讲稿

p4

接下来我将以“快问块答”的形式简单介绍LoongArch。

第一个问题：“什么是LoongArch?”
答：“LoongArch由龙芯公司推出的一种 RISC 风格的指令系统架构”

第二个问题：“LoongArch 的意义？”
答：“打破外国指令集垄断，建立完全自主的信息产业体系”

第三个问题：“LoongArch 的发展情况？”
答：“硬件平台基本完善，基础软件获得开源社区广泛认可”

p5

在硬件平台，基于LoongArch的CPU产品包括面向桌面与服务器的龙芯3号5000系列，以及面向工控与中断的龙芯2K0500和2K1000LA等。

此外，龙芯公司还推出了7A1000与7A2000桥片，与龙芯3号CPU搭配组成完整的硬件系统

p6

此外，LoongArch已经支持了多种主流的基础软件。

BIOS与内核支持PMON，UEFI和Linux

编译器支持GCC，LLVM，Golang

基础库支持glibc

拥有了这些硬件平台和基础软件的支持，LoongArch生态已经达到了一个“能用”的状态

p7

本课题所研究的对象是LoongArch上运行的实时操作系统。

目前的发展情况是龙芯公司针对嵌入式产品提供了闭源的 loongOS

但是还没有开源的 LoongArch RTOS 支持

本课题所做的工作是将开源实时项目 Xenomai 移植到 LoongArch 平台，并建立了实时任务模型进行评估和优化，

最终的产物是提供了基于龙芯 3A5000+7A2000 平台的 Xenomai 开源代码和有利于实时场景配置

p8

接下来背景部分，也就是关于Xenomai项目的介绍

p9

Xenomai项目是一个基于ADEOS的“双内核”系统

ADEOS指“自适应多域复用环境”是一种虚拟化技术，用于提供多个内核同时运行的环境

双内核指的是Xenomai系统中cobalt实时内核和Linux通用内核协同工作，提升实时性

左侧是Xenomai项目的架构图，主要包含了Xenomai项目的3大模块：I-pipe，cobalt和libcobalt

I-pipe是ADEOS的一个具体实现，提供了双内核运行基础

cobalt是专用的实时内核，用于提高系统实时性

libcobalt是Xenomai提供的用户实时库，用于编写实时程序，获取实时内核提供的实时服务

p10

Xenomai项目的优点在于

一，能够提供较好的实时性，官方宣称能够提供硬实时保障

二，带来了较好的兼容性，得益于libcobalt中实现了若干与主流API相兼容的API，官方宣称的效果是只需要对实时程序代码进行很小的改动，就能将一个基于实时框架的实时程序无缝迁移到Xenomai系统中

p11

下一部分是为了研究和评估Xenomai系统所建立的实时任务模型

p12

实时任务模型可以表述为一个二元组ti和一个三元组ek，其中ti用于描述实时任务，ek用于描述外部事件

p13

接下来通过一个例子来说明，在时间轴上，有一个外部事件在tai时刻到达，经过实时操作系统的一系列过程，响应它的实时任务ti在txi时刻开始执行

经过一段时间后，ti于tendi时刻运行完毕

在实时任务模型中，定义了两个量描述实时任务的运行过程，第一个是延迟delta tli，表示从外部事件到达，到操作系统调度实时任务开始执行的时间

另一个是运行时间delta tri，表示从实时任务开始执行，到运行完毕的时间

p14

实时任务模型的另一部分是外部事件ek，我们将响应ek的任务集合记为tiek

时间轴上的tdi是ek的截止时间，如果响应ek的实时任务ti在tdi还没有运行完毕，则整个系统的运行可能发生问题，甚至影响物理世界

ek三元组中的第一个分量cek1刻画了外部事件的截止时间

第二、三个分量作为一个范围的起始和终止点刻画了外部事件的周期性，也就是说每相邻两次ek发生的时间间隔都包含在了cek2和cek3所限定的范围内

p15

在实时系统中，希望响应ek的实时任务ti在tdi前运行完毕，也就是tendi小于等于tdi，这个条件等价于delta tli与delta tri之和小于等于cek1

当这个条件不成立，称系统发生了超时故障

p16

通过实时任务模型，Xenomai实现实时的原理核心在于缩小delta tli与delta tri之和，并使之变得具有确定

它使用专用的实时内核进行调度，结果就是缩短了延迟，实时任务能够尽快被调度

实时内核为实时应用提供的系统调用，结果是缩短了运行时间，保障了实时任务运行时能够尽快运行完毕

最后它通过libcobalt的方式向程序员提供实时API，使得编程时能够区分实时和非实时的API，相当于从宏观上缩小了delta tli与delta tri之和

p17

下一部分是移植过程

p18

Xenomai到LoongArch平台的移植基本上按照从底层向上层的顺序

首先移植双内核的基础I-pipe，就需要对龙芯3A5000的中断结构，LoongArch Linux的中断处理，以及I-pipe的实现原理有一定的了解

移植成功I-pipe后，开始尝试在系统中加入第二个实时内核coalt，这一部需要了解LoongArch中的时间硬件资源，包括定时器和计时器。接下来需要通过阅读代码的形式学习cobalt工作的原理，并找到哪些体系结构相关的接口需要进行移植

最后一部分是对libcobalt的移植，移植的难点主要在对于LoongArch ABI的理解，保证系统能够正确地进行系统调用，并传递参数

p19

移植成功后，就进入了性能优化的工作

p20

实时操作系统的性能主要包括两个部分，实时性和稳定性

因此定义了两个统计量对其进行评价，时间轴中间的部分表示外部事件多次到达，实时任务开始执行

实际执行时，实时任务的延迟将落在一个范围内，这个范围的大小称为抖动tj，它反映了实时系统的稳定性

而大量实时任务的延迟具有一个数学期望，称为延迟期望te，它间接反映了实时系统的实时性

p21

由于Xenomai是硬实时系统，不允许超时故障发生，因此还需要一个额外的约束

也就是最大的delta tli与delta tri之和小于等于cek1

p22

优化Xenomai系统所采用的测试工具是Xenomai测试集中自带的latency测试

它可以测量响应定时器中断事件el=(tperiod,tperiod,tperiod)时实时任务的延迟

最终得到的测试数据时运行大量实时任务样本后，延迟的最大值tmax，最小值tmin和平均值tavg

得到抖动为tmax和tmin之差，te可以使用tavg来估计

p23

另外，在latency测试中，实时任务运行的时间很短，可以认为delta tri约等于0，从而硬实时的约束条件就变为了tmax小于等于tperiod

p24

测试时，将latency的周期设置为100us，el，硬实时约束为

使用的物理平台是

测试使用了两种负载，无负载和访存负载，访存负载使用stress程序添加

p25

经过多次测试，最终发现在LoongArch平台上影响系统性能的因素有三个，对应进行优化的方法为：

关闭页迁移，主要选项是透明巨页THP；

关闭高延迟外设，包括声卡与实时时钟；

最后进行关键路径的优化，我主要做的是将定时器中断路径上的两个函数进行了体系结构相关优化，减少了指令数量

p26

页面中无负载条件下的测试结果图a是RTLinux下运行latency测试，作为对照，bcd分别对Xenomai进行了不同程度的优化

可以发现，无负载下Xenomai的实时性和稳定性均优于RTLinux，并且优化方法对无负载下的系统性能影响不大，两种系统均满足了硬实时约束

p27

接下来是访存负载下的测试结果可以发现图e，f中的高延迟样本数量相较于无负载的情况增加明显，无法满足硬实时约束，实时性的稳定性都很差

通过fg的对比可以看出页迁移对Xenomai的平均延迟影响很大，实时性在关闭页迁移后得到了很大的提升，但是仍然有超过100us的延迟

当关闭了高延迟外设后，g中的高延迟样本消失，变为了h中的图像，提升了系统的稳定性，并满足了硬实时约束

p28

最终的结论是经过一系列优化方法，可以使LoongArch平台上的Xenomai系统达到硬实时要求