哈爾濱工業大學(深圳)
HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

CHRONIX：面向多核平台异步架构高性能操作系统

Chronix 团队 ： 肖嘉誉、欧阳天麟、周立诚
指导老师：夏文、仇洁婷

目录/CONTENTS

PART 01

系统定位

Lorem ipsum dolor sit amet, consectetuer dipiscing elit, sed diam nonummy nibheuismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam

Chronix 是什么？

-基于 Rust 编写的类 Unix 多核操作系统：

Rust 语言提供内存安全与并发优势，更少的内核级 bug 与安全漏洞

-支持 RISC-V 和 LoongArch64 双架构：

面向国产自主指令集和开源硬件生态，安全可控与利于国产生态建设

-强调异步并发、高效资源管理与多核支持与管理：

-异步机制降低上下文切换成本，高效资源管理与多核支持管理提升IO性能

-注重可维护性、开源社区友好：

较低长期运维成本，为开源社区添砖加瓦

Chronix亮点

introduction and highlight

进程管理

- 异步无栈协程
- 支持多核调度
- Pelt算法均衡负载
- 统一进程/线程模型

内存管理

- 应用加载按需读取、写时复制、懒分配
- 全局使用 SLAB 内存分配器、
- 支持零页分配

架构管理

自研硬件抽象层、支持 Risc-V、Loongarch 双架构

文件系统

- 支持挂载、支持Ext4、Fat32 多磁盘文件系统
- 支持内存文件系统、进程文件系统、设备文件系统

设备驱动

支持硬件中断、MMIO 驱动、PCI 驱动、串口驱动。

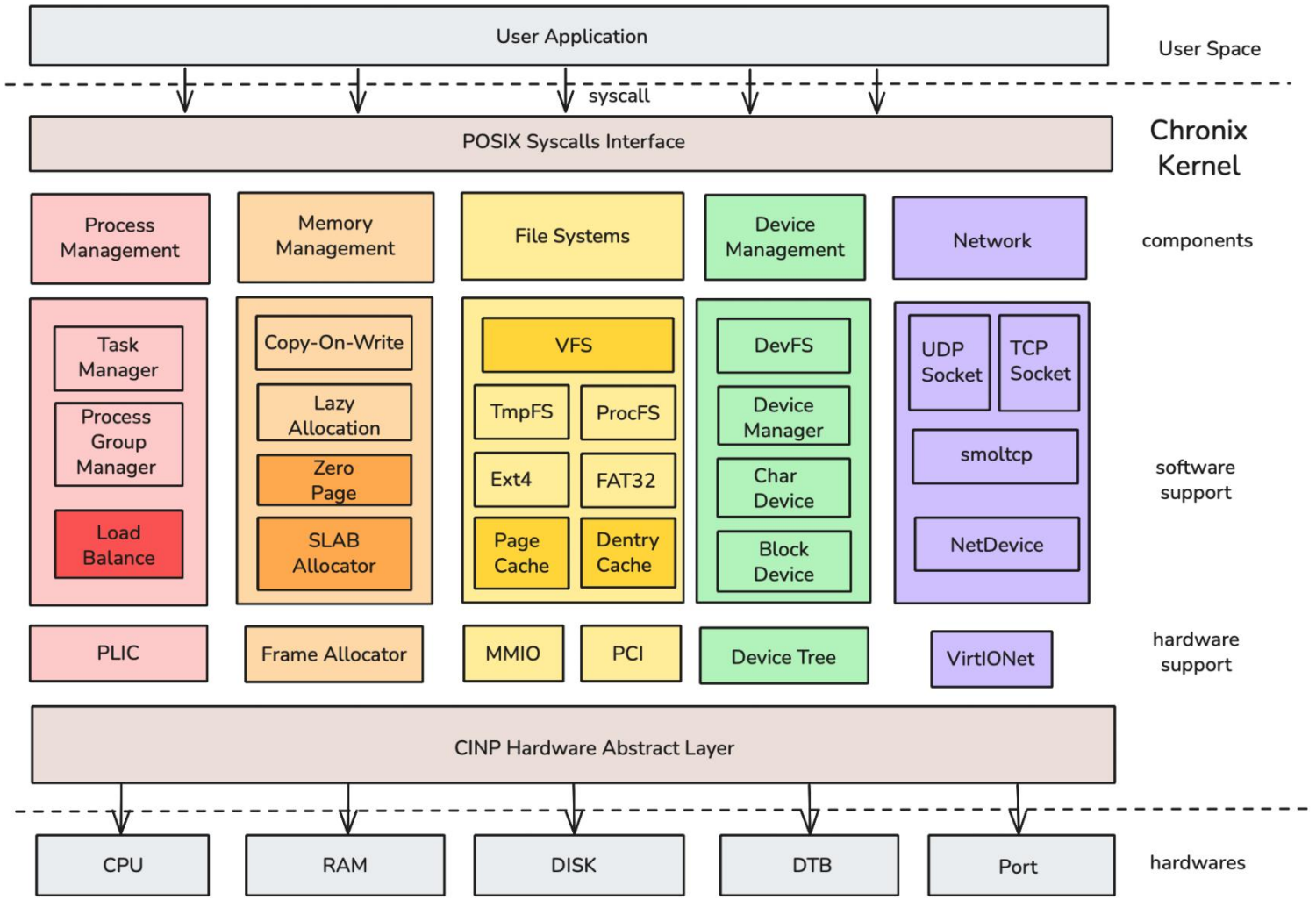
支持设备树解析

PART 02

系统介绍

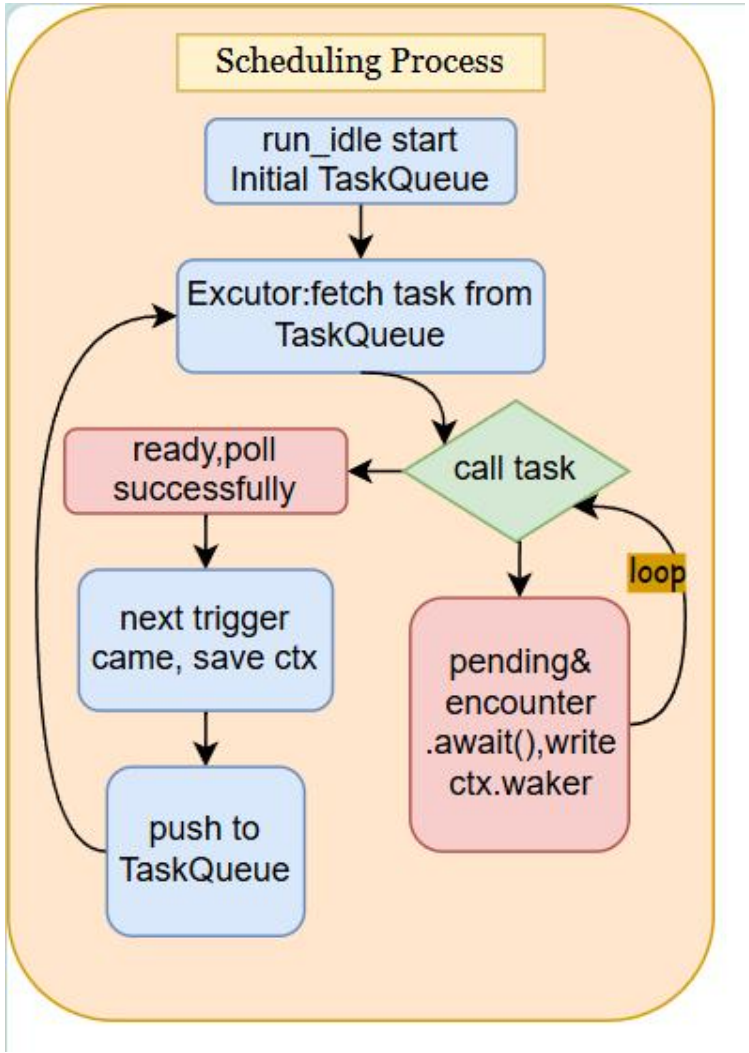
Lorem ipsum dolor sit amet, consectetuer dipiscing elit, sed diam nonummy nibheuismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam

系统框架


User Application
User Space
syscall
POSIX Syscalls Interface
Chronix Kernel
Process Management
Memory Management
File Systems
Device Management
Network
components
Task Manager
Process Group Manager
Load Balance
Copy-On-Write
Lazy Allocation
Zero Page
SLAB Allocator
VFS
TmpFS
ProcFS
Ext4
FAT32
Page Cache
Dentry Cache
DevFS
Device Manager
Char Device
Block Device
UDP Socket
TCP Socket
smoltcp
NetDevice
software support
PLIC
Frame Allocator
MMIO
PCI
Device Tree
VirtIONet
hardware support
CINP Hardware Abstract Layer
CPU
RAM
DISK
DTB
Port
hardwares

进程管理

Company introduction and corporate publicity



统一进程与线程管理：TaskControlBlock

异步驱动任务调度：借助Rust异步生态将转为

任务调度器使用 async_task + Rust trait 系统

高效无栈协程模型，内存占用低，切换代价小

多核调度与负载平衡

Company introduction and corporate publicity

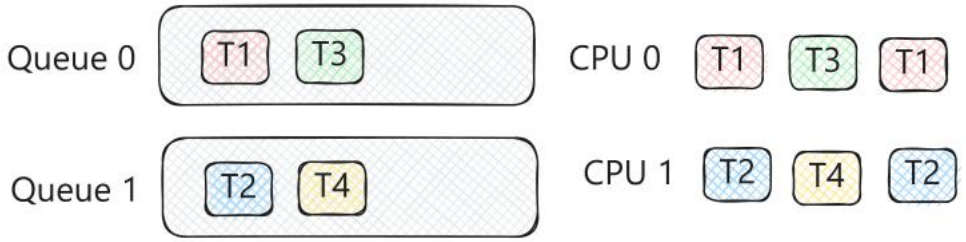


图 2-3：多队列调度模型

任务的负载计算公式为：

$$\text{Load} = \sum_{i=0}^{n} \text{load}_i y^i \tag{2-1}$$

其中：

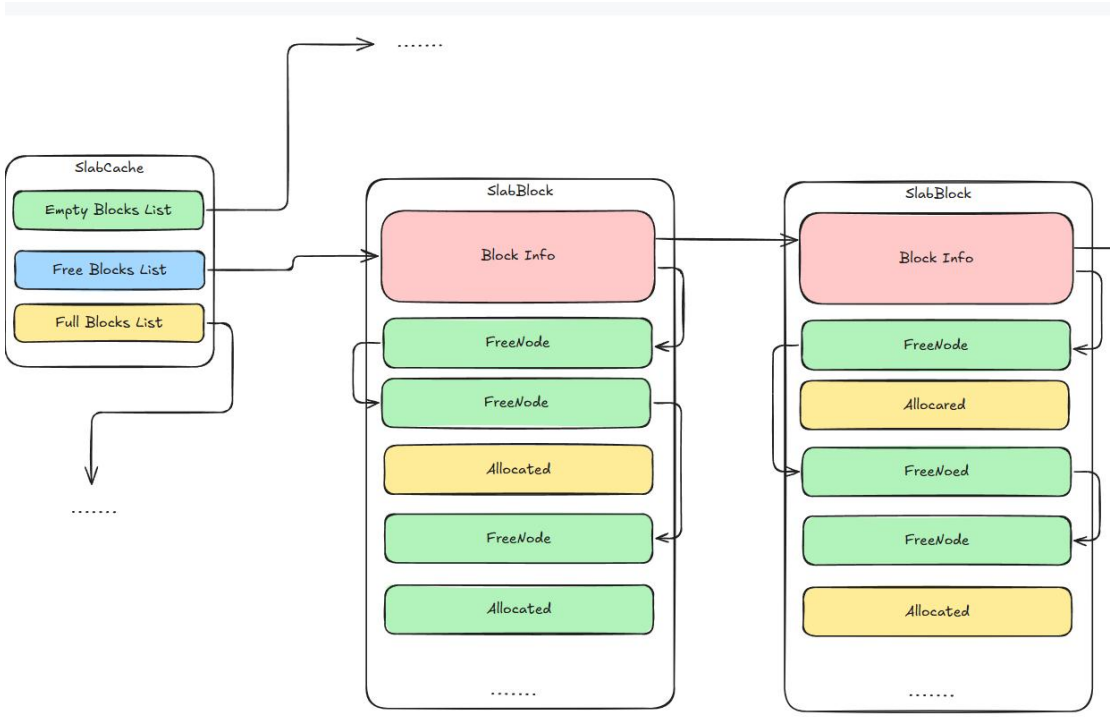
- load_i 表示第 i 个时间窗口内的负载贡献。
- $y = 0.5^{\frac{1}{1024}}$，即指数衰减因子。

- Chronix 参考 Linux 3.8 版本，引入了简化的 PELT 算法来每一个 sched entity 对负载的贡献.

- PELT （Per-Entity Load Tracking）是一种用于动态跟踪任务（进程/线程）和 CPU 负载的算法，旨在为调度器提供更精细的负载信息。它通过指数衰减历史负载的方式，综合考虑任务的当前和历史运行状态，从而更精确地反映系统的实际负载情况。

内存管理

Company introduction and corporate publicity

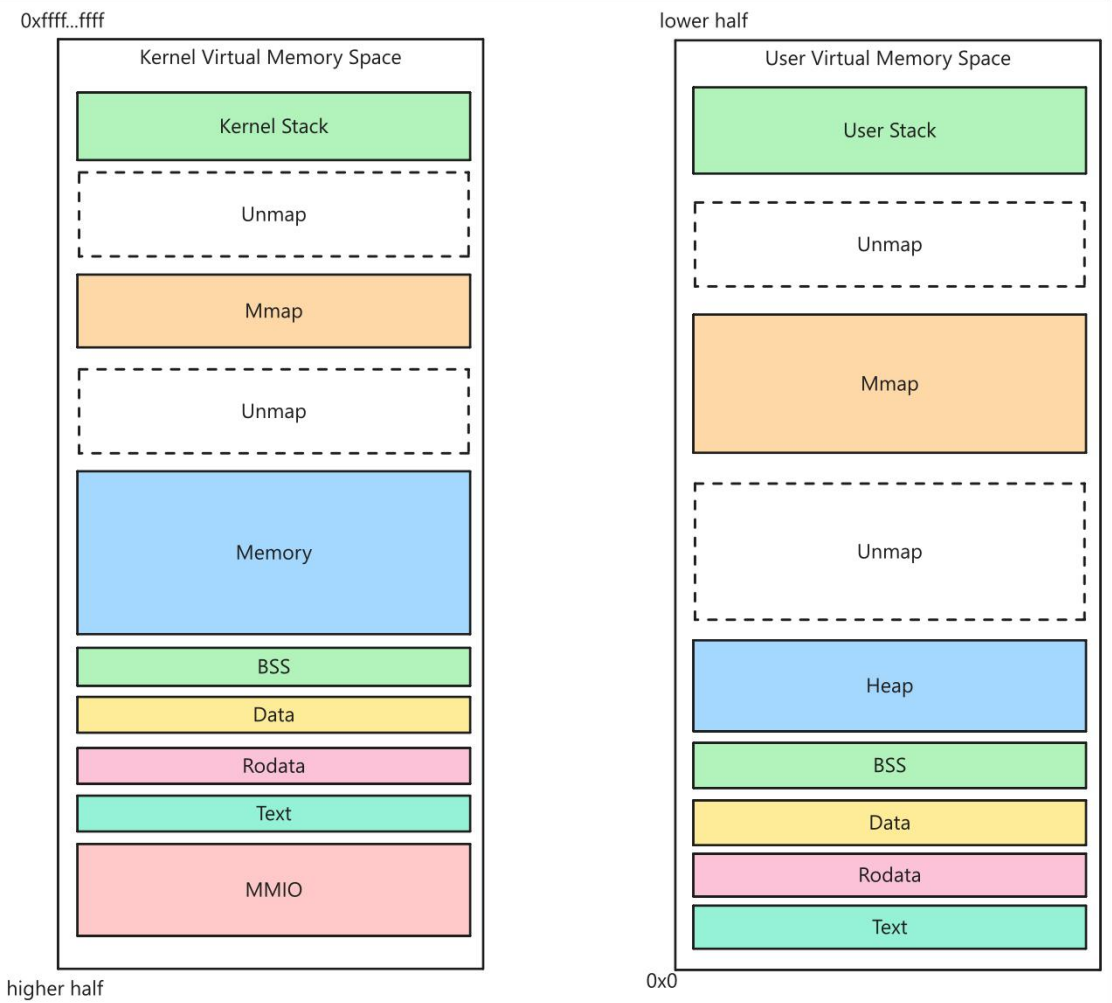


使用了 **BitmapAllocator** 分配页帧，不同于伙伴分配器基于空闲链表实现，而是使用了线段树和位图。查找、分配和回收页帧时，这个分配器的空间局部性更好，性能更高。

自制 **Slab** 缓存，高效实现小对象分配和回收，专门用来分配固定大小的小对象，这类对象大小通常在 200 字节内，可以连续存储在一页内存上。Slab 分配器管理了一系列 Slab 缓存，每个 Slab 缓存只负责分配和回收同一大小的内存块。

内存管理

Company introduction and corporate publicity



- **Chronix** 地址空间的设计如图所示：

- **Chronix** 内核态页表保存在全局内核地址空间 **KVMSPACE** 中，用户地址空间共享内核二级页表。

- 对于内核地址空间，为了方便管理所有物理地址，采用偏移映射的方式将物理地址以加上偏移量的方式映射为虚拟地址

- 对于用户地址空间，为了利用分页机制的灵活性，消除外部碎片，采用随机映射的方式将

- 虚拟页随机映射到空闲物理内存中任意一页。

信号机制

Company introduction and corporate publicity

- 定义**SigAction**： 注册信号处理的行为.
- Chronix 的每个进程都会持有一个 **SigManager** 。 SigManager 给进程间提供了简洁的收发信息的调用接口。 SigManager 几乎包含了一个进程所有信号相关的结构。
- 遵循了 linux 的标准。将信号分为了标准信号和实时信号两种类型。
- 在进程控制块的级别，设计了接收信号的接口。**SigInfo** 结构体包含了该信号的编号以及一些额外信息，方便后续的处理。

网络模块

Company introduction and corporate publicity

```rust
pub trait NetDevice: Send + Sync + Any {
    // ! smoltcp demands that the device must have below trait
    ///Get a description of device capabilities.
    fn capabilities(&self) -> DeviceCapabilities;
    /// Construct a token pair consisting of one receive token and one transmit token.
    fn receive(&mut self) ->  DevResult<Box<dyn NetBufPtrTrait>>;
    /// Transmits a packet in the buffer to the network, without blocking,
    fn transmit(&mut self, tx_buf: Box<dyn NetBufPtrTrait>) -> DevResult;
    // ! method in implementing a network device concering buffer management
    /// allocate a tx buffer
    fn alloc_tx_buffer(&mut self, size: usize) -> DevResult<Box<dyn NetBufPtrTrait>>;
    /// recycle buf when rx complete
    fn recycle_rx_buffer(&mut self, rx_buf: Box<dyn NetBufPtrTrait>) -> DevResult;
    /// recycle used tx buffer
    fn recycle_tx_buffer(&mut self) -> DevResult;
    /// ethernet address of the NIC
    fn mac_address(&self) -> EthernetAddress;
}
```

- Chronix修改并使用使用了 **smoltcp** 库作为对 TCP/IP 协议栈支持的工具库
- 对多种网络设备提出了统一的抽象 **NetDevice** trait，统一的设备层接口以支持多种网络设备并方便与上层基于 smoltcp 的协议栈层进行交互。
- 在设备层设计提供了缓冲层管理，用于管理网络数据包的高效分配和回收，缓冲层的设计参考了 smoltcp 的缓冲区管理，提供了 **NetBuf** ，用于描述网络数据包缓冲区：

设备驱动

Company introduction and corporate publicity

- **Chronix** 目前支持**块设备(Block Device)**、**网络设备(Network Device)**，**串口(Char Device)**和平台级**中断控制器（irq_ctrl)**
- 通过**设备树**解析实现同一份内核二进制在不同的硬件上启动

PART 03

当前成果

Lorem ipsum dolor sit amet, consectetuer dipiscing elit, sed diam nonummy nibheuismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam

工作总结

1. 通过初赛除了部分 LTP 测试外的所有测试点
2. 实现了无栈协程的任务调度。
3. 支持多核调度，采取了更先进的调度算法。
4. 优化可执行文件的加载。
5. 实现 COW 懒分配等基础内存分配优化。
6. 实现 SLAB 内存分配器，零页分配等高效内存分配方式。
7. 实现了高效的用户指针检查。
8. 实现较为完备的硬件抽象层，支持两个架构的正常运行。
9. 实现网络模块，本地回环设备稳定运行。
10. 实现磁盘文件系统挂载
11. 实现虚拟文件系统，便利新的文件系统接入，提供简洁的抽象层

实时排名

Company introduction and corporate publicity

1	T202510213995926	火箭队/ 哈尔滨工业大学	27	2025-06-30 06:59:52	102.0	102.0	102.0	102.0	53.0	53.0	53.0	53.0	6.60489917750369
2	T202518123995568	Chronix/ 哈尔滨工业大学（深圳）	30	2025-06-29 18:26:46	101.0	102.0	101.0	102.0	53.0	53.0	53.0	53.0	4.0
3	T202510003996120	StarryMix/ 清华大学	54	2025-06-30 02:37:16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	T202510336995214	NoAxiom/ 杭州电	49	2025-	102.0	102.0	102.0	102.0	54.0	54.0	54.0	54.0	0.0

PART 04

发展规划

Lorem ipsum dolor sit amet, consectetuer dipiscing elit, sed diam nonummy nibheuismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam

发展目标

基于Chronix对开源社区贡献

优化文件系统读写、路径查找的性能

支持更完备的网络

支持开发板

运行现实应用

哈爾濱工業大學(深圳)

HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

THANK . YOU