MASTER THESIS XIDIAN UNIVERSITY

西安电子科技大学

硕士学位论文

基于IEEE1588的虚拟集群任务同步测量技术研究

作者姓名 罗祥帆

学校导师姓名、职称 姚明昕 副教授

校外导师姓名、职称 高毅 高工

申请学位类别 工程硕士

西安电子科技大学
学位论文独创性（或创新性）声明

秉承学校严谨的学风和优良的科学道德，本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同事对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文若有不实之处，本人承担一切法律责任。

本人签名：罗祥帆　　日　　期：2017.6.8

西安电子科技大学
关于论文使用授权的说明

本人完全了解西安电子科技大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权属于西安电子科技大学。学校有权保留送交论文的复印件，允许查阅、借阅论文；学校可以公布论文的全部或部分内容，允许采用影印、缩印或其它复制手段保存论文。同时本人保证，结合学位论文研究成果完成的论文、发明专利等成果，署名单位为西安电子科技大学。

保密的学位论文在___年解密后适用本授权书。

本人签名：罗祥帆　　导师签名：

日　　期：2017.6.8　　日　　期：2017.6.9

学校代码 10701　　学　号 1401120572
分 类 号 TN91　　密　级 公开

西安电子科技大学

硕士学位论文

基于IEEE1588的虚拟集群任务同步测量技术研究

作者姓名：罗祥帆

领　　域：电子与通信工程

学位类别：工程硕士

校内导师姓名、职称：姚明昨 副教授

企业导师姓名、职称： 高毅　 高工

学　　院：通信工程学院

提交日期：2017 年 6 月

Study on the IEEE1588 Based Multi-task Time Synchronization Measurement in Virtual Clusters

A thesis submitted to

XIDIAN UNIVERSITY

in partial fulfillment of the requirements

for the degree of Master

in Electronics and Communications Engineering

By

Luo Xiangfan

Supervisor: Yao Mingwu Title: Associate Professor

Supervisor: Gao Yi Title: Senior Engineer

June 2017

摘要

近年来，虚拟化技术得到了快速的发展，在此基础上，多台物理机上的多台虚拟机组成的虚拟集群网络得到了广泛应用。为保证虚拟集群上实时性任务的服务质量，各虚实计算节点间需要通过 IEEE1588 协议实现时间同步。IEEE1588 协议也称精确时间协议，其通过主从时钟设备的报文交互实现时间同步。然而在虚拟环境下，由于现有虚拟化平台的特性使 IEEE1588 协议的同步效果难以保证。并且，现有技术难以有效实现虚拟集群中多虚拟机间任务的时间同步测量。

为实现虚拟集群中任务的高精度同步测量，本文提出了基于超调用的虚拟集群硬件 PTP 板卡辅助同步测量方案，该方案通过硬件时间服务器为通过 PCIe 接口安装在物理机上的 PTP 板卡授时，其硬件时间同步精度可达纳秒级，可作为虚拟机时间同步测量的标尺。在 KVM 虚拟化平台上通过超调用方法实现一台物理机上多台虚拟机共享硬件 PTP 板卡时间，解决了多台虚拟机无法共享 PCIe 板卡的问题。实验证明这一方法的误差范围在 20μs 以内。在多台虚拟机中基于这一方法，设计实现了虚拟机系统时间同步误差的测量方案。该方案既可实现长时间的持续测量，也可通过设计的可视化软件实现测量结果实时、直观的显示，该方案不仅精度较高，而且实现简单，占用资源少，可满足大多数虚拟集群中各种时间同步系统的的同步测量。

本文使用所提出的同步测量方案分别对软硬件的虚拟机同步系统的同步精度进行了测量，其中软件同步系统为广泛应用的 PTPd 软件，硬件同步系统使用超调用时间读取方法实现同步。为研究虚拟化环境对软硬件同步方法效果的影响，分别对物理机和虚拟机上的软硬件同步系统的同步精度进行了测量，为研究计算机负载对虚拟机和物理机同步系统的影响，在每组实验中都加入了 50%CPU 负载的对照实验。最终，经过多组实验大量测试，验证硬件同步系统的同步精度可达 20μs 以内，且受负载和虚拟化环境影响较小，而软件同步方式同步精度较低，且受虚拟化环境影响影响，尤其是在 CPU 负载增大时其同步误差明显增大，通过分析发现其原因是高负载情况下上层虚拟操作系统无法及时处理同步报文，进而影响时间戳精度。

关 键 词：虚拟化技术， KVM， 时间同步， 同步测量， PTPd

ABSTRACT

In recent years, virtualization technology has been rapidly developed, on this basis, virtual cluster, which is composed of multiple virtual machines on multiple physical machines, has been widely used. In order to ensure the quality of service for real-time tasks on virtual clusters, the nodes in a virtual cluster need to achieves time synchronization through various hardware and software applications based on IEEE1588 protocol. IEEE1588 protocol, also called precision time protocol(PTP), realizes time synchronization by message exchange between a master and a slave clock. In virtual environment, however, because of some intrinsic characteristics of current virtual platforms, it's hard to ensure the performance of IEEE1588 protocol. Besides, it is difficult to measure the offset between tasks over multiple physical machines using present technology for the similar reasons.

To achieve high precision synchronization measurement for the tasks in virtual clusters, we propose a synchronization measurement scheme for the tasks in virtual clusters based on hardware PTP equipment. The scheme use hardware time server to synchronize hardware PTP equipment installed on physical machine via PCIe interface. The synchronize offset between different hardware PTP equipment is verified in nanosecond level, sufficient for synchronization measurement as a reference. We use hypercall to make multiple virtual machines on the same physical machine to share the hardware PTP equipment, and prove the delay of hypercall within 20μs by experiment. We propose and design a scheme which can either achieve long term measurement or real-time display to meet the needs of different scenarios. The proposed scheme not only has high precision, but also is simple to implement and take up less resources, it can be applied in most virtual clusters to achieve synchronization measurement.

With the proposed scheme, we measure the offset of hardware and software synchronization applications based on IEEE1588 protocol. The software synchronization application uses PTPd, a widely used software daemon implementing IEEE1588 protocol, and the hardware synchronization application read the time of hardware PTP equipment by hypercall. In order to research the influence of virtualization on synchronization, we implemented different experiment in physical and virtual machine. In each experiment, the 50%CPU load was added to the control experiment. We found that the hardware

synchronization application based on IEEE1588 protocol has better accuracy and is less impacted by virtualization and CPU load, with an offset less than 20μs. On the other hand, the software synchronization application PTPd achieves lower accuracy and is heavily impacted by virtualization. Especially, in the condition with higher CPU load, the offset significantly increased. By analyzing the experimental result we found that in such conditions, the upper layer software in virtual machine operating systems cannot handle the synchronization message timely, resulting in inaccurate timestamp, degrading the software synchronization performance.

Keywords: virtualization technology, KVM, time synchronization, synchronization measurement, PTPd

插图索引

表格索引

符号对照表

符号	符号名称
ms	毫秒
μs	微秒
ns	纳秒
HZ	赫兹
GHZ	吉赫兹

缩略语对照表

缩略语	英文全称	中文对照
ACPI	Advanced Configuration and Power Management Interface	高级配置和电源管理接口
APIC	Advanced Programmable Interrupt Controller	高级可编程中断控制器
BC	Boundary Clock	边界时钟
BNC	Bayonet Nut Connector	卡扣配合型连接器
DMA	Direct Memory Access	直接内存存取
EPT	Extend Page Table	增强页表
E2E	End to End	端到端
HPET	High Precision Event Timer	高精度时间时钟
KVM	Kernel based Virtual Machine	基于内核虚拟机
NTP	Network Time Protocol	网络时间协议
OC	Ordinary Clock	普通时钟
PCIe	Peripheral Component Interconnect Express	总线和接口标准
PIT	Programmable Interval Timer	可编程间隔时钟
PTP	Precision Time Protocol	精确时间协议
P2P	Peer to Peer	点到点
PPS	Pulses Per Second	秒脉冲
RTC	Real Time Clock	实时时钟
TC	Transparent Clock	透明时钟
TCP	Transmission Control Protocol	传输控制协议
TSC	Time Stamp Counter	时间戳计数器
VCPU	Virtral CPU	虚拟 CPU
VMCX	Virtual Machine Control Structure	虚拟机控制数据结构
VM	Virtual Machine	虚拟机
VMM	Virtual Machine Monitor	虚拟机监视器
VMX	Virtual Machine eXtensions instructions	虚拟机功能扩展指令
VT-d	INTEL Virtualization Technology for Direct I/O	因特尔直接 I/O 虚拟化技术
VT-x	INTEL Virtualization Technology for x86	因特尔 X86 虚拟化技术

目录

第一章 绪论

1.1 研究背景

近年来，各种计算机硬件性能不断提升，为了更高效的利用物理计算机的各种硬件资源，虚拟化技术得到了快速的发展和广泛的应用。虚拟化技术是指在一台物理计算机上运行多台虚拟机，通过在底层的硬件设备和上层的操作系统之间添加一个虚拟化层，虚拟化层将计算机的硬盘、CPU、I/O 和内存等硬件资源虚拟化，供上层的虚拟机使用，从而使得多台虚拟机能够时分的共享一台物理计算机上的各种资源，同时各个虚拟机之间相互独立，各自运行自己的操作系统。

虚拟化技术具有诸多优点，其中最重要的是虚拟化技术的应用可大大提高物理资源的利用率，而且虚拟化技术的隔离性提升了虚拟机的安全性，此外，虚拟化技术的动态迁移功能可使虚拟机在不关机的情况下从一台物理计算机上迁移到另一台物理计算机上。因为这些优点，在服务器中经常使用虚拟化集群，所谓虚拟化集群是指在通过网络连接的多台物理机上分别运行多台虚拟机所形成的集群，服务器虚拟集群的应用可实现服务器物理资源的灵活部署，提高资源利用率，提升安全性，降低维护成本，并通过虚拟机动态迁移实现不关机的服务器迁移。如今，虚拟集群在各种服务器上得到了广泛的应用。

在另一方面，随着网络的发展，分布式控制系统在很多领域中得到了广泛应用。分布式控制系统中各计算节点协同工作提供服务，这就要求各计算节点需保持高精度的时间同步，否则系统就难以进行协调、管理和控制。如今越来越多的分布式控制系统建立在虚拟集群之上，相比物理机组成的物理机群，建立在虚拟化技术基础上的虚拟集群存在时钟漂移、时钟同步精度低和同步稳定性差等问题，这就使得在虚拟集群上要实现高精度的时间同步更加困难。

目前，各种网络中为实现各节点的时间同步主要采用基于 IEEE1588 协议[1]的各种软硬件同步应用方案，IEEE1588 协议也叫精确时间协议（PTP，Precision Time Protocol），它通过网络中主从时钟的同步报文交换，计算主从时钟的时间同步误差，并将同步误差消除，理论上，通过 IEEE1588 协议可实现网络中各个设备时间的高精度同步，但实际应用时，基于 IEEE1588 协议的不同方案所能达到的同步精度有所不同，尤其是在虚拟集群中，受虚拟化环境的影响，使用相同或类似的同步方案时，虚拟机间的同步精度往往比物理机更差[2]。因此，在实际应用时，为保证虚拟集群的时间同步要求，应对虚拟机上的各种时间同步方案进行测量。

基于 1588 协议的硬件同步系统可实现硬件设备间纳秒级的高精度的时间同步，

在物理集群中，物理计算机操作系统可通过安装基于 1588 协议的硬件同步设备获得精确地时间源，从而实现物理机上任务的同步测量。但在虚拟集群中，由于虚拟化平台的限制，一台物理机上的多台虚拟机无法有效地共享硬件 PTP 设备的时间，这就给虚拟集群中的虚拟机时间同步测量带来了困难，而且，硬件同步设备可通过输出秒脉冲（PPS，Pulses Per Second）实现同步误差的精确直观测量，但对于上层虚拟机操作系统上运行任务的同步误差测量精度无法保证，且测量结果无法直观显示。因此，需要一种可行的虚拟集群同步测量技术方案来实现虚拟集群任务的同步测量，并且保证测量的精确性、稳定性和直观性。

1.2 选题意义

虚拟集群任务同步测量技术有着重要的意义，其可应用于各种有时间同步要求的虚拟集群中，例如，舰载计算机中心机房虚拟集群就是虚拟集群任务同步测量技术的一种具体的应用场景。

海湾战争后，信息化战争已成为现代战争的必然趋势，对于现代化海军战舰，舰船综合平台管理系统是实现舰船信息化的关键[3]，而综合平台管理系统的核心是舰载计算机中心机房，舰载计算机中心机房由多台计算机组成，多台计算机间通过多级交换网络连接，为综合平台管理系统提供计算资源。由于虚拟集群在物理资源共享等方面上的优势，为进一步提升综合平台管理系统的性能，可在舰载计算机中心机房计算机上部署虚拟集群。

由于舰船综合平台管理控制着舰船的动力系统、驾驶系统、电力系统和火控系统，因此需要保证计算机中心机房网络上各节点间的时间同步。当在计算机中心机房网络上部署虚拟集群时，其中各虚拟机上运行着不同的任务，为保证舰上各系统在航行和海战中的正常运行，虚拟集群中的各虚拟机操作系统必须达到其上运行任务的同步精度要求，因此在设计舰载计算机中心虚拟集群的同步方案时，需要一种虚拟集群任务的同步测量技术来测量各种软硬件同步方案在虚拟机上能够达到的同步精度。

本文针对如舰载计算机中心机房虚拟集群等各种虚拟集群中的同步测量问题，分析了基于 IEEE1588 协议的同步系统在虚拟化环境中应用的各种实际问题，提出了的基于 IEEE1588 协议的虚拟集群任务同步测量方案，实现了虚拟集群任务的高精度同步测量，并且通过理论分析和实验数据验证了软硬件同步方案的性能，从而满足了虚拟集群中虚拟机上任务的时间同步测量需求。

1.3 国内外研究现状

时间同步一直是计算机网络中面临的问题之一，早期的网络时间协议（NTP，

Network Time Protocol）可达到毫秒级的同步[4],2002 年精确时间协议 PTP 诞生[5], PTP 也称 IEEE1588 协议，相比于 NTP，PTP 使用软硬件结合的同步方式，其同步精度明显提升，2008 年推出的升级版 IEEE1588 V2 改进了同步协议[1]，进一步提高了同步精度。在此基础上，基于 PTP 的多种软硬件应用被用于各个领域，

PTP 的应用包括了基于软件和硬件的 PTP 同步系统，他们之间的区别在于基于硬件的 PTP 同步系统的同步协议运行在底层硬件上，而基于软件的 PTP 同步系统的同步协议运行在上层软件上,因此带来了软硬件 PTP 同步系统同步效果的巨大差异，其中，基于硬件的 PTP 同步系统的同步精度较高，一般可达纳秒级，随着硬件设备性能的提升，基于硬件的 PTP 同步系统精度不断提升，一些高精度的 PTP 应用已经可以实现精度在亚纳秒级的时间同步[6]。而基于软件的 PTP 同步系统的同步精度较低，例如，PTPd 是一种被广泛使用的纯软件 PTP 实现方式，其同步精度受限于时间戳的精度[7]，因此其同步精度一般只能达到微秒级[8]。

计算机中运行软件 PTPd 的同步精度主要取决于时间戳的精度,而在虚拟机中，由于虚拟机的非实时性[9]和网络时延[10]所引起的时间戳错误会造成 PTPd 性能的降低，而且现有的虚拟化平台自身的时间维持也存在问题[11]，为解决这些问题，文献[12][13]通过调度方法对 KVM（Kernel based Virtual Machine）进行了实时性优化，文献[14]对 Xen 进行了实时性优化,文献[15]通过半虚拟化方式调度 VCPU 增强了 KVM 实时性，文献[16]对云环境下中虚拟机网络时延进行了研究。文献[17]通过实验测量了虚拟化环境对 NTP 的影响。

1.4 本文的主要工作和结构安排

根据本章前两节介绍的背景技术和选题意义，本文的研究方向是虚拟集群任务的同步测量技术，研究目标是提出一种可行的虚拟集群任务同步测量方案，并通过该方案对虚拟机软硬件同步系统的同步效果进行测量。为实现这个目标，本文深入研究了虚拟化技术和 IEEE1588 同步协议的原理，总结了虚拟化环境中给 IEEE1588 协议带来的各种问题，安装并运行了现有的 PTPd 同步软件和硬件 PTP 板卡。在此基础上，本文结合了 IEEE1588 同步技术和虚拟化技术，针对虚拟集群中同步测量技术的各种需求，完成了以下的创新性工作：

（1）研究并实现了基于超调用技术的虚拟机硬件 PTP 板卡时间读取方法，实现了一台物理机上多台虚拟机同时读取一块硬件 PTP 板卡时间，以此作为时间基准，从而使得虚拟化平台上多虚拟机时间同步测量成为了可能。

（2）设计并实现了基于 IEEE1588 协议的虚拟集群任务同步测量方案，并通过实验验证了该测量方案的测量精度和稳定性，同时，设计了时间同步测量软件，通过

可视化界面便于用户观察、测量和记录结果。

（3）通过基于 IEEE1588 协议的虚拟集群任务同步测量方案测量了虚拟集群中多虚拟机上应用基于 IEEE1588 协议的软硬件同步方法的同步性能，获得了大量的实验数据，并从同步误差、稳定性和高负载环境下的影响等几方面分别分析了软硬件同步方法的性能。

本文的组织结构如下：

第一章介绍了本文的背景、意义、研究现状和主要工作。

第二章介绍了虚拟化技术，着重介绍了本文用到的 KVM 平台实现虚拟化的原理和虚拟机的时间维持原理。

第三章深入分析了 IEEE1588 协议的原理，分别介绍了本文实验中用到的基于 IEEE1588 的软硬件同步技术，并对 IEEE1588 协议应用时的几种同步误差产生原因进行了分析。

第四章实现了虚拟机基于超调用的 PTP 板卡时间读取技术，针对定时器误差进行了分析和优化，并通过实验验证了该测量方法的误差范围，设计了本地记录和时间同步测量软件两种测量方式，分别满足了长时间持续测量和实时显示的功能需求，并设计了基于超调用的虚拟集群硬件 PTP 板卡辅助测量方案。

第五章通过第四章介绍的技术方案，分别设计了虚拟机软硬件同步方式的测量方案，并根据方案进行了虚拟机和物理机软硬件同步方法的测量实验，得到了软硬件同步方法在虚拟机和物理机上的同步误差数据并进行了分析，通过添加 CPU 负载研究了高负载对软硬件同步方法同步效果的影响。

第六章对本文的研究成果进行了总结，并对未来的研究方向进行了展望。

第二章 虚拟化技术

2.1 虚拟化技术的概念

虚拟化技术通过虚拟化计算机硬件设备，实现在一台硬件计算机上运行多个操作系统[18]，不同操作系统可通过虚拟化技术共享计算机的 CPU、I/O 和内存等硬件资源。实现这一功能的虚拟化层叫做虚拟机监视器（VMM，Virtual Machine Monitor）。VMM[19]通过对硬盘、内存、CPU 等底层硬件设备模拟，向上虚拟出多组计算单元，因此可在 VMM 之上运行多个虚拟操作系统，每个虚拟操作系统都能共享使用物理计算机上的所有计算单元，VMM 负责对各种物理资源进行调度，分配各多个虚拟操作系统。运行在 VMM 之上的操作系统叫做客户操作系统（guest os），物理计算机叫做宿主机（host machine），在宿主机上直接运行的操作系统叫做宿主操作系统（host os），运行在一个 VMM 上的多个客户操作系统互相隔离，不同客户操作系统间相互独立。

2.1.1 虚拟化技术的发展

虚拟化技术的概念最初于上世纪五十年代末被提出[20]，第一台虚拟机是 IBM 公司于 1965 年设计开发的 System/360 Model 40 VM,最初，它的设计目标是将当时最先进的虚拟内存技术应用于其他计算机子系统，通过时分共享的方式实现计算机资源的分配，从而实现不同用户能够共享昂贵的物理计算机资源。第一个成熟虚拟化设备的是 1980 年的 IBM VM/370，IBM VM/370 运行在 IBM System/370 大型机上[21]，IBM VM/370 的许多标志性的设计理念仍应用与如今的大型机上，但是，由于其应用的背景与如今广泛应用的服务器与微型计算机有所不同，所以和如今的虚拟化技术有所不同。由于计算机硬件设备的发展，大型机逐渐被取代，虚拟化技术一度不再被重视。直到上世纪 90 年代，随着计算机技术飞速发展，个人计算机的性能不断提升，其性能逐渐达到了可支持多个操作系统同时运行的程度，虚拟化技术开始在小型机和微机上广泛应用。

从 90 年代末至今，虚拟化技术在 X86 平台上得到了长足的发展和应用[22]，VMware Work Station[23]、KVM[24]和 Xen[25]等虚拟机软件被广泛使用。虚拟化可分为服务器虚拟化、网络虚拟化和存储虚拟化，近年来，服务器虚拟化技术逐渐成为了广泛关注和研究的热点，根据虚拟化的层次，服务器虚拟化又可分为编程语言级虚拟化、操作系统虚拟化、ISA 层虚拟化和硬件抽象层虚拟化，其中，硬件抽象层虚拟化是当今商用服务器和个人计算机上所采用的最主流的虚拟化技术，也是本文研究中所使用的的虚拟化平台。

2.1.2 VMM 的分类

VMM 根据实现结构分类可分为监控模型(Hypervisor Model)、宿主模型(Host-based Model）和混合模型(Hybrid Model)。

图2.1 监控模型 VMM

如图 2.1，在监控模型中，VMM 作为一个独立的操作系统运行在物理计算机之上，拥有物理机的所有硬件资源，并将内存、CPU 和硬盘等各种资源分配给运行在其上的多个客户操作系统。由于 VMM 虚拟化并分配所有硬件资源，因此监控模式的物理资源虚拟化效率较高，而且由于监控模式的安全性只受 VMM 的安全性影响而不像宿主模式下受宿主操作系统影响，所以安全性较高。但是由于监控模式中 VMM 需要包涵所有硬件设备的 I/O 驱动，这加大了 VMM 软件的开发工作量。

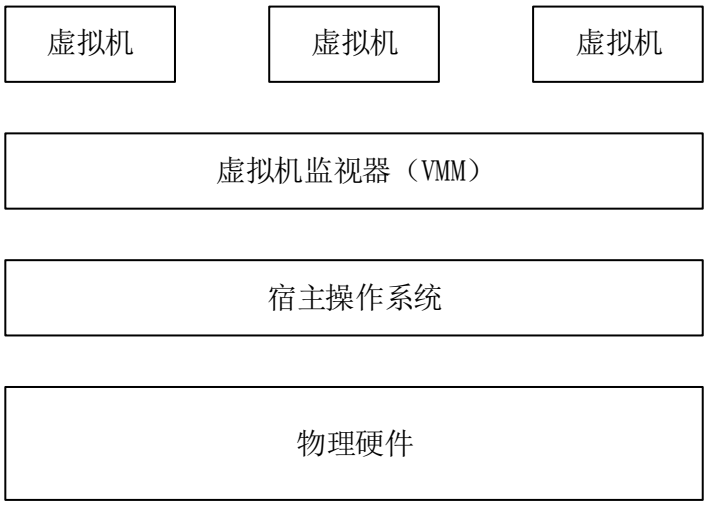

图2.2 宿主模型 VMM

如图 2.2,宿主模式与监控模式的区别在于 VMM 运行在宿主操作系统之上，宿主操作系统可以是 LInux、Windows 等普通操作系统，宿主操作系统控制所有硬件资源，VMM 通过调用宿主操作系统提供的服务为其上运行的客户操作系统提供虚拟化。宿

主模式与监控模式的优缺点相反，其开发工作量较小但效率和安全性低于监控模式。

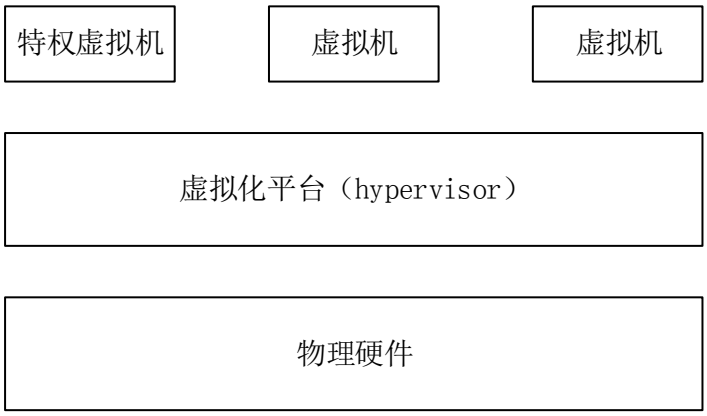

图2.3 混合模型 VMM

如图 2.3,混合模式中 VMM 与监控模式一样位于底层，但与监控模式不同，混合模式的 VMM 上运行一个特权操作系统，设备驱动模块包含在这个特权操作系统中，因此混合模式可兼具高效和开发量小的优点。混合模式的缺点在于特性操作系统带来了一定的上下文切换开销。

VMM 根据虚拟平台类型又可分为全虚拟化和类虚拟化。

全虚拟化中 VMM 为客户操作系统提供的环境和普通操作系统运行环境相同，VMM“欺骗”客户操作系统使其完全不知道运行在虚拟的环境之上，因此，运行的客户操作系统是未作修改的普通操作系统。全虚拟化技术又可包括软件辅助的全虚拟化和硬件辅助的全虚拟化，软件辅助的全虚拟化的实现方法是二进制代码动态翻译技术，当客户操作系统执行一些特权指令时由 VMM 将这些指令翻译成可执行的指令。硬件辅助的全虚拟化对硬件本身进行增加虚拟化改进，使其在客户操作系统运行敏感指令或访问敏感资源时以报告 VMM，INTEL 的 VT-x 就是这一技术的代表，运行在 Linux 系统下的 KVM 就是基于硬件辅助虚拟化的实现方案。下一节将具体介绍硬件辅助虚拟化技术的实现原理。

类虚拟化技术通过修改客户操作系统的源代码，使得客户操作系统在了解虚拟化环境，与 VMM 相配合，实现虚拟化。类虚拟化技术可以降低虚拟化技术多带来的性能开销，提高虚拟机运行的性能，其缺点在于必须对客户操作系统进行修改，这增加了使用的工作量且对于像 Windows 这种源码不开源的操作系统难以实现,由英国剑桥大学开发的虚拟化平台 Xen[25]是类虚拟化技术的代表，一些全虚拟化平台如 KVM 也包含类虚拟化接口。

2.2 基于 KVM 的虚拟化技术

KVM（Kernel-based Virtual Machine）是一款开源的系统虚拟化软件，它最初由 Qumrant 于 2006 年开发，2007 年 2 月发布的 Linux 2.6.20 首次将 KVM 模块集成与内核之中。KVM 是一种基于 X86 的硬件辅助虚拟化技术，由 Intel VT 技术提供的硬件辅助虚拟化支持，其中由 VT-x（INTEL Virtualization Technology for x86）支持 CPU 虚拟化[26]，由 VT-d（INTEL Virtualization Technology for Direct I/O）支持 I/O 虚拟化[27]。从结构上来看，KVM 是一种宿主模式的虚拟化技术，但由于 Linux 加入了越来越多的虚拟化功能，所以也有人认为 KVM 是监控模式。KVM 是全虚拟化解决方案，其上可安装未修改的操作系统，KVM 也可通过加入类虚拟化补丁实现类虚拟化功能。

2.2.1 KVM 的系统结构

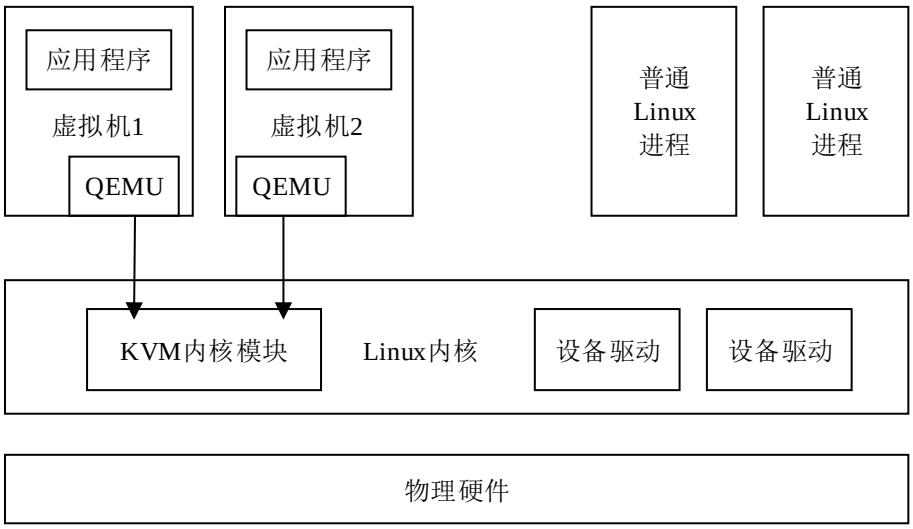

图2.4　KVM 结构

如图 2.4 所示，KVM 可分为 KVM 内核模块和工作在用户空间的 QEMU 模块。

KVM 内核模块集成与 Linux 内核之中，它使得 Linux 系统成为一个 VMM，每一个运行的 KVM 虚拟机都是一个 Linux 进程，与普通的 Linux 进程类似，被 Linux 系统调度，由于 KVM 工作于 LInux 的内核空间，因此 KVM 被认为是一种监控模型的虚拟化技术，而工作在宿主机操作系统用户空间上的虚拟化软件，如 VMware Work Station，属于宿主模型的虚拟化技术。KVM 内核模块基于 Intel VT 技术提供的硬件辅助虚拟化支持，实现 CPU 与内存的虚拟化，但 KVM 内核模块不支持其他设备的虚拟化，因此它无法单独工作，需要与用户空间的 QEMU 模块协同工作实现虚拟化。

QEMU 是一种工作在用户空间的纯软件方式实现的虚拟化方案，它可单独使用实现完整计算机物理环境的虚拟化，也就是说，QEMU 不与 KVM 协同工作，也可以

在其上运行虚拟机，但这种模式下运行的虚拟机由于缺少 KVM 内核模块的支持，其性能非常有限，几乎无法正常使用，因此 QEMU 常与 KVM 内核模块协同使用实现虚拟化。KVM 虚拟机由 QEMU 创建、配置和动态迁移，同时由 QEMU 实现各种设备的 I/O 虚拟化，虚拟机运行时，虚拟机通过 KVM 内核模块提供的系统调用接口从宿主 Linux 操作系统的用户空间进入内核空间，从而实现虚拟化加速。

2.2.2 KVM 的 CPU 虚拟化

KVM 的虚拟化包括 CPU 虚拟化、I/O 虚拟化和内存虚拟化三种主要技术，其中 CPU 虚拟化是最核心的部分，I/O 虚拟化和内存虚拟化都依赖于 CPU 虚拟化来实现敏感指令的处理。CPU 虚拟化是通过 VMM 陷入方式来实现的[28]。

VT-x 定义了两种种虚拟机功能扩展指令(VMX，Virtual Machine eXtensions instructions）操作模式，分别是 VMX 根模式和 VMX 非根模式，VMX 根模式就是 VMM 所处的模式，而 VMX 非根模式就是虚拟机所处的模式。VMM 陷入就是操作模式在根模式和非根模式之间切换的过程。

有三种情况会触发 VMM 陷入：

（1）虚拟机执行敏感指令时会判断是否可以在虚拟环境下执行，如果不行就需要陷入到 VMM 根模式下执行；

（2）外部中断引起虚拟机陷入；

（3）虚拟机主动执行 VMCALL 指令陷入到 VMM 中。

VMM 陷入的过程如图 2.5 所示：

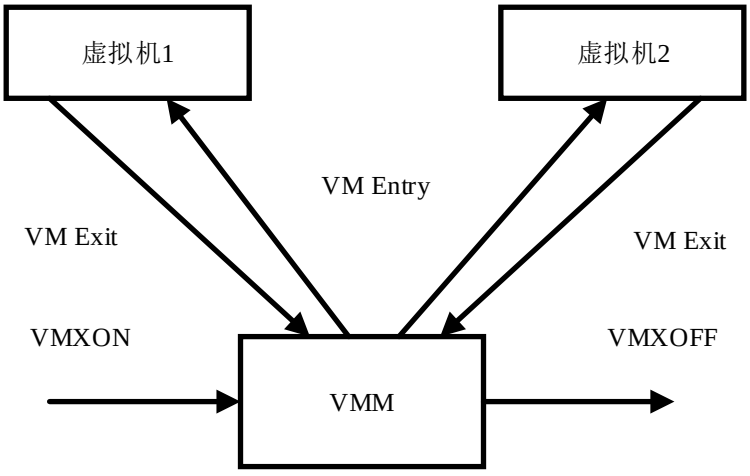

图2.5 VMM 陷入

（1）KVM 执行 VMXON 指令进入 VMX 操作模式；

（2）虚拟机执行 VMLAUNCH 或 VMRESUME 指令运行虚拟机，从而产生 VM Entry，操作模式由根模式转换为非根模式；

（3）虚拟机因上述几种原因产生 VMM 陷入，从而产生 VM Exit，操作模式由非根模式转换为根模式，VMM 根据客户虚拟机的 VM Exit 原因调用相应的处理函数进行处理，处理跳转到过程（2）完成后产生 VM Entry；

（4）当 VMX 决定退出时 KVM 执行 VMOFF 指令，退出 VMX 操作模式。

每个虚拟机拥有一个 VMCS（Virtua Machine Control Structure）数据结构，VMCS 由 VT-x 提供，保存了虚拟机和 VMM 的状态信息。VT-x 可根据虚拟机的 VMCS 数据域执行虚拟机陷入的上下文切换。

硬件辅助虚拟化使用 VCPU（Virtral CPU）来描述虚拟的 CPU，VCPU 描述符是一个结构体，包括了 VCPU 标示信息、虚拟寄存器信息、VCPU 状态信息、额外寄存器/状态信息及其他信息，当运行虚拟机时，首先应运行 VCPU，然后在一个或多个 VCPU 之上运行客户虚拟机。

2.2.3 KVM 的 I/O 虚拟化

I/O 虚拟化也是虚拟化技术的重要组成部分[29]，I/O 虚拟化可使多个客户虚拟机共享计算机的 I/O 设备，I/O 的性能对于虚拟机的性能至关重要。I/O 虚拟化的性能就是虚拟化后 I/O 操作的延迟，越接近普通环境越好，此外，通用性也是重要的标准，通用性就是对客户虚拟机的透明程度。

现有的虚拟化技术主要通过三种方式实现 I/O 的虚拟化，分别是全虚拟化软件模拟、类虚拟化 Virtio 和设备直通。

在 KVM 中全虚拟化软件模拟通过纯 QEMU 软件方式实现模拟 I/O 指令的方式，其实现过程为当虚拟机操作系统执行 I/O 指令时触发 VM Exit 陷入到 VMM 也就是 KVM 内核中，VMM 通过客户机的 VMCS 数据结构获取相关信息，内核直接调用物理 I/O 设备，得到结果后返回虚拟机。全虚拟化软件模拟的优点在于在虚拟机客户操作系统中可通过普通操作系统的各种 I/O 驱动程序实现 I/O 操作，从而保证了全虚拟化的无需修改操作系统的特性。但其缺点在于需要频繁造成陷入。从而对 VMM 产生了大量开销，大大影响了 I/O 的性能。

为解决软件模拟方式性能较差的问题，产生了 Virtio 技术[30]，Virtio 技术是一种类虚拟化技术，客户机知道自己运行在虚拟化平台之上，并在其操作系统上安装修改后的 I/O 驱动程序，在 Virtio 中客户机上的驱动程序叫做前端驱动，在 VMM 中同样安装一个驱动程序，叫做后端驱动，当虚拟机使用 I/O 设备时，前端驱动和后端驱动相互配合，完成 I/O 的虚拟化。Virtio 方式的性能非常出色，几乎达到了运行在物理机上的普通操作系统的 I/O 性能，但它需要在客户机操作系统和 VMM 中分别加入前

/后端驱动，加大了开发难度。

设备直通是一种基于 INTEL VT-d 的技术，VT-d 通过 DMA（ Direct Memory Access，直接内存存取）重映射技术实现了 I/O 设备的透传，在这种模式下，可使运行在 VMM 上众多虚拟机中的一个独占某个物理 I/O 设备，这时，该 I/O 设备对其他虚拟机不可见，独占 I/O 设备的虚拟机在使用该 I/O 设备时可直接通过 VMM，访问 I/O 设备。这种方式下独占 I/O 设备的虚拟机可获得运行在物理机上的普通操作系统的 I/O 性能，但其他虚拟机无法共享该 I/O 设备。KVM 中的 PCI 设备透传（PassThrough）就是通过这种方式实现的。

2.2.4 KVM 的内存虚拟化

除了 I/O 虚拟化和 CPU 虚拟化外，内存的虚拟化也是虚拟化技术的重要组成部分之一，VMM 通过内存虚拟化技术将物理内存地址分配给其上运行的虚拟机使用，KVM 内存虚拟化的实现方式包括了软件方式和硬件方式。

常用的软件内存虚拟化方式是影子页表（Shadow Page Table），影子列表通过多次地址映射实现了虚拟机内存空间中虚拟地址到物理内存地址的转换，影子列表技术可实现虚拟化平台上物理内存地址的动态分配，但这种纯软件方式开销较大，会降低虚拟化平台的性能。

为了提升内存虚拟化的性能，Intel VT-x 中加入了 EPT（Extend Page Table，增强页表）技术，当 KVM 支持 EPT 时，VMM 维护并更新一个 EPT 页表，通过硬件方式根据 EPT 页表进行虚拟内存地址和物理内存地址之间的映射，因其硬件转换方式降低了开销，从而提升了内存虚拟化的性能。

由于本文研究的虚拟集群任务同步测量技术对于内存虚拟化技术的使用较少，因此对于 KVM 的内存虚拟化技术不作详细介绍。

2.3 虚拟机的系统时钟组成及维持原理

2.3.1 计算机系统时钟的组成和维持原理

在 X86 架构的计算机中有多种时钟源，目前使用最多的是以下三种：

（1）PIT（Programmable Interval Timer，可编程间隔时钟）

PIT 是最早的计算机计时设备之一，其计时原理是以固定的间隔产生时钟中断，并通过计数器计算一段时间内产生的中断数量来实现计时，在 Linux 系统中的定时器采用 PIT 触发，但是，PIT 产生中断的频率是 1000HZ，也就是说两次时钟中断的间隔为 1 毫秒，触发时延较大，这就意味着 PIT 的定时器触发精度并不高，PIT 已逐渐被精度更高的 HPET 所取代。

（2）RTC(Real Time Clock，实时时钟）

RTC 是一种集成与 CMOS 之中的时钟，它由 CMOS 电池供电，因此可以在计算机关机时继续工作。RTC 的频率范围是 2～8192Hz，由于其可在计算机关机时继续工作，因此被作为计算机关机后维持时间的时钟。当计算机开机时，操作系统首先通过 BIOS 从 CMOS 中读取 RTC 时间，然后通过 PIT 等其他的计时设备维持系统时间。

（3）TSC(Time Stamp Counter，时间戳计数器）

TSC 与其它时钟设备不同，它本身并没有产生固定周期中断的晶振，而是与 CPU 主频相关，TSC 本质上是 CPU 上的一个 64 位的寄存器，自开机后 CPU 每个工作周期其值加 1，因此它的频率与精度取决于 CPU 的主频。例如，一台计算机的主频为 1GHZ，那么这个 CPU 上的 TSC 每 1ns 加 1，现在，计算机的主频一般在 2GHZ 以上，这就意味着 TSC 的计时精度很高，可以精确到 ns。在操作系统系统中，可通过 RDTSC 指令读取 TSC 寄存器的值来获取精确的系统时间。

此外，随着计算机时钟技术的发展，又出现了 APIC（Advanced Programmable Interrupt Controller，高级可编程中断控制器）、HPET（High Precision Event Timer，高精度时间时钟）、ACPI（Advanced Configuration and Power Management Interface，高级配置和电源管理接口）等更高精度的时钟设备，这些高精度的时钟设备将逐渐取代原有的低精度时钟设备，由于频率过高的高精度时钟中断会影响计算机的性能，因此常使用高精度时钟设备来模拟丁精度时钟设备。

在 Linux 系统中，可通过 clock_gettime 函数读取时间，并将时间通过 timespec 结构体返回，其中 timespec 的结构如下：

```
struct timespec{
__time_t tv_sec;   /*seconds  秒*/
long int tv_nsec;   /*nanoseconds  纳秒*/
}
```

timespec 结构体包括了两个成员，分别代表当前时间的秒位和 ns 位，当前时间是指 1970 年 1 月 1 日到现在的时间，由于 clock_gettime 函数是通过 TSC 寄存器获取当前时间，因此可获得精确到 ns 的高精度时间。

2.3.2 虚拟机系统时钟的维持原理

如上一节所述，计算机系统时钟的维持方式是在开机时操作系统读取 RTC 时钟获取开机时间，操作系统运行时通过计算 PIT 或 HPET 等时钟中断设备产生的中断计算开机后的运行时间，将开机时间加上开机后的运行时间就得到了当前时间。因此在虚拟操作系统中，为实现虚拟机的系统时间维持，VMM 就需要通过虚拟化 PIT、RTC、TSC 和 HPET 等时钟设备。

VMM 通过软件定时器模拟 PIT，为上层虚拟机提供一个虚拟的 PIT，虚拟机可通过 VMM 提供的接口设置 PIT 的中断频率，VMM 以设置的频率向虚拟机注入虚拟中断，虚拟操作系统可像物理操作系统一样靠将一段时间内的 PIT 中断数量乘以中断间隔来计算时间。其他时钟设备的虚拟化方式与 PIT 相似。

在虚拟化环境中，多个虚拟机运行在一个 VMM 之上，虚拟机在运行时会频繁的执行陷入，当一台虚拟机陷入时，VMM 或在同一物理核上运行的其他虚拟机获得控制权，这台虚拟机被挂起，此时，这台虚拟机无法接收时钟中断，无法更新时间，为维持虚拟机的时间维持，在虚拟机被挂起的时间里，VMM 会储存本应送给虚拟机的时钟中断，在虚拟机被调度回来的时候再一次性把这一段时间的时钟中断都送进来，这就保证了虚拟机能够继续维持时间。

虚拟机的时间维持与物理机最大的不同之处就在于时钟中断的虚拟化，但有时，VMM 可能无法正确地传递虚拟中断，造成中断丢失，使得一段时间内虚拟机接收的时钟中断少于产生的虚拟时钟中断，进而造成虚拟机操作系统根据时钟中断数量计算的系统时间比实际时间落后，在虚拟操作系统设置的中断频率越大时这种情况就会越严重，Linux 系统的默认中断频率是 1000HZ，即每秒需传递 1000 个时钟中断，经观察，在 KVM 上运行 Linux 操作系统的虚拟机时，其系统时钟在连续运行一天后可落后数秒。

第三章　IEEE1588 协议原理与应用

精确时间同步对自动化技术尤其是对于分布式系统非常重要，随着分布式系统在各个工业领域的广泛应用，由于原有的网络时间协议 NTP（Network Time Protocol）的时间同步精度仅能达到毫秒级别[4]，难以达到越来越高的时间精度同步要求，而直接通过 GPS 为网络中的各个时钟授时虽可保证高精度的时间同步需求，但是这种同步方法的开销过大，亦无法满足应用的要求，因此 IEEE1588 协议 PTP（Precision Time Protocol）应运而生。

PTP 的想法诞生于上世纪 90 年代末的美国安捷伦科技公司，2002 年底 PTP 被命名为“网络测量和控制系统的精密时钟同步协议标准”，其目标是实现网络中各设备之间亚微秒级的同步，同时减低网络时间同步协议所产生开销。2008 年正式颁布了 IEEE1588 的升级版本 IEEE1588 V2，这一版本改进了原有的协议，提高了同步精度，IEEE1588v2 推出以来，得到了很大发展，在网络中交换节点充分支持，终端和时钟性能优越的情况下，同步精度可以达到纳秒级甚至亚纳秒级[6]。

3.1 IEEE1588 协议原理

IEEE1588 协议中的时钟包括主时钟和从时钟，在一个同步网络中只有一个主时钟，通过最佳主时钟算法或手动设置决定，主时钟接收 GPS 或北斗等高精度时间源，同步网络中的从时钟通过交换 1588 同步报文与主时钟同步。1588 协议的同步过程可分为两个阶段，分别是偏移测量阶段和延迟测量阶段，其同步原理如图 3.1 所示。

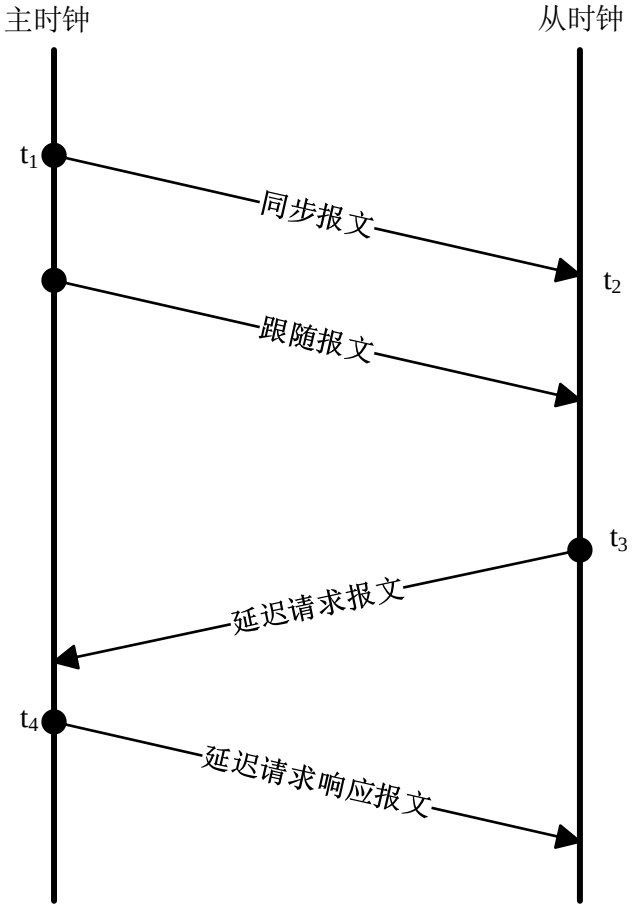

图3.1 IEEE1588 协议同步原理

（1）偏移测量阶段

时间偏移（OFFSET）指的是从时钟与主时钟同步前之间的时间差，1588 协议的同步过程就是消除主从时钟间时间偏移的过程。同步过程是由主时钟发起的，在偏移测量阶段，主时钟首先以设置的固定频率周期的通过组播向网络中的从时钟发送同步报文（Sync 报文），同步报文中可以包含有主时钟当前时间的时间戳，也可以不包含，在同步报文离开主时钟时，主时钟记录同步报文的离开时间戳 t_1。紧接着主时钟向从时钟发送跟随报文(Follow_Up 报文)，跟随报文中包含有同步报文的离开的时间戳 t_1。从时钟收到同步报文后通过本地时钟记录同步报文的到达时间戳 t_2。

（2）延迟测量阶段

偏移测量阶段后，从时钟分别收到了主时钟发送的同步报文和延迟请求报文，但由于网络中的延迟，主时钟向从时钟发送的时间无法直接用于计算主从时间偏移，因此，需要在延迟测量阶段测量主从时钟间的网络时延。在从时钟接收到跟随报文后开始延迟请求阶段，从时钟可选择合适的时间通过单播发送延迟请求报文(Delay_Req 报文)，并记录延迟请求报文的离开时间戳 t_3，主时钟通过延迟请求响应报文（Delay_Resp 报文）将延迟请求报文到达时间戳 t_4 发送给从时钟。

至此，从时钟获得了四个时间戳：同步报文到达时间 t_1、同步报文发送时间 t_2、延迟请求报文发送时间 t_3 和延迟请求报文到达时间 t_4，从时钟可通过这个四个时间戳计算主从时间的时延，具体方法如公式（3-1）（3-2）所示，首先计算出主从时间的双向时延，再将双向时延除 2，得到单向时延 DELAY，如式（3-1）所示。

$$DELAY = [(t_2 - t_1) + (t_4 - t_3)]/2 \tag{3-1}$$

通过 t1 和 DELAY 可计算出 OFFSET，如式（3-2）所示。

$$OFFSET = [(t_2 - t_1) - (t_4 - t_3)]/2 \tag{3-2}$$

在理想条件下，通过计算并消除主从时钟时间偏移 OFFSET 后即可实现主从时钟件的时钟同步，但在实际网络中，主从时钟间的双向网络时延往往是不对称的，且时延具有不确定，这也是 IEEE1588 协议同步误差产生的主要原因之一。

3.2 IEEE1588 时钟类型

IEEE1588 协议的时钟类型包括普通时钟（OC，Ordinary Clock）、边界时钟（BC，Boundary Clock）和透明时钟（TC，Transparent Clock）。

（1）普通时钟

IEEE1588 同步网络中的大部分时钟都是普通时钟，普通时钟可以是主时钟或从时钟，普通时钟是同步网络中的一般节点，可以是普通计算机或其他支持 IEEE1588 协议的网络设备，其端口都作为主时钟或从时钟连接网络。

（2）边界时钟

当同步网络的节点较多时，由于多层交换网络带来的网络时延不对称性会影响主从时钟的同步效果，因此在多级 IEEE1588 交换网络中引入了边界时钟，作为边界时钟的设备有多个 PTP 端口连接网络，其中一些 PTP 端口作为从时钟与上层网络中的主时钟通过 PTP 协议同步，而其他的一些端口作为主时钟为下层作为从时钟的设备授时，因此，边界时钟本身既是主时钟也是从时钟。

（3）透明时钟

透明时钟是在 IEEE1588 V2 中引入的，透明时钟包括了 E2E(End to End, 端到端)透明时钟和 P2P(Peer to Peer，点到点)透明时钟两种。

E2E 透明时钟与普通时钟和边界时钟不同，透明时钟本身不作为主时钟或从时钟与网络中的其他设备同步，而是网络中的交换转发设备，如支持 PTP 的交换机或路由器，E2E 透明时钟不仅转发包括 PTP 报文在内的各种网络报文，还会测量 PTP 报

文的驻留时间，并将驻留时间写入跟随报文（Follow_Up 报文）的特定字段 correctionField（时间修正域）中，从时钟在计算 OFFSET 时就可以通过读取 correctionField 中记录的修正时间对计算的 OFFSET 进行修正。

P2P 透明时钟与 E2E 透明时钟的功能原理基本相同，不同之处在于 P2P 透明时钟在计算修正时间时还会计算链路的端到端时延。

3.3 基于 IEEE1588 协议的硬件时间同步

3.3.1 基于 IEEE1588 协议的硬件时间同步网络结构

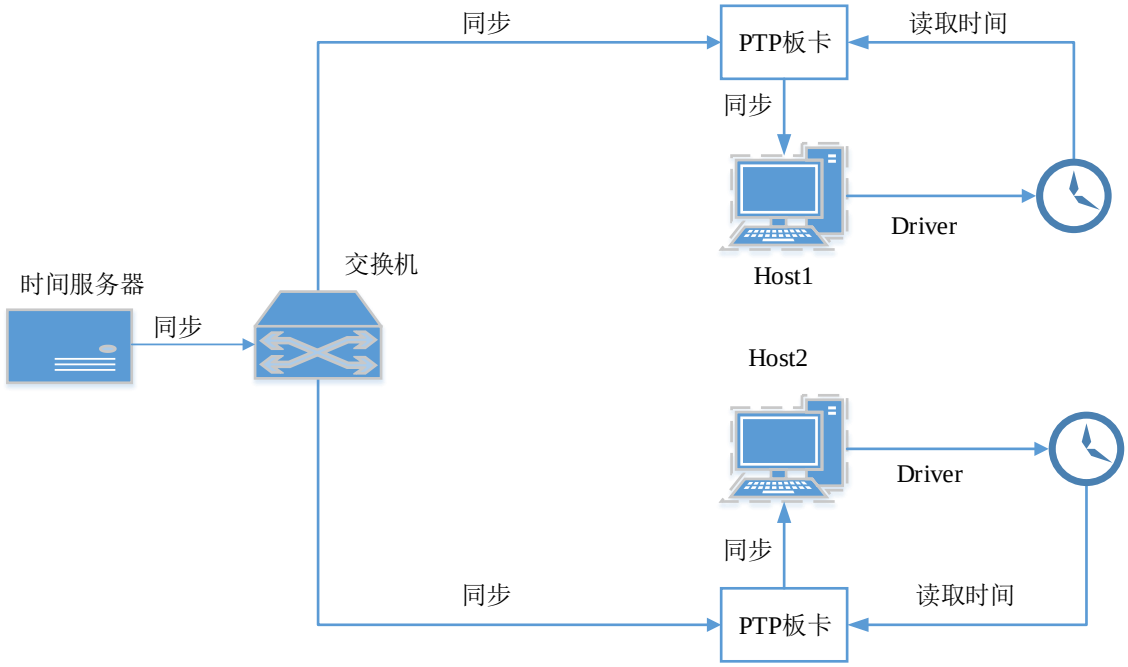

图3.2 基于 IEEE1588 协议的硬件时间同步系统网络结构

基于 IEEE1588 协议的硬件时间同步系统网络结构如图 3.2 所示，时间服务器作为 PTP 主时钟，接收外部时钟源，并为同一网络中所有作为从时钟的 PTP 板卡授时，每个 PTP 板卡和时间服务器都维护一个本地时钟，当 1588 协议报文到达和离开 PTP 板卡或时间服务器时，PTP 板卡或时间服务器可通过本地时钟为报文在物理层打上较精确的硬件时间戳，记录报文到达或离开设备的时间，完成 1588 协议所需的报文交换后作为从时钟的 PTP 板卡就获得了 4 个时间戳，根据 1588 协议 PTP 板卡可计算出其与作为主时钟的时间服务器之间的 offset，PTP 板卡再根据计算的 offset 对其本地时钟进行偏差补偿，主从时钟以一定的频率不断重复此过程，进而实现板卡与时间服务器之间的精确同步。多个 PTP 板卡通过 PCIe 接口与需要同步的设备连接，使得与之连接的计算机或其他设备可通过 PCIe 接口读取时间，实现各设备间的时间同步。

根据时间同步网络的精度要求，网络中的交换机可使用普通交换机或支持 1588 协议的交换机以实现边界时钟或透明时钟功能，当使用普通交换机时，由于受网络中时延非对称性影响，根据 1588 协议，会带来较大的同步误差，而使用支持 1588 协议的交换机时，由于其支持边界时钟或透明时钟功能，可大大减少网络时延非对称性给时间同步系统所带来的影响。

因此，在理想的条件下，理论上基于 IEEE1588 协议的硬件时间同步系统可实现各 PTP 板卡之间纳秒级甚至亚纳秒级的高精度同步。

3.3.2 硬件时间同步设备

（1）时间服务器

本文实验采用的时间服务器是用于实验室的 PTPGrand-100 型时间服务器。PTPGrand-100 时间同步系统可同时接收北斗卫星导航系统、GPS、PTP 网络时间报文及地面授时信号，产生与 UTC 保持同步的脉冲信号、串口时间报文、NTP/PTP 网络时间报文、地面授时信号等时间信息，全面支持最新的 IEEE 1588-2008 时间同步协议，采用基于硬件的时间标记，保证时间信号输出的有效性，提供稳定、可靠、精确的时间和频率基准。如图 3.3 所示，时间服务器上的接口有网口和 BNC 接口，时间服务器可通过网口与网络相连实现同步，并可通过连接在同一网络的计算机进入配置界面，BNC 接口输出 PPS（秒脉冲），可用于硬件同步误差的测量。

图3.3 PTPGrand-100 时间服务器实物图

（2）PTP 板卡

本文实验采用的 PTP 板卡是 KW-800，IEEE1588v2 PCIe 板卡 KW-800，主要应用在金融、工业控制及数据采集等行业。通过 PCI-E 接口直接为服务器或 PC 机授时。通过内存 I/O 映射和专有操作系统时间同步算法，实现业界领先的应用程序授时精度；支持即插即用（Plug and Play）。为多种流行操作系统提供驱动程序和管理工具。和时间服务器一样，PTP 板卡上也有网口和 PPS 输出接口，分别用于硬件的同步和测量。

图3.4　KW-800 PTP 板卡实物图

（3）交换机

由于本文所研究的虚拟集群同步精度为微秒级，且其误差主要是由操作系统中的硬件 PTP 板卡读取时延产生的，因此系统中硬件 PTP 板卡的同步精度只要达到纳秒级即可达到要求，所以在本文的实验中，同步系统中的交换机使用普通交换机而并未使用支持 IEEE1588 边界时钟和透明时钟的交换机。

（4）硬件时间同步系统的同步精度测量

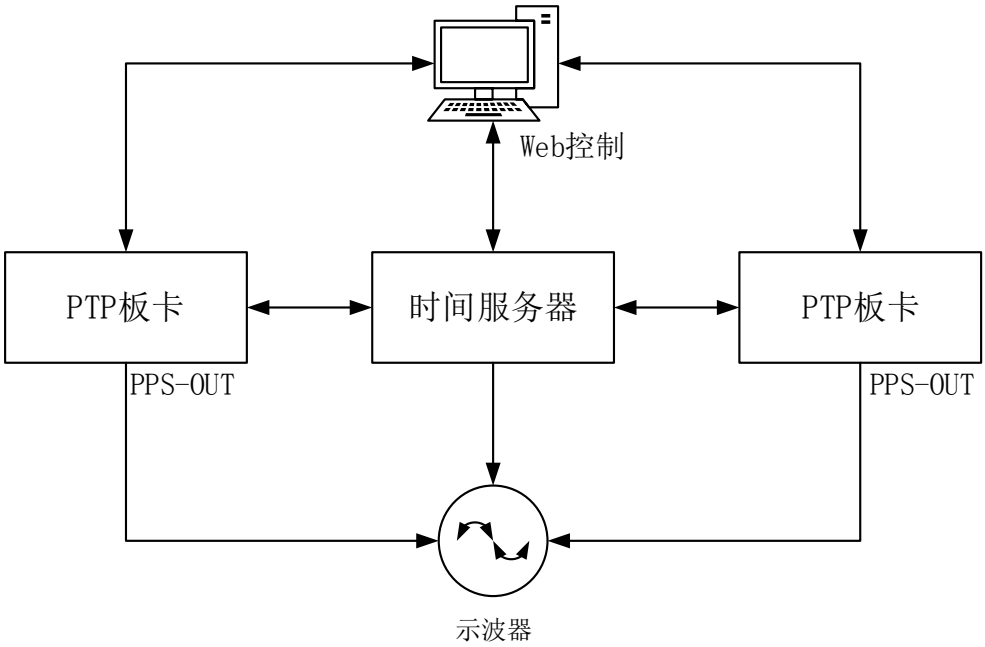

图3.5　硬件 PTP 同步系统精度测量

本文中基于 IEEE1588 协议的硬件时间同步设备同步误差测量方式如图 3.5 所示，将 PTP 时间服务器用作主时钟，PTP 板卡用作从时钟。主时钟和从时钟通过普通交换机相连。然后通过 Web 浏览器分别对主从时钟进行配置，使主时钟以 1 秒为周期

运行 1588 同步协议同步从时钟。然后将主从时钟的 PPS 输出接口与示波器相连，以测试其同步精度。测试过程中的交换网络中无其他网络报文。

图3.6 PTP 硬件系统时间误差测量图

如图 3.6 所示，黄色波形(中间)为时间服务器（主时钟）PPS 输出波形，蓝色波形（最左）和紫色波形（最右）为 PTP 板卡（从时钟）的 PPS 输出波形，经过长期实验证明，PTP 板卡与时间服务器之间的误差维持在 150ns 以内。上述实验结果说明，即使在网络中交换机不直接支持 IEEE1588 协议的情况下，两物理板卡间的时间误差精度相对于计算机上层软件微秒级的同步误差也已足够小，可忽略不计，因此可以证明硬件 PTP 板卡可作为测量物理计算机和虚拟机上同步误差的标尺。

3.3.3 硬件板卡支持虚拟机同步关键技术

安装到实体计算机上的 PTP 板卡通过局域网与 IEEE1588 精准时间服务器进行持续的时钟偏差测量和纠正，能够获得准确的同步时间。PTP 板卡的时间误差精度主要取决于服务器和板卡的时间准确度和对时性能，以及网络对时间信息帧准确传递的支持程度。由于板卡硬件同步机制的时间戳获取与插入，以及时间误差的滤波与计算都在板卡硬件内完成，基本不会受到计算机任务负载的影响，因而所提供的精度远高于虚拟机同步精度要求。所以，关键在于如何合理而可靠地将板卡上的时间信息准确传递给虚拟机。

3.4 基于 IEEE1588 协议的软件同步

3.4.1 PTPd 软件

PTPd 是 IEEE1588 协议的一种被普遍应用的纯软件实现方式，PTPd 是开源软件，包括了两个版本：Version1.x.x 和 Version2.x.x。本文采用的版本为 Version 2.3.1。因

为 V2 版支持 IEEEl588.2008，提供了更好的精度及引入透明时钟。

PTPd 软件包括 Master 和 Slave 两部分，分别工作在作为主时钟和从时钟的设备上计算机上，主从时钟通过网络相互发送 IEEE1588 协议报文，并在发送和接收时通过上层软件打上时间戳，主从时钟的协议引擎中包括了基于 IEEE1588 协议的状态机，协议报文的收发会引起状态机的转移，从而使得 IEEE1588 协议在主从时钟之间工作，实现主从时钟的同步。

PTPd 软件与基于 IEEE1588 的硬件同步系统都使用 IEEE1588 协议报文，因此可通过软硬件结合的方式运行 IEEE1588 协议，如使用高精度的硬件时间服务器作为主时钟，并使运行 PTPd SLave 的计算机作为从时钟通过网络与之连接，通过交互 IEEE1588 协议报文实现主从时钟之间的同步。

3.4.2 软硬件同步方式的区别

PTPd 的同步原理与硬件 PTP 同步系统相同，都是通过运行 IEEE1588 协议计算主从时钟之间的同步误差，然后消除误差，进而实现主从时钟的时间同步，但实际应用时，PTPd 软件的同步精度要远远低于硬件 PTP 同步系统[33]，其主要原因在于时间戳的精度，运行 PTPd 软件同步时，同步报文由计算机的网卡接收和发送，由于计算机的普通网卡没有时钟设备，时间戳需要由上层软件方式读取时钟设备后产生，这就带来了同步报文延迟的不确定性，根据 IEEE1588 协议，这会产生同步误差。而且当计算机 CPU 负载增大时这种情况会更加严重，另一方面，计算机连接网络时经常会产生大量的网络流量，下行和上行的网络流量增大时会造成同步报文的收发时间戳延迟。

理论上，时间戳的产生位置越靠近底层硬件，IEEE1588 协议的同步精度越高。使用纯硬件主从时钟设备的 PTP 同步网络的时间戳均由物理层产生，因此相比纯软件的同步方式，硬件同步方式的时间戳精度更高，因此其同步误差更低且同步效果更稳定。

3.5 IEEE1588 协议的误差分析

理论上，IEEE1588 V2 协议的同步精度在理想条件下能够达到纳秒级甚至亚纳秒级，但在其各种实际应用场景中，由于软硬件条件和网络环境等因素的不同，它们时间同步精度会有很大区别[34]。因此，本节总结了 IEEE1588 协议应用时的误差原因分析和优化方法。

3.5.1 时间戳精度

如上节 PTPd 软件同步中所介绍的，IEEE1588 协议同步报文时间戳的产生位置

对同步精度有着重要的影响，不仅是 PTPd 软件，IEEE1588 协议的各种应用为达到更高的同步精度，都应使时间戳尽可能地接近物理层硬件，因为运行 IEEE1588 协议时从时钟靠计算主从时钟收发同步报文的时间计算双向总延迟，理论上，准确的收发同步报文时间应是报文到达或离开硬件网卡的时间，为得到并记录这个时间，主从设备将会读取其自身的时钟，而从同步报文到达网口的时刻到读取到时钟时间的这段时间就是时间戳的误差。使用纯软件方式的 PTPd 同步时需要上层软件读取系统时间，由于操作系统中的进程调度等原因，会造成时间戳不准，进而影响同步精度。此外，主从时钟设备上的性能和 CPU 负载也会影响时间戳的精度。

3.5.2 链路双向延时不同

在 IEEE1588 协议中，为计算同步报文（Sync 报文）从主时钟到从时钟的时延，在延时测量阶段计算主从时钟的双向总延时，即报文从主时钟到从时钟的时间加上从从时钟到主时钟的时间，再将双向总延时除 2 得到单向时延，认为这就是从主时钟到从时钟的时延。

在一条网线上，两个方向的传播时延是基本相同的，但在网络拓扑结构中，由于交换机和路由器的存在，网络数据报文交换转发时延具有不确定性，造成从时钟和主时钟之间的双向时延不同，而在一个局域网络中，往往存在多层的交换机和路由器，使得双向时延不同的影响更加严重。

为降低同步网络中多级交换机和路由器对同步精度的影响，IEEE1588 V2 中引入了透明时钟，透明时钟可计算同步报文在交换机或路由器中的驻留时间，并将计算的时间写入到同步报文的 correctionField（时间修正域）里，这样，从时钟在计算单向时延时就可以根据 correctionField 中的值消除透明时钟交换机或路由器中双向报文驻留时间不等的影响。

透明时钟的引入可以解决交换机和路由器双向报文转发时延不等问题，但使用透明时钟需要要求作为透明时钟的交换机或路由器支持 IEEE1588 协议，增加了同步系统的成本和复杂度，因此在同步精度要求不高时一些基于 IEEE1588 的同步网络仍然使用普通交换机和路由器，此时，连接主从时钟和交换机的网线也有可能影响同步效果，应在交换机的所有接口上均使用百兆或均使用千兆网线。如一边百兆一边千兆，会导致上下行时延的不对称，其原因是交换机一般采用二层转发方式实现数据转发，二级转发使用分组转发机制，当交换机端口收到长 80 字节的同步报文时，交换机会将接收的同步报文存储到缓存中，等到 80 字节的报文完全接收完毕后再转发到目的端口，有些交换机支持百兆和千兆端口，当端口连接百兆或千兆网线时，端口速率会自动调整为百兆或千兆，当主时钟和从时钟分别通过百兆和千兆连接交换机时，就会造成交换机上下行存储转发时延不同，具体计算结果如下：

百兆：收 1bit 需要 10ns，1 字节为 80ns；

千兆：收 1bit 需要 1ns，1 字节为 8ns；

一个 80 字节的报文从百兆到千兆的延时为：80*80 = 6400ns；

一个 80 字节的报文从千兆到百兆的延时为：80*8 = 640ns；

因此与式(3-2)不同，实际的 OFFSET 如式(3-3)所示：

$$OFFSET = ((t4 - t3) - (t2 - t1))/2 + (6400 - 640)/2 \qquad (3\text{-}3)$$

如果上下行对称则加号右边即(6400-640)/2 部分应接近为 0，由于百兆和千兆之间的转化造成较大不对称性导致与实际的 offset 相差 3μs 左右。在硬件同步系统中可通过示波器观测到这种现象，主从时钟间同步误差为 3μs 左右。

3.5.3 网络时延影响

当基于 IEEE1588 协议不使用专用网络时，网络上除同步报文外还有各种其他报文，网络上的其他报文与同步报文之间的碰撞也会造成双向时延的不确定性，影响同步精度。

在同步精度要求较高时，应使用专用网络运行 IEEE1588 协议，从而完全避免与网络其他报文碰撞的可能。在非专用网络上运行 IEEE1588 协议时，为提高同步精度，也可通过降低同步报文与网络其他碰撞概率的方式减少双向时延不确定性。在支持优先级调度的网络中，可提升同步报文的优先级，并增加抢占机制，使得同步报文不受网络其他报文的影响，由于同步报文以固定的频率通过网络在主从时钟间交互，也可通过周期性任务调度算法保证同步报文的服务质量。

在一些特殊的 IEEE1588 协议应用场景中，网络中其他任务也具有周期性或规律性，例如，在电网网络中，传输着包括同步报文在内的各种控制报文，且各类电网控制报文普遍具有周期性或规律性，此时，可通过网络自学习方法优化电网同步性能，其具体方法是：在电网 IEEE1588 同步模块上添加自学习模块，记录网络其他报文的到达时间，再通过数据统计、机器学习等方法学习之前一段时间网络其他报文到达时间的到达时间，预测未来一段时间发送延迟请求（Delay_Req）报文的碰撞概率，根据 IEEE1588 协议，从时钟发送延迟请求报文的发送时间并不影响同步误差的计算，因此可选择最佳的发送时间发送延迟请求报文，以减少其与网络其他报文的碰撞概率。

3.5.4 虚拟环境对 PTPd 软件的影响

IEEE1588 协议的同步效果与时间戳的产生位置有关，时间戳与物理层硬件越近同步误差越小。而在 KVM 虚拟化平台上，与物理机相比，虚拟机与底层硬件之间加

入了 VMM，由于 PTPd 软件时间戳在上层软件产生，因此虚拟机操作系统中时间戳的产生位置离物理层硬件更远。此外，VMM 上常常运行这不只一台虚拟机，为使多台虚拟机同时运行，VMM 通过调度使多台虚拟机以时分的方式共享虚拟机时间，即使 VMM 上只有一台虚拟机，Host 系统也会抢占虚拟机的资源，这就会使虚拟机有时会被挂起，无法使用 CPU，当同步报文到达时如果虚拟机正在被挂起，那么同步报文只能被阻塞，等到虚拟机通过 VM Entry 重新唤醒后才能再被处理，打上时间戳。因此，在虚拟机上使用 PTPd 作为从时钟与主时钟同步时同步精度较低。

在 KVM 虚拟化平台上，VMM 将物理网卡虚拟成多个虚拟网卡，虚拟机通过虚拟网卡连接网络，从另一个角度来看，各个虚拟网卡组成了一个虚拟局域网，物理网卡作为网桥连接虚拟局域网和外部网络，因此，相比于物理机上基于 IEEE1588 的软件同步系统，虚拟集群上的同步系统中，由于作为从时钟的虚拟机与主时钟之间又多了一层交换机，而与物理交换机相似，虚拟网桥的报文上下行驻留时间不同会造成双向时延不同，影响同步性能。为改善虚拟机上 PTPd 软件的同步性能，也可将物理交换机上的透明时钟概念应用于虚拟网卡上，即将同步报文从物理网卡到虚拟网卡的转发时延记录到 correctionField 中，将普通虚拟网桥变为透明网桥。

第四章　基于 IEEE1588 的虚拟集群任务同步测量技术方案

如上一章所述，基于 IEEE1588 协议的硬件时间同步系统中多个 PTP 同步板卡通过一跳交换机与时间服务器连接，各板卡间同步误差远小于 1μs，因此同步误差在微秒级的多虚拟机与物理机直接可以通过安装在物理机上的 PTP 板卡时间作为标尺，实现时间同步和测量。PTP 板卡通过 PCIe 接口安装在物理计算机上，在物理机操作系统中，可通过安装 PTP 板卡的驱动程序直接读取板卡时间，而在虚拟机中，需通过 I/O 虚拟化技术读取 PTP 板卡时间，KVM 中的 PCI 设备透传可直接实现一台虚拟机使用并独占 PCIe 设备，但这种 PTP 板卡时间读取方法并不适用于多虚拟机的同步测量。

4.1 基于透传的虚拟机硬件 PTP 板卡时间读取方法

PTP 板卡实现与时间服务器的同步之后，其内部时间计数器持续更新时间接口寄存器中的计数值。在物理机操作系统上安装板卡驱动程序后，板卡驱动程序运行在所安装的物理机宿主操作系统的内核中，定时的从板卡上读取最新时间，并可及时更新计算机操作系统的时间。在虚拟机操作系统中，无法直接利用 PTP 板卡上的时间信息更新自身的系统时间。

透传（PassThrough）是一种设备直通方式的硬件辅助 I/O 虚拟化技术[35]，这种技术是通过英特尔 VT-d 的 DMA 重映射实现的。通过透传，可以使运行在 VMM 之上的一台客户虚拟机完全独占该 I/O 设备，同时将该 I/O 设备与运行在 VMM 之上的其他虚拟机隔离，使得该设备对其他虚拟机不可见。使用透传设置后，在这台独占虚拟机上可以通过与物理机上完全一样的方式使用该 I/O 设备，而且由于设备物理地址映射到客户虚拟操作系统之上，因此在其性能与在物理机上直接使用该设备十分接近。

基于透传的虚拟机硬件 PTP 板卡读取方法就是将作为 PCIe 设备的硬件 PTP 板卡通过透传方式与一台客户虚拟机绑定，使这台客户虚拟机能够直接访问 PTP 板卡并读取时间，但此时 PTP 板卡对其他虚拟机不可见，其他虚拟机无法访问该设备。使用透传方法后，与 PTP 板卡绑定的客户虚拟机可独享 PTP 板卡，PTP 板卡硬件信息会直接提供给虚拟机客户操作系统识别，因而可以直接在虚拟机上安装驱动程序，安装驱动程序后可由驱动提供的接口读取 PTP 板卡上的时间。

透传方式的优点是直接由虚拟机操作系统支持，实现简单。但其缺点是：需要给每一台虚拟机指定一个专门的 PTP 板卡，并且当虚拟机上运行的是 VxWorks、BlueCat 等操作系统时，需要定制多种驱动程序。这两点都意味着较高的实现成本，并且即使

能够承受，物理计算机上也难以提供足够的板卡插槽，因此在透传方式难以应用于虚拟集群中虚拟机见的同步测量。

4.2 基于超调用的虚拟机硬件 PTP 板卡时间读取方法

由于基于透传方式读取时间的虚拟机硬件 PTP 板卡时间读取方法只能使一台虚拟机完全独占一个硬件 PTP 板卡，因此这种方法不适合虚拟机群中多台虚拟机的同步和测量。而本章提出的基于超调用（Hypercall）的虚拟机时间读取方法可解决这一问题，实现多台虚拟机通过一块 PTP 板卡读取时间，因此这种方法适用于虚拟机群中多台虚拟机的同步和测量。超调用的实现机制与操作系统中的系统调用类似。

4.2.1 Linux 中的系统调用

系统调用[36]是底层操作系统内核为上层应用程序提供的接口，系统调用的实现方式与虚拟机的超调用十分相似。

英特尔 X86 处理器为操作系统设定了 Ring0-Ring3 四种权限级别，其中 Ring0 最高，Ring3 最低，Linux 使用 Ring0 和 Ring3 分别作为内核态和用户态。

当程序运行在内核态时，可直接执行特权指令，可自由的访问计算机的存储设备和外设设备，而当程序运行在用户态时因处于 Ring3，权限级别不够，所以不能执行特权指令，不能直接访问这些设备。操作系统中的大多数程序都运行在用户态，当这些程序需要访问这些设备时就需要进行用户态和内核态的切换，这种切换的实现方式就是系统调用。

操作系统中每个系统调用都有一个系统调用号，当运行在用户态的程序需要使用系统调用时，通过指令造成内中断，从而引起用户态到内核态的陷入。陷入时，用户态程序可通过将系统调用号作为参数写入寄存器 eax 中，陷入到内核态后，内核的系统调用处理函数就可以通过读取系统调用号来判断用户态的程序需要执行的操作。除系统调用号外，系统调用还可以通过除 eax 外另外 5 个寄存器传递其他参数，如需传递更多的参数，也可以通过一个寄存器传递一个用户空间存储地址的指针。当内核态完成操作后返回用户空间，可将返回值通过寄存器返回。

4.2.2 KVM 中的超调用

超调用是一种类虚拟化平台中客户虚拟机与 VMM 的通信方式，如图 4.1 所示，它的原理和实现方式与操作系统中的系统调用非常相似。不同之处在于系统调用的陷入是从用户空间陷入到内核空间执行相关操作后返回，而超调用是从客户虚拟操作系统陷入到 VMM 中执行相关操作后返回。

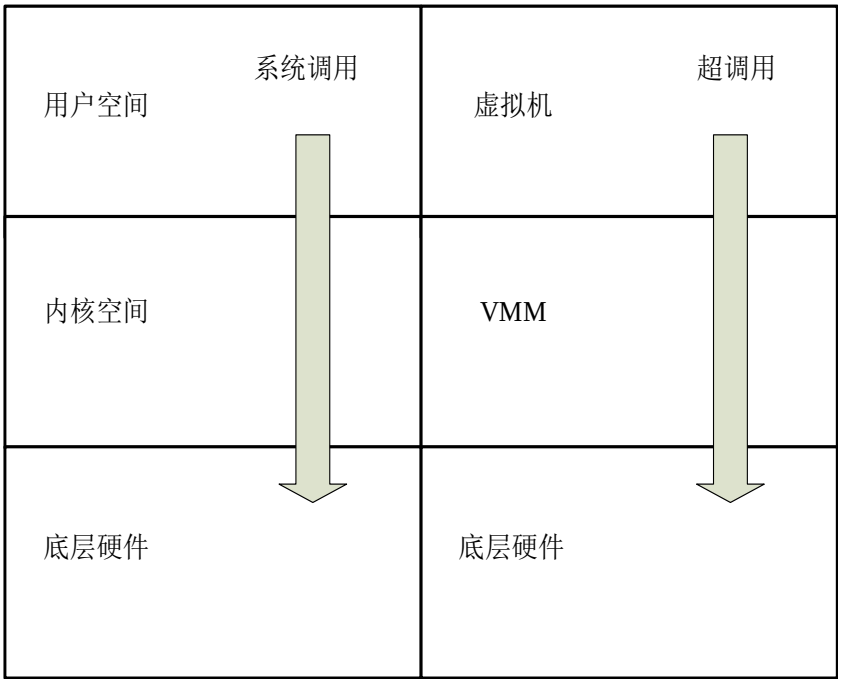

图4.1 超调用与系统调用

每个基于 KVM 的虚拟机都作为一个进程运行于 Linux 系统的用户空间，其权限级别是 Ring3，VMX 处于非根模式，所以如果执行特权指令，需要通过 VM Exit 陷入到 VMM 中执行特权指令并通过 VM Entry 返回虚拟机操作系统，外部中断也会引起陷入，因此虚拟机在运行时会频繁的执行陷入过程。除此之外，还有一种由虚拟机主动执行的陷入方式，这就是虚拟机的超调用。

KVM 内核中提供了超调用的接口，超调用通过 X86 架构的 VMCALL 指令实现[36]，VMCALL 指令是 0x0f，0x01，0xc1，当 Linux 作为客户操作系统运行在 KVM 平台上时通过执行这 3 个字节的指令，可以触发 VMCALL，执行 VM Exit 陷入到 VMM 中，Linux 源码中 arch/x86/include/asm/kvm_para.h 文件中定义了 4 个超调用接口函数，与系统调用类似，超调用函数的第一个参数是超调用号，4 种超调用函数可分别再传入 0-3 个参数，超调用函数将输入的每个参数分别存入一个 32 位寄存器中。虚拟机发起超调用后作为 VMM 的 KVM 内核模块接管控制权，如果 KVM 内核模块通过 VCPU 描述符中的信息判断虚拟机 VM Exit 的原因，如果发现 VM Exit 是由 VMCALL 引起的，则调用超调用处理函数 kvm_emulate_hypercall 来处理，kvm_emulate_hypercall 的输入为指向 VCPU 描述符的指针，kvm_emulate_hypercall 首先读取 VCPU 的 5 个寄存器中存储的参数，其中第一个寄存器中的参数作为超调用号，然后通过对超调用号的 switch 判断执行已定义的相应操作，并将结果存入 VCPU 寄存器，最后通过 VM Entry 返回到客户虚拟操作系统中。虚拟机中的超调用函数是同步方式执行的，虚拟机执行超调用函数陷入到 VMM 后阻塞，等到 VMM 执行完超调用处理函数后，超调

用函数通过读取存储多个寄存器中的值并返回。

超调用是一种由虚拟机主动陷入到 VMM 的方式，在这种方式下虚拟机与 VMM 可通过读写寄存器进行通信，通过修改 KVM 内核添加超调用号实现相关操作，虚拟机自身知道其运行在虚拟平台之上，打破了客户机和宿主之间的隔离性，因此可以认为在这种情况下 KVM 由全虚拟化平台变为类虚拟化平台。

4.2.3 虚拟机基于超调用的时间读取方法

基于超调用的时间读取方法在 Host 上加载 PTP 板卡驱动，修改 KVM 内核代码中的超调用处理函数 kvm_emulate_hypercall，首先删除 VCPU 权限级别的判断语句，因为该语句禁止 VCPU 权限级别非 Ring0 时使用寄存器，然后添加读取板卡时间的超调用号和读取板卡时间的程序，最后将读取到的 PTP 板卡时间存储到寄存器中，由于读取的 PTP 板卡时间是一个 64 位 long long int 变量，而每个寄存器的存储空间是 32 位，因此将秒位和 ns 位分别存储到寄存器 RAX 和 RBX 中并返回。在客户虚拟机操作系统中通过 VMCALL 指令陷入，并将获取板卡时间的调用号存储在 VCPU 的寄存器中。发生超调用引起 VM Exit 后，KVM 内核读取到获取板卡时间的调用号，将获取的板卡时间存储到寄存器中，在 VM Entry 时返回给虚拟机，虚拟机通过读取 RAX、RBX 两个寄存器存储的秒和 ns 位时间获得 PTP 板卡时间，进而通过同步系统时间或打印来实现虚拟机操作系统时间的同步与误差测量。

4.2.4 基于超调用的虚拟机硬件 PTP 板卡时间读取方法的优点

虚拟机基于超调用的硬件 PTP 板卡时间读取方法可实现多台虚拟机共享硬件 PTP 板卡时间，解决了透传方式下仅能使一台虚拟机独占一个 PTP 板卡的问题。这种方法使用了 KVM 的类虚拟化接口，相比于通过 virtio 实现类虚拟化，其实现简单，不需要开发前端驱动和后端驱动，而且由于虚拟机通过 VMCALL 指令触发陷入后，直接进入了 KVM 内核 PTP 板卡读取时间，其性能几乎不受 Host 操作系统的影响，误差范围较小，在下一节中将会通过实验对这种方法的误差进行测试和分析。因此，可以通过这种方法实现虚拟机群中多台虚拟机之间的时间同步与误差测量。

4.3 测量方法的误差分析及优化方法

如上一章所述，通过与时间服务器同步，多个硬件 PTP 板卡之间可达到纳秒级的同步，对于同步误差为微秒级的同步系统其纳秒级误差可忽略不计，因此，硬件 PTP 板卡可作为同步测量的标尺，测量时使多个待测设备通过系统定时器在约定的系统时间读取 PTP 板卡时间，理论上，每两台待测物理机或虚拟机在相同系统时间读取到的 PTP 板卡时间就是两台设备之间的同步误差。但是实际中，由于待测的操作

系统无法在理论时间读取到硬件 PTP 板卡的时间，因此会造成测量误差。

4.3.1 误差原因分析

虚拟机通过定时器触发超调用读取硬件 PTP 板卡时间的误差包括定时器触发延迟和读取延迟。

定时器触发延迟是指 Linux 系统定时器的实际触发时间和理论触发时间的误差，由于 Linux 的系统定时器由精度时钟设备 PIT 产生，因此定时器每次的触发延迟较大且不确定，经实验测试一般在数百微秒左右，有时可达 1 毫秒以上，物理机操作系统读取 PTP 板卡时也会受系统定时器触发延迟的影响。因此测量时需通过读取触发时的系统时间消除定时器误差。

读取延迟包括客户虚拟机陷入时延和在KVM内核中读取PTP板卡时间寄存器的延迟，读取延迟无法准确测量，因此无法消除，这也是本方法的主要误差产生原因，但是由于 KVM 是监控模型的 VMM，VMM 拥有所有底层硬件的使用权，所以读取延迟不受 Host 系统的影响，其值较小，本章中的虚拟机超调用时延测量实验证明其延迟一般在 20us 以内，因此本方法的测量误差范围在 20μs 以内。

4.3.2 定时器触发延迟的消除方法

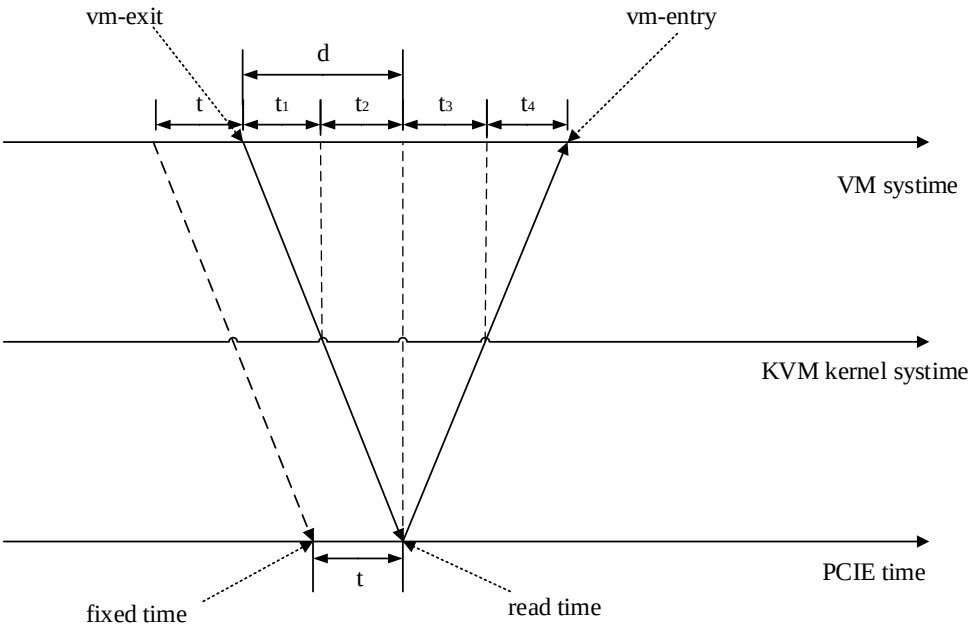

图4.2 定时器触发误差的消除方法

PTP 板卡通过 PCIe 接口安装在物理机上，物理机上通过 KVM 创建并运行虚拟

机 VM，虚拟机启动定时器以一定时间间隔读取硬件 PTP 板卡时间，如图 4.2 所示，在理论触发时间后经过时间 t 定时器实际触发，触发后经过时间 t_1 陷入到内核中，又经过时间 t_2 后读取到 PTP 板卡时间，再经过时间 t_3 和 t_4 返回到虚拟机中，此时，虚拟机读取到了一个 PTP 板卡时间。

图中 t 为定时器触发时延，$d=t_1+t_2$ 为测量误差即读取时延，t_1+t_4 为虚拟机陷入的时延，t_2+t_3 为读取 PTP 板卡寄存器的时延。在虚拟机中定时器触发时首先读取虚拟机的系统时间，将读取的系统时间减去定时器的理论触发时间得到定时器触发时延 t，再将读取到的板卡时间（read time）减去 t 得到修正后的时间，即可消除定时器不准的影响，这个修正后的时间（fixed time）就被认为是虚拟机操作系统理论触发时间时的 PTP 板卡时间。

读取时延 d 造成的误差由于在测量时无法确定，所以无法消除，但可通过实验证明其误差范围。

4.3.3 虚拟机超调用时延测量实验

为证明基于超调用的虚拟机硬件 PTP 板卡时间读取方法的误差范围，进行虚拟机超调用时延测量实验。根据 4.3.1 中所述的虚拟机时间读取误差分析，由系统定时器引起的误差可以消除，但虚拟机陷入时延和读取硬件板卡时延所造成的误差由于在测量时无法确定，所以无法消除。这部分误差就是图 4.2 中的 d，虽然无法直接测量出 d，但是由于 $d=t_1+t_2$,且 $t_1+t_2<t_1+t_2+t_3+t_4$，因此可以通过测量 $t_1+t_2+t_3+t_4$ 证明虚拟机超调用时延上限。

具体方法是在虚拟机中运行读取 PTP 板卡时间超调用，获得 PTP 板卡时间，但不以此改变系统时间，在运行读取 PTP 板卡时间超调用语句前后分别读取虚拟机的系统时间，以虚拟机系统时间作为基准，通过前后两次读取的虚拟机操作系统时间间隔计算超调用时延。

为测试虚拟机超调用时延误差的分布范围，在虚拟机中运行以两秒为间隔的定时器，每次定时器触发时执行上述操作，运行约 9 小时后，得到 16000 个左右的时延数据，对这些数据处理后得到了时延的频数分布图，实验结果如图 4.3 所示。

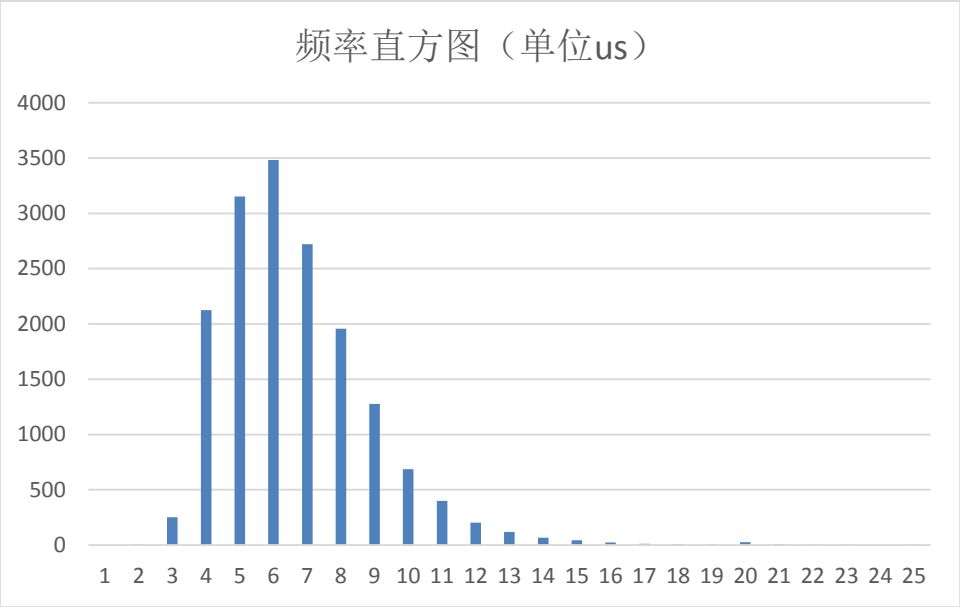

图4.3 虚拟机超调用时延测量实验结果

图 4.3 为虚拟机超调用时延测量实验结果，其中横轴为每次定时器触发后得到的超调用时延，单位为 μs，纵轴是在实验数据在每一微秒的分布数，根据时延结果显示，超调用时延频数分布中心为 6μs，时延数据最大值为 25μs，且绝大多数时延数据分布在 15μs 以内。根据实验结果可证明虚拟机超调用时延的极值与分布中心的误差在 20μs 以内，因此可证明虚拟机基于超调用的硬件 PTP 板卡测量方法误差在 20μs 以内。

4.4 同步误差测量方式

4.4.1 本地记录

本地记录方法是使两台不同待测设备在一定时间段内通过系统定时器以相同间隔不断读取硬件 PTP 板卡时间并存储，再通过 Excel 对数据处理，将两台设备在相同系统时间读取到的硬件 PTP 板卡时间两两相减，得到一段时间内两待测设备的同步误差分布。这种方法可获得大量的同步误差数据，并获得同步误差分布图。为得到长时间持续测量结果，本文的实验数据均通过这种方法获得。

4.4.2 时间同步测量软件

本地记录方法可长时间持续记录待测设备之间的同步误差，但无法实现同步误差的实时测量。因此设计了时间同步测量软件，可在实际应用时实时显示待测设备的同步误差。该软件已上传 github 网站[37]。

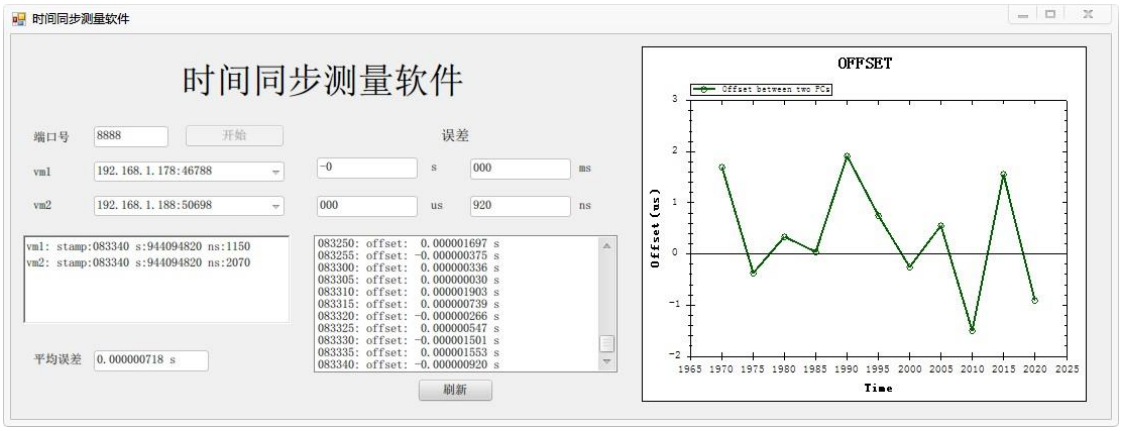

图4.4　时间同步测量软件界面

时间同步测量软件用 C#编写，在 windows 系统下运行，运行在非待测设备的电脑上，时间同步测量软件是一个 Socket 服务器，其界面如图 4.4 所示，测量时，各待测操作系统上运行 Socket 客户端程序，通过 TCP 与服务器连接，在约定的相同系统时间，以相同间隔不断读取硬件板卡时间，再将读取到的时间和系统时间组帧后发送给服务器，时间同步测量软件将选择的两个客户端相同系统时间读取到的 PTP 板卡时间相减得到误差，通过数字和波形图显示并不断刷新，计算最近一段时间内的平均误差，从而实现实时可视化误差测量。

通过时间同步测量软件，可以比较直观的观察两个待测操作系统的时间同步误差，而且可以实现同步误差的实时监控，方便现场测试。由于这种测量方式与本地记录方式一样，都是通过两待测操作系统在约定系统时间读取硬件 PTP 板卡时间，时间同步测量软件只是接收这两个时间并相减，因此测量精度与网络的时延无关。

4.5 基于超调用的虚拟集群硬件 PTP 板卡辅助同步测量方案

在本章之前几节介绍的虚拟机同步测量技术基础上，本节以软件 PTP 同步的测量为例介绍基于超调用的虚拟集群硬件 PTP 板卡辅助同步测量方案，测量方案如图 4.5 所示。

虚拟集群中包括 4 台虚拟机 VM1、VM2、VM3 和 VM4，分别运行于两台物理机之上（一台物理机上可运行任意数量的虚拟机，在 PTP 板卡数量足够时可实现多台物理机上虚拟机的同步测量），两台物理机分别通过 PCIe 接口安装两块硬件 PTP 板卡，硬件 PTP 板卡自身维持一个时钟，并作为从时钟通过硬件同步网络与时间服务器同步，时间服务器和 PTP 板卡除网口外还各有一个 PPS 输出接口，可输出秒脉冲，因此可通过同轴电缆(图中粗线)将各 PTP 板卡及时间服务器的 PPS 输出接口与示波器连接，通过示波器测量 PTP 板卡与时间服务器之间的同步误差（如图 3.6），待

PTP 板卡稳定同步后开始虚拟机的同步测量。

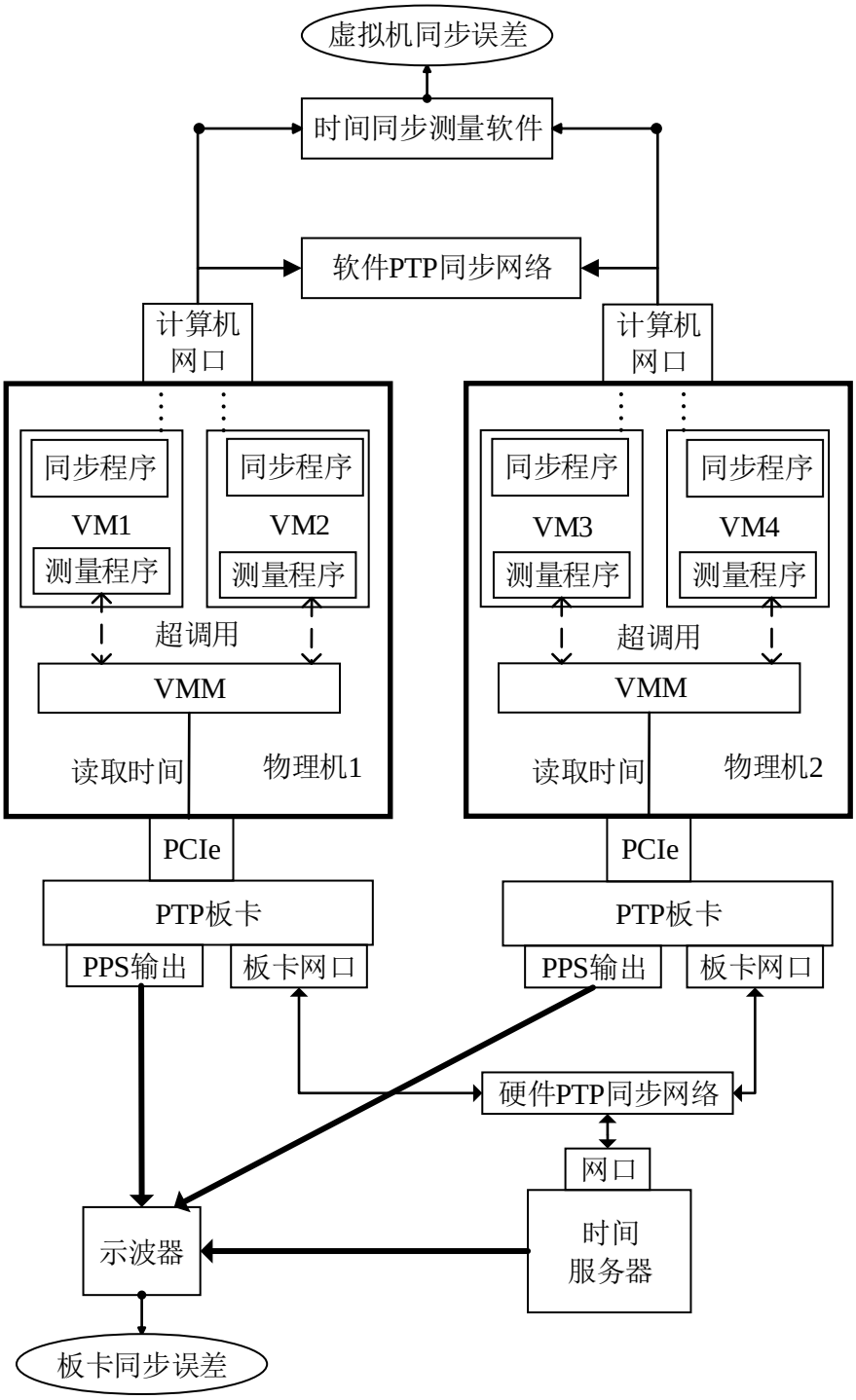

图4.5 基于超调用的虚拟集群硬件 PTP 板卡辅助同步测量方案

各虚拟机上分别运行同步程序和测量程序，同步程序为待测的同步方案，如当使用 PTPd 软件时，虚拟机上的同步软件就是 PTPd 从时钟软件，它通过虚拟网卡与软件 PTP 网络中的主时钟同步。开始测量后，各虚拟机上的测量程序通过定时器在相

同的系统时间通过超调用方式读取 PTP 板卡上的时间，并将读取到的时间通过 Socket 发送给运行在一台 Windows 系统计算机上的时间同步测量软件，时间同步测量软件接收来自多个虚拟机上的时间，在其可视化界面上选择其中两台虚拟机后测量软件可显示选中的两台虚拟机的同步误差，实现软件 PTP 同步效果的测量。

图 4.5 中使用时间同步测量软件作为测量方式，可实现同步误差测量结果的实时显示，如需长期测量记录同步误差可使用本地测量方式，即将两台被测虚拟机上读取到的时间在本地保存，在完成测量后再处理两台虚拟机上的数据得到测量结果。

第五章 软硬件同步方法测量实验和性能分析

通过第三章介绍的基于 1588 协议的硬件同步系统，安装在物理计算机上作为1588 从时钟的硬件 PTP 板卡可与网络中作为 1588 主时钟的时间服务器实现纳秒级的同步。通过第四章介绍的基于超调用的硬件 PTP 板卡时间读取方法可在物理机上多台虚拟机操作系统中分别读取硬件 PTP 板卡时间，并可保证读取时间的误差在 20μs 内。本章通过实验测试了基于软硬件的同步方法在物理机和虚拟机上同步效果，并对实验结果进行了分析。

5.1 实验环境

5.1.1 实验设备

同步实验中的时间服务器和硬件 PTP 板卡为第三章中所介绍的 PTPGrand-100 时间服务器和 KW-800 板卡，实验中使用的网线全部为千兆网线，经过一跳交换机同步后其同步精度可达到 150ns。实验中两块 KW-800 板卡通过 PCIe 接口分别安装到两台支持英特尔 VT-d 的计算机上。两台计算机上分别安装修改了内核的 Linux-3.13 系统，安装 KW-800 板卡的驱动程序，并通过 QEMU-KVM 运行多台虚拟机，虚拟机上安装的操作系统为普通的 Linux 系统。

5.1.2 软硬件同步方法

实验中首先将硬件 PTP 板卡与时间服务器同步，板卡与服务器开始同步后约 10 分钟达到稳定，经示波器测量板卡与服务器误差小于 150ns。

实验中的硬件同步方法是在虚拟机上运行一个定时器，每 0.1 秒通过超调用方式读取一次 PTP 板卡时间，然后将虚拟机操作系统时间更新为读取到的板卡时间。在物理机上的硬件同步方法直接通过 PTP 板卡驱动程序接口读取板卡时间，系统时钟同步方式与虚拟机中相同，通过以 0.1 秒为间隔的定时器不断更新系统时间。

软件同步方法是使用基于 1588 v2 的 PTPd 软件实现的，在被测物理或虚拟操作系统上运行 PTPd 作为 slave，时间服务器作为 master，被测操作系统通过网卡与时间服务器连接到同一网络并运行 PTPd 程序与之同步。使用软件方法同步时，完全通过计算机的网卡运行 1588 同步协议，硬件 PTP 板卡只作为测量工具。

5.1.3 同步误差测量方法

在待测设备与时间服务器稳定同步后开始测量同步误差，测量方法使用第四章介

绍的本地记录方法，在两个待测操作系统中运行定时器，从相同的系统时间开始以2s 为间隔不断读取得到硬件 PTP 板卡时间，记录一段时间的测量结果后，将得到的板卡时间两两相减，得到同步误差。在虚拟机上，通过超调用方式读取板卡时间，在物理机上直接通过 PTP 板卡的驱动程序接口读取板卡时间。在测量硬件同步方法的同步效果时，同步方法与测量方法相同，但同步程序和测量程序运行时相互独立，因此测量得到的同步误差可反映同步效果。

对各组测量实验的结果统计处理后制作了误差时间频数分布图，各组频数分布图的横轴是同步误差时间，单位均为 μs，纵轴为同步误差值在每 1μs 上的分布频数。

5.1.4 实验中的负载

同步系统运行时，操作系统中其他任务的运行会带来内存负载的增加，计算机的CPU 负载会对同步性能有所影响，为测试软硬件同步方法在计算机较高负载条件下的同步性能，每组同步误差测量实验都增加了有负载的情况。

实验中添加背景内存负载的方法为在 Host 系统中运行高频率定时器进行运算，由于通过 QEMU-KVM 创建的虚拟机均运行在一个物理核上，可通过 Linux 系统设置将定时器程序进程运行在指定的核上，并通过调整定时器的运算频率以设置该核上的CPU 使用率，这样就可以为虚拟机添加背景 CPU 负载，实验中的背景 CPU 负载均为50%。

5.1.5 实验方案

IEEE1588 同步协议在应用时受多种条件影响同步效果有所不同，因此在以上实验环境基础上，本文针对虚拟化环境中影响同步效果的多种因素，设计了多个同步测量实验方案。为研究虚拟机软硬件同步方案的同步误差范围和稳定性，分别测量了软硬件同步方法在虚拟机上一段时间的同步误差；为研究虚拟化环境对软硬件同步方法产生的影响，分别测量了软硬件同步方法应用在物理机上的同步效果，并与虚拟机上的实验结果进行比较；为研究多物理机和单物理机软件同步的同步效果差异，分别测量了在两台物理机和一台物理机上的两台虚拟机实验 PTPd 软件的同步误差；此外，以上每组实验方案均测量了无负载和 50%CPU 负载条件下的同步误差，以研究 CPU 负载对各个实验方案的影响。

5.2 硬件同步测量实验

5.2.1 物理机硬件同步测量实验

首先通过多物理机硬件板卡同步测量实验测量物理机操作系统通过硬件板卡的

同步误差，作为参考。该实验中，在两台物理机上分别安装硬件同步板卡，板卡通过时间服务器同步，物理机通过驱动程序的软件接口读取网卡时间并同步系统时间，两物理机分别通过定时器每隔 2 秒读取板卡时间，获得 3000 个数据，两两相减后得到同步误差测量结果。

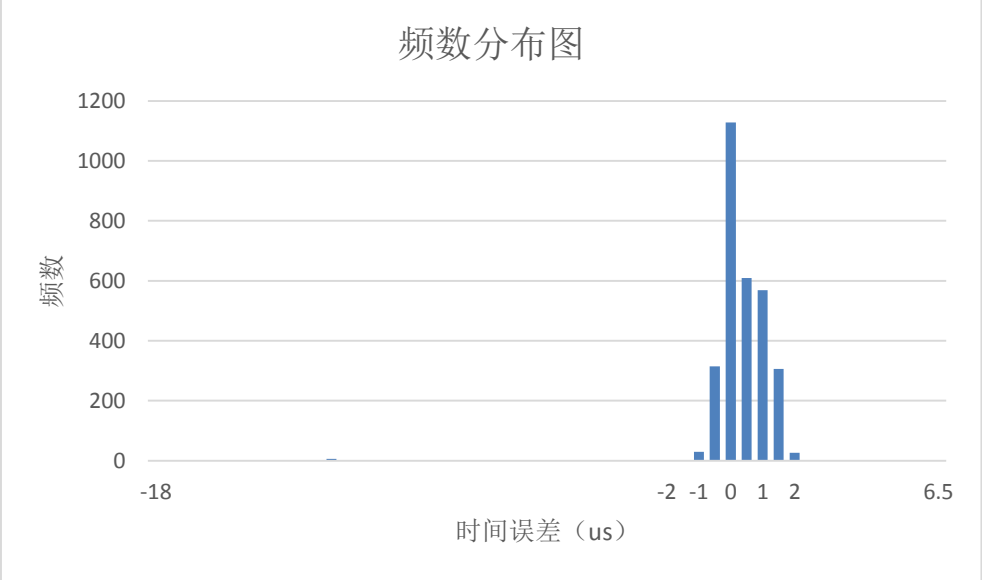

图5.1　多物理机上硬件同步时间误差（未加负载）直方图

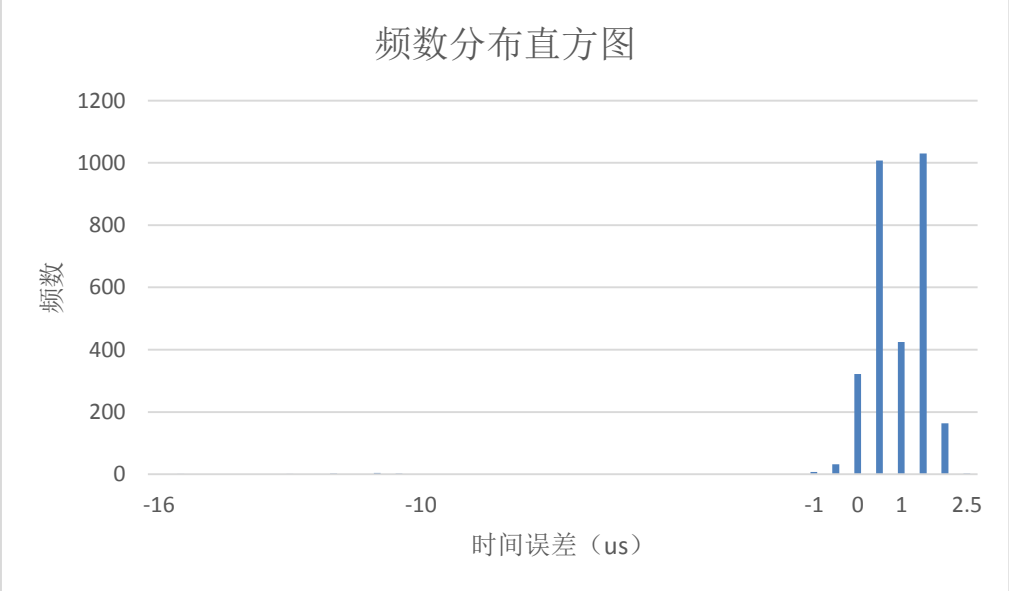

图5.2　多物理机硬件同步时间误差（加 50%负载）直方图

如图 5.1 和图 5.2 所示，实验结果显示，物理机上硬件同步误差分布中心偏移为 0.5μs，主要集中于分布中心附近 2μs 以内，最大偏差在分布中心附近 18μs 以内，加负载后影响较小，分布中心偏移为 1μs，主要集中于分布中心附近 2μs 以内，最大偏

差在分布中心附近 16μs 以内，误差主要是由测量误差（不同计算机读取板卡时延不同）引起的。

5.2.2 虚拟机硬件同步测量实验

如图 5.3 所示，该实验测量运行在两台不同的物理机上的虚拟机通过硬件同步方法同步后的同步误差，图中 Host 代表物理计算机，VM 代表虚拟机，该方案中两台虚拟机 VM1 和 VM2 分别运行在两台不同的物理机 Host1 和 Host2 上，两台物理机上分别安装 PTP 板卡，两个 PTP 板卡通过一级交换机与时间服务器相连，将时间服务器作为 master，两个 PTP 板卡作为 slave，通过交换机使两个 PTP 板卡与时间服务器的主时钟同步。两台虚拟机 VM1 和 VM2 上分别通过基于超调用的硬件同步方法将系统时间与 PTP 板卡时间同步。在保持稳定同步后，在 VM1 和 VM2 上分别同时运行定时器，每隔 2 秒读取板卡时间，获得 3000 个数据，两两相减后得到同步误差测量结果。

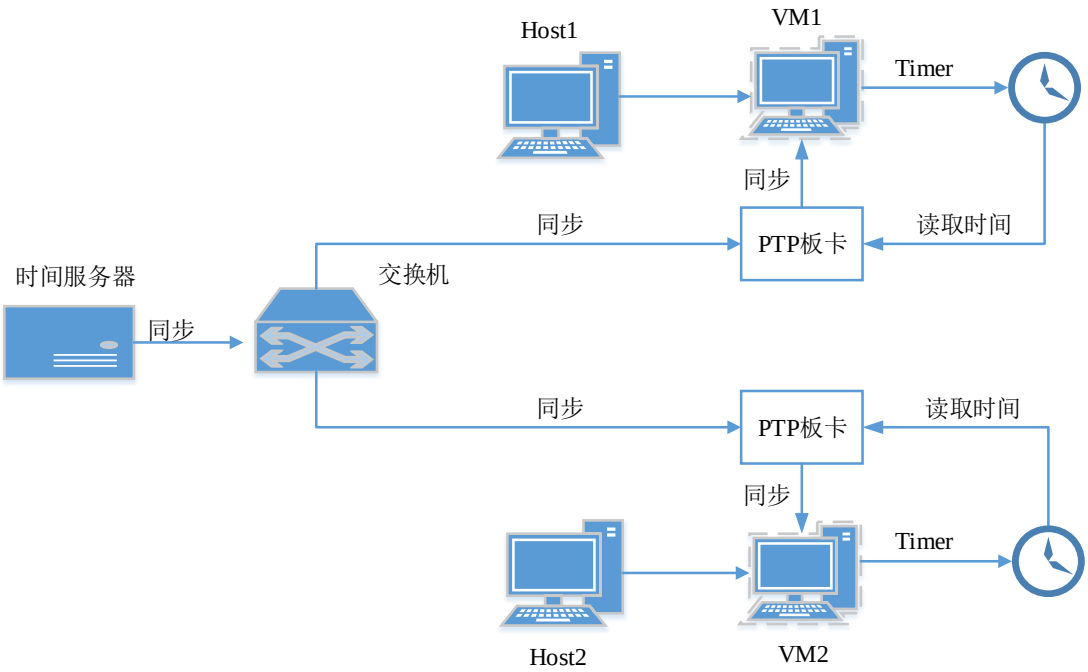

图5.3　虚拟机硬件同步方案

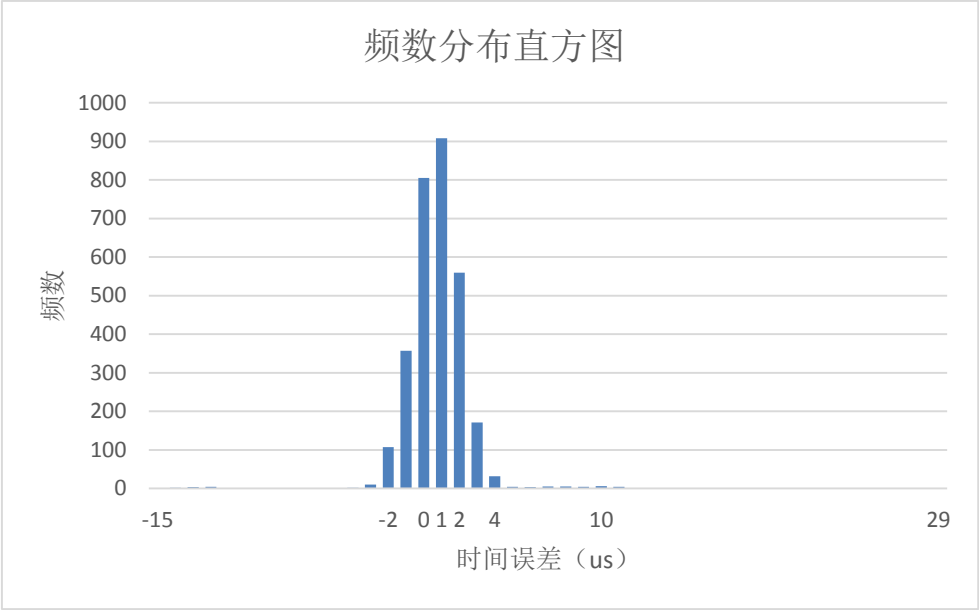

图5.4 多物理机上虚拟机间硬件同步时间误差（未加负载）直方图

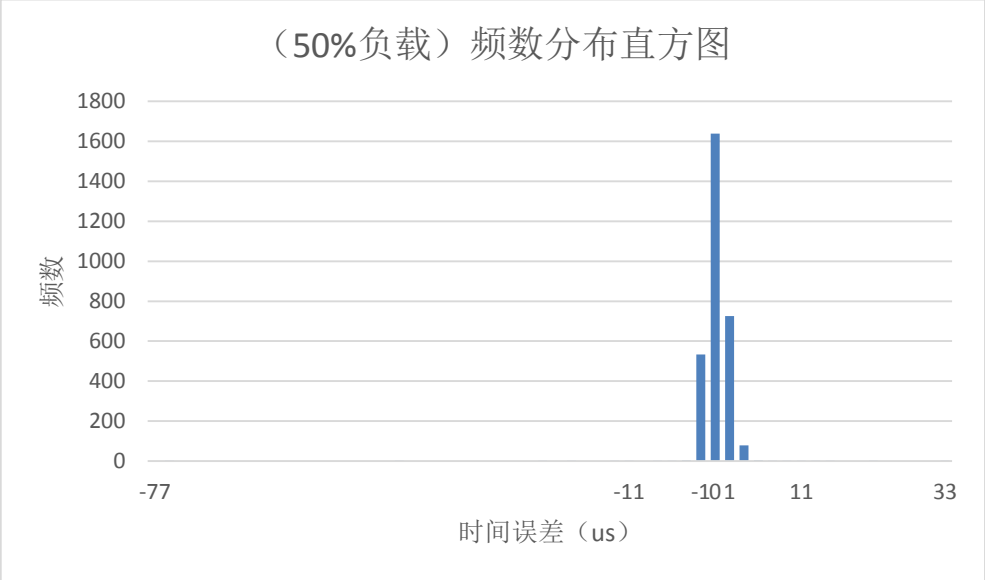

图5.5 多物理机上虚拟机间硬件同步时间误差（加 50%负载）直方图

如图 5.4 和 5.5 所示，多物理机上虚拟机硬件同步误差分布中心偏移为 0μs，主要集中于分布中心附近 2μs 以内，最大偏差在分布中心附近 30μs 以内。加负载后同步误差分布中心偏移 0μs，主要集中于分布中心附近 8μs 以内，最大偏差在分布中心附近 80μs 以内。

通过虚拟机和物理机硬件同步测量实验证明通过超调用方式实现的虚拟机基于硬件 PTP 板卡同步方法的同步效果与物理机接近，且其同步精度在高 CPU 负载下受影响较小。

5.3 软件同步测量实验

软件同步测量实验对物理机和虚拟机通过 PTPd 软件同步的误差进行了测量，为研究同步误差分布中心偏移的产生原因，分别进行了多物理机上和单物理机上虚拟机间的软件同步祥测量实验。

PTPd 软件同步需要一段时间才能实现稳定同步，从 0.5s 逐渐同步到稳定需要 20 分钟，实验数据均为同步稳定后一段时间的同步误差。处理软件同步测量数据时，为观察 PTPd 软件的同步时间稳定性，除了制作同步误差频数分布直方图外，还制作了同步误差时间折线图，其横轴为误差测量结果的序号，即距离测量开始的时间，纵轴为同步误差，单位为 ns，同步时间折线图可以反映软件同步误差变化的时间规律。

5.3.1 物理机软件同步测量实验

首先，测量两台物理机上使用 PTPd 软件的同步效果作为对比。

实验结果如图 5.6 至 5.9 所示，测量了有无负载条件下两台物理机使用 PTPd 软件同步的同步误差。

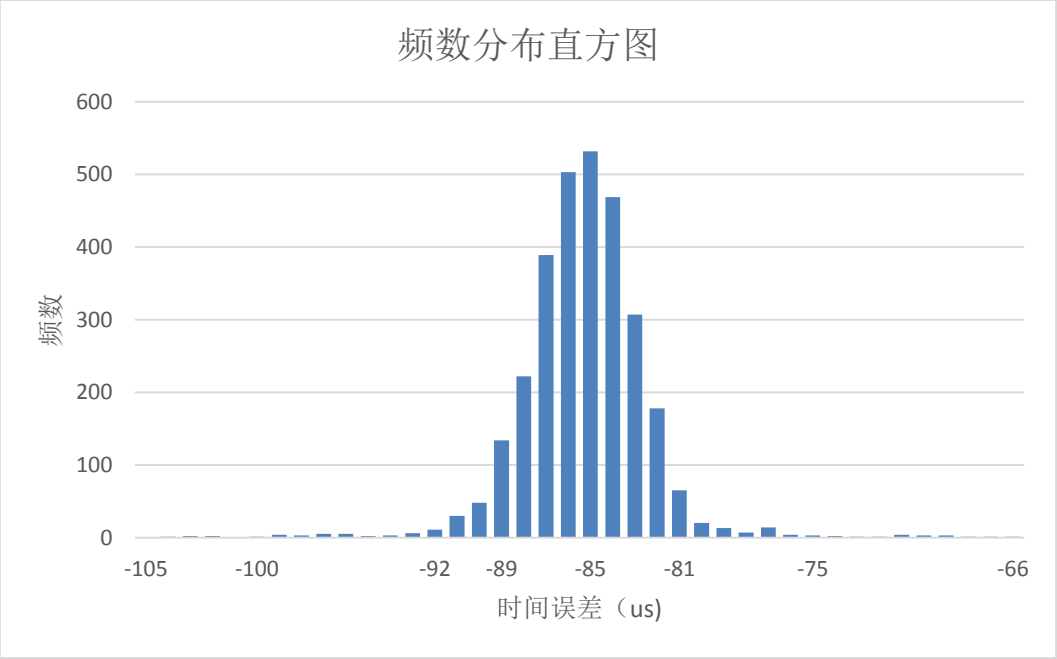

图5.6　多物理机间软件同步时间误差（未加负载稳定后）直方图

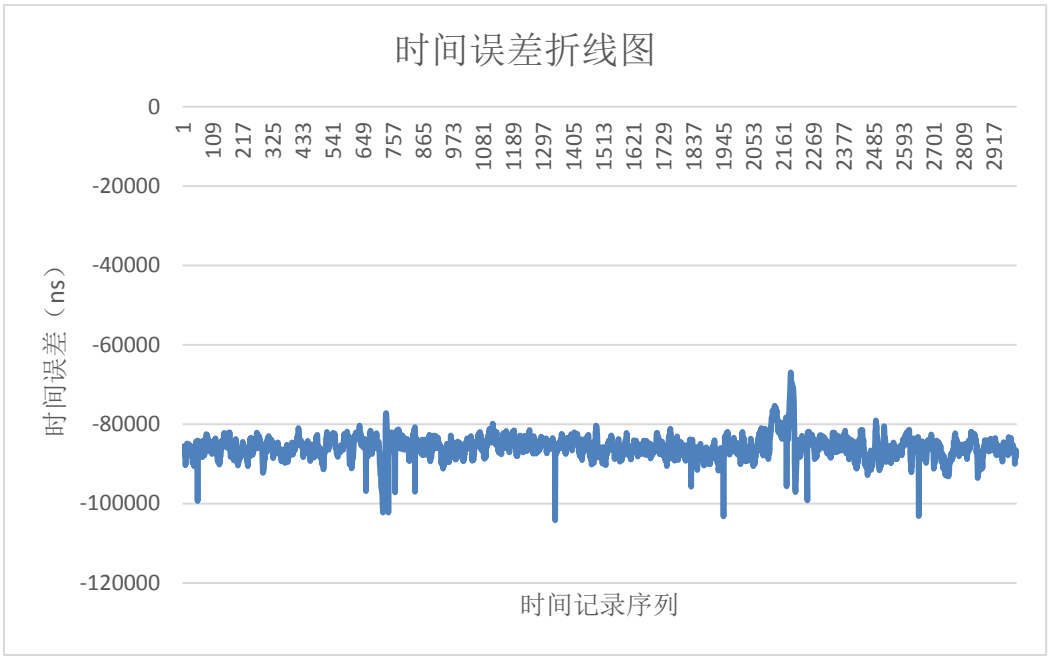

图5.7 多物理机间软件同步时间误差（未加负载稳定后）折线图

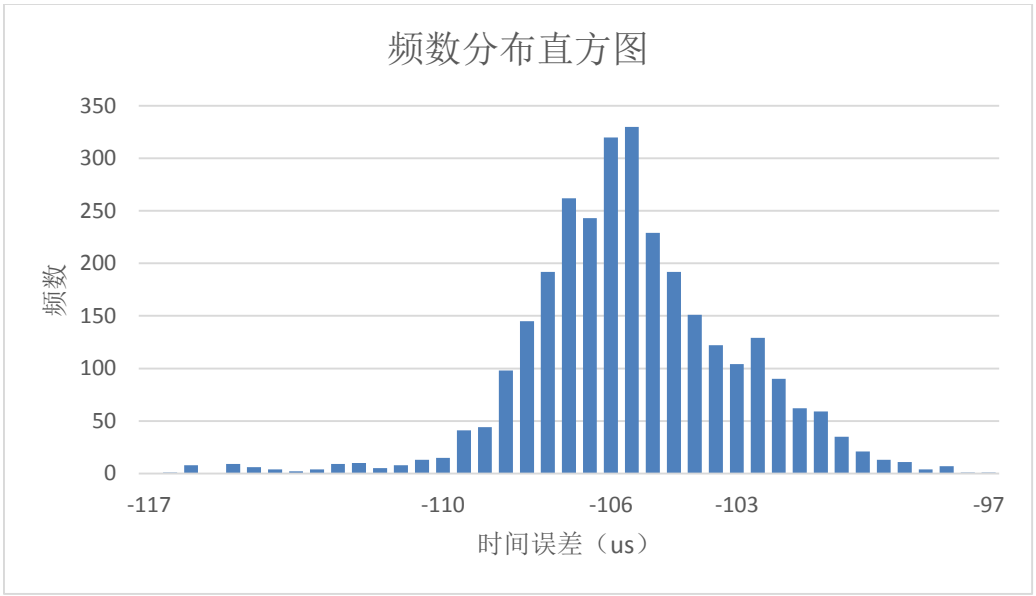

图5.8 多物理机间软件同步时间误差（加 50%负载稳定后）直方图

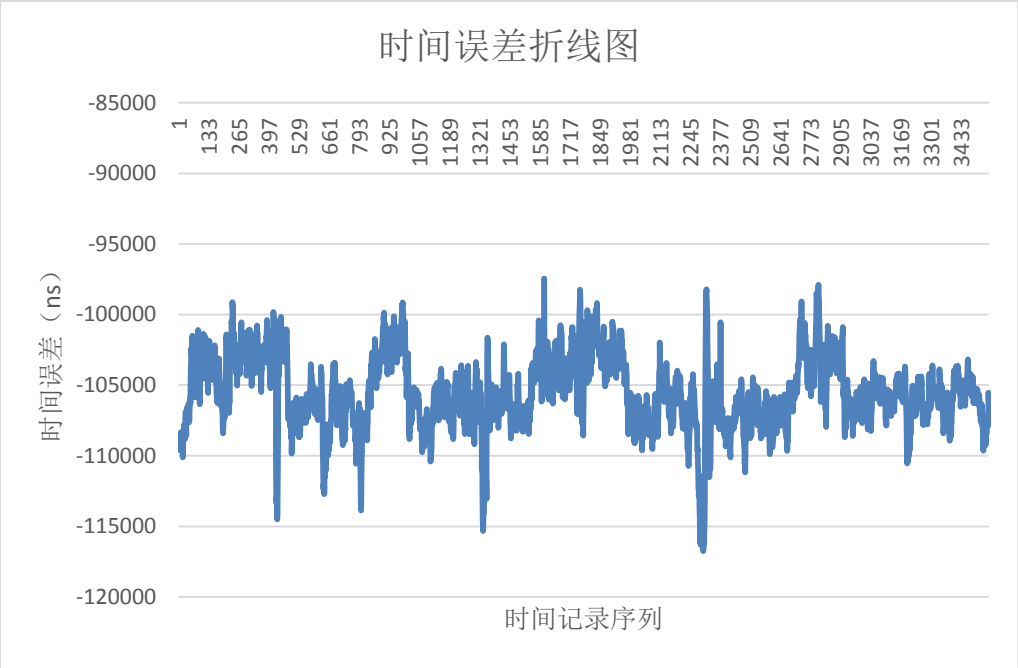

图5.9　多物理机间软件同步时间误差（加 50%负载）折线图

如图 5.6 至图 5.9 所示，实验结果显示通过 PTPd 软件同步，两台物理机的同步误差分布中心偏移为 86ns，主要分布于分布中心附近 5μs 以内，与分布中心偏差最大值在 20μs 以内。加负载后分布中心偏移为 106μs，主要分布于分布中心附近 5μs 以内，与分布中心偏差最大值在 10μs 以内。

通过测量物理机上 PTPd 软件同步效果发现同步分布中心出现较大偏差，根据 IEEE1588 协议同步原理，可证明软件同步方式存在比较固定的双向时延差。在有无负载的情况下，同步误差均基本维持在分布中心 5μs 以内，这证明物理机上 PTPd 同步效果比较稳定，且受 CPU 负载影响较小。

5.3.2　多物理机上虚拟机软件同步测量实验

图 5.10 为多物理机上虚拟机间软件同步测量方案，将两个 PTP 板卡与时间服务器同步，虚拟机 VM1，VM2 的时间利用 PTPd 软件同步，将时间服务器作为 master，将 VM1、VM2 作为 Slave 同步。VM1，VM2 分别读取硬件 PTP 板卡的时间作为测量的标尺，每隔 2 秒读取板卡时间，获得 3000 个数据，从而得到测量结果。测量结果如图 5.11 至图 5.14 所示。

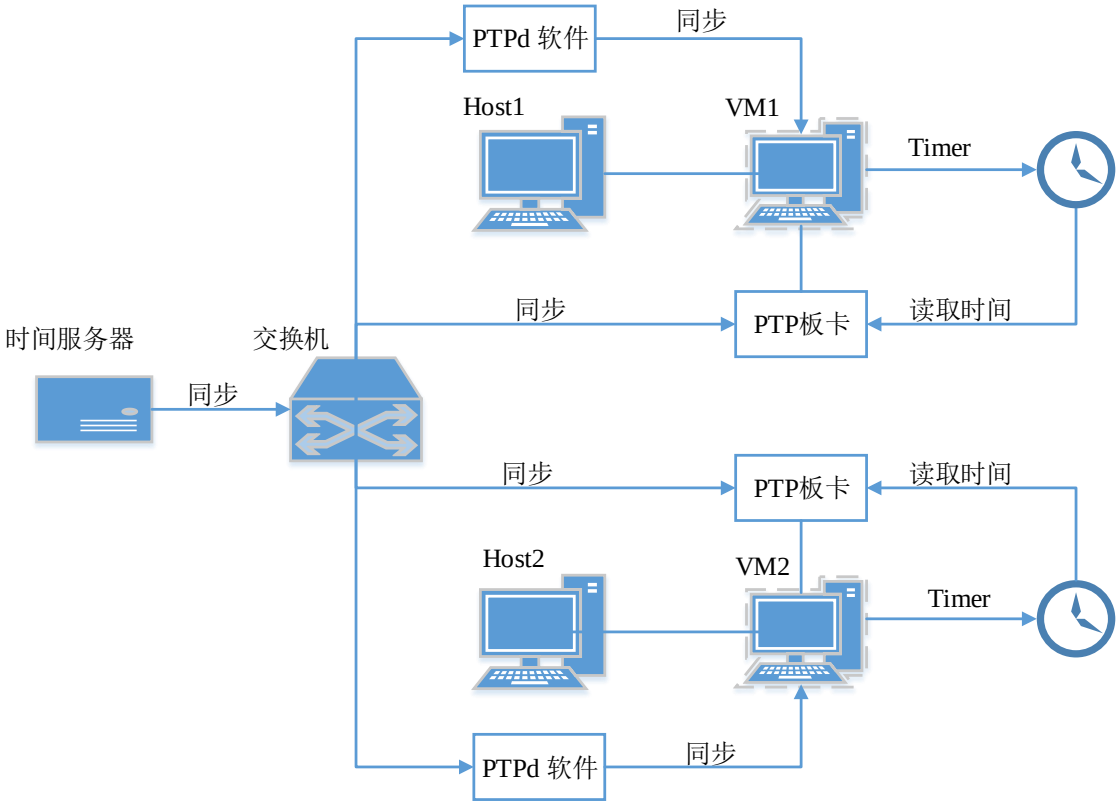

图5.10 多物理机上虚拟机间软件同步测量方案

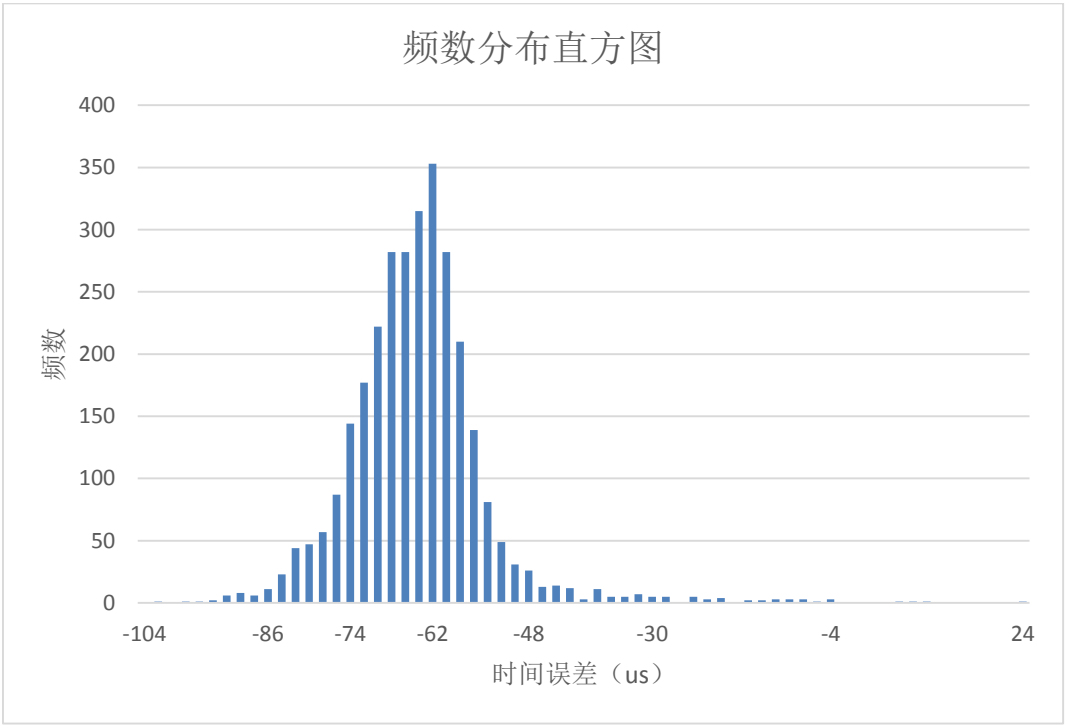

图5.11 多物理机上虚拟机间软件同步时间误差（未加负载）直方图

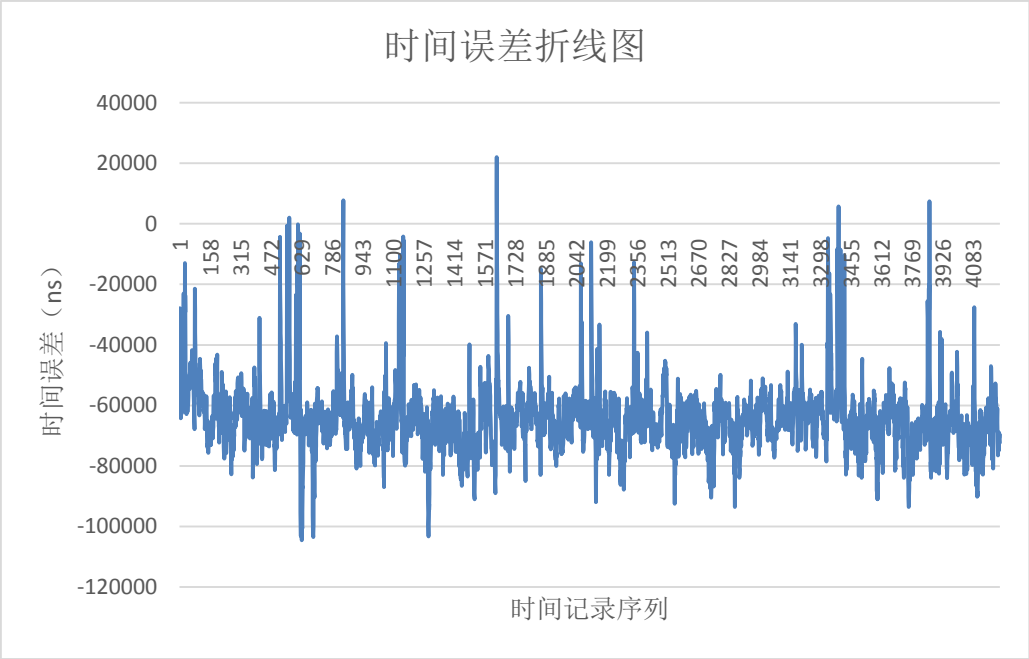

图5.12 多物理机上虚拟机间软件同步时间误差（未加负载）折线图

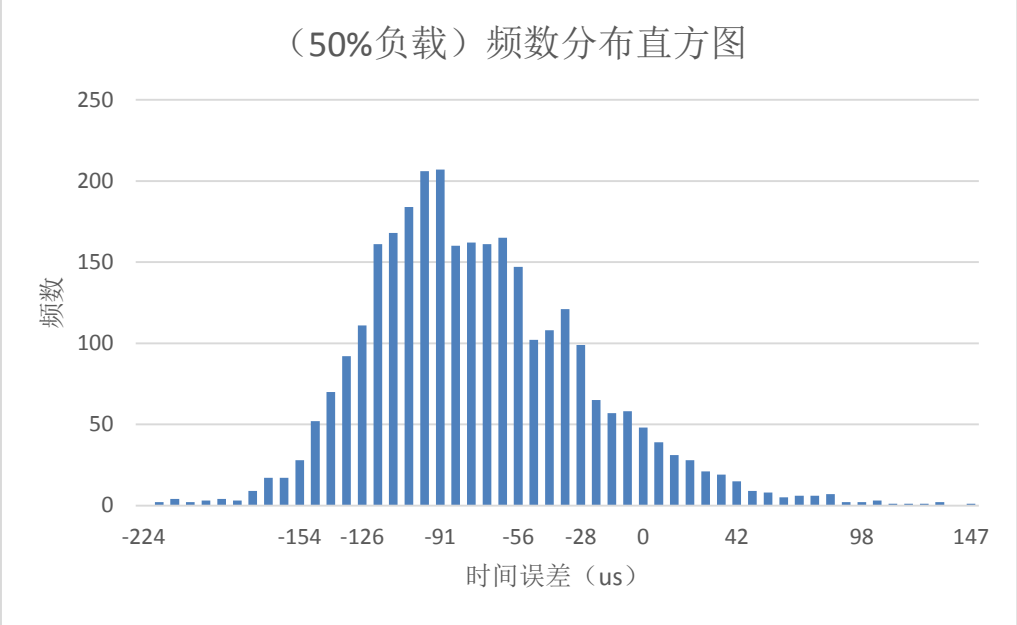

图5.13 多物理机上虚拟机间软件同步时间误差（加 50%负载）直方图

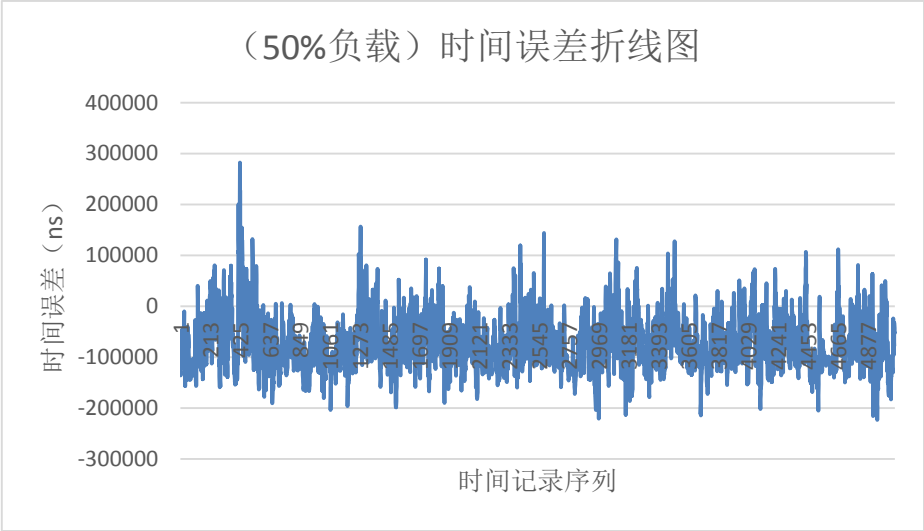

图5.14 多物理机上虚拟机间软件同步时间误差（加 50%负载）折线图

如图 5.11 至图 5.14 所示，实验结果显示，两台物理机上的虚拟机的同步误差分布中心偏移为 65μs，主要分布于分布中心附近 20μs 以内，与分布中心偏差最大值在 100μs 以内。加负载后同步误差分布中心偏移为 80μs，主要分布于分布中心附近 100μs 以内，与分布中心偏差最大值在 200μs 以内。

分析数据发现，在无负载的情况下，虚拟机上运行 PTPd 软件的同步误差与物理机上接近，而增加负载后，虚拟机上 PTPd 软件同步误差明显增大，物理机上同步误差没有明显增大，这就说明在虚拟机上通过软件方式同步时同步效果受负载影响较大，这也验证了第三章中的理论分析：在较高负载下同步报文无法得到上层软件的及时处理，造成了时间戳误差增大，从而引起同步误差的剧烈波动。

5.3.3 单物理机上虚拟机软件同步测量实验

图 5.15 为单物理机上虚拟机间软件同步测量方案，该方案中测量一台物理机上两个虚拟机间的时间误差。同一物理机上的 VM1 和 VM2 通过 PTPd 软件同步，通过超调用方法每隔 2 秒读取同一板卡上的时间测量同步效果。测量结果如图 5.16 至图 5.19 所示。

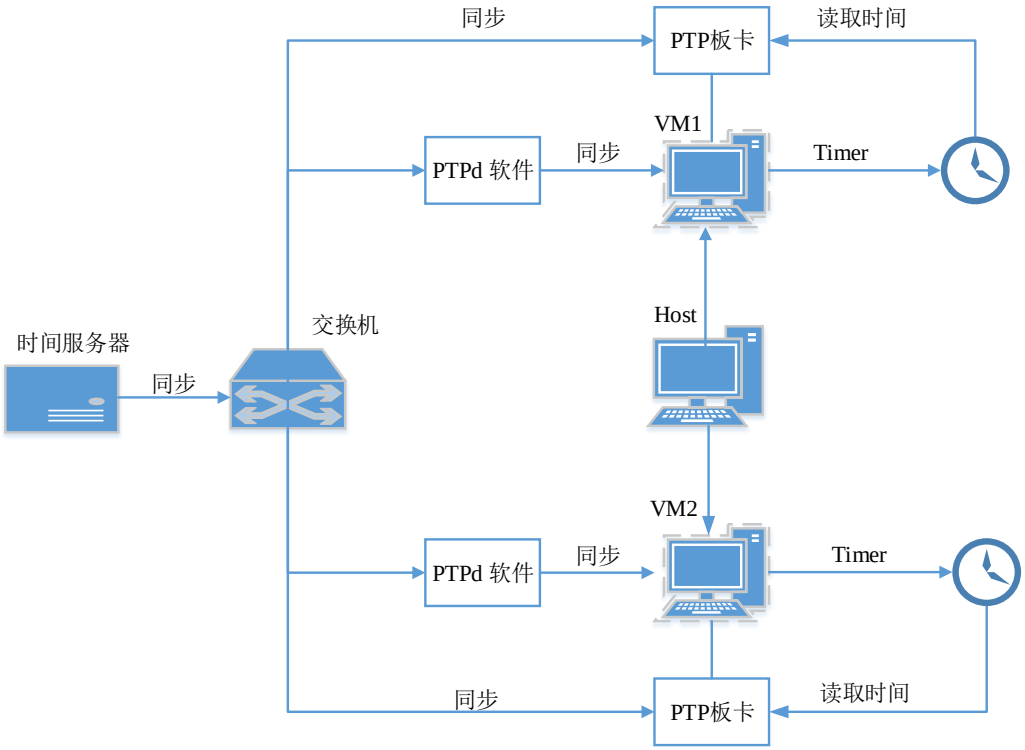

图5.15 单物理机上虚拟机间软件同步测量方案

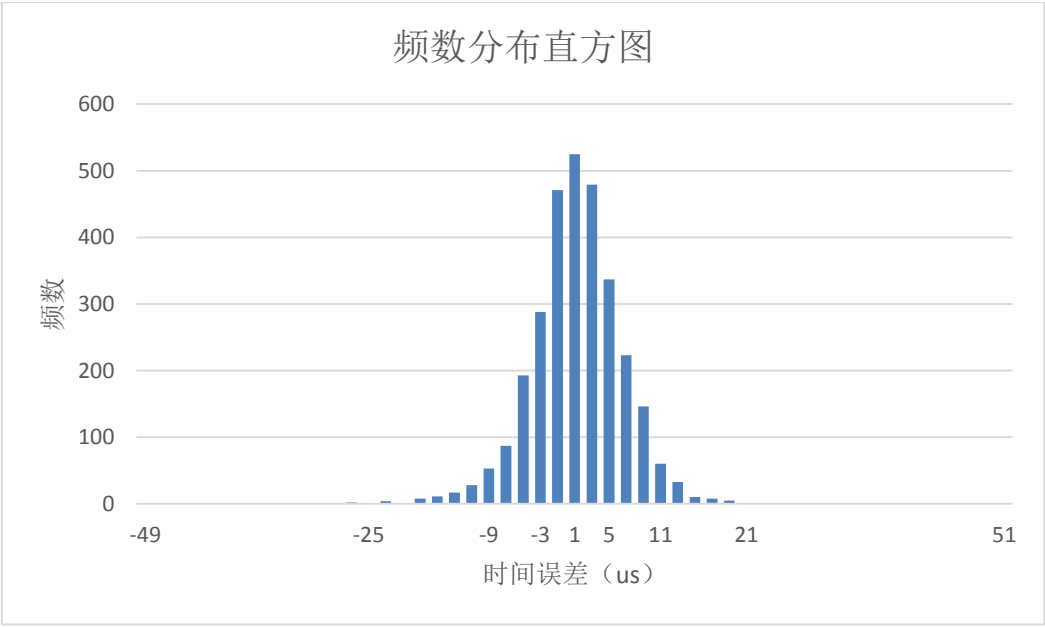

图5.16 单物理机上虚拟机间软件同步时间误差（未加负载）直方图

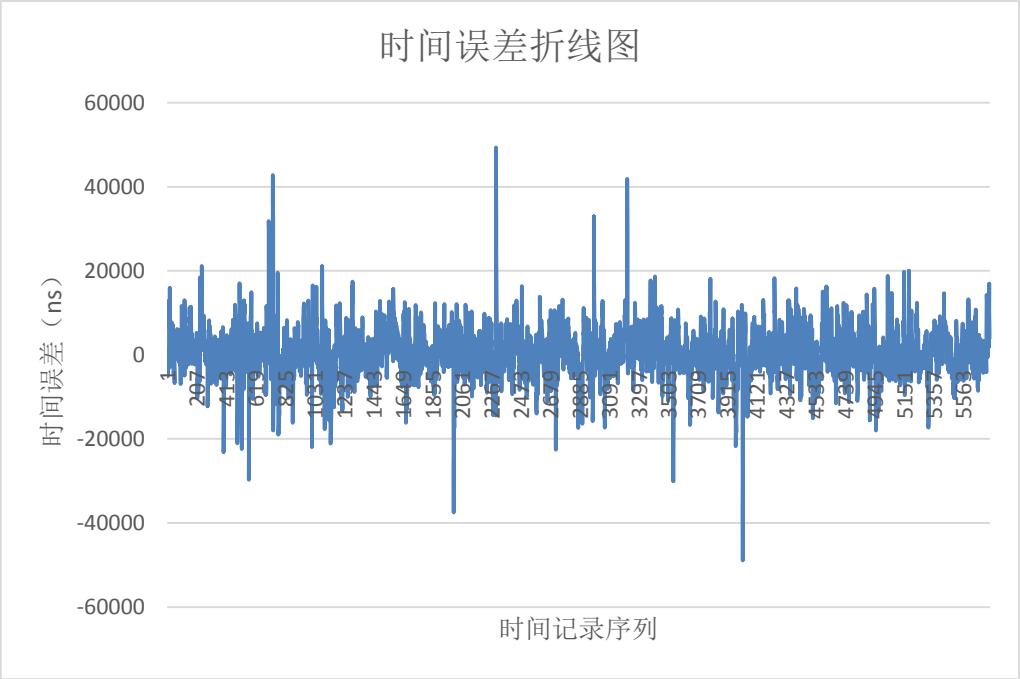

图5.17 单物理机上虚拟机间软件同步时间误差（未加负载）折线图

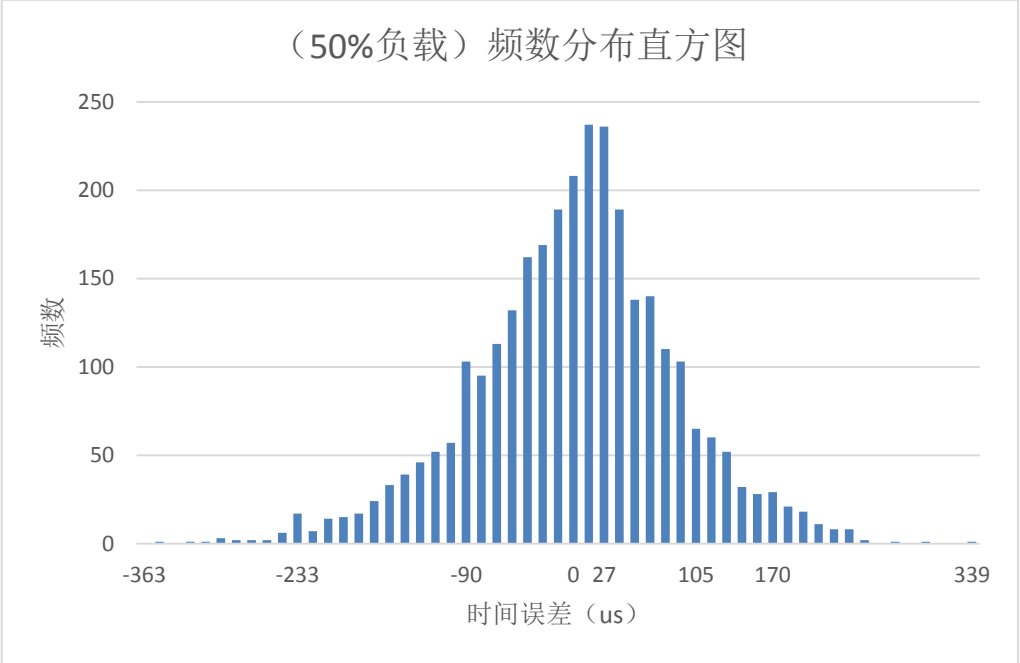

图5.18 单物理机上虚拟机间软件同步时间误差（加 50%负载）直方图

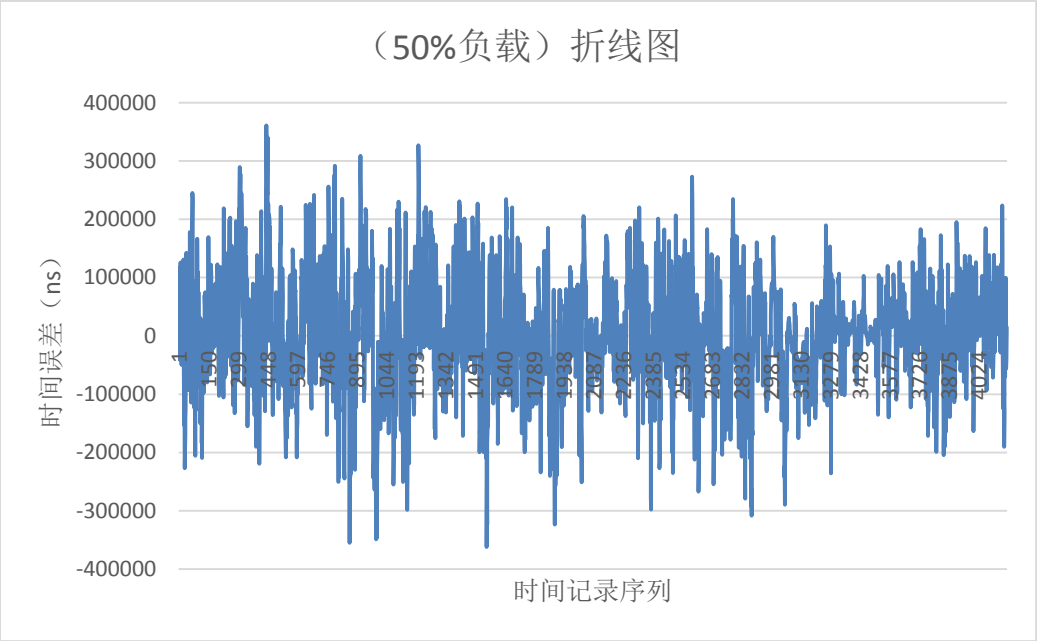

图5.19 单物理机上虚拟机间软件同步时间误差（加负载）折线图

如图 5.16 至图 5.19 所示，单物理机上多虚拟机间软件同步时间误差结果显示，单物理机上多虚拟机间使用 PTPd 软件同步误差分布中心偏差为 0μs，主要集中于分布中心附近 10μs 以内，偏差最大值在 50μs 以内。加负载后误差分布中心偏差为 0μs，主要集中于分布中心附近 200μs 以内，偏差最大值在 400μs 以内。

实验结果表明，单物理机上多虚拟机间使用 PTPd 软件同步的同步误差没有出现多物理机上虚拟机间同步误差的分布中心偏移，说明同一物理机上多虚拟机与主时钟的同步误差偏移相同，这证明分布中心偏移是由物理网卡收发数据时的时间戳固定误差引起的。在加负载后同步误差明显增大，再次证明在虚拟机上，高负载对 PTPd 软件报文时间戳的误差产生了较大的随机性影响。

5.4 测量结果总结

将上述 10 个实验中时间误差的均值、标准差以及绝对值最大值统计如表 5.1 所示：

表5.1时间误差绝对值统计表

实验名	实验平台	同步方法	CPU 负载	分布中心	标准差	最大误差
多物理机间硬件同步	物理机	硬件	无	≈0us	312us	18μs
多物理机间硬件同步	物理机	硬件	50%	≈0us	310us	16 μs
多物理机上虚拟机间硬件同步	虚拟机	硬件	无	≈0us	1097us	29 μs
多物理机上虚拟机间硬件同步	虚拟机	硬件	50%	≈0us	943us	77 μs
多物理机上软件同步	物理机	软件	无	86μs	2860us	104 μs
多物理机上软件同步	物理机	软件	50%	106μs	2500us	117 μs
多物理机上虚拟机间软件同步	虚拟机	软件	无	65μs	10316us	103 μs
多物理机上虚拟机间软件同步	虚拟机	软件	50%	80μs	48817us	223 μs
单物理机上虚拟机间软件同步	虚拟机	软件	无	≈0us	5621us	49 μs
单物理机上虚拟机间软件同步	虚拟机	软件	50%	≈0us	85859us	362 μs

综合上述图表及数据分析可证明：

（1）硬件同步精度高于 PTPd 软件同步精度。表 5.1 中各组实验中硬件同步的误差最大值和标准差都远小于软件同步，证明硬件同步的同步误差和波动性均小于软件同步，且在 50%负载情况下受影响较小。其原因是 IEEE1588 协议的同步精度取决于时间戳的精度，硬件 PTP 板卡上有一个硬件时钟，可在同步报文到达硬件 PTP 板卡后立即读取本地时钟得到精确时间戳，硬件同步方式时间戳不受上层计算机操作系统影响，其同步精度仅受操作系统上读取 PTP 板卡的时延影响，而如 3.5.4 小节所述，通过 PTPd 软件同步时，底层物理网卡收到同步报文后无法像硬件 PTP 板卡一样直接打上时间戳，需要等待上层的 PTPd 从时钟软件收到报文后读取 TSC 时间才能获得时间戳，这就带来了时间戳误差的不确定性，影响了同步精度。

（2）硬件同步精度受虚拟化平台影响较小。对比物理机和虚拟机上的硬件同步

误差发现，虚拟机平台对硬件同步方式的同步效果有一定影响，其误差产生原因主要是虚拟机超调用时延，但其同步误差的波动较小，且受 CPU 负载影响较小，可证明硬件同步的同步误差基本在 20μs 以内。

（3）软件同步精度受虚拟化平台影响，且 CPU 负载增加时影响较大。观察表 1 中软件同步方式实验结果发现，在无负载情况下，虚拟机上软件同步误差最大值与物理机上接近，但标准差较大，在有负载情况下误差最大值和标准差均明显增大，这证明虚拟机上使用 PTPd 软件同步时同步误差波动比物理机更大，尤其是在负载增大时，同步误差出现剧烈波动。其原因是相比于物理机操作系统，由于 VMM 的存在，虚拟机操作系统距底层物理网卡更远，而且由于 KVM 的 CPU 虚拟化原理，VMM 和其上多台虚拟机时分地共享物理 CPU，一台虚拟机无法得到所有的 CPU 时间，运行在虚拟机上的 PTPd 从时钟软件有时无法及时接收并处理同步报文，因此（1）中所述的软件时间戳误差就会更大，而在 CPU 负载增大时，从同步报文抵达物理网卡到虚拟机上层软件读取时间戳的时延就会进一步增大，这就带来了同步误差的明显波动。在本文实验场景中，PTPd 的同步精度在 400μs 以内。

（4）多物理机间 PTPd 软件同步误差出现固定的分布中心偏移。观察表 5.1 发现，处于多个物理计算机上的物理机和虚拟机操作系统使用软件同步时误差分布中心都不为 0，而硬件同步和单物理机上多虚拟机间软件同步的误差分布中心均约等于 0，这种固定的分布中心偏移是由主从时钟间交换网络的固定双向时延不对称以及不同计算机物理网卡时延不同造成的。在实际应用软硬件 PTP 同步方案时可使用 3.2 中介绍的边界时钟或透明时钟消除交换机双向时延不对称的影响。

第六章 总结与展望

本文结合本实验室承担的“1588 虚拟机对时系统研究”项目，致力于实现虚拟集群中任务的高精度同步测量方案。

本文第二章首先介绍了虚拟化技术，针对 KVM 虚拟化平台的系统结构和关键技术的实现原理进行了分析，并介绍了物理机和虚拟机操作系统的时间维持原理。第三章首先研究了 IEEE1588 协议的原理，介绍了基于 IEEE1588 协议的软硬件同步系统实现方案，然后对基于 IEEE1588 协议的硬件同步系统的同步精度进行了测量，验证硬件同步系统各板卡间的同步精度在 150ns 以内，从而证明了硬件 PTP 板卡可作为操作系统上任务级同步测量的标尺，最后从理论上分析了 IEEE1588 协议的实际应用时的误差产生原因和优化方案。

在第四章中，提出了基于超调用的虚拟机硬件 PTP 板卡时间读取方法，这一方法解决了同一物理机上不同虚拟机无法共享硬件 PTP 板卡的问题，从而使得了一台物理机上多虚拟机可同时通过一块硬件 PTP 板卡实现时间同步和测量，并通过超调用时延测量实验验证了虚拟机基于超调用的硬件 PTP 板卡测量方法误差在 20μs 以内。在这一方法的基础上，设计了基于硬件 PTP 板卡的虚拟集群任务同步测量方案，该方案可实现虚拟集群中各虚拟机和物理机上任务的高精度同步测量。为实现同步误差的直观显示，设计了可视化同步测量软件和相关方案，方便实际测量时的实时显示。

最后，在第五章中，基于提出的同步测量方案，通过实验对虚拟机上基于软硬件的同步方式的同步误差分别进行了测量，实验中通过本地记录方法对同步误差进行了长时间的测量，并在每组实验中加入了高 CPU 负载情况下的同步测量，实验结果显示基于硬件的同步方式的同步误差远远小于基于软件的同步方式，且同步效果更稳定，可达到接近物理机的同步效果，而在虚拟机上基于软件的同步方式在负载增加时同步误差明显增大，验证了第三章中的理论分析，但可验证其同步精度在 400μs 以内，可以用于一些同步精度要求较低的场景。

本文设计的虚拟集群任务测量方案可应用于KVM虚拟化平台虚拟机上的同步测量，在未来的工作中，可在此基础上研究适用于 Windows 等其他操作系统的同步测量方案。本文对虚拟机上基于 PTPd 软件的同步方案的同步误差进行了时延测量，发现高负载情况下虚拟机上 PTPd 软件的同步误差明显大于物理机，在未来，可研究 PTPd 软件针对虚拟化平台的优化方法，如通过 VMM 调度的优化及在 VMM 上透明网桥等方法降低虚拟化平台对 PTPd 软件的影响。此外，还应研究将本文提出的同步测量方案应用到各种同步系统的设备上，实现精度更高，更稳定的同步测量实施方案。

参考文献

[1] “IEEE 1588-2008: IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Syste.New York, NY, 2008.

[2] Chauhan J, Makaroff D, Arkles A. VM clock synchronization measurements[J].Performance Computing and Communications Conference. 2011:1-2.

[3] 王彦东, 邵英, 王黎明,等. 基于舰船综合平台的精确同步数据采集设计[J]. 舰船科学技术, 2015(4):70-75.

[4] D. L. Mills, “RFC-958: Network Time Protocol (NTP),” September,1985.

[5] IEEE, “IEEE 1588-2002: IEEE Standard for a Precision Clock Synchronization

[6] Lipinski M, Wlostowski T, Serrano J, et al. White rabbit: a PTP application for robust sub-nanosecond synchronization[C]// International IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication. IEEE, 2011:25-30.

[7] Ferrari P, Flammini A, Rinaldi S, et al. Evaluation of timestamping uncertainty in a software-based IEEE1588 implementation[C]// Conference Record - IEEE Instrumentation and Measurement Technology Conference. IEEE, 2011:604-609.

[8] 苏 宇,胡 珩,张 涛,等.基于 PTPd 的精准时钟同步技术的研究[J].计算机技术与发展，2016，26（1）：175-180.

[9] Henning S, Adam L, Alexander W.Faith Virtualization on a Real-Time Operating System[J]. RTLWS11, 2009.

[10] Whiteaker J, Schneider F, Teixeira R. Explaining packet delays under virtualization[J]. Acm Sigcomm Computer Communication Review, 2011, 41(1):38-44.

[11] Broomhead T, Cremean L, Ridoux J,et al.Virtualize everything but time[J]. Proceedings of Msenix Symposium on Operating Systems Design & Implementation. 2010: 1-6

[12] Cucinotta T, Anastasi G, Abeni L. Real-Time Virtual Machines[J]. 29TH IEEE REAL-TIME SYSTEMS SYMPOSIUM, WORK-IN-PROGRESS SESSION, 2008, 27(4):220.

[13] Zuo B, Chen K, Liang A, et al. Performance Tuning Towards a KVM-Based Low Latency Virtualization System[J]. IEEE, 2010:1-4.

[14] Xi S, Xu M, Lu C, et al. Real-time multi-core virtual machine scheduling in Xen[C]// International Conference. 2014:1-10.

[15] Kiszka J. Towards Linux as a Real-Time Hypervisor[J]. In Proc. of the 11th Real-Time Linux Workshop, 2009.

[16] Shea R, Wang F, Wang H, et al. A deep investigation into network performance in virtual

machine based cloud environments[C]//IEEE INFOCOM 2014 - IEEE Conference on Computer Communications. IEEE, 2014:1285-1293.

[17] Jagmohan C, Dwight Anthony A.Is Doing Clock Synchronization in a VM a Good Idea?. IEEE, 2016.

[18] Goldberg R.P., Survey of Virtual Machine Research, IEEE Computer, June 1974, 34-45.

[19] Rosenblum M., Garfinkel T. Virtual Machine Monitors: Current technology and future trends.IEEE Computer Society Press.Computer 38(5),2005:39-47.

[20] 陈鹏. 浅谈虚拟化技术进化[J]. 信息系统工程, 2010(4):102-103.

[21] Seawright L. H., MacKinnon R. A, VM/370-a study of multiplicity and μsefulness, IBM System Journal, 1979, 18 (1):4-17.

[22] Yang Z. A Summary of X86-based Virtualization Technology Research[J]. Journal of Luzhou Vocational Technical College, 2012.

[23] Waldspurger C,.Memory Resource Management in VMware ESX Server, Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI).ACM SIGOPS Operating Systems Review, 2002, Winter 2002 Special Issue: 181-194.

[24]]Kivity A, Kamay Y., Laor D, Lublin U. KVM: the Linux Virtual Machine Monitor.Linux Symposium, 2007, 225-230.

[25] Barham P., Xen and the Art of Virtualization, In: Proceedings of the ACM Symposium on Operating Systems Principles (SOSP), 2003.

[26] Intel .IntelVan Technology for IA-32 Processors(VT-x). Intel Corporation .2005.

[27] Intel.Intel Virtualization Technology for Directed I／O. Intel Corporation,2006.

[28] 英特尔开源软件技术中心. 系统虚拟化 ：原理与实现[M]. 清华大学出版社, 2009.

[29] 杨洪波. 高性能网络虚拟化技术研究[D]. 上海交通大学, 2012.

[30] Jones M T. Virtio: An I/O virtualization framework for Linux[J]. American Journal of Human Biology, 2010, 4(6):793-793.

[31] Han J, Jeong D K. A Practical Implementation of IEEE 1588-2008 Transparent Clock for Distributed Measurement and Control Systems[J]. IEEE Transactions on Instrumentation & Measurement, 2010, 59(2):433-439.

[32] 徐子昂. 1588v2 端到端透明时钟的分析与应用[D]. 西安电子科技大学, 2014.

[33] 苏宇，胡珩，张涛,等. 基于 PTPd 的精准时钟同步技术的研究[J]. 计算机技术与发展, 2016(1):175-180.

[34] Eidson J C. Measurement, Control, and Communication Msing IEEE 1588[M]. Springer London, 2006.

[35] Ng B H, Lau B, Prakash A. Direct Access to Graphics Card Leveraging VT-d Technical

Report[J]. 2009.

[36] Wartell R, Mohan V, Hamlen K W, et al. Binary stirring:self-randomizing instruction addresses of legacy x86 binary code[C]//ACM Conference on Computer and Communications Security. ACM, 2012:157-168.

[37] bossluo888.SyncMeasureProject.https://github.com/bossluo888/SyncMeasureProject.

致谢

在论文完成之时，我首先要衷心感谢的是我的导师姚明昨老师，在从大四开始的三年多时间里，姚老师在学术上和生活中都给予了我极大的指导和帮助，本文从选题到方案设计，再到技术实现，直至撰写论文中的每一步都离不开姚老师的悉心指导，不仅如此，在这三年多里我取得的每一点进步都离不开姚老师的栽培，在硕士期间的科研工作中，姚老师渊博的知识、非凡的智慧、迎难而上的精神和严谨治学的态度给我带留下了深刻的印象，也给我带来了深远的影响，在生活中，姚老师品德高尚，平易近人，是我终生学习的楷模。在此，我要对姚老师表达我最崇高的敬意和最真挚的感谢。

感谢刘增基教授和李峰老师多年来对我的巨大帮助和鼓励，感谢实验室的杨清海老师、沈中老师、王勇超老师和易湘老师为我提供良好的科研环境，感谢各位老师对我论文的悉心指导。

感谢我的父母，在我求学的道路上，你们一直都是我最坚强的后盾，你们永远都是我奋斗拼搏的动力，没有你们就没有我的成长，感谢你们给予我一如既往的支持和鼓励，感谢你们为我付出的牺牲。

感谢实验室的师兄刘吉龙博士和王啸阳博士，感谢张建、宋婉莹、李超、郭天昊、边策、张诚、刘涵宇、晏雷、常前锋、黄艳丽、苏宝霞、吴山丹等师兄师姐的指导，特别感谢刘涵宇师兄在项目中和生活上给予我的帮助和鼓励，感谢黄振林、高晨、史春燕和宋吉庆等同学三年多来对我的帮助，感谢钱玥妍、刘犇、茹旭隆、张亮、董林峰、李珂、杜静、许博文和王莹等师弟师妹对我的支持。

感谢舍友黄振林、贾栋和刘海峰在研究生期间的陪伴和帮助。

感谢西安电子科技大学七年来对我的培养。

最后，感谢所有在生活上，科研工作中以及撰写论文的过程中给予我指导和帮助的老师和同学。

作者简介

1. 基本情况

罗祥帆，男，北京人，1992 年 8 月出生，西安电子科技大学通信学院电子与通信工程专业 2014 级硕士研究生。

2. 教育背景

2010.08～2014.07 西安电子科技大学，本科，专业：信息工程
2014.08～ 西安电子科技大学，硕士研究生，专业：电子与通信工程

3. 攻读硕士学位期间的研究成果

3.1 申请（授权）专利

[1] 姚明昨, 罗祥帆, 黄振林,高晨. 基于网络自学习改进的电网时间同步测量系统和方法: 中华人民共和国,201710006019.0[P]. 2017.01.

3.2 参与科研项目及获奖

[1] 地铁列车数据传输项目, 2015.03~2015.11, 已完成, 完成上位机软件的设计、开发和调试。

[2] IEEE1588 虚拟机对时系统项目, 2015.11~2016.11, 已完成, 深入参与项目调研、开发、调试和现场测试。

西安電子科技大學
XIDIAN UNIVERSITY

地址：西安市太白南路2号
邮编：710071
网址：www.xidian.edu.cn