一种轻量级软件划分系统①

李利锋[1]，吴俊敏[1,2]，朱晓东[1]，黄　景[1]，蒋　楠[1]

[1](中国科学技术大学计算机科学与技术学院，合肥 230027)
[2](中国科学技术大学苏州研究院，苏州 215123)

摘　要：提出一种应用在多核网络处理器平台上的轻量级软件划分方案，以支持多个异构 OS 在同一物理主机中同时运行，在尽量满足整个网络处理系统可靠性和安全性的前提下提高系统的业务融合能力。首先分析传统系统虚拟化技术在网络处理器平台中应用的缺陷和不足，并提出一种轻量级的软件划分方案；接着分别从系统物理资源划分、运行时控制系统和分层运行结构几个方面描述系统设计中的关键问题；最后通过测试数据验证了整体系统的性能。

关键词：软件划分;虚拟化;多核网络处理器;虚拟机监视器;客户操作系统;可靠性

Lightweight Software Partition System

LI Li-Feng[1], WU Jun-Min[1,2], ZHU Xiao-Dong[1], HUANG Jing[1], JIANG Nan[1]

[1](School of Computer Science and Technology, USTC, Hefei 230027, China)
[2](SuZhou institute for Advanced Study, USTC, Suzhou 215123, China)

Abstract：Nowadays, the NP of Multi-core architecture is being used widely. We bring up a new lightweight software partition method, which improves the capability of integration of different network tasks on one host. Thanks to this system, many different OS can run correctly in the same time on one physical host. First of all, in this article, after the comparison to the mainstream VMM, we propose our idea. And then, we detail our work in three aspect: the partition, the control and the security. Finally, by running the prototype system, we test it and finish the analysis roughly.

Key words：software partition; virtualization; network processor; hypervisor; guest OS; reliabilit

随着多核网络处理器的发展，基于多核 NP（Network Processor）架构的产品不断涌现[1]。为了满足某些业务需要，缩短开发时间，有必要在一套物理主机上同时运行多个 OS 或者 BM（Bare Metal），这样就导致高度集成的 SoC 系统中的软件环境也越来越复杂，从而引起安全性，可管理性，可靠性等方面的问题。为解决上述问题，本文提出一种针对多核网络处理器应用环境下的软件划分方案。论文首先分析传统虚拟机技术的特点，通过对比归结出我们的软件划分方案模型；其次，深入分析所提出的模型在多核网络处理器应用环境下的各种优缺点；再次，论文描述了所提出的软件划分方案在设计和实现中所遇到的主要问题。

1　软件划分方案模型

1.1 传统虚拟化技术

传统虚拟化技术以 Vmware、Xen 为代表，提供了在一台物理主机中同时运行多套 Guest OS 的能力。Vmware 以二进制翻译技术为基础，使 Guest OS 可以无修改的运行在虚拟主机上，即所谓的全虚拟化技术；Xen 通过修改 Guest OS，使多个 Guest OS 能同时运行在同一物理主机上而不相互干扰，即所谓的半虚拟化技术[2]。虽然传统虚拟化技术支持多个异构 OS 在同一物理主机的同时运行，但是却不适合在网络数据处理环境中的应用。主要原因包括：难以避免的性能开销，系统设计的复杂性，难以提供良好的实时性支持等。以 IBM 的 Z 系列产品为代表，通过软硬件协同设计，

① 基金项目:中央高校基本科研业务费专项资金资助,资助号 WK0110000020.
收稿时间:2011-09-09;收到修改稿时间:2011-10-14

提供在同一物理主机中同时运行多个 OS 的能力，但是这种方案受限于硬件/固件，缺乏灵活性且设计成本较高[3]。另外，IBM 在其 AIX SYSTEM 中实现了一种纯软件的划分方案，可以支持在一个宿主 AIX 中运行多个负载分区[4]，每个负载分区运行一个隔离的系统镜像。但是这样的设计方案仅能限于运行 AIX 这样的系统，目前对其他系统缺乏支持。采用这样基于容器设计方案的系统级虚拟化技术的还有如 Solaris10, Virtuozzo for Linux, and Linux-VServer 等，也面临同样的问题。

1.2 一种轻量级的软件划分方案模型

针对多核网络处理器应用环境下的各种问题，我们提出了一种轻量级的软件划分方案，其模型图示如下：

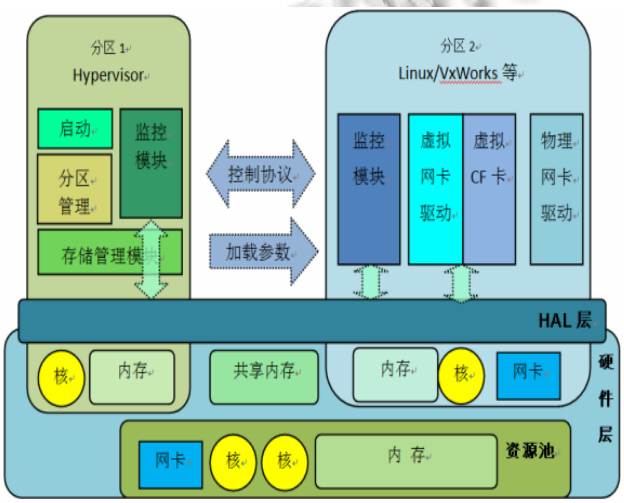

图 1 软件划分方案模型

这里提出所谓分区的概念，分区是物理硬件资源的集合，在每个分区上可以运行独立的 OS 或 BM。上述软件划分方案中，主机中的各种硬件资源由原型系统命名为 KMON(Kernel-Monitor)。负责监管，KMON 本身占用某个分区(图 1 中为占用分区 1 中的内存资源和 CPU 核心)，始终保持运行。KMON 负责所有分区的管理与资源分配（例如分区的创建撤销，CPU 物理核、内存、外设等得分配），OS 和 BM 的加载与启动（从外设加载 OS kernel 到内存并跳转执行），OS 和 BM 的人机交互等（通过共享内存实现的简单字符输入输出），并通过控制协议管理 OS 和 BM 的运行。从硬件层面上看，KMON 系统的核心任务在于将整机的物理资源，如 CPU 的物理核心、物理内存、特殊外设等划分到不同的资源分区，使分区中运行的 OS 或 BM 仅能使用本身所在分区中的物理资源而不会对其他分区中运行的应用有任何影响；从软件层面上看，KMON 系统的核心任务在于如何高效、灵活、安全的完成面向多 OS 的物理资源分配，如何有效管理同时运行在一台物理宿主中的多个 OS，为上层应用的任务划分提供良好支持。

相对于传统系统虚拟机，这种轻量级的软件划分方案主要特点如下：

(1) 灵活性

KMON 监管所有本机物理资源的使用，负责分区的创建和撤销，负责 OS 的加载与启动，关闭，重启等。既可以在 OS 启动前根据配置文件指定对应分区使用的物理资源，也可以在 OS 运行时通过控制协议对某些资源进行动态调整。

(2) 高性能

KMON 本身仅需要极少的物理内存但却需要占用某个独立的物理核心。分区上运行的 OS 或 BM 被设计为可直接使用属于该分区的关键物理资源，例如物理核，内存，关键 I/O 设备等，其使用方式与普通的方式没有任何区别，此外，通常仅有简单的低速外设是被模拟的。相对于传统虚拟机，这种对物理资源的使用策略显然是低效的，但是却有设计的复杂性以及额外的性能开销，这一特点对于某些实时性应用提供了天然的支持。例如，对于某个外设的 I/O 中断信号可以直接被路由到使用该外设的 OS 或 BM 所占有的某个物理核，触发对设备中断信号的处理而不引入任何额外的性能开销。

(3) 安全性

相比于传统虚拟机各种复杂的设计，这里的 KMON 可以设计的简洁和精巧（实现中的原型系统代码量在 1 万行左右），从而容易获得更高的可靠性。传统的基于宿主的系统虚拟机通常是一种紧耦合的关系，一旦系统虚拟机出问题则整个系统中的所有应用都会出错，而 KMON 则不然，因为它并不干涉分区上 OS 对物理资源的使用，这样就极大的降低了整个系统的安全风险。同时，对于各种 OS 和 BM，系统采用了分层的划分方案，兼顾了性能和安全性需求（稍后详解）。

2 关键问题设计与实现

系统的设计主要从以下三个方面进行阐述，物理

资源划分，运行时控制系统，分层的运行结构。系统基于 UBOOT 进行二次开发，通过添加上述模块形成 KMON 系统原型。

2.1 物理资源划分

从 KMON 的观点，系统的物理资源可分为三大类：物理核、内存、外设和 I/O.多核系统上通常以二进制的一个 bit 标示某一个物理核，这样的描述方式不仅方便进行物理寻址而且可以方便的表示物理核的集合，KMON 的实现中也遵从了这一观点，在 KMON 数据结构中使用长整形的 bit 位标示系统中存在的物理核。不同于传统系统虚拟机，KMON 系统中所有 OS 均固定使用某些物理核。

对于物理内存的管理和使用，是传统虚拟机技术中最核心的功能之一。典型的虚拟机实现中，对于物理内存的管理通常可以精细到页框，即被广泛采用“影子页表”技术[5]。Guest OS 中的页表由其本身维护，负责完成从 Guest OS 的线性地址到 Guest OS 的物理地址的转换；虚拟机负责维护“影子页表”，负责完成从物理地址到机器地址（真正的物理地址）的转换；两者协同工作，保证 Guest OS 可以正确的读写自己所需数据。采用影子页表机制使各个 Guest OS 使用的物理内存相互隔离，在此基础上的“气球驱动”，页共享机制以及其他优化技术极大的提高了系统物理内存的利用率.但是，这项技术实现复杂，且有不可避免的性能开销，在某些实时系统上并不适用。

KMON 没有采用复杂的影子页表机制，其设计思想为：KMON 仅负责系统资源的分配与监管，如何高效利用物理资源是 Guest OS 自己的事情。在目前的实现方案中，KMON 实现了自己的物理内存分配器，可根据配置文件为将要被加载的 OS 或 BM 分配某块物理内存区域。对于 Linux 系统，通过修改内核启动部分代码，使其能根据传入的参数初始化其内存子系统，SMP 相关系统等；对于其他的简单系统（试验系统中采用 Vxworks 和 BM），通过设置固定 TLB 映射[5]（目前的实验平台为 MIPS64 架构）保证其不会因地址出错而导致分区间的访问越界，进而破坏整个系统的稳定运行。整体来看，分区内运行系统对内存的使用方式与正常系统没有任何区别，对自己拥有的合法内存的直接使用并不会导致任何额外的性能开销。KMON 系统中实现了 KMON 与分区 OS、分区 OS 与分区 OS 之间的共享内存通信机制，这些功能通过在系统中添加额外的系统调用来实现（实验系统中为 Linux），对于不能添加系统调用的情形（试验系统中为 BM），通过在其调用的库函数中添加类似接口来实现。

对于外设，试验系统中以字符终端和以太网卡为例，字符终端代表低速外设，以太网卡代表高速复杂外设。对于每个 OS，使用虚拟的字符终端和分配给其的小块儿专用内存缓冲区与 KMON 交互，KMON 负责提供统一的人机交互界面。通过移植伪 TTY 到 KMON，使其具有了简单的文本交互接口。对系统性能有重大影响的关键外设，通常被划分为 OS 私有的。这里以以太网卡为例说明。KMON 负责以太网卡的初始化，包括记录其加电初始化后所使用的中断号，I/O 地址空间，内存区域等。这里的初始化并不是为了使用以太网卡做准备，而是为了划分做准备。比如以太网卡 A 被划分给 OS 1，那么需要修改 KMON 维护的设备资源表使其不被再次分配给第三方使用，需要修改外部中断路由（试验平台上是修改 CIU，X86 平台上可以通过修改 APIC），使设备外部中断可以正确路由到正在使用该设备的 OS 所占用的物理 CPU 核上。而分区上的 OS 根据系统启动时传入参数决定是否加载相应的设备驱动，即是否使用该这设备。分区 OS 独占该设备，使用原生设备驱动驱动外设，保证了正确性和高效性。很多情况下，分区间 OS 可以通过共享内存机制极大的提高对外设的利用率。

单个 OS 通过分区启动时传入的固定格式参数表初始化其所分配到的物理资源，通常包括所分配到的物理核心（CPU Core）掩码、物理内存区域、外设资源（如图 2），形成逻辑上独立运行的单个系统，而系统本身并不会感知自己与普通情况下的运行有何差别。在图 2 中，KMON 建立整个物理主机中的全局物理资源视图，在划分形成单个分区时将对应资源记录到分区对应的私有资源表中，该表即为单个分区上 OS 可见的物理资源视图，包含了分区上 OS 正确使用这些物理资源的所有必须信息。

2.2 运行时控制系统

传统系统虚拟机（以 Xen 为例）中，通过超级调用和虚拟中断完成 Guest OS 和 VMM 的交互[6]。Guest OS 通过超级调用，陷入核心态从而完成某些需要在特权态下执行的任务，如更新影子页表等；VMM 模仿中断机制，向 Guest OS 注入中断从而通知 Guest OS 完成某个任务。在试验系统中，KMON 通过协议机制

完成与 OS 的交互。协议机制以核间中断和共享内存机制为基础，完成格式化的控制信息在 KMON 和分区 OS 间的传递。例如，系统 A 与系统 B 通信，需要首先通过超级系统调用向 KMON 申请注册消息句柄（描述用于数据传输的所有必须属性），完成数据填充后需要再通过超级系统调用向对应的物理核发起核间中断。如果系统 A 和 B 在加载时并没有通过配置文件添加通信属性，则 KMON 不允许两者通信，因为 KMON 在 A 和 B 启动时并没有在其启动信息段中填充双方的信息。KMON 与各个分区 OS 的交互仅是分区 OS 间信息交换的一个特例，每个分区 OS 启动时必须注册与 KMON 间用于通信的共享内存和完成必要的初始化。

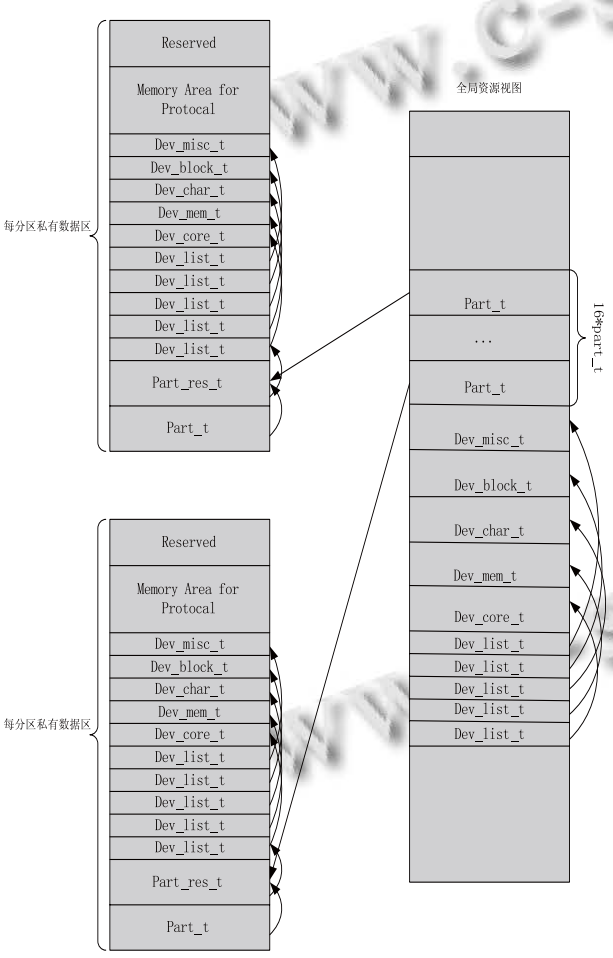

图 2 资源划分示意图

从效率上来看，一次最简单的控制信息传递需要经过写寄存器（消息发送方）-读内存（消息接收方）中断处理（接收方）-写内存（接收方）几步；这样控制功能可由分区 OS 异步完成。传统虚拟机中的超级调用通常设计为同步完成，并且涉及复杂的内核态-内核态（次特权）-用户态之间的状态转换，除了系统开销和设计复杂性这两点外，这些对 Guest OS 的修改在某些实时系统中是难以接受的，因为这样很可能影响原有系统性能和稳定性。协议机制的优点是灵活，每一条协议都可以独立设计。例如可以设计需要满足同步要求的某条协议，设计可以用以完成延时任务的某条协议等。实验系统中仅完成了用于虚拟终端切换和单系统重启两个简单控制协议。

2.3 分层的运行结构

传统系统虚拟机保证 Guest OS 间的严格隔离性，提供系统资源的动态调度提高系统资源利用率，但是却很难对实时应用提供良好支持。KMON 采用了如下图（图 3）所示的分层运行结构以达到隔离性和单系统性能的折中。

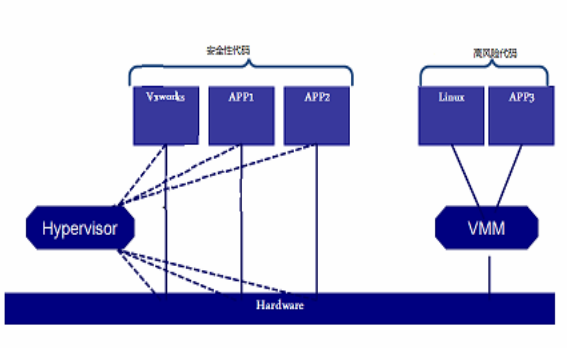

图 3 系统架构示意图

在此设计方案中，一些高吞吐量和时间至关重要应用被划分到独立的分区中，它们直接使用私有的物理设备以满足性能需求。这些系统通常被认为是简单 OS（比如 Vxworks、BM，图 3 中左侧）通过简单设置静态的地址映射[7]（实验系统中通过设置静态 TLB 固定映射）能够保证在单一系统出错或故障的情况（例如内存越界，CPU core 挂死）下不会影响到整个系统的稳定运行。这些系统运行在 CPU 的核心态。对于另外一些需要执行复杂业务应用的 OS（比如 Linux，图 3 中右侧部分，很多控制面开发需要使用其丰富的系统功能），被设计为运行在次特权级。这些系统代码更庞杂，提供丰富的系统调用，从而也更容易出错和遭到恶意入侵，从整个系统的稳定性和安全性考虑，认为这部分属于高风险代码。它们被设计运行在次特权

态，以受限的方式使用分区的物理资源，以满足整个系统的安全性需求。KMON 本身运行在 CPU 的核心态，负责全局物理资源的划分。

3 测试与验证

3.1 实验环境

硬件：Cavium Network 公司的开发板，搭配 12 颗 MIPS 核心 800Mhz，4GB 物理内存，2 串口，1GB CF 卡，1 百兆以太网卡，4 万兆以太网卡。PC 机 2 台，各搭配百兆以太网卡一块。软件：KMON，Uboot(MIPS),Linux/MIPS(2.6.27), Vxworks5(MIPS)。

3.2 测试方案

首先，通过 Uboot（MIPS）在开发板上启动单个 Linux 内核（2.6.27）和 Vxworks5。Linux 内核性能数据由 LmBench 性能测试套件[8]测定。Linux 内核编译为仅使用 256MB 物理内存，使用单个物理核心，1GB CF 卡作为文件系统载体，Vxworks5 被编译为使用单个物理核心，百兆以太网卡。

然后，通过 KMON 在同一开发板上同时启动和运行 Linux 和 Vxworks。混合启动配置为 4 个 Linux，1 个 Vxworks，其中某个 Linux 内核各使用 1 个物理核心，256MB 物理内存，虚拟控制台一个，1GB CF 卡作为根文件系统载体，其余 Linux 内核各使用 1 个物理核心，256MB 物理内存，虚拟控制台一个，使用内存挂在根文件系统，Vxworks5 配置同上。

3.3 测试结果

如表 1 所示，L-Up 标示出数据为仅启动单个 Linux 时测得的数据，KMON 标示出数据为同时启动运行 4 个 Linux 和一个 Vxworks 时测得的第一个 Linux 内核性能数据（因为实验测试的设计用例中仅第一个 Linux 内核与 L-Up 中 Linux 内核运行配置的物理资源相同）。

表 1　LmBench 测试数据

Config	Simple syscall	Simple read	Simple write	Simple stat	Simple open/Close	Signal instal	Signal handler	fork+ Exit	fork+ execve	fork+ /bin/sh
L-Up	0.2818 0.2817	0.6976 0.6969	0.6225 0.6225	4.5664 4.5674	6.7365 6.7414	0.7118 0.7118	8.0699 8.0683	292.3158 292.1579	1227.6000 1228.4000	5868.0000 5853.0000
KMON	0.2818 0.2817	0.6968 0.6968	0.6225 0.6218	4.3803 4.3726	6.8509 6.8298	0.7119 0.7119	8.1391 8.1312	299.9444 300.8889	1252.2000 1255.8000	5968.0000 5966.0000
Config	Prot fault	Pipe latency	Page faults	UDP latency	TCP latency	TCP/IP connection	Pipe bandwidth	AF_UNIX sock stream bandwidth		
L-Up	1.0759 1.0990	14.5024 15.4499	3.1092 2.9147	42.0668 42.3117	60.1380 86.4710	397.1538 169.4668	660.53(MB/sec) 666.29(MB/sec)	823.62(MB/sec) 822.75(MB/sec)		
KMON	1.1098 1.0759	14.7708 14.5024	3.1089 3.1092	42.5269 42.9787	65.6487 70.7905	171.5806 172.3226	667.65(MB/sec) 654.38(MB/sec)	829.47(MB/sec) 829.55(MB/sec)		

总共测的 4 组数据，L-Up 和 KMON 条件下各两组，测试数据主要包括系统调用、网络、管道三个方面，综合反应了系统的 CPU、内存、网络、I/O 性能。表中没有表明单位的数据单位均为 microseconds。由于 KMON 系统仅进行资源的划分而不影响 OS 对资源的使用，因此并不会对系统性能产生影响，测试结果验证了这个观点，L-Up 和 KMON 中的测试数据基本持平，并没有出现某方面的显著性能差异,充分体现了 KMON 系统高性能的优点。

4 总结

本文提出一种面向多核网络处理器平台的轻量级的软件划分方案。各种应用需求，根据特性和运行环境的不同被部署到不同的 OS 中，各个 OS 之间通过 OS 之间的共享内存交换和共享数据。不同的 OS 根据应用需求

(下转第 50 页)

⑥ 特殊打印功能。形式上具有续打及选页打印；内容上具有原件打印和清洁打印，清洁打印即屏蔽修改后记录，使文件版面整洁清晰。

⑦ 扩展功能。按照国际通用标准进行模块化扩展，如与 PACS 接口(DICOM3.0)，远程会诊和教学接口(TCP/IP)，兼容 HL7 接口，语音识别输入接口，以及与医疗保险和个人信息服务等系统的接口。

4 结语

本文基于 Web 技术，提出了基于 XML 技术的 EMR 系统，并给出了 EMR 架构设计方案和系统功能特性，实现了 EMR 与医院其他医疗系统的数据集成。相对纸质病历而言，EMR 具有较强的实时性，在一定程度上减轻了医护人员的工作负担，提升了工作效率，同时，对于国家正在推行的“医疗实名制”，EMR 的实施也起到了重要的驱动力。

参考文献

1 薛万国.我国电子病历研究进展.中国医院管理,2005,25(2):17–19.

2 王兴林,姚军.电子病历潜在的法律责任分析及研究.中华医院管理, 2007,23(2):140–142

3 王炳胜,王景明,李永申,等.电子病历存储方法探讨.中国医院管理, 2007,27(12):86–89.

4 Health Level Seven (HL7) Version 3.0,Michigan:Health Level Seven Inc, 2003.

5 Digital Image and Communication in Medicine (DICOM) Version 3.0,Germany: ACR/NEMA, 2007.

6 Integrating the Healthcare Enterprise:IHE Technical Framework, Revision 4.0,HIMSS/RSNA, 2007.

7 李昊旻,薛万国,段会龙,等.电子病历与标准化和结构化.中国数字医学, 2009,3(10):9–12.

8 郑重,薛万国.XML 签名在电子病历系统安全中的应用.计算机系统应用, 2005,14(2):15–17.

9 吴伶俐,刘洪星.基于 XML 的结构化电子病历系统设计,计算机工程与设计, 2007,28(1):473–476.

10 胡业发,陈娟,陶飞,等.基于 XML 的电子病历数据模式研究,计算机工程与设计, 2007,28(4):914–916.

11 唐维新.病历书写规范.东南大学出版社, 2003.132–134.

(上接第 10 页)

的不同，通过灵活的配置获取不同的物理资源，使各种系统同时运行在同一物理主机中。通过分层设计的划分方案，在满足某些高吞吐量和时间至关重要应用需求的前提下，尽量保证了整个系统的安全性和可靠性。

目前的原型系统中，Linux 还没有成功运行在次特权态，完善分层的运行结构将是下一步将要完成的工作重点。

参考文献

1 Kyueun Yi, Jean–Luc Gaudiot. Features of Future Network Processor Architectures.jva. IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing (JVA’06), 2006.69–76.

2 Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield. Xen and the art of virtualization. Proc. of the Nineteenth ACM symposium on Operating Systems Principles (SOSP19). 164–177.

3 Bugnion E, Devine S, Rosenblum M. Disco: running commodity operating systems on scalable multiprocessors. Proc. of the Sixteenth ACM Symposium on Operating System Principles. October 1997.

4 Satish Kharat, Rajeev Mishra, Ranadip Das, Srikanth Vishwanathan. Migration of software partition in UNIX system. Proc. of the 1st Bangalore Annual Compute Conference(COMPUTE’08).2008.

5 Intel Corporation,英特尔开源软件技术中心,复旦大学并行处理器研究所.系统虚拟化-原理与实现.北京:清华大学出版社, 2009.

6 石磊,邹德清,金海.Xen 虚拟化技术.武汉:华中科技大学出版社, 2009.

7 (英)斯威特曼著.赵俊良,等译.MIPS 处理器设计透视.北京:北京航空航天大学出版社.

8 LMbench–Tools for Performance Analysis. http://www.bitmover.com/lmbench/.