Ridiculous-IncludeOS

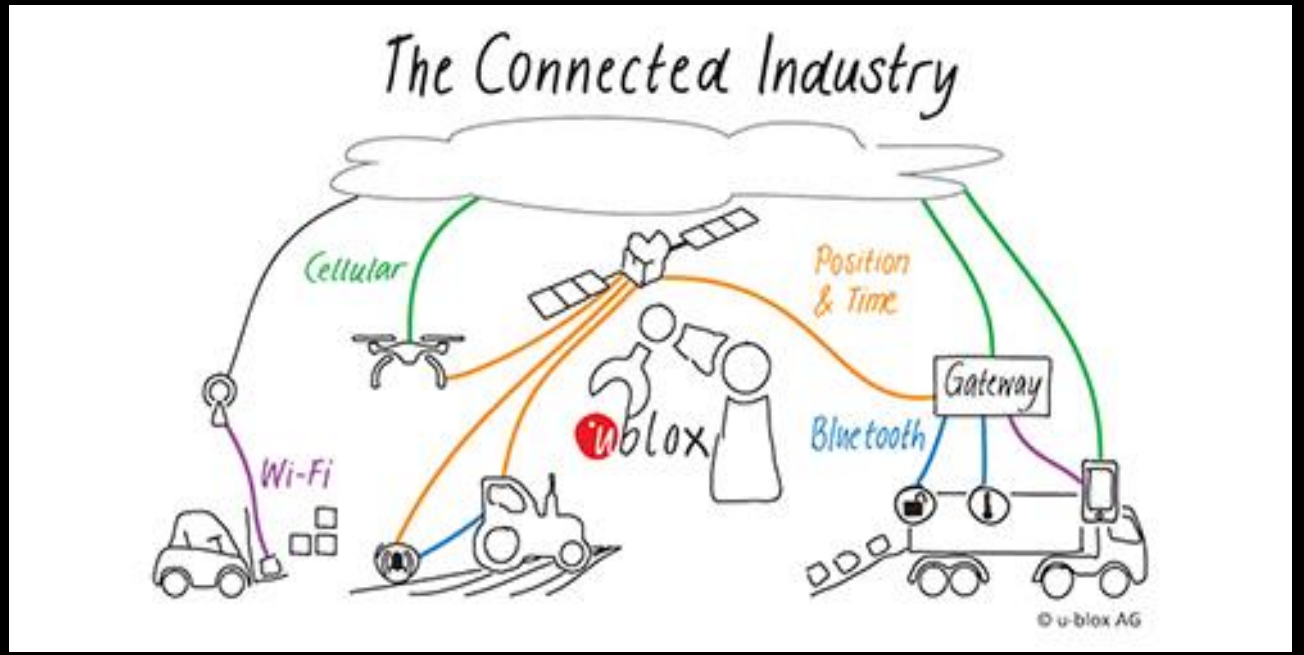
小组成员：刘紫檀 张博文 虞佳焕 汤兖霖

IoT 简介

- IoT(Internet of Things)即物联网,它的定义为把所有物品通过射频识别等信息传感设备与互联网连接起来，实现智能化识别和管理。
- 要求：可靠传递,通过各种电信网络与互联网的融合,将物体的信息**实时准确**地传递出去。

IoT 目前存在的问题

- 物联网有很多领域对延迟和通信的可靠性具有很高的要求，延迟高或者通信不可靠都会造成比较严重的后果。
- 如果延迟和通信问题不能解决，很多概念将很难投入到真实的应用场景中。
- 安全性有待提升：容易泄露敏感信息，容易造成安全问题 。

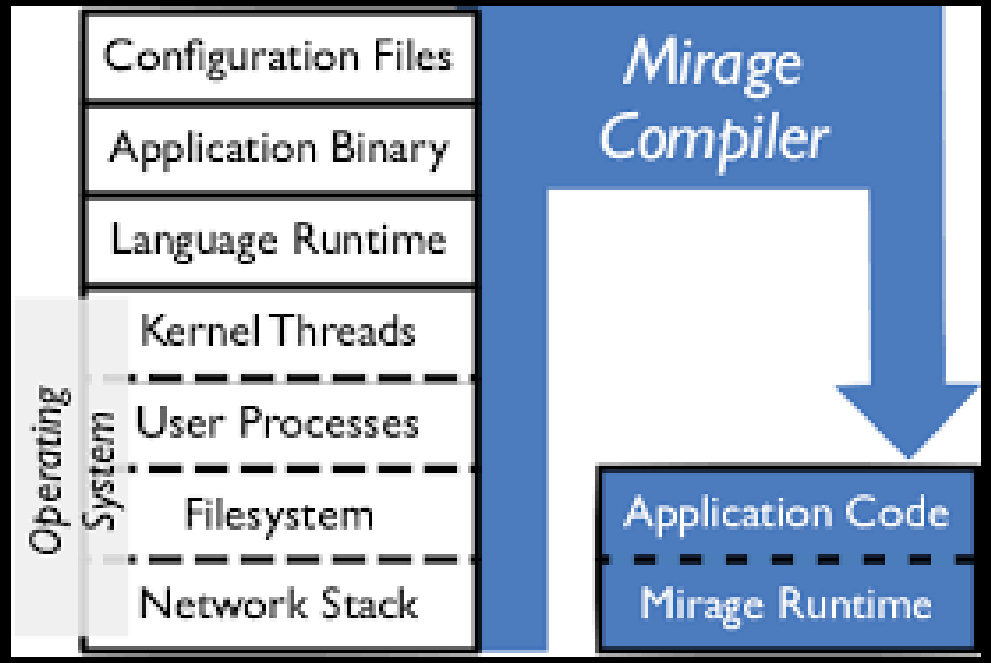


解决方案

- 5G
- 操作系统

什么是 Unikernel ？

- 它是专用的，单地址空间的，使用 library OS 构建出来的镜像。
- Unikernel 不仅可以运行在 bare-metal 上，也可以运行在虚拟机上。



Unikernel 的项目

ClickOS

A high-performance, virtualized software middle box platform based on open source virtualization. Early performance analysis shows that ClickOS VMs are small (5MB), boot quickly (as little as 20 milliseconds), add little delay (45 microseconds) and more than 100 can be concurrently run while saturating a 10Gb pipe on an inexpensive commodity server.

cnp.neclab.eu

Clive

An operating system designed to work in distributed and cloud computing environments, written in the Go programming language.

lsub.org

Drawbridge

A research prototype of a new form of virtualization for application sandboxing. Drawbridge combines two core technologies: a picoprocess, which is a process-based isolation container with a minimal kernel API surface, and a library OS, which is a version of Windows enlightened to run efficiently within a picoprocess.

research.microsoft.com

HaLVM

A port of the Glasgow Haskell Compiler tool suite that enables developers to write high-level, lightweight VMs that can run directly on the Xen hypervisor.

galois.com

IncludeOS

A minimal, service oriented, includeable library operating system for cloud services. Currently a research project for running C++ code on virtual hardware.

includeos.org

LING

A unikernel based on the Erlang/OTP and understands .beam files. Developers can create code in Erlang and deploy it as LING unikernels. LING removes the majority of vector files, uses only three external libraries and no OpenSSL.

erlangonxen.org

MirageOS

A clean-slate library operating system that constructs unikernels for secure, high-performance network applications across a variety of cloud computing and mobile platforms. There are now almost 100 MirageOS libraries and a growing number of compatible libraries within the wider OCaml ecosystem.

mirage.io

OSv

A new OS designed specifically for cloud VMs from Cloudius Systems. Able to boot in less than a second, OSv is designed from the ground up to execute a single application on top of any hypervisor, resulting in superior performance, speed and effortless management. Support for C, JVM, Ruby and Node.js application stacks is available.

osv.io

Rumprun

A software stack which enables running existing unmodified POSIX software as a unikernel. Rumprun supports multiple platforms, including bare hardware and hypervisors such as Xen and KVM. It is based on rump kernels which provide free, portable, componentized, kernel quality drivers such as file systems, POSIX system call handlers, PCI device drivers, a SCSI protocol stack, virtio and a TCP/IP stack.

rumpkernel.org

runtime.js

An open-source library operating system for the cloud that runs JavaScript, could be bundled up with an application and deployed as a lightweight and immutable VM image. It's built on V8 JavaScript engine and uses event-driven and non-blocking I/O model inspired by Node.js. At the moment KVM is the only supported hypervisor.

runtimejs.org

UniK

为什么选择 includeOS？

- 性能优良，启动迅速，能在几十毫秒之内启动。
- 体积小，只需很小的磁盘和内存。
- 安全性高，没有冗余代码。
- 支持在裸机上运行。
- 延迟很低。
- 对网络的支持很好，与 Linux 相比表现出色。

为什么选择 includeOS？

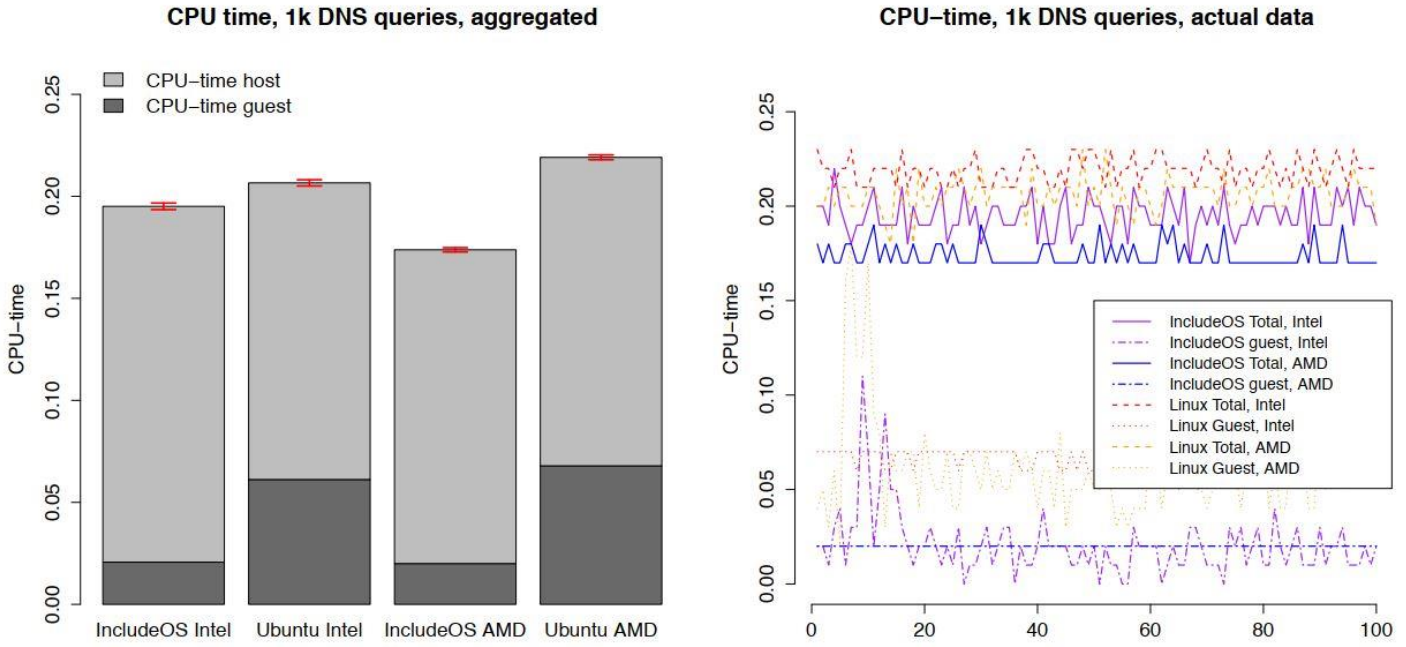
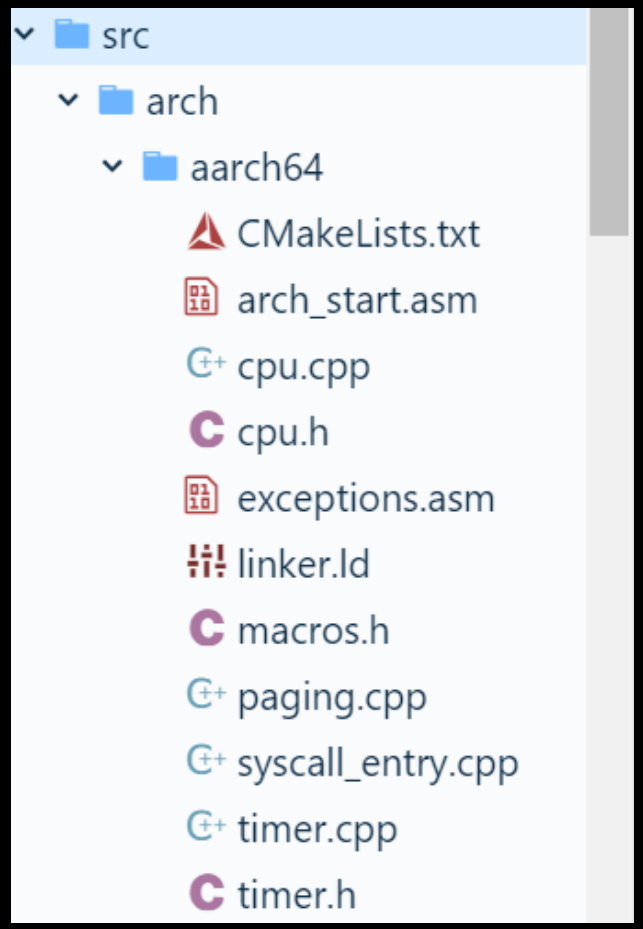


Fig. 4. *CPU-time required to execute 1000 DNS-requests, on IncludeOS and Ubuntu, running on Intel- and AMD systems. IncludeOS and Ubuntu are running the same DNS sevice binary. On AMD, IncludeOS uses 20% fewer CPU-ticks on average total, and 70% fewer ticks on average spent inside the guest CPU. On Intel IncludeOS uses 5.5% fewer CPU-ticks on average total, and 66% fewer ticks inside the guest. Error bars represent the 95% confidence interval over 100 samples.*

分工安排

- 张博文：了解目前 includeOS 官方在 AArch64 上的工作
- 虞佳焕：熟悉AArch64体系结构
- 汤兖霖：熟悉代码构建工具
- 刘紫檀：阅读源码结构

了解 includeOS 官方目前的工作



了解 includeOS 官方目前的工作

- 已完成启动平台初始化、关机、中断和异常处理、初始化 C 运行时库
- 分页、系统调用、内存管理等大量工作未完成

```
#include "cpu.h"


extern "C"
long syscall_SYS_set_thread_area(struct user_desc *u_info)
{
  set_tpidr(u_info);
  /*
  if (UNLIKELY(!u_info)) return -EINVAL;
#ifdef __x86_64__
  x86::CPU::set_fs(u_info);
#else
  x86::CPU::set_gs(u_info);
#endif*/
  return 0;
}
```

```
27  namespace os {
28  namespace mem {
29    __attribute__((weak))
30    Map map(Map m, const char* name) {
31      return {};
32    }
33
```

了解 includeOS 官方目前的工作

- 下一步目标：以官方工作作为参考，继续完成剩余的工作

熟悉 AArch64

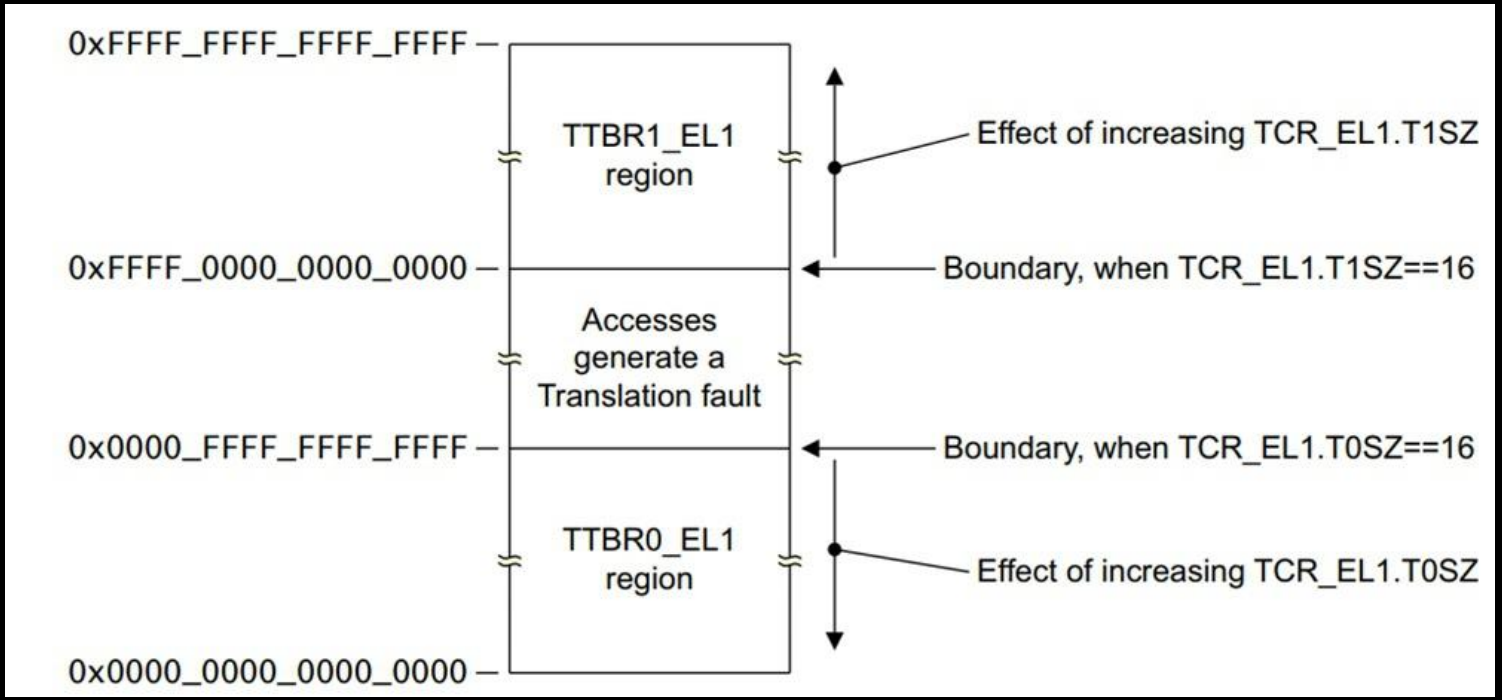
- 汇编
- MMU
- System registers
- Interrupt
- Device

AArch64 MMU

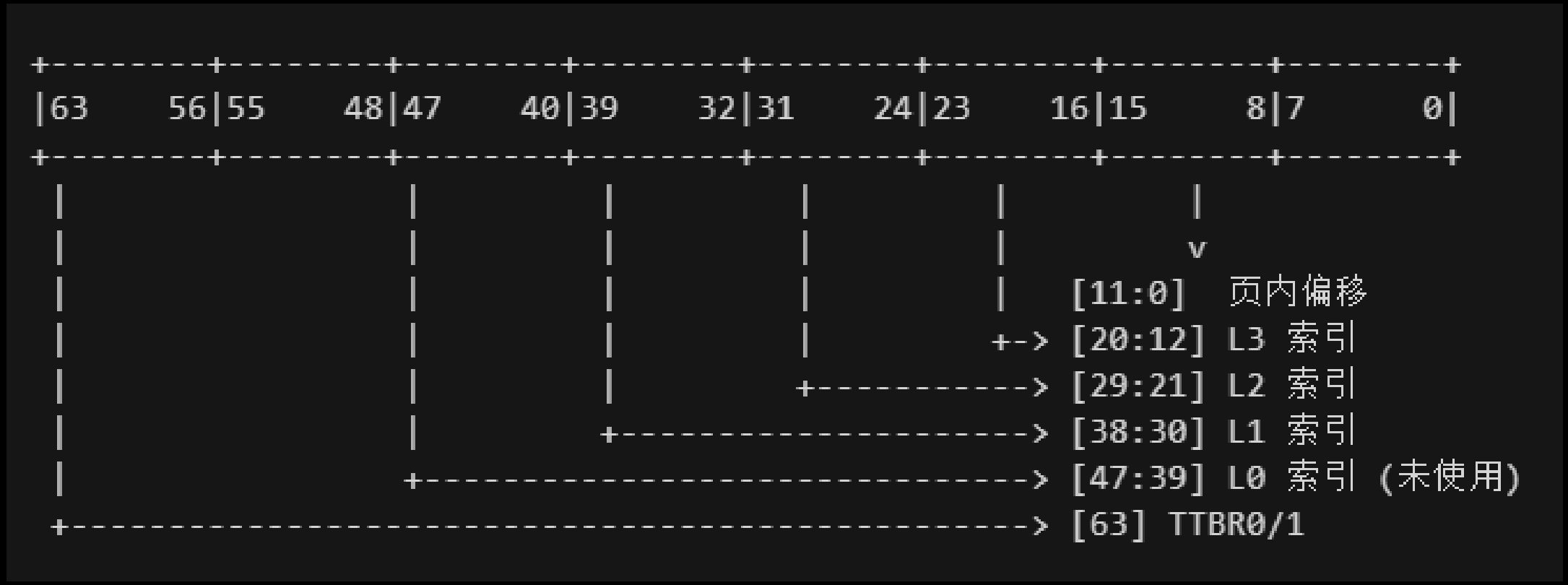
- MMU (Memory Management Unit)
- VA => PA

AArch64 MMU

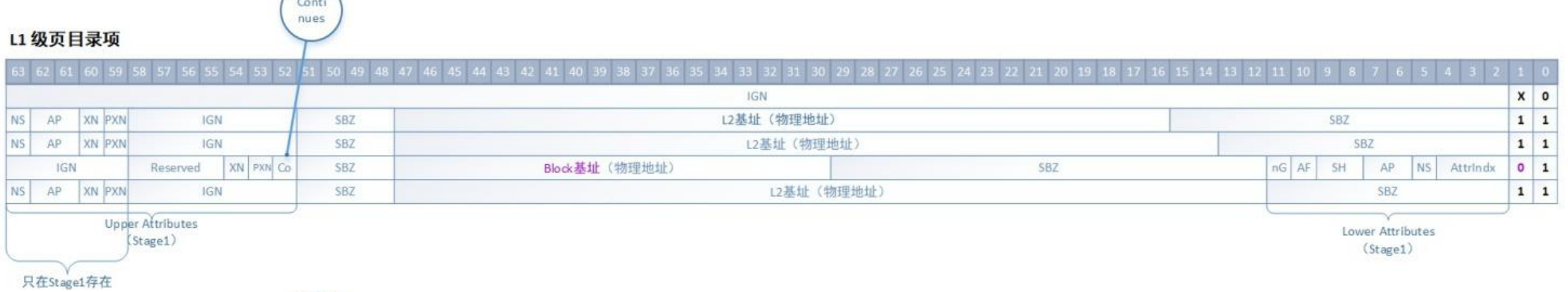
- AArch64 Linux 一般使用页大小为 4KB 的 3 级转换表配置。
- User space 和 kernel space 都有 39-bit (512GB) 的虚拟地址空间。用户地址空间的 63:39 位为 0，而内核地址空间的相应位为 1。



AArch64 MMU

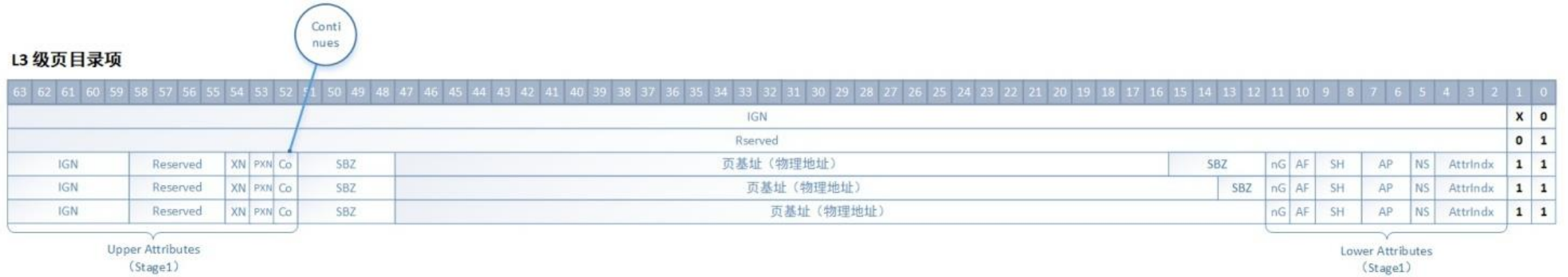

63 56 55 48 47 40 39 32 31 24 23 16 15 8 7 0
[11:0] 页内偏移
[20:12] L3 索引
[29:21] L2 索引
[38:30] L1 索引
[47:39] L0 索引 (未使用)
[63] TTBR0/1

L1 级页目录项

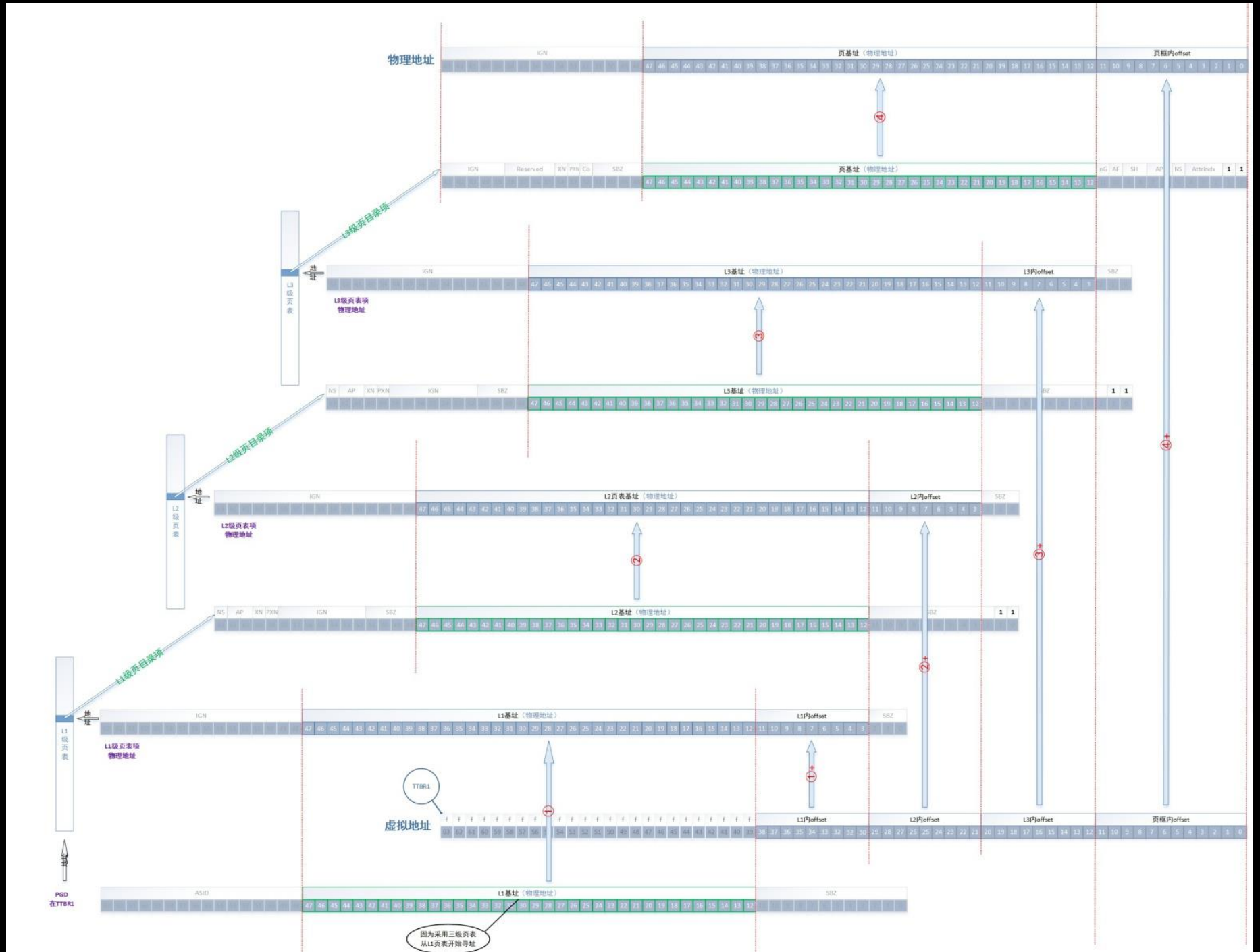


L2 级页目录项

L3 级页目录项



SBZ: Should-Be-Zero fileds IGN: Ignored


物理地址
IGN
页基址（物理地址）
页框内offset
Reserved
XN
PXN
Co
SBZ
nG
AF
SH
AP
NS
AttrIndx
L3级页目录项
L3级页表
地址
L3级页表项
物理地址
L3基址（物理地址）
L3内offset
L2级页目录项
L2级页表
L2级页表项
L2页表基址（物理地址）
L2内offset
L2基址（物理地址）
L1级页目录项
L1级页表
L1级页表项
L1基址（物理地址）
L1内offset
TTBR1
虚拟地址
ASID
PGD
在TTBR1
因为采用三级页表
从L1页表开始寻址

源码结构

- 阅读源码：Understand (C/C++) + Github repository search (asm)
  - Generated report: http://home.ustc.edu.cn/~jauntyliu/includeOS_understand_html/
- 工具链一览
  - 构建工具：CMake
  - 单元测试：Lest
  - IncludeOS 配置生成：NaCl
  - 安装和虚拟机交互：Shell + Python (+ CMake)

源码结构

- 外部项目一览
  - musl libc
  - Botan
  - http-parser && rapidjson && uzlib

源码结构

_start => **__arch_start && long_mode** => **kernel_start**

1. __init_serial1()
2. __init_sanity_checks()
3. _move_symbols
4. _init_bss()
5. OS::init_heap(free_mem_begin, memory_end)
6. _init_syscalls()
7. x86::idt_initialize_for_cpu(0);
8. _init_elf_parser()
9. Elf_binary<Elf64>
10. RNG::init()
11. __libc_start_main(kernel_main, argc, argv.data())

源码结构

kernel_main

1. OS::start(__grub_magic,__grub_addr);
2. OS::post_start();
3. kernel_sanity_checks();
4. OS::event_loop();

Thanks.