2018

读书报告

报告人：蒋丽婷

时间：2018.6.5

CONTENTS

Singularity: Rethinking the Software Stack
重新思考软件堆栈

题目

Galen C. Hunt 和 James R. Larus

作者

操作系统、安全编程语言、程序验证、程序规范、密封进程架构、密封内核、软件隔离进程、硬件保护域、MBP、不安全代码税

关键词

本文描述了singularity项目根据编程语言和验证工具的进步重新审核这些设计所付出的努力。这篇论文总体可分为3部分，第一部分介绍了该项目的起源。第二部分描述了该系统的内核实现，最后一部分则是总结过去所做的工作。

总结


广泛的安全漏洞
缺乏稳健性
应用程序间的意外交互
错误的扩展、插件和驱动程序导致的失败

如果软件平台是从头开始设计

其主要目标是改进可靠性和确实性

那么软件平台会是什么样子？

singularity操作系统是基于软件隔离整合的新软件架构使用的编程语言是sing#，该语言是c#的扩展，为操作系统通信原语提供了可验证的一流支持，并为系统编程和代码因子提供了强有力的支持。

软件独立进程

基于合同的渠道

基于清单的程序

软件独立进程（SIP）

SIP与操作系统中的其他进程共享许多属性。每个用户程序的执行发生在SIP的上下文中。与SIP关联的是一组包含代码和数据的内存页面。SIP包含一个或多个执行线程。SIP以安全身份执行具有关联的操作系统安全属性。最后，SIP提供信息隐藏和故障隔离。

软件独立进程

基于合同的渠道

基于清单的程序

基于合同的渠道

Singularity中SIP之间的所有通信都通过合同渠道进行流动，一个通道是具有两个端点的双向信息通道。一个通道提供了一个无损的。有序的消息队列。在语义上，每个端点都有一个接收队列，在一个端点发送信息，在另一个端点排队接收信息。通道之间的通信由通道合同来描述。

软件独立进程

基于合同的渠道

基于清单的程序

基于清单的程序（MBP）

MBP是由静态清单定义的程序，没有代码被允许在没有清单的情况下在singularity系统上运行。清单描述了MBP的代码资源，所需的系统资源，所需的功能以及对其他程序的依赖关系。在系统上安装MBP时，清单将用于识别并验证MBP代码是否满足所有必须的安全属性，以确保可以满足所有MBP的系统依赖关系，并防止MBP的安装受到之前安装的MBP的干扰。

CONTENTS

题目

Disco- Running Commodity Operating Systems on Scalable Multiprocessors
Disco:在可扩展的多处理器上运行商品操作系统

作者

EDOUARD BUGNION, SCOTT DEVINE, KINSHUK GOVIL, and MENDEL ROSENBLUM

问题

可扩展共享内存多处理器之类的硬件创新需要对操作系统软件 进行重大改变。这些变化有巨大的成本，它可能会带来不稳定性，不可靠性和 不良数据共享

总结

本文试图通过引入一层虚拟机监视器来解决 这个问题，该监视器能够通过多路复用硬件来运行多个商品操作系统以实现最 大的利用率。

Disco

Disco 是一款虚拟机监 视器，它通过在操作系统和硬件之间增加一层额外的系统软件，在可扩展的共 享内存多处理器系统上运行许多商品操作系统。由于代码库较小，与商品操作 系统相比，它可以更容易地演变和修改。

作者通过大量工作负载(包括工程，科学和数据库)提供了详细的实施评 估结果。比较 IRIX 和迪斯科，与不使用 VMM(主要来自陷阱和 TLB 重新加载) 相比，VMM 产生的开销在 3%和 16%之间。内存在虚拟机之间进行分区，NFS 提供的内存比可用内存更多。动态页面迁移和复制图表显示，与商品操作系统 相比，性能提高了约 33%。

CONTENTS

Hyperkernel Push-Bu on Verification of an OS Kernel
Hyperkernel:操作系统内核的按钮验证

题目

Luke Nelson, Helgi Sigurbjarnarson, Kaiyuan Zhang, Dylan Johnson, James Bornholt, Emina Torlak, and Xi Wang

作者

系统接口、c、状态机、 Hyperkernel、 Z3 SMT 解算器

关键词

论文 供了:一个低开销的方法来构造一个验证过的操作系统 内核、一种便于SMT求解的接口设计方法、一个有合适的性能的操作系统(Hyperkernel)的实现。

总结

论文通过两种规范来描述“正确的”内核接口

包含更多的细节

状态机规范出了一个内核调用和中断处理函数的状态机，以及这个规范的“正确的”C 实现;

声明式规范 用具体的语言描述规范

便于理解

三个挑战

界面设计

内核代码中的虚拟内存管理

用 c 编写，会使形式化推理变复杂

1

2

3

三个思想

限 定了内核接口以避免无限循环或递归

分离内核和用户地址空间以简化虚拟内存 的推理

在 LLVM 中间表示级别执行验证以避免建模复杂的 C 语言

题目

The Scalable Commutativity Rule: Designing Scalable Software for Multicore Processors
可扩展交换规则:为多核处理器设计可扩展软件

作者

Austin T. Clements, M. Frans Kaashoek, Nickolai Zeldovich, Robert T. Morris, and Eddie Kohler†

问题

如果把接口和实现良好地分离开来, 那接口的设计中是不是就蕴含了一些天然的限制, 使得无论怎样实现, 都无法突这些天然的限制

总结

本文提供了一个非常好的交换性观点，并 供了现有接口的具体示例来帮助理解问题和解决方案。 这些例子使得理解这些概念变得更容易。此外，性 能评估清楚地表明交换性有助于实现可扩展性

SIM-交换性规则

如果一个历史 Y 在 H 中交换，存 在一个正确的实现，其中区域 Y 是无冲突的。这意味着内存访问是缓存行的互 斥区域，这意味着该接口会缩放。 SIM 交换特别意味着操作必须是状态依赖的， 基于接口的和单调的。

如果没有 COMMUTER，确定接口的可扩展性并 改进它的常见方法是在不同数量的内核上使用给定的工作负载对其进行测试，然 后使用差异分析等工具查找瓶颈并在重新测试之前对其进行改进。

C OMMUTER

实验结果

在论文最后部分，作者给出了详细的评估。 在 COMMUTER 中对几个 POSIX 文件系统和虚拟内存调用进行建模后，用于评估 Linux 的可伸缩性。 将 18 个 POSIX API 与 sv6 API 进行比较。C OMMUTER 确定 sv6 在 13,664 次测试中的 13,528 次是无冲突的，而 Linux 在 9,389 次测试中没有冲突。

CONTENTS


Lottery Scheduling- Flexible Proportional-Share Resource Management
彩票计划:灵活的比例分享资源管理
题目
Carl A. Waldspurger William E. Weihl
作者
必须共享 稀缺资源以支持具有不同重要性的请求，并且选择处理此问题的策略可能会对响应时间和吞吐量产生巨大影响
问题
这篇论文提出了一个基于随机数和比例分配的资源分配 机制的彩票调度器。
总结

主要问题

01

对于交互式计算，用户需要能够快速将可用资源集中在当前重要的任务上。很少的通用计划可以提供对服务费率的这种响应控制

02

虽然使用了优先权的概念，但 据观察，优先权和动态优先权调整方案的分配通常是特设的

03

某些能够解决某些 问题的调度程序会伴随着太多的假设和开销，因此会限制系统仅对较长时间的计 算进行相对粗略的控制

根据每个过程的优先级为每个过程提供一定数量的 票据。门票编号从 1 到 n。一个随机数发生器选取一个进程来执行一个时间片， 其票号与生成的随机数匹配。因此，拥有更多票证的流程会有更多的机会执行。

这些票据可以在诸如客户端和服务器之类的信任过程之间进行转移以协同 执行。票据通胀应该动态控制流程中的执行率，但应谨慎使用。在保护边界内，货币就像是控制边界内相对执行率的本地票据。它可以根据传输速率转换为通用 票据(基础货币)。补偿标签将给予仅使用一小部分获奖时间片的流程的票数膨胀，以确保该流程的实际执行时间与授予的票数成正比。

CONTENTS

题目

The Multikernel: A new OS architecture for scalable multicore systems
多内核:可扩展多核系统的新操作系统架构

作者

Andrew Baumann*, Paul Barham†, Pierre-Evariste Dagand‡, Tim Harris†, Rebecca Isaacs†, Simon Peter*, Timothy Roscoe*, Adrian Schüpbach*, and Akhilesh Singhania*

问题

传统的单片 操作系统对解决这种可扩展性问题和优化各种硬件结构提出了巨大的挑战。

总结

本文作者提出了 Multikernel 及其原型 Barrelfish，这是一种新的操作系 统体系结构，它继承了分布式系统的主要特性和见解，旨在实现跨不同多核硬件 系统的更好的可扩展性。

3

Multikernel设计原则

明确核心通信

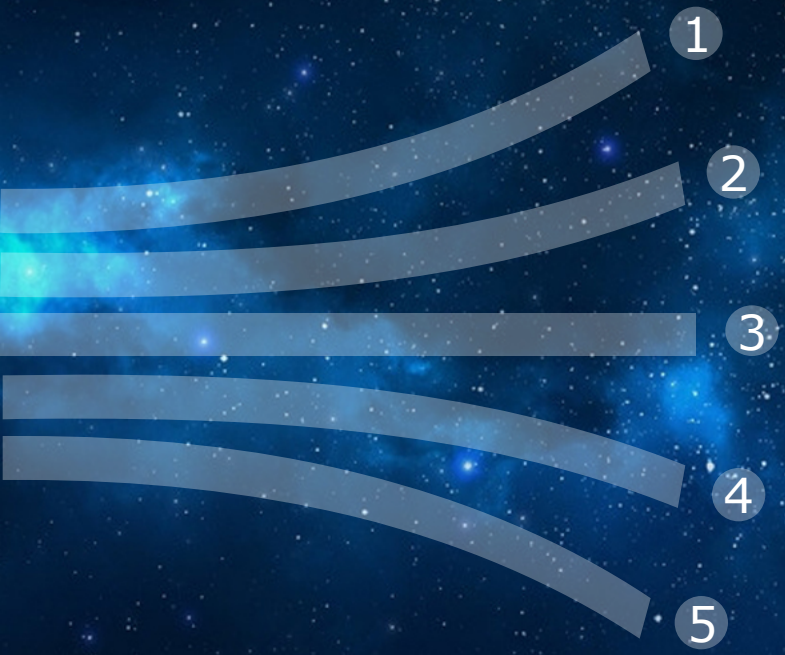
明确所有核心间通信，而不是在通信内核之间共享内存，所有通信都使用消息传递完成。这提供了模块化，因为 通信已经完成了定义良好的接口。

os结构硬件中立

使 OS 结构硬件中立，如上所述，这使 得 OS 具有可扩展性和未来的可行性。消息传递机制和硬件接口是依赖于硬件的;。

视为复制状态

将状态视为复制而不是共享，每个内核都有自己的状态副本。这些状态之 间保持一致。这个副本靠近核心可用，因此具有更好的性能。



他们的实验表明，通过扩展线程/内核数量，消息传递证明性能比共享内存模 型更好。系统结构具有提供硬件结构访问权限的低级别 CPU 驱动程序和负责监 视内核之间的 RPC 的用户监视器。在内核之间复制内核状态会使系统中具有不 同内核的可能性，从而不能针对单个体系结构优化操作系统。

THANK YOU

汇报人：蒋丽婷
2018.6.5