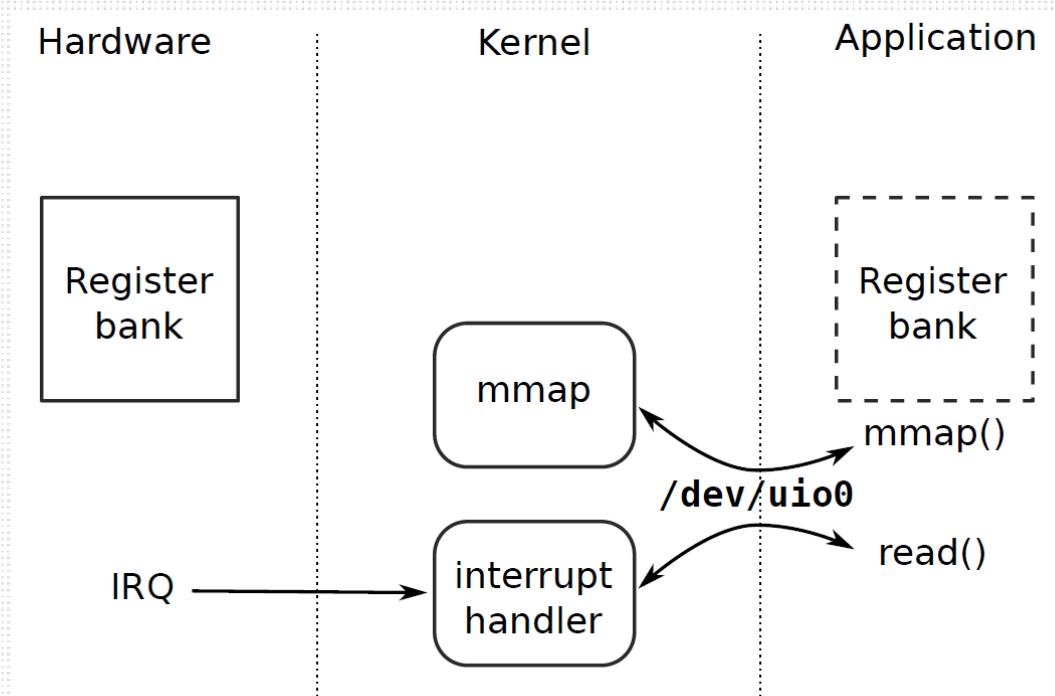


HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

Reuse drivers

- **不快的“快速”尝试**
  - 在LKL的编译配置中启用驱动程序，并且填充好它依赖但LKL没有提供的功能
  - 使用UIO框架复用LKL的驱动程序，有些波折但结果证明这条路是基本可行的
- **用户态驱动的挑战: A new building block ?**
  - 置于用户态的额外安全隐患？Why ?
  - 驱动“进程”如何与调度器交互？依赖倒置
  - 多驱动进程如何处理依赖：跨进程IPC ?
    - EventIntf -> Input driver -> USBHID -> EHCI -> PCIE -> ACPI -> APIC
  - 系统级组件和驱动进程的依赖。
  - 中断处理的要求多种多样，定制的进程调度模型？
    - X86: local interrupt, 外设中断, IPI
    - ARM: PPI, SPI, SGI, LPI
  - 如何安全高效地处理DMA
    - 物理地址连续性信息，SGIO
    - 操作缓存
  - 安全性：IOMMU?
    - 针对恶意硬件（或者bugs），不能防御恶意APP或者恶意驱动
    - 高速设备的性能损失
  - 性能：无明确证据表明用户态驱动的性能会是瓶颈



HUAWEI TECHNOLOGIES CO., LTD.
HUAWEI

So, next ?

SMP prototype is workable at x86_64 and AArch64 platform.

- **优点**
  - 省略1000字
- **缺点**
  - Still Poor performance & control plane
  - LKL内部很难直接调用外部的host routines（by host ops），反之则通过LKL syscalls。
  - 不支持fork
  - 内存开销不低，尤其是内存受限系统内，但是可配置。
  - PRELOAD DSO调试不太容易。
  - 作为DSO/静态库，license需要明确的边界界定。

- Sources
  - Upstream https://github.com/lkl/linux
  - SMP prototype https://github.com/Rover-Yu/lkl-linux
  - Drivers-reusing prototype https://github.com/alephman/linux
- Others
  - Search and research it using Web search engine, ☺
  - lkl team @ slack

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

欢迎加入华为OS内核实验室

华为操作系统部：华为端、管、云核心的OS软件基础设施

OS Kernel Lab
- Linux内核（ARM/x86/异构等）的技术研发与创新
- 低时延、高安全、高可靠、高智能的下一代OS内核技术的研究和成果转化

招聘岗位

下一代操作系统研究员/高级工程师

形式化技术研究员/高级工程师

Linux内核架构师/高级工程师

工作地

杭州、北京、上海

简历投递

Tel：王先生/18658102676

Email：hr.kernel@huawei.com

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI