**CPUPowerGUI 面向对象分析设计文档**

作者：moriv4

目录

**[1. 项目简介](#_Toc76751911_WPSOffice_Level1)** **[1](#_Toc76751911_WPSOffice_Level1)**

[1.1. 原理分析](#_Toc2055551697_WPSOffice_Level2) [2](#_Toc2055551697_WPSOffice_Level2)

[1.2. 模块介绍](#_Toc1948426868_WPSOffice_Level2) [3](#_Toc1948426868_WPSOffice_Level2)

**[2. 对象分析](#_Toc2055551697_WPSOffice_Level1)** **[5](#_Toc2055551697_WPSOffice_Level1)**

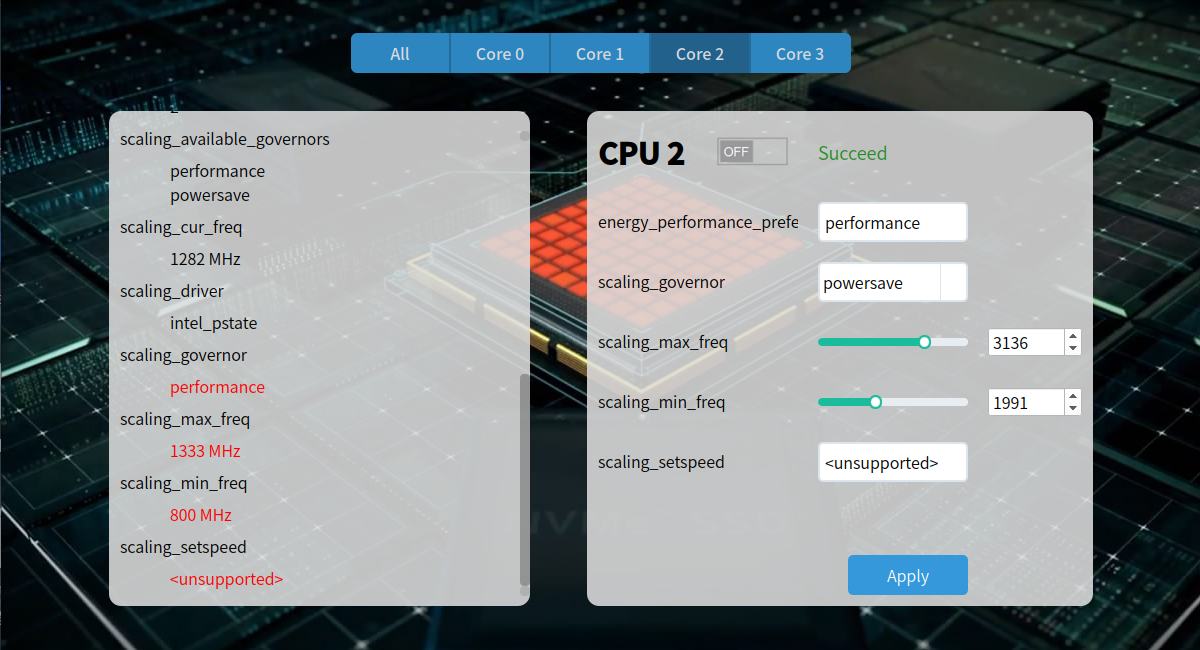
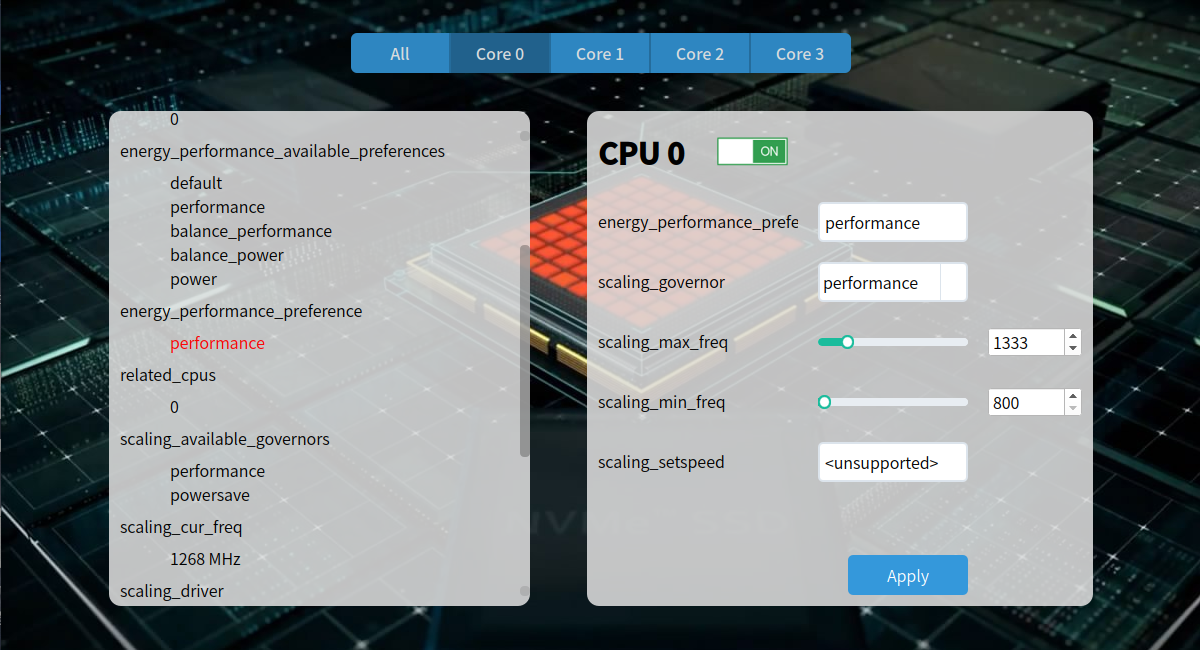
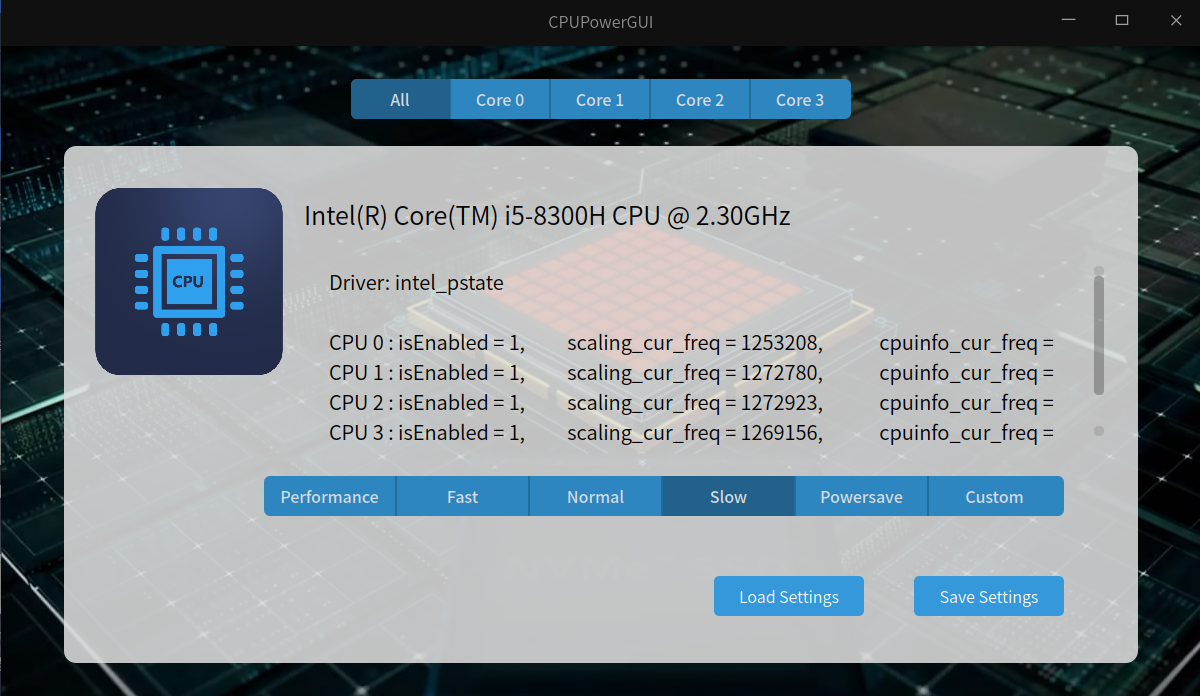
[2.1. 模型分析](#_Toc497549785_WPSOffice_Level2) [5](#_Toc497549785_WPSOffice_Level2)

[2.2. 视图分析](#_Toc1042699986_WPSOffice_Level2) [5](#_Toc1042699986_WPSOffice_Level2)

**[3. 对象设计](#_Toc1948426868_WPSOffice_Level1)** **[6](#_Toc1948426868_WPSOffice_Level1)**

# 项目简介

CPUPowerGUI是Linux系统上带图形界面的CPU频率调节工具。它的功能有：查看CPU信息、修改频率、开启或关闭CPU核心等。



## 原理分析

内核提供的CPU相关信息放在 /sys/devices/system/cpu/\* 中。程序做的是把这个文件目录结构映射为图形界面元素。程序每秒读取一次信息并刷新界面，如果用户修改CPU选项就把相关内容写到/sys/devices/system/cpu/\* 对应的文件里。

文件目录结构说明：

/sys/devices/system/cpu/cpu[0..1..2..3..n] 文件夹代表CPU有多少个核心

/sys/devices/system/cpu/cpu0/topology/\* 记录核心ID、超线程信息等

/sys/devices/system/cpu/cpu0/cpufreq/\* 频率调节、控制等

/sys/devices/system/cpu/cpu1/online 开启、关闭CPU核心

相关文档：

<https://www.kernel.org/doc/Documentation/cputopology.txt>

<https://www.kernel.org/doc/Documentation/cpu-freq/user-guide.txt>

<https://www.kernel.org/doc/html/v4.14/admin-guide/pm/cpufreq.html>

图形界面使用Qt5 Widgets。由于每个人的CPU不同，编程时不知道/sys/devices/system/cpu/里面有什么文件，所以界面上的控件大部分是动态创建的。得益于QObject的优秀设计（父子关系和对象名称），动态创建控件十分容易。

## 模块介绍

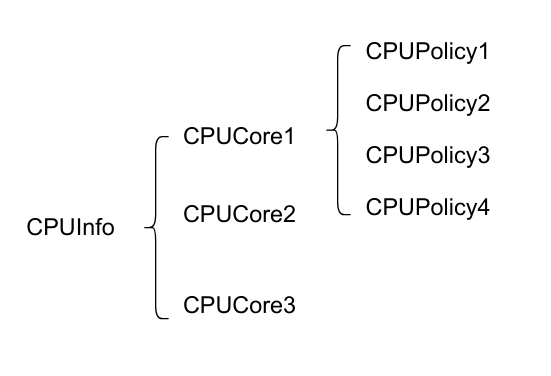
软件使用MVC框架设计模块。

模型：

CPUInfo ： 代表整个CPU

CPUCore： 代表一个CPU核心

CPUPolicy ：记录一条CPU频率策略信息，包括它的名字、值、读写性。



视图：

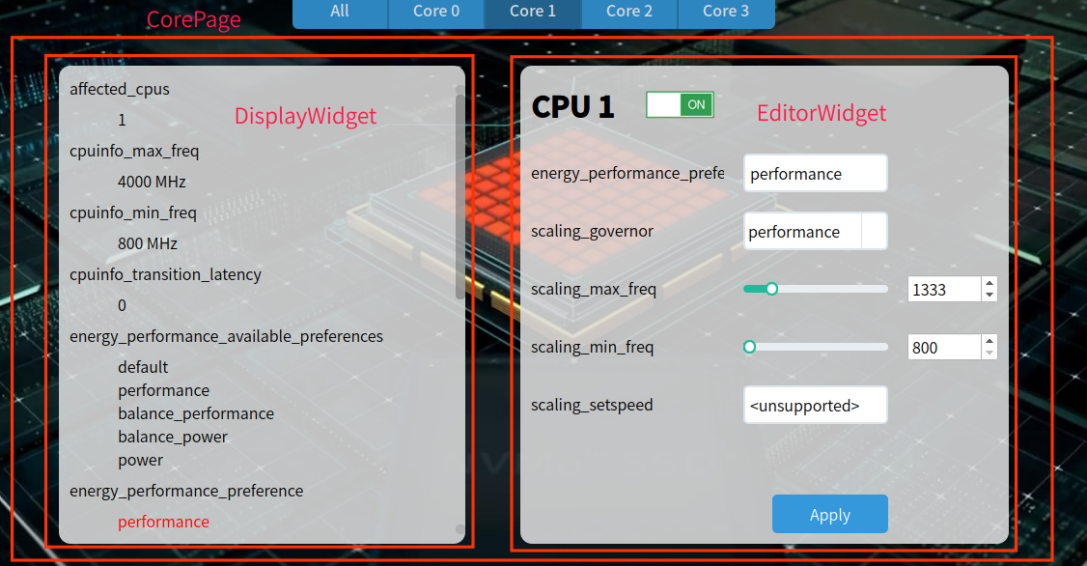
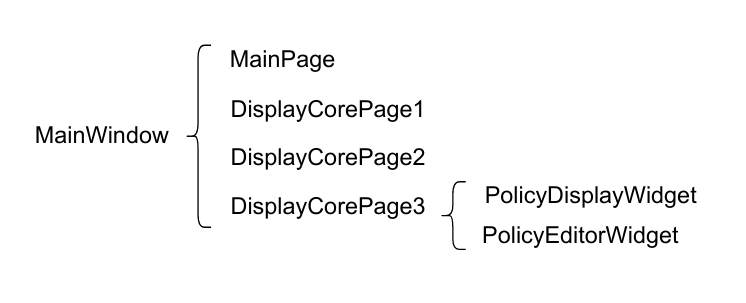
DisplayCorePage ： 显示某个CPU核心的信息

PolicyDisplayWidget ： 显示某个CPU核心的所有策略

PolicyEditorWidget ： 某个CPU核心的策略编辑器

MainPage ： 程序的首页面，显示CPU整体信息

MainWindow ： 程序的主窗口





控制器：

使用面向对象方法，控制器体现在上述模块的类的成员函数中。

# 对象分析

## 模型分析

CPUPolicy是和频率调节有关的选项。

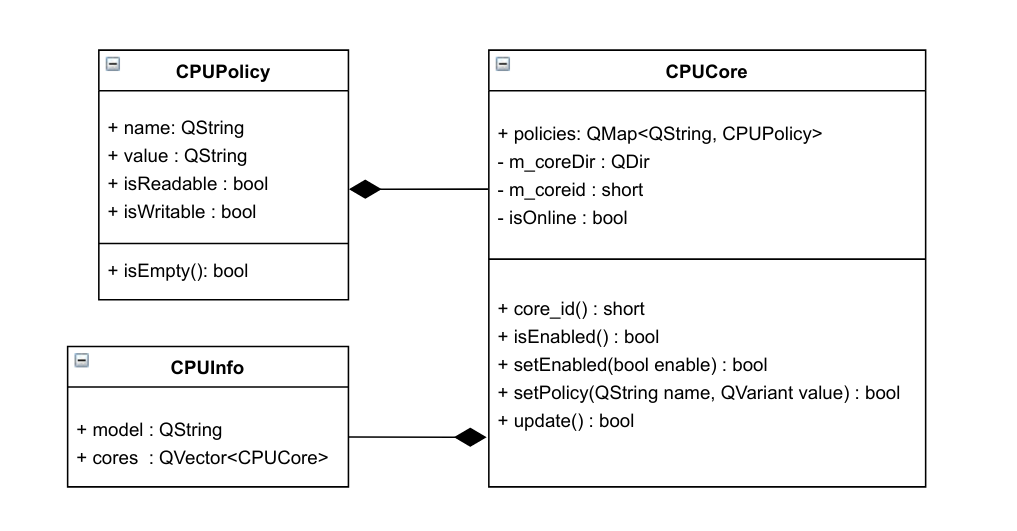
从文件的角度看，例如 /sys/devices/system/cpu/cpu1/cpufreq/scaling\_min\_freq文件代表第二核心的最小频率，文件内容是 800000 (800MHz)，文件属性是对root用户可读可写。

从数据结构的角度看，可以把它抽象成键值对和一些属性的集合，成员变量name是选项名，value是值，isReadable isWritable 是可读可写的布尔变量。这样就和文件对应上了。

CPUCore是对一个核心的抽象。它有编号id、是否启用isEnabled、可用的CPUPolicy等数据，有修改CPUPolicy、刷新数据等功能。

CPUInfo 代表整个CPU，记录CPU的型号和几个核心。

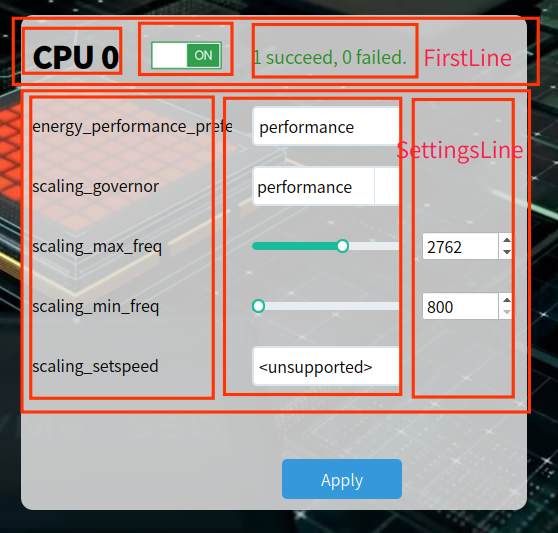
模型部分UML：



## 视图分析

PolicyDisplayWidget 把CPUCore中的policies信息映射为图形界面中的控件，内部控件是在它初始化时动态创建的，每秒刷新控件的内容。我在这里使用的是FormLayout和一堆QLabel。更好的方案是用RichTextBox之类的控件，希望有人帮我改进一下。因为内容很多，所以这里放了一个QScrollArea滚动区域。

PolicyEditorWidget是选项编辑器，它把CPUCore中可写的policies信息映射为可编辑控件。它的内部控件同样是在初始化时动态创建的。中间布局的是GridLayout，上面是HBoxLayout。getEdiorPolicyValueWidget函数根据参数中的CPUPolicy类型选择合适的编辑控件。



DisplayCorePage 把PolicyEditorWidget 和 PolicyDisplayWidget组合起来形成一个整体。

MainPage显示CPU的整体信息，也是每秒刷新一次。

MainWindow 有一个SwitchButton和StackLayout，可以选择不同的页面显示。实际的刷新操作是在这里进行的。只有这里有QTimer，选择哪个页面，每秒刷新哪个页面。

# 对象设计